From Click Imitation to Transition Equivalence: Rethinking Supervision for GUI Agents

Zhiming Lin[1*] **Tianxiang Xu**[2*] **Zizhao Zhang**[3] **Yixue Liu**[4] **Yumeng Ma**[5]
Yihao Zhong[6] **Shaoyan He**[7] **Rongru Chen**[8] **Siyue Liu**[1†]
[1]Independent Researcher [2]Peking University
[3]University of Michigan–Ann Arbor
[4]Mellon College of Science, Carnegie Mellon University
[5]University of California, San Diego [6]New York University
[7]Ant Group Co., Ltd. [8]Jiangsu University

Abstract

GUI agents are increasingly expected to complete real tasks through language-based interaction, yet most supervised training still treats a single demonstrated click or action as the unique ground truth. This creates a mismatch between how agents are trained and how they are evaluated: task success depends on reaching the intended interface state, not on reproducing the demonstrator's exact surface action. We address this gap by introducing *Transition-Equivalent Supervision* (TES), a framework that trains GUI agents around goal-relevant state changes rather than raw action identity. TES converts demonstrated interactions into transition descriptors, supervises agents to predict the intended transition, and learns an executor that selects any verified action capable of realizing that transition. At inference time, the agent further verifies whether the executed action produces the intended state change, reducing repeated no-op or erroneous behavior. Across offline web prediction, executable web navigation, mobile control, and desktop-use benchmarks, TES consistently improves equivalent-step success, alternative-action recall, and online task completion while reducing NOOP/ERROR transitions. These results suggest that GUI-agent learning should move beyond click imitation toward transition-level reasoning, offering a more robust supervision principle for agents operating in diverse and dynamic interfaces.

1 Introduction

Graphical user interfaces (GUIs) are the primary medium through which humans operate digital services, from web search and online shopping to mobile applications and desktop productivity software. Building autonomous GUI agents is therefore a central step toward language-based computer use, where a user can specify an intent in natural language and delegate multi-step interaction to an agent [Xiao et al., 2026b]. This problem is challenging because successful GUI control requires joint reasoning over visual layouts, textual content, structured interface elements, and long-horizon state changes under dynamic and often underspecified environments [Nguyen et al., 2025, Fan et al., 2026].

Recent work has substantially advanced the evaluation and modeling of GUI agents. Web benchmarks have evolved from offline action prediction in Mind2Web [Deng et al., 2023] to executable environments such as WebArena [Zhou et al., 2024], visually grounded web tasks in VisualWebArena [Koh et al., 2024], and real-website navigation in WebVoyager [He et al., 2024]. Parallel efforts

*These authors contributed equally to this work.
†Corresponding author: lsy.canglang@gmail.com.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

extend GUI agents to mobile and desktop settings, including Android in the Wild [Rawles et al., 2023], AndroidWorld [Rawles et al., 2025], OSWorld [Xie et al., 2024], and WindowsAgentArena [Bonatti et al., 2025]. These benchmarks make GUI-agent evaluation increasingly execution-oriented: an agent is ultimately judged by whether it reaches the intended application state or completes the user goal.

However, the supervision used to train GUI agents often remains action-identity oriented. Many imitation-based systems learn to reproduce the recorded click, tap, coordinate, or action token in a demonstration, even though task success is defined by the resulting state change rather than the demonstrator's particular physical realization. GUI grounding models such as CogAgent [Hong et al., 2024], SeeClick [Cheng et al., 2024], UGround [Gou et al., 2025b], ShowUI [Lin et al., 2025], and OS-ATLAS [Wu et al., 2025] improve the ability to localize actionable interface elements, but their supervision still typically treats one element or coordinate as the target. Feedback-driven agents and exploration-based systems, including AgentOccam [Yang et al., 2025a], Agent S [Agashe et al., 2025], DigiRL [Bai et al., 2024], OpenWebVoyager [He et al., 2025], and Explorer [Pahuja et al., 2025], reduce reliance on static demonstrations by using execution feedback or mined trajectories; yet sparse success signals do not directly resolve which local alternatives preserve the same task progress. As a result, current methods face a fundamental mismatch: evaluation rewards state-level progress, while supervision often penalizes any deviation from a single observed surface action.

This mismatch is especially problematic in realistic GUIs, where multiple operations can induce the same meaningful transition. A search can be submitted by pressing Enter or clicking a button; a filter can be applied through different widgets; a dialog can be opened through several equivalent menu paths. Exact action imitation can therefore over-specify arbitrary demonstration choices, while naively treating visually nearby or semantically similar actions as positives can include harmful NoOp or erroneous operations. The central gap is thus not merely insufficient perception or insufficient exploration, but the absence of a supervision principle that separates task-relevant progress from incidental action realization.

This paper asks: *how can GUI agents learn from demonstrations without treating the demonstrated surface action as the only valid target?* We propose **Transition-Equivalent Supervision** (TES), a model-agnostic supervision framework that anchors learning to the goal-relevant state change induced by an action. Instead of asking an agent only to imitate what the demonstrator clicked, TES encourages the agent to reason about the intended transition, realize it through an executable action, and verify whether the post-action observation matches the intended progress.

Our contributions are summarized as follows:

- We identify and empirically diagnose an action-progress mismatch in GUI demonstrations: many GUI states admit multiple non-demonstrated actions that preserve the same task-relevant outcome, indicating that exact action identity is an unstable supervision target.
- We introduce TES, which reframes GUI supervision around transition equivalence rather than surface-action identity. This provides a principled bridge between demonstration learning and execution-based verification while remaining compatible with existing visual grounding and GUI-agent backbones.
- We show that transition-centered supervision improves the agent's ability to recognize valid alternative realizations, avoid harmful non-progress actions, and generalize beyond the specific interaction paths observed in demonstrations.

2 Related Work

Executable GUI-agent evaluation. Early web-agent environments such as World of Bits framed webpages as interactive environments with screen/DOM observations and keyboard/mouse actions [Shi et al., 2017]. Recent benchmarks broadened this line from offline web-action prediction in Mind2Web [Deng et al., 2023] to self-hosted or live web interaction in WebArena [Zhou et al., 2024], visually grounded web tasks in VisualWebArena [Koh et al., 2024], real-website execution in WebVoyager [He et al., 2024], and more realistic online evaluation in Online-Mind2Web [Xue et al., 2025]. WebLINX further studies multi-turn dialogue-conditioned navigation [Lu et al., 2024a], while Mind2Web 2 extends web-agent evaluation to long-horizon agentic search with judge-based assessment [Gou et al., 2025a]. Similar executable settings now cover mobile and desktop control,

including AITW [Rawles et al., 2023], AndroidWorld [Rawles et al., 2025], OSWorld [Xie et al., 2024], WindowsAgentArena [Bonatti et al., 2025], and WorkArena [Drouin et al., 2024]. Verified synthetic web environments further ground agent learning in executable backend states and task-progress predicates [Zhang et al., 2026a]. These benchmarks make task success closer to deployment, but supervised signals are still often recorded as one demonstrated next action or one successful trajectory. In contrast, TES treats an action as a realization of an intended state transition, thereby supervising task progress rather than path identity. In healthcare, episode-level evaluations track workflow-state continuity and test whether agents escalate when evidence is missing or conflicting [Yao et al., 2026, Zheng et al., 2026].

Grounding GUI observations into actions. As agents move beyond HTML-rich webpages, GUI research has increasingly focused on visual perception [Chang et al., 2025, Zhang et al., 2023, 2024a, 2026d] and grounding. ScreenAI and CogAgent improve screen/UI understanding [Baechler et al., 2024, Hong et al., 2024], while SeeClick identifies GUI grounding as a central bottleneck for screenshot-only agents [Cheng et al., 2024]; UGround further scales universal visual grounding across web, desktop, and mobile interfaces [Gou et al., 2025b]. More recent action models such as ShowUI and OS-ATLAS integrate grounding with action prediction and establish strong generalist GUI-agent backbones [Lin et al., 2025, Wu et al., 2025]. These models address where to act, but their training objectives usually reward agreement with a coordinate, element, or action token. This is too restrictive when multiple clicks, shortcuts, or UI paths induce the same task-relevant transition. TES is complementary to these backbones: stronger grounders can serve as executors, while transition-equivalent labels provide a less over-specified supervision target. Outside GUIs, Touch-Line Transformer combines language and gestural cues with geometric consistency to locate object referents [Li et al., 2023]. For multi-vendor OpenROADM networks, Xiao et al. [2026c] use YANG schemas to guide and validate LLM-generated configurations.

Beyond GUI control, hidden evidence-use forgetting shows that stable answer accuracy can conceal changes in multimodal grounding [Chen et al., 2026], motivating separate attention to evidence use and final outcomes [Liu et al., 2026, Sun et al., 2026]. In model provenance audits, pairwise similarity can indicate relatedness while leaving lineage direction unresolved, so Witness Overlap uses a third checkpoint to infer that direction [Li et al., 2026b].

Feedback-driven and adaptive GUI agents. Recent systems also improve GUI agents through better action-space design, exploration, or memory. AgentOccam shows that aligning observation and action spaces with LLM priors yields a strong zero-shot web agent [Yang et al., 2025a]; Agent S combines planning, memory, and experience retrieval for OS-level tasks [Agashe et al., 2025]; Mobile-Agent-v2 and GUI-explorer stabilize mobile interaction through collaboration or transition-aware exploration [Wang et al., 2024, Xie et al., 2025]. Training-oriented work such as DigiRL [Bai et al., 2024], OpenWebVoyager [He et al., 2025], and Explorer [Pahuja et al., 2025] uses reinforcement learning, real-world exploration, or synthetic trajectories to scale feedback. However, these approaches typically learn from sparse task success, mined trajectories, or exact action imitation, leaving local equivalence among successful actions implicit. TES makes this equivalence explicit by extracting goal-conditioned transition descriptors, training a transition-conditioned executor, and verifying whether the realized post-action state matches the intended progress. In medical decision support, Hu [2026] uses risk screening and evidence checks to defend agents against adversarial inputs before they issue recommendations.

Retrieval diversity also improves multi-turn intent understanding under fixed token budgets [Lin et al., 2026a], providing a complementary perspective on selecting context for language-based interaction. In histology, VOICE combines direct gene expression predictions with measurements retrieved from similar reference cells [Luo et al., 2026]. Beyond GUI tasks, CEC-Zero uses semantic similarity and candidate agreement to construct self-generated rewards for character error correction [Lin et al., 2026b], illustrating supervision beyond a single annotated output. In robotic assembly, graph transformers encode heterogeneous part relations to predict assembly sequences [Ma et al., 2023], offering a related perspective on structured action planning. In optical network diagnostics, Xiao et al. [2025] use multi-head graph attention to correlate alarms and identify the initiating alarm.

Structured and partially specified supervision. Affordance-First Decomposition separates stable interaction representations from adaptation in continual video-language understanding [Xu et al., 2026a]. COMPO-REALIGN uses reversible composition and multi-positive alignment for continual

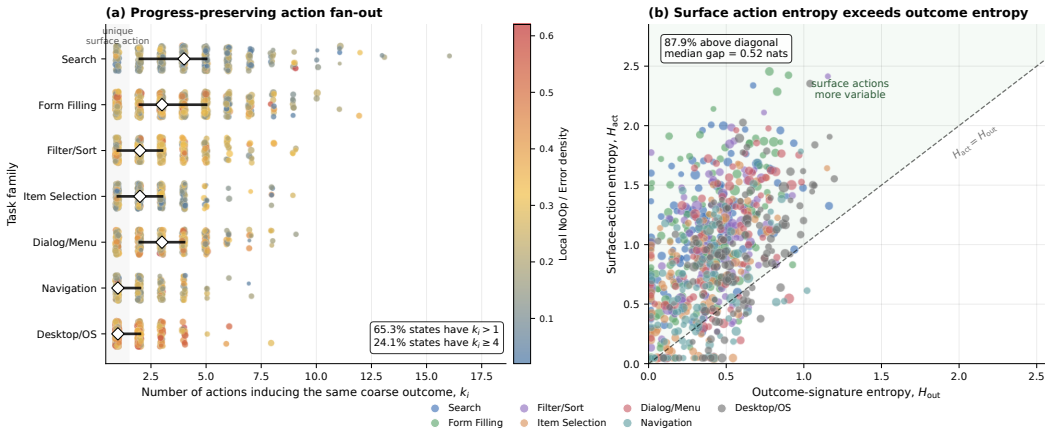


Figure 1: **Action identity is less stable than task progress. (a)** Many GUI states admit multiple actions that induce the same non-trivial outcome as the demonstration. **(b)** Surface-action entropy is usually higher than outcome-signature entropy, suggesting that exact click imitation over-specifies GUI supervision.

vision-language learning [Xiao et al., 2026a]. CoMem stores concepts and relations in a graph and rehearses in feature space [Zhou et al., 2026]. Text as Partial Constraint models captions as incomplete supervision through core-residual alignment [Yu et al., 2026]. In autonomous driving, Hint-AD aligns language with intermediate perception, prediction, and planning outputs [Ding et al., 2025], providing a related example of grounding semantic descriptions in structured system outputs. Prior work provides related perspectives on structuring supervision [Yu et al., 2026, Wang et al., 2026a, Li et al., 2026a, Zhang et al., 2026c] and enforcing structural constraints on foundation-model predictions [Wang et al., 2026b]. TES defines valid action realizations through verified GUI transitions.

3 Preliminary Study

Motivation. A GUI demonstration records one concrete surface action, e.g., a click, tap, shortcut, or text-entry confirmation. Yet task success depends on whether the GUI reaches the intended state, not on whether the agent exactly reproduces the demonstrator's action. We therefore ask: *is exact action identity a stable supervision target for GUI agents, or does it over-specify one arbitrary realization of task progress?*

Diagnostic setup. We conduct a replay-based diagnostic study without training any new model or using any component of our proposed method. For each demonstrated state-action transition, we replay a set of local candidate actions from the same GUI observation and compare their induced post-action changes with that of the demonstration. We summarize each post-action change by a coarse outcome signature covering navigation, visible-text, form-value, selection, focus, and completion changes. Full details of candidate construction and metric computation are provided in Appendix B.1.

We report two quantities. First, we measure the number k_i of distinct candidate actions that induce the same non-trivial outcome as the demonstrated action. Large k_i indicates that exact imitation would falsely penalize alternative actions that preserve task progress. Second, for each GUI template group G, we compare the entropy of surface action labels $H_{\text{act}}(G)$ with the entropy of outcome signatures $H_{\text{out}}(G)$. A positive gap $H_{\text{act}}(G) - H_{\text{out}}(G)$ indicates that demonstrators vary more in *how* they act than in *what state change* they induce.

Findings. Figure 1 shows two consistent patterns. **First**, progress-preserving action fan-out is common: 65.3% of states admit at least one non-demonstrated action with the same coarse outcome, and 24.1% admit at least four such realizations. This is especially visible in search, form filling, filtering, and dialog operations, where different GUI operations can realize the same task-relevant change. **Second**, action labels are substantially less stable than outcome changes. In 89.5% of GUI template groups, $H_{\text{act}}(G) > H_{\text{out}}(G)$, with a median entropy gap of 0.54 nats. However, Figure 1(a) also shows non-negligible NoOp/Error density around dense GUI states. Thus, one should not

simply treat all nearby or visually similar actions as positives; valid alternatives must be distinguished from harmful actions by their realized state change.

Implication. This study suggests that GUI supervision should be anchored to task-relevant transition progress rather than exact surface-action identity. This motivates the next section, where we introduce *Transition-Equivalent Supervision*: demonstrated transitions are converted into goal-conditioned transition descriptors, the agent predicts the intended transition before selecting an executable action, and execution is verified by comparing the realized post-action change with the intended progress.

4 Method

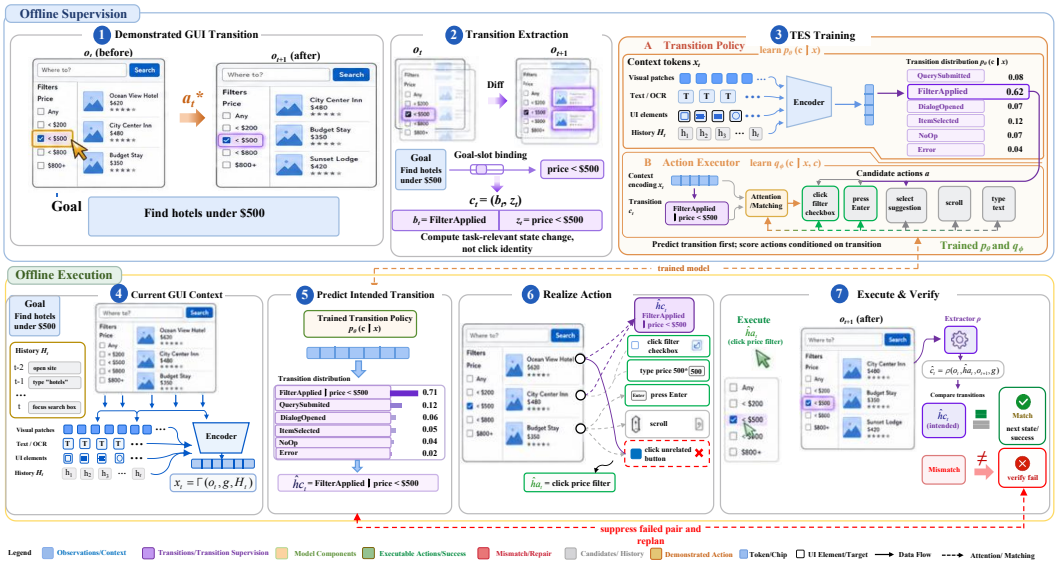


Figure 2: Pipeline of Transition-Equivalent Supervision. Demonstrated GUI transitions are converted into goal-conditioned transition descriptors. The agent learns to predict the intended transition and then selects an executable action that realizes it. Verification compares the intended transition with the realized transition after execution.

We propose **Transition-Equivalent Supervision** (TES), a supervision framework for GUI agents that learns task progress rather than click identity. Given a demonstrated GUI transition, TES extracts a goal-conditioned transition descriptor that specifies the semantic state change produced by the action. The agent is then trained to predict this transition descriptor before selecting a concrete executable action. At inference time, the predicted transition constrains action selection, and the post-action observation verifies whether the intended state change has been realized. Figure 2 summarizes the framework.

4.1 GUI Interaction and Click-Imitation Mismatch

A GUI task is specified by a goal g. At step t, the agent observes a GUI observation $o_t = (v_t, \ell_t, r_t)$, where v_t is the visual screen representation, ℓ_t is the visible text representation, and r_t is the structured interface representation. A benchmark-specific mask marks unavailable views as null inputs. The interaction history is denoted by $\mathcal{H}_t = \{(o_i, a_i, o_{i+1})\}_{i<t}$. The decision context and executable action are represented as

$$x_t = \Gamma(o_t, g, \mathcal{H}_t), \qquad a_t = (\kappa_t, u_t, \chi_t) \in \mathcal{A}(o_t), \tag{1}$$

where Γ is the GUI context encoder, κ_t is the primitive operation type, u_t is the target descriptor, χ_t is the action argument, and $\mathcal{A}(o_t)$ is the finite candidate action set.

The training data consists of demonstrated trajectories

$$\mathcal{D} = \left\{ \zeta^{(n)} = (g^{(n)}, o_0^{(n)}, a_0^{(n)\star}, o_1^{(n)}, \ldots, a_{T_n-1}^{(n)\star}, o_{T_n}^{(n)}) \right\}_{n=1}^{N}, \qquad M = \sum_{n=1}^{N} T_n. \tag{2}$$

Here M is the number of demonstrated state-action-state transitions. Standard click imitation optimizes

$$\mathcal{L}_{\text{act}}(\theta) = -\frac{1}{M} \sum_{n=1}^{N} \sum_{t=0}^{T_n-1} \log \pi_\theta(a_t^{(n)\star} \mid x_t^{(n)}). \tag{3}$$

This objective assigns probability mass to one recorded surface action. In GUI tasks, the same goal-relevant progress is often produced by several distinct clicks, taps, shortcuts, and navigation paths. Exact action matching therefore confounds task progress with demonstrator-specific realization.

4.2 Task-Relevant Transition Abstraction

TES replaces action identity with a goal-conditioned transition descriptor. For each demonstrated tuple $(g, o_t, a_t^\star, o_{t+1})$, we extract

$$c_t = \rho(o_t, a_t^\star, o_{t+1}, g) = (b_t, z_t) \in \mathcal{C}, \tag{4}$$

where ρ is the transition extractor, $b_t \in \mathcal{B}$ is the transition type, and z_t is the goal-relevant semantic slot. The type b_t captures the form of progress, such as QUERYSUBMITTED, FILTERAPPLIED, DIALOGOPENED, ITEMSELECTED, FORMSAVED, PAGENAVIGATED, NOOP, and ERROR. The slot z_t specifies the relevant content of the transition, such as the submitted query, the selected item, the applied filter value, or the opened dialog name. Thus, $(\text{FILTERAPPLIED}, \texttt{price} < 500)$ and $(\text{FILTERAPPLIED}, \texttt{rating} > 4)$ are different transition descriptors.

The extractor ρ uses observable changes between o_t and o_{t+1}, including visual differences, visible-text changes, structured-element changes, navigation changes, focus/selection changes, and task success signals. The deterministic extraction rules and soft-label construction are given in Appendix A.1. This design makes the learning target describe what changed for the task, not how the demonstrated user physically caused the change. Rules and a fixed vocabulary are implementation choices: learned and visual-only extractors are evaluated in Appendix D.

4.3 Transition Equivalence

For a fixed observation-goal pair (o, g), two executable actions are transition-equivalent when their observed post-action states yield the same transition descriptor:

$$a_i \sim_{o,g} a_j \quad \Longleftrightarrow \quad \rho(o, a_i, o_i^+, g) = \rho(o, a_j, o_j^+, g), \tag{5}$$

where o_i^+ and o_j^+ are the post-action observations obtained by executing a_i and a_j from o. This definition avoids assuming a deterministic environment transition function. The distributional extension for stochastic GUI environments is provided in Appendix A.3.

Xu et al. [2026b] distinguish changes in executed plans from changes in operational outcomes. They compare independent runs to check agreement in execution and use a specified tolerance to assess cost equivalence, which can hold even when the selected policies differ. TES makes this distinction at the level of one GUI step for a fixed observation and goal. Pressing Enter and clicking Search are equivalent if both produce the same QUERYSUBMITTED descriptor, while submitting different queries yields distinct descriptors even if both actions open a results page.

Each transition descriptor induces a set of valid realizations:

$$\mathcal{A}_c(o, g) = \left\{ a \in \mathcal{A}(o) : \rho(o, a, o^+(a), g) = c \right\}, \tag{6}$$

where $o^+(a)$ is the observed state after executing a from o. The demonstrated action $a_t^\star$ is one positive realization of c_t, not the only valid target. The verified positive set $\mathcal{P}_t \subseteq \mathcal{A}_{c_t}(o_t, g)$ always contains $a_t^\star$ and is expanded through environment replay and deterministic GUI rules described in Appendix A.2.

4.4 Transition-Equivalent Supervision

TES factorizes GUI decision making into transition prediction and transition-conditioned action realization. The transition policy predicts the intended state change, while the executor scores candidate actions under that intended transition:

$$p_\theta(c \mid x_t) = \mathrm{softmax}(W_c f_\theta(x_t))_c, \qquad q_\phi(a \mid x_t, c) = \frac{\exp r_\phi(x_t, a, c)}{\sum_{a' \in \mathcal{A}(o_t)} \exp r_\phi(x_t, a', c)}. \tag{7}$$

Here f_θ is the GUI context encoder, W_c maps the context representation to transition logits, and r_ϕ is the executor score for realizing transition c with action a.

Let $y_{t,c}^{(n)}$ denote the transition target distribution produced by the extractor for the t-th transition in trajectory n. Deterministic extraction yields a one-hot target; ambiguous extraction yields a calibrated soft target as defined in Appendix A.1. The transition loss is

$$\mathcal{L}_{\mathrm{trans}}(\theta) = -\frac{1}{M} \sum_{n=1}^{N} \sum_{t=0}^{T_n - 1} \sum_{c \in \mathcal{C}} y_{t,c}^{(n)} \log p_\theta(c \mid x_t^{(n)}). \tag{8}$$

The executor is trained with all verified positive realizations:

$$\mathcal{L}_{\mathrm{exec}}(\phi) = -\frac{1}{M} \sum_{n=1}^{N} \sum_{t=0}^{T_n - 1} \log \sum_{a \in \mathcal{P}_t^{(n)}} q_\phi(a \mid x_t^{(n)}, c_t^{(n)}). \tag{9}$$

The final objective is

$$\mathcal{L}_{\mathrm{TES}}(\theta, \phi) = \mathcal{L}_{\mathrm{trans}}(\theta) + \lambda_{\mathrm{exec}} \mathcal{L}_{\mathrm{exec}}(\phi), \tag{10}$$

where λ_{exec} controls the contribution of action realization. The transition loss trains the agent to recognize the intended state change; the executor loss trains the agent to realize that state change through available GUI actions.

4.5 Transition-Conditioned Execution and Verification

At inference time, the agent jointly selects an intended transition and a concrete action:

$$(\hat{c}_t, \hat{a}_t) = \underset{c \in \mathcal{C},\, a \in \mathcal{A}(o_t)}{\arg\max} \left[\log p_\theta(c \mid x_t) + \log q_\phi(a \mid x_t, c) \right]. \tag{11}$$

After executing $\hat{a}_t$, the agent observes o_{t+1} and extracts the realized transition:

$$\bar{c}_t = \rho(o_t, \hat{a}_t, o_{t+1}, g), \qquad m_t = \mathbf{1}[\bar{c}_t = \hat{c}_t]\, \mathbf{1}[\bar{c}_t \notin \mathcal{C}_{\mathrm{bad}}], \tag{12}$$

where $\mathcal{C}_{\mathrm{bad}} = \{\textsc{NoOp}, \textsc{Error}\}$. If $m_t = 1$, the step is accepted and the history is updated with $(o_t, \hat{a}_t, o_{t+1})$. If $m_t = 0$, the failed transition-action pair is recorded and the next decoding step suppresses repeated failures. The repair scoring rule is given in Appendix A.5.

5 Experiments

5.1 Experimental Setup

Benchmarks and metrics. We evaluate TES across four GUI-agent settings: offline web action prediction, executable web navigation, mobile control, and desktop computer use. Specifically, we use Mind2Web-style offline benchmarks [Deng et al., 2023, Pahuja et al., 2025], executable web environments [Zhou et al., 2024, Koh et al., 2024, He et al., 2024, Xue et al., 2025, Gou et al., 2025a], mobile-control benchmarks [Rawles et al., 2023, 2025, Lu et al., 2025, Chai et al., 2025], and desktop/general-computer benchmarks [Xie et al., 2024, Bonatti et al., 2025, Drouin et al., 2024, Nayak et al., 2025, Li et al., 2025]. We report standard GUI-agent metrics, including element accuracy, operation F1, step success, task success, grounding accuracy, completion rate, and execution cost. Since TES targets transition-equivalent supervision rather than exact action identity, we additionally report transition accuracy, equivalent-step success, alternative-action recall, NoOp/Error rate, and verification agreement. Full benchmark coverage and metric definitions are given in Appendix B.2.

Table 1: **Main offline, grounding, and mobile-control results.** MM-M2W denotes Multimodal-Mind2Web. CT/CW/CD denote cross-task, cross-website, and cross-domain splits. SS-Pro and SS-v2 denote ScreenSpot-Pro and ScreenSpot-v2 grounding accuracy. AC-H denotes AndroidControl-High success rate, and GUI-Ody denotes GUI-Odyssey success rate. EqStep measures transition-equivalent step success, while BadRate measures realized NOOP/ERROR transitions. Best and second-best results are bold and underlined, respectively; TES rows are shaded light blue.

Method	**Backbone**	**MM-M2W Step SR ↑**			**Grounding Acc. ↑**		**Mobile Step SR ↑**		**TES Metrics**	
		CT	CW	CD	SS-Pro	SS-v2	AC-H	GUI-Ody	EqStep ↑	BadRate ↓
Action imitation and general GUI grounding/action models										
Action Imit.	Qwen2.5-VL-7B	62.9	58.5	57.0	28.0	86.3	69.2	74.7	65.6	12.7
CogAgent [Hong et al., 2024]	CogAgent-18B	62.3	54.0	59.4	7.7	54.9	48.7	42.0	58.8	16.5
SeeClick [Cheng et al., 2024]	Qwen-VL-9.6B	41.4	37.8	39.7	1.1	55.1	59.1	53.9	55.5	18.8
ShowUI [Lin et al., 2025]	Qwen2-VL-2B	48.7	43.6	45.3	7.7	71.2	63.7	58.6	59.5	15.4
Aria-UI [Yang et al., 2025b]	Aria-UI-3B	55.1	51.5	52.7	11.3	78.5	10.2	36.5	61.0	14.1
UGround [Gou et al., 2025b]	Qwen2-VL-7B	58.5	54.2	55.8	31.1	87.6	65.2	60.9	64.3	13.0
OS-ATLAS [Wu et al., 2025]	OS-ATLAS-7B	61.8	57.3	58.8	18.9	84.1	71.2	62.0	66.6	11.6
Aguvis [Xu et al., 2024]	Aguvis-7B	60.4	54.6	56.6	22.9	86.0	61.5	64.3	66.8	11.5
Aguvis [Xu et al., 2024]	Aguvis-72B	64.0	56.5	58.2	34.5	88.2	66.4	70.4	69.7	9.7
UI-TARS [Qin et al., 2025]	Qwen2-VL-7B	67.1	61.7	60.5	35.7	91.6	72.5	87.0	72.5	8.2
UI-TARS [Qin et al., 2025]	Qwen2-VL-72B	68.6	63.5	62.1	38.1	90.3	74.7	88.6	73.0	7.8
GUI-Actor+Verifier	Qwen2.5-VL-7B	66.1	62.8	61.9	47.7	92.5	73.8	86.5	72.6	8.0
POINTS-GUI-G [Zhao et al., 2026]	POINTS-7B	67.9	63.2	62.5	46.5	92.1	74.1	87.9	73.6	7.7
Ours: Transition-Equivalent Supervision										
TES	Phi-3.5-V-4B	69.4	64.7	63.5	40.2	91.9	76.2	89.8	76.7	6.8
TES	Qwen2.5-VL-7B	<u>72.7</u>	<u>68.4</u>	<u>67.0</u>	<u>49.2</u>	<u>93.4</u>	<u>78.4</u>	<u>91.3</u>	<u>81.8</u>	<u>4.9</u>
TES	OS-ATLAS-7B	**74.0**	**69.7**	**68.2**	**50.8**	**94.1**	**80.3**	**92.5**	**83.5**	**4.2**

Comparison protocol. We compare TES with direct action imitation, GUI grounding/action models, feedback-driven agents, exploration-based agents, and recent generalist GUI agents [Hong et al., 2024, Cheng et al., 2024, Gou et al., 2025b, Lin et al., 2025, Wu et al., 2025, Yang et al., 2025a, Agashe et al., 2025, Bai et al., 2024, Xie et al., 2025, He et al., 2025, Pahuja et al., 2025]. We also include ablations that remove transition prediction, verified positive sets, replay positives, rule positives, soft transition targets, verification, and repair memory. To avoid unfair comparisons across heterogeneous GUI agents, we report results separately under screenshot-only, structured-observation, parser-augmented, and closed-source API-reference regimes. All trainable methods use matched train/validation splits, candidate-action budgets, maximum step budgets, and evaluation scripts whenever the benchmark allows [Yang et al., 2026b]. Heterogeneous systems provide context; attribution to TES rests on comparisons with matched backbones, demonstrations, candidate generators, and training/execution budgets. Details are provided in Appendix B.3.

Backbones and implementation. We instantiate TES on three representative backbones: Qwen2.5-VL-7B-Instruct [Bai et al., 2025], Phi-3.5-Vision-Instruct [Abdin et al., 2024], and OS-ATLAS-7B [Wu et al., 2025]. For each backbone, we train both a direct action-imitation baseline and a TES variant under the same data and optimization budget. Unless otherwise specified, we fine-tune with LoRA adapters and train only the TES transition/executor heads plus lightweight adapter parameters. Offline experiments are run with three random seeds, while online execution results are reported with bootstrap confidence intervals. Implementation details, hyperparameters, candidate-action construction, and compute resources are described in Appendix B.4.

5.2 Main Results

Overall comparison. Tables 1 and 2 show that TES consistently improves over direct action imitation and strong GUI-agent baselines across offline, mobile, web, and desktop settings. The gains are most pronounced on transition-sensitive metrics, including EqStep, VAgree, and BadRate, indicating that modeling goal-relevant state changes provides a more stable target than exact surface-action matching. TES also benefits both general VLM and GUI-specialized backbones: the OS-ATLAS-based variant performs best, while the Qwen2.5-VL variant remains consistently competitive.

Table 2: **Main online execution results.** We report task success rate on executable web, mobile, and desktop benchmarks. WV denotes WebVoyager, VWA denotes VisualWebArena, AW denotes AndroidWorld, OSW denotes OSWorld, and WAA denotes WindowsAgentArena. VAgree is verification agreement between intended and realized transitions.

Method	Backbone	WebArena ↑	WV ↑	VWA ↑	AITW ↑	AW ↑	OSW ↑	WAA ↑	VAgree ↑	Steps ↓
Prompted, feedback-driven, and exploration-based agents										
WebVoyager [He et al., 2024]	GPT-4V	31.9	59.1	15.2	—	—	—	—	61.6	13.3
OpenWebVoyager [He et al., 2025]	Idefics2-8B	23.5	27.4	13.5	—	—	—	—	58.8	12.8
AgentOccam [Yang et al., 2025a]	GPT-4-Turbo	43.1	55.7	18.5	—	—	—	—	64.7	11.5
Agent S [Agashe et al., 2025]	GPT-4o	—	—	—	—	—	20.6	18.2	66.0	14.7
Mobile-Agent-v2 [Wang et al., 2024]	GPT-4o	—	—	—	61.3	42.5	—	—	63.3	15.0
DigiRL [Bai et al., 2024]	PaLI-X-1.5B	—	—	—	67.2	0.9	—	—	55.8	16.4
GUI-Explorer [Xie et al., 2025]	GPT-4o	—	—	—	—	47.4	—	—	69.3	13.7
RAG-GUI [Xu et al., 2025]	GPT-4o	—	—	—	—	49.4	—	—	70.9	13.0
Generalist GUI action models										
GPT-4o + UGround [Gou et al., 2025b]	GPT-4o	39.4	52.3	18.0	—	32.8	—	—	65.1	13.0
GPT-4o + Aria-UI [Yang et al., 2025b]	GPT-4o	40.8	53.7	18.7	—	44.8	15.2	—	66.8	12.7
GPT-4o + OS-ATLAS [Wu et al., 2025]	GPT-4o	41.1	54.2	18.8	—	41.6	14.6	—	66.3	12.9
Aguvis [Xu et al., 2024]	Aguvis-72B	38.7	51.0	17.1	—	26.1	10.3	—	63.2	14.3
UI-TARS-SFT [Qin et al., 2025]	Qwen2-VL-7B	42.3	56.3	19.6	69.9	33.0	17.7	—	70.5	12.0
UI-TARS-DPO [Qin et al., 2025]	Qwen2-VL-7B	43.6	56.9	20.0	70.3	35.9	18.7	—	71.1	11.7
UI-TARS-SFT [Qin et al., 2025]	Qwen2-VL-72B	46.1	60.2	21.5	72.0	46.6	18.8	—	73.5	11.2
UI-TARS-DPO [Qin et al., 2025]	Qwen2-VL-72B	47.0	61.3	22.0	72.7	48.4	22.7	—	74.3	10.8
OpenCUA [Wang et al., 2025]	OpenCUA-32B	49.6	62.5	22.8	73.5	51.9	34.8	22.5	75.0	10.6
Ours: Transition-Equivalent Supervision										
TES	Phi-3.5-V-4B	51.8	63.7	23.6	73.9	52.1	33.2	23.1	77.9	10.4
TES	Qwen2.5-VL-7B	54.6	65.4	25.2	75.6	54.8	36.9	25.9	81.6	9.8
TES	OS-ATLAS-7B	**56.1**	**66.8**	**26.4**	**76.8**	**56.3**	**38.6**	**27.4**	**83.2**	**9.3**

Table 3: **Single-factor ablation of TES.** All variants use OS-ATLAS-7B as the backbone. CT/CW/CD denote the three Multimodal-Mind2Web splits. EqStep measures transition-equivalent step success, VAgree measures agreement between intended and realized transitions, BadRate measures NOOP/ERROR transitions, and Steps denotes average online execution steps. Blue numbers indicate degradation relative to the full model.

Variant	MM-M2W Step SR ↑			WebArena ↑	AndroidWorld ↑	OSWorld ↑	EqStep ↑	VAgree ↑	BadRate ↓
	CT	CW	CD						
Full TES	**74.0**	**69.7**	**68.2**	**56.1**	**56.3**	**38.6**	**83.5**	**83.2**	**4.2**
w/o transition prediction	67.5 (↓6.5)	62.9 (↓6.8)	61.4 (↓6.8)	50.3 (↓5.8)	50.8 (↓5.5)	34.3 (↓4.3)	73.3 (↓10.2)	71.7 (↓11.5)	7.8 (↑3.6)
w/o transition slot z_t	70.9 (↓3.1)	66.3 (↓3.4)	64.9 (↓3.3)	53.2 (↓2.9)	53.6 (↓2.7)	36.1 (↓2.5)	78.9 (↓4.6)	78.6 (↓4.6)	5.9 (↑1.7)
w/o transition-conditioned executor	68.6 (↓5.4)	64.1 (↓5.6)	62.7 (↓5.5)	51.8 (↓4.3)	51.9 (↓4.4)	35.2 (↓3.4)	75.1 (↓8.4)	74.9 (↓8.3)	7.0 (↑2.8)
w/o positive-set marginalization	69.8 (↓4.2)	65.4 (↓4.3)	64.0 (↓4.2)	52.6 (↓3.5)	52.8 (↓3.5)	35.9 (↓2.7)	77.1 (↓6.4)	76.8 (↓6.4)	6.4 (↑2.2)
w/o replay positives	71.8 (↓2.2)	67.4 (↓2.3)	65.8 (↓2.4)	53.9 (↓2.2)	54.1 (↓2.2)	36.7 (↓1.9)	80.3 (↓3.2)	80.1 (↓3.1)	5.3 (↑1.1)
w/o rule positives	72.4 (↓1.6)	67.9 (↓1.8)	66.3 (↓1.9)	54.4 (↓1.7)	54.6 (↓1.7)	37.1 (↓1.5)	81.0 (↓2.5)	80.8 (↓2.4)	5.0 (↑0.8)
w/o soft transition targets	72.2 (↓1.8)	68.0 (↓1.7)	66.5 (↓1.7)	54.9 (↓1.2)	54.8 (↓1.5)	37.3 (↓1.3)	81.2 (↓2.3)	81.1 (↓2.1)	4.9 (↑0.7)
w/o verification	71.1 (↓2.9)	66.6 (↓3.1)	65.0 (↓3.2)	51.9 (↓4.2)	52.6 (↓3.7)	35.0 (↓3.6)	78.0 (↓5.5)	75.5 (↓7.7)	7.1 (↑2.9)
w/o repair memory	72.6 (↓1.4)	68.2 (↓1.5)	66.7 (↓1.5)	54.0 (↓2.1)	54.2 (↓2.1)	36.6 (↓2.0)	80.8 (↓2.7)	80.0 (↓3.2)	5.7 (↑1.5)
w/o bad-transition filtering	71.5 (↓2.5)	67.0 (↓2.7)	65.6 (↓2.6)	52.8 (↓3.3)	53.0 (↓3.3)	35.7 (↓2.9)	78.4 (↓5.1)	77.9 (↓5.3)	7.4 (↑3.2)

5.3 Ablation and Analysis

We ablate one TES component at a time under the same OS-ATLAS-7B backbone, data, candidate budget, and inference budget; blue numbers in Table 3 denote degradation from the full model. The results show that TES mainly benefits from three factors: transition intent modeling, multi-positive action realization, and execution-time verification. Removing transition prediction causes the largest drop, reducing EqStep from 83.5 to 73.3 and VAgree from 83.2 to 71.7. Removing the transition-conditioned executor or positive-set marginalization also substantially hurts EqStep by 8.4 and 6.4 points, respectively. Verification-related modules improve online reliability, with BadRate rising from 4.2 to 7.4 without bad-transition filtering, validating the predict–realize–verify design of TES.

5.4 Does TES help more when action fan-out is larger?

If action-level imitation is mismatched because multiple surface actions can realize the same task-relevant transition, TES should be most beneficial when the verified action fan-out is large. We group held-out GUI states by action fan-out $f_i = |\mathcal{P}_i|$, where $\mathcal{P}_i$ is the verified positive set, and compare Action Imitation with TES using the same OS-ATLAS-7B backbone, training data, candidate budget,

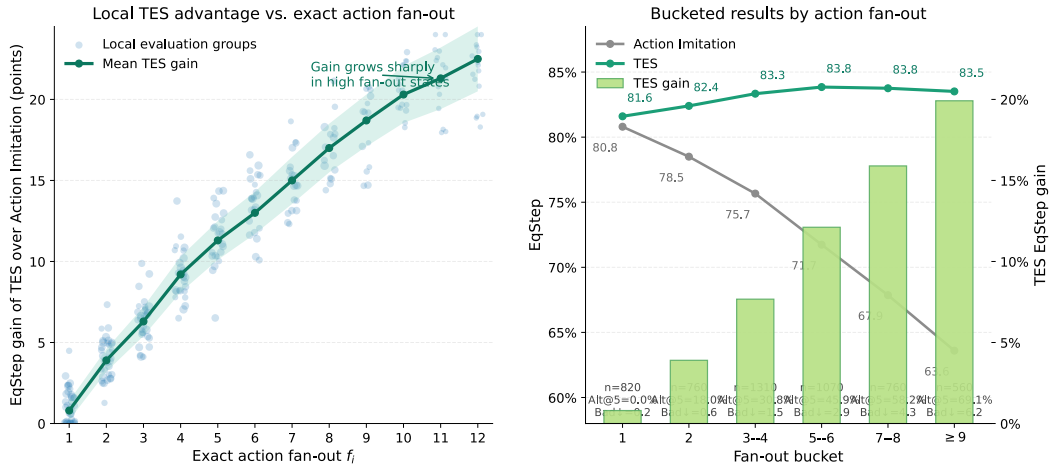


Figure 3: **TES benefits more when action fan-out is larger.** Left: local EqStep gain of TES over Action Imitation across exact fan-out values. Right: bucketed EqStep comparison, with green bars showing TES gain and annotations reporting sample size, AltRecall@5, and BadRate reduction. Details are in the action fan-out analysis paragraph of Appendix B.6.

and inference budget. We report EqStep as the primary metric, together with AltRecall@5 and BadRate reduction; full protocol details are provided in the action fan-out analysis paragraph of Appendix B.6.

Results and interpretation. Figure 3 shows that TES brings only a small gain when the demonstrated action is nearly unique, improving EqStep by $+0.8$ points at $f_i = 1$. The gain increases steadily with action fan-out, reaching $+7.7$, $+12.1$, $+15.9$, and $+19.9$ points for the 3–4, 5–6, 7–8, and ≥ 9 buckets, respectively. Meanwhile, AltRecall@5 rises to 69.0% in the highest-fan-out bucket, and BadRate reduction grows from 0.2 to 6.1 points. This supports the central hypothesis: TES is most useful precisely when exact click imitation over-specifies one arbitrary realization of a broader transition-equivalence class.

Extractor dependence and replay cost. Controlled tests retain 82.9 EqStep with a learned extractor and 81.6 with visual-only extraction, versus 83.5 with default rules. Limiting replay to four attempts per state retains 82.7 EqStep at 17.4 rather than 36.0 CPU hours per 10,000 states. With no replay, demo-plus-rule and demonstration-only training reach 80.3 and 77.1, respectively. Appendix D reports these results, matched comparisons, and the limits of post-action verification.

6 Conclusion

Transition-Equivalent Supervision (TES) aligns GUI-agent training with execution-based evaluation by using goal-relevant state changes as supervision targets. TES predicts the intended transition, selects an action to realize it, and verifies the resulting state. Training on verified transition-equivalent actions helps agents recognize valid alternatives and reject NoOp or erroneous transitions. Across offline web prediction, executable web navigation, mobile control, and desktop-use benchmarks, TES improves transition-level metrics and task success, with the largest gains when many actions can realize the same transition.

Acknowledgments and Disclosure of Funding

Funding. This work received no external funding.

Competing interests. The authors declare no additional competing interests.

References

Marah Abdin, Jyoti Aneja, Hany Awadalla, Ahmed Awadallah, Ammar Ahmad Awan, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Jianmin Bao, Harkirat Behl, et al. Phi-3 technical report: A highly capable language model locally on your phone. *arXiv preprint arXiv:2404.14219*, 2024.

Saaket Agashe, Jiuzhou Han, Shuyu Gan, Jiachen Yang, Ang Li, and Xin Eric Wang. Agent s: An open agentic framework that uses computers like a human. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=lIVRgt4nLv.

Gilles Baechler, Srinivas Sunkara, Maria Wang, Fedir Zubach, Hassan Mansoor, Vincent Etter, Victor Cărbune, Jason Lin, Jindong Chen, and Abhanshu Sharma. Screenai: a vision-language model for ui and infographics understanding. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, pages 3058–3068, 2024.

Hao Bai, Yifei Zhou, Mert Cemri, Jiayi Pan, Alane Suhr, Sergey Levine, and Aviral Kumar. Digirl: Training in-the-wild device-control agents with autonomous reinforcement learning. *Advances in Neural Information Processing Systems*, 37:12461–12495, 2024.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. Qwen2.5-vl technical report. *arXiv preprint arXiv:2502.13923*, 2025.

Rogerio Bonatti, Dan Zhao, Francesco Bonacci, Dillon Dupont, Sara Abdali, Yinheng Li, Yadong Lu, Justin Wagle, Kazuhito Koishida, Arthur Bucker, et al. Windows agent arena: Evaluating multi-modal os agents at scale. In *International Conference on Machine Learning*, pages 4874–4910. PMLR, 2025.

Yuxiang Chai, Siyuan Huang, Yazhe Niu, Han Xiao, Liang Liu, Guozhi Wang, Dingyu Zhang, Shuai Ren, and Hongsheng Li. Amex: Android multi-annotation expo dataset for mobile gui agents. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 2138–2156, 2025.

ChiaHua Chang, M. Fu, X. Chen, S. Feng, M. Zhang, X. Zhou, J. J. Chuang, J. Pan, and Z. Liu. Research on PDU-Net lung nodule segmentation algorithm based on path aggregation and dual attention. In *2025 4th International Conference on Image Processing, Computer Vision and Machine Learning (ICICML)*, Chongqing, China, 2025. IEEE. ISBN 979-8-3315-6581-7. doi: 10.1109/ICICML67980.2025.11333821. URL https://doi.org/10.1109/ICICML67980.2025.11333821.

Dongping Chen, Yue Huang, Siyuan Wu, Jingyu Tang, Huichi Zhou, Qihui Zhang, Zhigang He, Yilin Bai, Chujie Gao, Liuyi Chen, et al. Gui-world: A video benchmark and dataset for multimodal gui-oriented understanding. In *The Thirteenth International Conference on Learning Representations*, 2025a. URL https://openreview.net/forum?id=QarKTT5brZ.

Jingxuan Chen, Derek Yuen, Bin Xie, Yuhao Yang, Gongwei Chen, Zhihao Wu, Li Yixing, Xurui Zhou, Weiwen Liu, Shuai Wang, et al. Spa-bench: A comprehensive benchmark for smartphone agent evaluation. In *NeurIPS 2024 Workshop on Open-World Agents*, 2024.

Qianyu Chen, Canran Xiao, and Runxuan Tang. Hidden forgetting in continual multimodal learning: When accuracy survives but grounding fails. *arXiv preprint arXiv:2607.02020*, 2026. URL https://arxiv.org/abs/2607.02020.

Wentong Chen, Junbo Cui, Jinyi Hu, Yujia Qin, Junjie Fang, Yue Zhao, Chongyi Wang, Jun Liu, Guirong Chen, Yupeng Huo, et al. Guicourse: From general vision language model to versatile gui agent. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 21936–21959, 2025b.

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)*, pages 9313–9332, 2024.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2web: Towards a generalist agent for the web. *Advances in Neural Information Processing Systems*, 36:28091–28114, 2023.

Kairui Ding, Boyuan Chen, Yuchen Su, Huan-ang Gao, Bu Jin, Chonghao Sima, Xiaohui Li, Wuqiang Zhang, Paul Barsch, Hongyang Li, and Hao Zhao. Hint-AD: Holistically aligned interpretability in end-to-end autonomous driving. In *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pages 3742–3765. PMLR, 2025. URL https://proceedings.mlr.press/v270/ding25a.html.

Alexandre Drouin, Maxime Gasse, Massimo Caccia, Issam H Laradji, Manuel Del Verme, Tom Marty, David Vazquez, Nicolas Chapados, and Alexandre Lacoste. Workarena: How capable are web agents at solving common knowledge work tasks? In *International Conference on Machine Learning*, pages 11642–11662. PMLR, 2024.

Enguang Fan, Anfeng Peng, Matthew Caesar, Jae Kim, Josh Eckhardt, Greg Kimberly, and Denis Osipychev. Towards effective swarm-based GPS spoofing detection in disadvantaged platforms. In *Proceedings of the 2023 IEEE Military Communications Conference (MILCOM)*, pages 722–728, 2023. doi: 10.1109/MILCOM58377.2023.10356314.

Enguang Fan, Yifan Chen, Zihan Shan, Klara Nahrstedt, Matthew Caesar, and Jae Kim. Communication-aware multi-agent reinforcement learning for decentralized cooperative UAV deployment. *arXiv preprint arXiv:2603.16141*, 2026.

Boyu Gou, Zanming Huang, Yuting Ning, Yu Gu, Michael Lin, Weijian Qi, Andrei Kopanev, Botao Yu, Bernal Jimenez Gutierrez, Yiheng Shu, et al. Mind2web 2: Evaluating agentic search with agent-as-a-judge. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2025a. URL https://openreview.net/forum?id=AUaW6DS9si.

Boyu Gou, 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 *The Thirteenth International Conference on Learning Representations*, 2025b. URL https://openreview.net/forum?id=kxnoqaisCT.

Hongliang He, Wenlin Yao, Kaixin Ma, Wenhao Yu, Yong Dai, Hongming Zhang, Zhenzhong Lan, and Dong Yu. Webvoyager: Building an end-to-end web agent with large multimodal models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6864–6890, 2024.

Hongliang He, Wenlin Yao, Kaixin Ma, Wenhao Yu, Hongming Zhang, Tianqing Fang, Zhenzhong Lan, and Dong Yu. Openwebvoyager: Building multimodal web agents via iterative real-world exploration, feedback and optimization. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 27545–27564, 2025.

Jiani He, Dingyan Shang, Yihua Xu, Shiqi Huang, Yan Lyu, Jize Li, and Shangjing Tang. Semantic signal-assisted inspection and recovery allocation in reverse logistics. *arXiv preprint arXiv:2609.02116*, 2026.

Wenyi Hong, Weihan Wang, Qingsong Lv, Jiazheng Xu, Wenmeng Yu, Junhui Ji, Yan Wang, Zihan Wang, Yuxiao Dong, Ming Ding, et al. Cogagent: A visual language model for gui agents. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 14281–14290, 2024.

Saisai Hu. Research on security enhancement methods for adversarial robust large language model intelligent agents for medical decision-making tasks. In *2026 7th International Seminar on Artificial Intelligence, Networking and Information Technology (AINIT)*, pages 599–603. IEEE, 2026.

Shiqi Huang, Jiani He, Dingyan Shang, Yihua Xu, Jize Li, Yan Lyu, and Lakshmi Muraleedharan Nair. DACRI: Decision-aware causal intervention ranking for critical supply chains. *arXiv preprint arXiv:2608.11154*, 2026.

Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Russ Salakhutdinov, and Daniel Fried. Visualwebarena: Evaluating multimodal agents on realistic visual web tasks. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 881–905, 2024.

Jize Li, Jiani He, Dishu Yang, Dingyan Shang, Jingjing Liu, and Shiqi Huang. Accuracy-preserving stability regularization for large-scale retail demand forecasting. *arXiv preprint arXiv:2607.13331*, 2026a.

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*, pages 8778–8786, 2025.

Siyuan Li, Haoxuan Zeng, Xin Luo, Fernando Jia, Florence Li, Zhengyang Geng, Zico Kolter, Tai Sing Lee, and Tianqin Li. Witness Overlap: Directional provenance inside open-weight model families. In *Advances in Neural Information Processing Systems*, 2026b.

Yang Li, Xiaoxue Chen, Hao Zhao, Jiangtao Gong, Guyue Zhou, Federico Rossano, and Yixin Zhu. Understanding embodied reference with Touch-Line Transformer. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=ugA1HX69sf.

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

Kevin Qinghong Lin, Linjie Li, Difei Gao, Zhengyuan Yang, Shiwei Wu, Zechen Bai, Stan Weixian Lei, Lijuan Wang, and Mike Zheng Shou. Showui: One vision-language-action model for gui visual agent. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 19498–19508, 2025.

Zhiming Lin, Canran Xiao, and Kai Zhao. Beyond more context: Retrieval diversity boosts multi-turn intent understanding. In *Proceedings of the ACM Web Conference 2026*, pages 2320–2329, 2026a. doi: 10.1145/3774904.3792567. URL https://doi.org/10.1145/3774904.3792567.

Zhiming Lin, Kai Zhao, Sophie Zhang, Peilai Yu, and Canran Xiao. CEC-Zero: Zero-supervision character error correction with self-generated rewards. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 23612–23620, 2026b. doi: 10.1609/aaai.v40i28.39534. URL https://doi.org/10.1609/aaai.v40i28.39534.

Jingjing Liu, Dishu Yang, Hongbo Qin, and Hansong Zhang. Passing tests, missing semantics: Evidence of interpretive lock-in in plan-based DataFrame code generation. *arXiv preprint arXiv:2610.15797*, 2026.

Quanfeng Lu, Wenqi Shao, Zitao Liu, Lingxiao Du, Fanqing Meng, Boxuan Li, Botong Chen, Siyuan Huang, Kaipeng Zhang, and Ping Luo. Guiodyssey: A comprehensive dataset for cross-app gui navigation on mobile devices. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 22404–22414, 2025.

Xing Han Lu, Zdeněk Kasner, and Siva Reddy. Weblinx: Real-world website navigation with multi-turn dialogue. In *International Conference on Machine Learning*, pages 33007–33056. PMLR, 2024a.

Yadong Lu, Jianwei Yang, Yelong Shen, and Ahmed Awadallah. Omniparser for pure vision based gui agent. *arXiv preprint arXiv:2408.00203*, 2024b.

Yao Lu, Tengfei Ma, Zeyu Wang, Zhuangzhi Chen, Dongwei Xu, Yun Lin, Qi Xuan, and Guan Gui. FCOS: A two-stage recoverable model pruning framework for automatic modulation recognition. *IEEE Transactions on Cognitive Communications and Networking*, 12:10764–10778, 2026. doi: 10.1109/TCCN.2026.3706560.

Xin Luo, Yicheng Tao, Haoxuan Zeng, Suyuan Wang, Chenzi Ouyang, Meiqi Zhu, Kai Liu, Shuibing Chen, and Jie Liu. VOICE: A vision-omics foundation model integrating direct and retrieval-based prediction of in-situ single-cell gene expression. *arXiv preprint arXiv:2608.08366*, 2026.

Lin Ma, Jiangtao Gong, Hao Xu, Hao Chen, Hao Zhao, Wenbing Huang, and Guyue Zhou. Planning assembly sequence with Graph Transformer. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 12395–12401. IEEE, 2023. URL https://arxiv.org/abs/2210.05236.

Mingjun Ma, Saisai Hu, Tiantian Zhu, Yitian Lin, Jiachen Sun, Zhengqiu Weng, Dehao Li, and Yajie Zhang. MuSK: Multi-scale knowledge learning for provenance-graph anomaly detection. *Computer Networks*, 289:112728, 2026. doi: 10.1016/j.comnet.2026.112728.

Shravan Nayak, Xiangru Jian, Kevin Qinghong Lin, Juan A Rodriguez, Montek Kalsi, Nicolas Chapados, M Tamer Özsu, Aishwarya Agrawal, David Vazquez, Christopher Pal, et al. Ui-vision: A desktop-centric gui benchmark for visual perception and interaction. In *International Conference on Machine Learning*, pages 45817–45851. PMLR, 2025.

Dang Nguyen, Jian Chen, Yu Wang, Gang Wu, Namyong Park, Zhengmian Hu, Hanjia Lyu, Junda Wu, Ryan Aponte, Yu Xia, et al. Gui agents: A survey. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 22522–22538, 2025.

Vardaan Pahuja, Yadong Lu, Corby Rosset, Boyu Gou, Arindam Mitra, Spencer Whitehead, Yu Su, and Ahmed Hassan. Explorer: Scaling exploration-driven web trajectory synthesis for multimodal web agents. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 6300–6323, 2025.

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.

Christopher Rawles, Alice Li, Daniel Rodriguez, Oriana Riva, and Timothy Lillicrap. Androidinthewild: A large-scale dataset for android device control. *Advances in Neural Information Processing Systems*, 36:59708–59728, 2023.

Christopher Rawles, Sarah Clinckemaillie, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth Fair, Alice Li, William E Bishop, Wei Li, Folawiyo Campbell-Ajala, et al. Androidworld: A dynamic benchmarking environment for autonomous agents. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=il5yUQsrjC.

Wanru Shao. Interpretable ensemble learning for network traffic anomaly detection: A SHAP-based explainable AI framework for embedded systems security. In *2026 14th International Symposium on Digital Forensics and Security (ISDFS)*, pages 1–4. IEEE, 2026a. doi: 10.1109/ISDFS69419.2026.11459069. URL https://doi.org/10.1109/ISDFS69419.2026.11459069.

Wanru Shao. Policy-driven vulnerability risk quantification framework for large-scale cloud infrastructure data security. In *Proceedings of the 2026 5th International Conference on Cyber Security, Artificial Intelligence and Digital Economy*, pages 40–45, New York, NY, USA, 2026b. Association for Computing Machinery. doi: 10.1145/3819836.3819842. URL https://doi.org/10.1145/3819836.3819842.

Tianlin Shi, Andrej Karpathy, Linxi Fan, Jonathan Hernandez, and Percy Liang. World of bits: An open-domain platform for web-based agents. In *International Conference on Machine Learning*, pages 3135–3144. PMLR, 2017.

Xian Sun, Wei Gao, Yingshuo Wang, Lingdong Kong, Yanhang Li, Zhichao Fan, Zexin Zhuang, Wenlong Dong, Zhiyuan Zheng, Hrishikesh Paranjape, Abhishek Mandal, and Johnny R. Zhang. Beyond accuracy: Measuring bias acknowledgment in chain-of-thought reasoning for responsible AI evaluation, 2026. URL https://arxiv.org/abs/2606.15127. ICML 2026 Workshop on Trustworthy AI for Good.

Jiacheng Wang, Liang Fan, Baihua Li, and Luyan Zhang. Forecasting with guidance: Representation-level supervision for time series forecasting, 2026a. URL https://arxiv.org/abs/2603.24262.

Junyang Wang, Haiyang Xu, Haitao Jia, Xi Zhang, Ming Yan, Weizhou Shen, Ji Zhang, Fei Huang, and Jitao Sang. Mobile-agent-v2: Mobile device operation assistant with effective navigation via multi-agent collaboration. *Advances in Neural Information Processing Systems*, 37:2686–2710, 2024.

Mengxiao Wang and Nitesh Saxena. Trading agents or market crashers? dissecting robustness and security failures in academic financial LLM trading schemes, 2026. URL https://arxiv.org/abs/2609.19705.

Xinyuan Wang, Bowen Wang, Dunjie Lu, Junlin Yang, Tianbao Xie, Junli Wang, Jiaqi Deng, Xiaole Guo, Yiheng Xu, Chen Henry Wu, et al. Opencua: Open foundations for computer-use agents. *arXiv preprint arXiv:2508.09123*, 2025.

Yingshuo Wang, Xian Sun, Yanhang Li, Zhichao Fan, and Zexin Zhuang. Embedding foundation model predictions in discrete-choice models with structural guarantees, 2026b. URL https://arxiv.org/abs/2606.26432. NeurIPS 2026.

Zijun Wang, Saisai Hu, Tiantian Zhu, Wenrui Cheng, Zhengqiu Weng, Haiting Chen, Xiangyang Zheng, and Suyu Zhang. Trident: Dual-stream APT attribution over heterogeneous threat knowledge graphs. *Computers & Security*, 171:105094, 2026c. doi: 10.1016/j.cose.2026.105094.

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 *The Thirteenth International Conference on Learning Representations*, 2025.

Canran Xiao, Tianxiang Xu, SiYuan Ma, Yiyang Jiang, Haoyu Gao, and Yuhan Wu. Reversible primitive–composition alignment for continual vision–language learning. In *The Fourteenth International Conference on Learning Representations*, 2026a. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/8f0d446441a938d9de420a8ab8d7fd36-Abstract-Conference.html.

Linqi Xiao, Venkateswarlu Gudepu, and Andrea Fumagalli. Joint alarm correlation and root alarm detection in optical transport networks with graph-attention. In *2025 IEEE 30th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, pages 1–6. IEEE, 2025. doi: 10.1109/CAMAD67323.2025.11229899.

Linqi Xiao, Venkateswarlu Gudepu, and Andrea Fumagalli. CORA: A conversational AI-agent for intelligent automation of OpenROADM multi-vendor optical transport networks. In *Applications of Machine Learning 2026*, volume 14218 of *Proceedings of SPIE*, page 142180Y. SPIE, 2026b. doi: 10.1117/12.3113861.

Linqi Xiao, Aparaajitha Gomathinayakam Latha, Venkateswarlu Gudepu, and Andrea Fumagalli. A YANG-grounded LLM agent supporting multi-vendor OpenROADM optical transport networks. In *Proceedings of the ACM SIGCOMM 2026 Conference*, pages 2109–2115, 2026c. doi: 10.1145/3789240.3822606.

Bin Xie, Rui Shao, Gongwei Chen, Kaiwen Zhou, Yinchuan Li, Jie Liu, Min Zhang, and Liqiang Nie. Gui-explorer: Autonomous exploration and mining of transition-aware knowledge for gui agent. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5650–5667, 2025.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh J Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, et al. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *Advances in Neural Information Processing Systems*, 37:52040–52094, 2024.

Mengzhu Xu, Hanzhi Liu, Ningkang Peng, Qianyu Chen, and Canran Xiao. Affordance-first decomposition for continual learning in video-language understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3908–3919, 2026a. URL https://openaccess.thecvf.com/content/CVPR2026/html/xu_Affordance-First_Decomposition_for_Continual_Learning_in_Video-Language_Understanding_CVPR_2026_paper.html.

Ran Xu, Kaixin Ma, Wenhao Yu, Hongming Zhang, Joyce C Ho, Carl Yang, and Dong Yu. Retrieval-augmented gui agents with generative guidelines. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 17877–17886, 2025.

Yiheng Xu, Zekun Wang, Junli Wang, Dunjie Lu, Tianbao Xie, Amrita Saha, Doyen Sahoo, Tao Yu, and Caiming Xiong. Aguvis: Unified pure vision agents for autonomous gui interaction. *arXiv preprint arXiv:2412.04454*, 2024.

Yihua Xu, Yan Lyu, Jize Li, Jiani He, You Lin, Yidan Hu, and Rubing Qiang. When does an added planning capability create operational value? an observable evidence standard. *arXiv preprint arXiv:2610.27305*, 2026b. doi: 10.48550/arXiv.2610.27305. URL https://arxiv.org/abs/2610.27305.

Tianci Xue, Weijian Qi, Tianneng Shi, Chan Hee Song, Boyu Gou, Dawn Song, Huan Sun, and Yu Su. An illusion of progress? assessing the current state of web agents. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=6jZi4HSs6o.

Dishu Yang, Jingjing Liu, and Jize Li. Balancing privacy, utility, and safety in LLM alignment through preference optimization. *arXiv preprint arXiv:2608.30141*, 2026a.

Dishu Yang, Qi Su, Hongbo Qin, and Hansong Zhang. Measurement-first auditing of agentic leaderboards: Contamination susceptibility, matched-control re-evaluation, and scorer validation. *arXiv preprint arXiv:2610.05830*, 2026b.

Ke Yang, Yao Liu, Sapana Chaudhary, Rasool Fakoor, Pratik Chaudhari, George Karypis, and Huzefa Rangwala. Agentoccam: A simple yet strong baseline for llm-based web agents. In *The Thirteenth International Conference on Learning Representations*, 2025a.

Yuhao Yang, Yue Wang, Dongxu Li, Ziyang Luo, Bei Chen, Chao Huang, and Junnan Li. Aria-ui: Visual grounding for gui instructions. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 22418–22433, 2025b.

Junyi Yao, Baichuan Li, Zihao Zheng, and Jiayu Long. Toward workflow-aware benchmarking for healthcare NLP agents, 2026. URL https://arxiv.org/abs/2609.00296.

Chengzhen Yu, Canran Xiao, Siyuan Ma, and Yang Liu. Text as partial constraint: Core-residual alignment for robust vision-language learning. *arXiv preprint arXiv:2607.03143*, 2026. URL https://arxiv.org/abs/2607.03143.

Chenghao Zhang, Yuxi Cheng, Saisai Hu, Canran Xiao, Yikai Mao, and Dan Roth. Training needs trustworthy worlds: Verified synthetic web environments for agent learning. *arXiv preprint arXiv:2608.21898*, 2026a. URL https://arxiv.org/abs/2608.21898.

Fan Zhang, Gongguan Chen, Hua Wang, Jinjiang Li, and Caiming Zhang. Multi-scale video super-resolution transformer with polynomial approximation. *IEEE Transactions on Circuits and Systems for Video Technology*, 33(9):4496–4506, 2023. doi: 10.1109/TCSVT.2023.3278131.

Fan Zhang, Gongguan Chen, Hua Wang, and Caiming Zhang. CF-DAN: Facial-expression recognition based on cross-fusion dual-attention network. *Computational Visual Media*, 10(3):593–608, 2024a. doi: 10.1007/s41095-023-0369-x.

Fan Zhang, Shiming Fan, and Hua Wang. Rethinking activation function: A simple path to efficient and accurate time series forecasting. *IEEE Transactions on Knowledge and Data Engineering*, 38 (10):6987–7004, 2026b. doi: 10.1109/TKDE.2026.3716656.

Fan Zhang, Shiming Fan, and Hua Wang. TimeSAF: Towards LLM-guided semantic asynchronous fusion for time series forecasting. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 26267–26284, 2026c. doi: 10.18653/v1/2026.acl-long.1208.

Fan Zhang, Zhiwei Gu, and Hua Wang. Decoding with structured awareness: Integrating directional, frequency-spatial, and structural attention for medical image segmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 12421–12429, 2026d. doi: 10.1609/aaai.v40i15.38235.

Li Zhang, Shihe Wang, Xianqing Jia, Zhihan Zheng, Yunhe Yan, Longxi Gao, Yuanchun Li, and Mengwei Xu. Llamatouch: A faithful and scalable testbed for mobile ui task automation. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology*, pages 1–13, 2024b.

Zhongyin Zhao, Yuan Liu, Yikun Liu, Haicheng Wang, Le Tian, Xiao Zhou, Yangxiu You, Zilin Yu, Yang Yu, and Jie Zhou. Points-gui-g: Gui-grounding journey. *arXiv preprint arXiv:2602.06391*, 2026.

Zihao Zheng, Junyi Yao, Baichuan Li, and Jiayu Long. Auditing escalation reliability in healthcare LLM agents: A perturbation-controlled benchmark of workflow-state failures, 2026. URL https://arxiv.org/abs/2610.21270.

Heng Zhou, Jing Tang, Juheng Zhang, Yanshu Li, Canran Xiao, Liwei Hou, Zong Ke, and Jiawei Yao. CoMem: Compositional concept-graph memory for vision–language adaptation. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/21fbdec9f82503196b4f984971ff3d2a-Abstract-Conference.html.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, et al. Webarena: A realistic web environment for building autonomous agents. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=oKn9c6ytLx.

A Additional Technical Details for Transition-Equivalent Supervision

A.1 Transition Extraction and Soft Targets

For each demonstrated tuple (g, o_t, a_t, o_{t+1}), the transition extractor ρ assigns a descriptor $c_t = (b_t, z_t)$. The type b_t identifies the form of GUI progress, and the slot z_t identifies the goal-relevant content of that progress. The extractor operates on a multi-view state-difference representation:

$$d_t = \Phi(o_t, o_{t+1}, g) = \left[d_t^{\text{vis}}; d_t^{\text{text}}; d_t^{\text{struct}}; d_t^{\text{nav}}; d_t^{\text{elem}}; d_t^{\text{goal}} \right]. \tag{13}$$

The visual component measures screen-level change,

$$d_t^{\text{vis}} = \frac{1}{|\mathcal{P}|} \sum_{p \in \mathcal{P}} \left\| \Omega(v_{t+1})_p - \Omega(v_t)_p \right\|_1, \tag{14}$$

where Ω is the normalized screen encoder and $\mathcal{P}$ is the set of normalized screen patches. The text component measures the change in visible text:

$$d_t^{\text{text}} = 1 - \frac{|\operatorname{Tok}(\ell_t) \cap \operatorname{Tok}(\ell_{t+1})|}{|\operatorname{Tok}(\ell_t) \cup \operatorname{Tok}(\ell_{t+1})| + \epsilon}. \tag{15}$$

The structured component compares element states:

$$d_t^{\text{struct}} = \frac{1}{|\mathcal{V}_t \cup \mathcal{V}_{t+1}| + \epsilon} \sum_{u \in \mathcal{V}_t \cup \mathcal{V}_{t+1}} \mathbf{1}[\sigma_t(u) \neq \sigma_{t+1}(u)], \tag{16}$$

where $\mathcal{V}_t$ is the set of visible GUI elements and $\sigma_t(u)$ contains the text, role, value, selected state, enabled state, focus state, and bounding box of element u. Navigation and element-state differences are represented as

$$\begin{aligned} d_t^{\text{nav}} &= [\mathbf{1}[\text{url}(o_t) \neq \text{url}(o_{t+1})], \mathbf{1}[\text{title}(o_t) \neq \text{title}(o_{t+1})]], \\ d_t^{\text{elem}} &= \left[d_t^{\text{focus}}, d_t^{\text{select}}, d_t^{\text{enable}}, d_t^{\text{dialog}} \right]. \end{aligned} \tag{17}$$

The goal-alignment component measures whether the observed state change matches the instruction:

$$d_t^{\text{goal}} = \text{sim}\left(E_g(g), E_\Delta(o_t, o_{t+1})\right), \tag{18}$$

where E_g encodes the goal and E_Δ encodes the textual and structural description of the state difference.

Each transition type $b \in \mathcal{B}$ has a rule score $s_b(d_t, a_t, g)$. For example, QUERYSUBMITTED receives high score when an input value is committed and the subsequent page, result list, or URL reflects the submitted query; FILTERAPPLIED receives high score when a control value changes and the visible results refresh under a goal-relevant attribute; DIALOGOPENED receives high score when a modal region appears and its title or contents match a goal-relevant entity. The type distribution is

$$p_\rho(b \mid o_t, a_t, o_{t+1}, g) = \text{softmax}(\{s_b(d_t, a_t, g)/\tau_\rho\}_{b \in \mathcal{B}}), \tag{19}$$

where τ_ρ is the extraction temperature. The slot z_t is filled from the changed query string, selected item name, applied attribute-value pair, opened dialog title, saved form field, or navigated page title. The resulting soft target is

$$y_{t,(b,z)} = p_\rho(b \mid o_t, a_t, o_{t+1}, g) \cdot \mathbf{1}[z = z_t(b)], \tag{20}$$

where $z_t(b)$ is the slot value extracted under type b. When rule confidence exceeds a fixed threshold γ_ρ, the soft target is converted into a one-hot label. In all experiments, label quality is reported on a human-validated diagnostic subset.

A.2 Candidate Actions and Verified Positive Sets

The candidate action set $\mathcal{A}(o_t)$ is generated from visible GUI targets and a fixed primitive operation set. Let $\mathcal{U}(o_t)$ be the set of candidate targets, including visible elements, text fields, clickable regions,

scrollable containers, and global keyboard targets. For each $u \in \mathcal{U}(o_t)$, the valid primitive operations are specified by its role and state:

$$\mathcal{A}(o_t) = \{(\kappa, u, \chi) : u \in \mathcal{U}(o_t), \kappa \in \mathcal{K}(u, o_t), \chi \in \mathcal{X}(\kappa, u, g)\}. \tag{21}$$

The verified positive set for a demonstrated transition contains three parts:

$$\mathcal{P}_t = \{a_t^\star\} \cup \mathcal{P}_t^{\text{replay}} \cup \mathcal{P}_t^{\text{rule}}. \tag{22}$$

The replay positives are obtained by executing candidate actions from the same pre-action state in a controlled environment:

$$\mathcal{P}_t^{\text{replay}} = \left\{a \in \mathcal{A}(o_t) : \rho(o_t, a, o^+(a), g) = c_t\right\}. \tag{23}$$

The rule positives are actions whose equivalence follows from deterministic GUI semantics. Examples include pressing `Enter` in a focused search field and clicking the associated submit button, clicking a menu item and activating the same item by keyboard focus, and selecting an autocomplete item whose value exactly matches the submitted query. Rule positives are added only when the source target, committed value, and resulting transition slot agree with c_t.

TES does not require exhaustive counterfactual enumeration. When replay produces no additional verified action, $\mathcal{P}_t = \{a_t^\star\}$, and transition supervision is still applied through $\mathcal{L}_{\text{trans}}$.

A.3 Stochastic Transition Equivalence

Real GUI environments contain stochasticity from delayed loading, asynchronous rendering, dynamic content, and nondeterministic service responses. Let $T(\cdot \mid o, a)$ be the distribution over post-action observations. An action a induces a distribution over transition descriptors:

$$P_\rho(c \mid o, a, g) = \mathbb{E}_{o^+ \sim T(\cdot \mid o, a)}\left[\mathbf{1}\left[\rho(o, a, o^+, g) = c\right]\right]. \tag{24}$$

Two actions are δ-transition-equivalent when the induced transition distributions are close:

$$a_i \sim_{o,g}^{\delta} a_j \quad \Longleftrightarrow \quad \text{JSD}\left(P_\rho(\cdot \mid o, a_i, g), P_\rho(\cdot \mid o, a_j, g)\right) \le \delta, \tag{25}$$

where JSD is Jensen-Shannon divergence. The main text uses the observed-transition definition because each offline demonstration provides one realized transition instance. The stochastic definition is used for repeated environment replay and robustness analysis.

A.4 Executor Score

The executor score in Eq. (7) is implemented as a compatibility function between the context, the transition descriptor, and the candidate action:

$$r_\phi(x_t, a, c) = h_t^\top W_r \left[e_a(a); e_c(c)\right] - \lambda_{\text{cost}} \, \text{Cost}(a), \qquad h_t = f_\theta(x_t). \tag{26}$$

Here $e_a(a)$ encodes the operation type, target role, target text, bounding box, and argument; $e_c(c)$ encodes the transition type and slot; W_r maps the action-transition representation into the context space. $\text{Cost}(a)$ assigns larger penalty to destructive actions, long navigation jumps, repeated failed targets, and actions outside the current visible region.

A.5 Verification and Repair

The verifier compares the intended transition $\hat{c}_t$ with the realized transition $\bar{c}_t = \rho(o_t, \hat{a}_t, o_{t+1}, g)$. A failed pair is stored in

$$\mathcal{F}_{t+1} = \mathcal{F}_t \cup \{(\hat{c}_t, \hat{a}_t) : \bar{c}_t \neq \hat{c}_t \text{ or } \bar{c}_t \in \mathcal{C}_{\text{bad}}\}. \tag{27}$$

During the next decoding step, failed pairs receive a penalty:

$$S_{\text{repair}}(c, a \mid x_t) = \log p_\theta(c \mid x_t) + \log q_\phi(a \mid x_t, c) - \mu \, \mathbf{1}[(c, a) \in \mathcal{F}_t], \tag{28}$$

where $\mu > 0$. This repair rule prevents the agent from repeatedly selecting a transition-action pair that has already produced a no-op, wrong transition, or error transition under the current context.

Algorithm 1 Full TES Training and Inference

Require: Demonstrations $\mathcal{D}$, extractor ρ, candidate generator $\mathcal{A}$, maximum steps $T_{\max}$

Transition label construction

for $n = 1, \ldots, N$ **do**
  for $t = 0, \ldots, T_n - 1$ **do**
    Build context $x_t^{(n)} = \Gamma(o_t^{(n)}, g^{(n)}, \mathcal{H}_t^{(n)})$
    Compute transition target $y_t^{(n)}$ by Eq. (20)
    Set $c_t^{(n)} = \arg\max_c y_{t,c}^{(n)}$
    Construct $\mathcal{P}_t^{(n)}$ by Eq. (22)
  end for
end for

Optimization

while not converged **do**
  Sample a minibatch of transition tuples
  Update θ using $\nabla_\theta \mathcal{L}_{\text{trans}}$
  Update ϕ using $\nabla_\phi \mathcal{L}_{\text{exec}}$
end while

Online inference

Initialize history $\mathcal{H}_0 = \emptyset$ and failure memory $\mathcal{F}_0 = \emptyset$
for $t = 0, \ldots, T_{\max} - 1$ **do**
  Encode $x_t = \Gamma(o_t, g, \mathcal{H}_t)$
  Select $(\hat{c}_t, \hat{a}_t) = \arg\max_{c,a} S_{\text{repair}}(c, a \mid x_t)$
  Execute $\hat{a}_t$, observe o_{t+1}, and compute $\bar{c}_t = \rho(o_t, \hat{a}_t, o_{t+1}, g)$
  if $\bar{c}_t = \hat{c}_t$ and $\bar{c}_t \notin \mathcal{C}_{\text{bad}}$ **then**
    Update $\mathcal{H}_{t+1} = \mathcal{H}_t \cup \{(o_t, \hat{a}_t, o_{t+1})\}$
    Set $\mathcal{F}_{t+1} = \mathcal{F}_t$
  else
    Update $\mathcal{F}_{t+1}$ by Eq. (27)
    Set $\mathcal{H}_{t+1} = \mathcal{H}_t$
  end if
  if the task success condition is satisfied **then**
    return success
  end if
end for
return failure

A.6 Training and Inference Details

Algorithm 1 gives the full training and inference procedure with soft transition targets, verified positives, and repair memory.

B Additional Experimental Details

B.1 Details of the Preliminary Study

Replay protocol. The preliminary study is designed as a diagnostic analysis. Given a demonstrated step

$$e_i = (g_i, o_i, a_i^\star, \tilde{o}_i),$$

where g_i is the task goal, o_i is the current GUI observation, $a_i^\star$ is the recorded demonstrator action, and $\tilde{o}_i$ is the demonstrated post-action observation, we construct a local candidate set $\mathcal{U}_i$ around the same observation. The candidate set contains visible clickable elements, nearby coordinate perturbations, keyboard shortcuts, equivalent menu operations, and text-entry confirmations when available. Each candidate action $a \in \mathcal{U}_i$ is replayed from o_i, producing a post-action observation $\hat{o}_i(a)$. Failed executions, unavailable actions, and environment-level errors are retained as diagnostic outcomes rather than discarded.

Outcome signature. To avoid introducing any learned transition model, we summarize the raw post-action difference with a coarse outcome signature:

$$s_i(a) = \psi(o_i, \hat{o}_i(a), g_i) = \big(\Delta_{\text{nav}}, \Delta_{\text{text}}, \Delta_{\text{value}}, \Delta_{\text{select}}, \Delta_{\text{focus}}, \Delta_{\text{done}}\big). \tag{29}$$

Here Δ_{nav} indicates page, URL, screen, or view change; Δ_{text} indicates visible-text change; Δ_{value} indicates form-value or input-buffer change; Δ_{select} indicates selection-state change; Δ_{focus} indicates focus or cursor-state change; and Δ_{done} indicates whether the task-completion predicate changes. The signature is used only for exploratory analysis and is not used as a supervision target in the proposed method.

Progress-preserving action fan-out. For each demonstrated step, we define the set of candidate actions that induce the same non-trivial coarse outcome as the demonstration:

$$\mathcal{E}_i = \{a \in \mathcal{U}_i : s_i(a) = s_i(a_i^\star),\ s_i(a) \notin \{\textsc{NoOp}, \textsc{Error}\}\}, \qquad k_i = |\mathcal{E}_i|. \tag{30}$$

The quantity k_i measures how many distinct surface actions preserve the same observed progress. When $k_i > 1$, exact action imitation would treat $k_i - 1$ progress-preserving realizations as negatives, although they produce the same coarse state change as the recorded action. We summarize this over-specification by the spurious-negative ratio

$$\eta_i = 1 - \frac{1}{\max(k_i, 1)}. \tag{31}$$

A larger η_i indicates that the demonstrated action is only one of many valid realizations of the same progress.

Template-level entropy comparison. We further group visually and semantically similar GUI states into template groups. A group G collects states with similar screen structure, task context, and available operation types. Within each group, we compare the entropy of recorded surface actions with the entropy of coarse outcome signatures:

$$H_{\text{act}}(G) = -\sum_{\alpha} p_G(\alpha) \log p_G(\alpha), \qquad H_{\text{out}}(G) = -\sum_{s} p_G(s) \log p_G(s), \tag{32}$$

where α ranges over normalized surface-action labels and s ranges over outcome signatures. We use $H_{\text{act}}(G) - H_{\text{out}}(G)$ as a stability gap. A positive gap means that, under similar GUI contexts, surface actions vary more than the state changes they induce.

NoOp and error density. For each state o_i, we also compute the local density of harmful or uninformative candidate actions:

$$r_i^{\text{bad}} = \frac{|\{a \in \mathcal{U}_i : s_i(a) \in \{\textsc{NoOp}, \textsc{Error}\}\}|}{|\mathcal{U}_i|}. \tag{33}$$

This statistic is visualized by the color intensity in Figure 1(a). It highlights an important caveat: although many non-demonstrated actions are valid alternatives, dense GUI states also contain many actions that do not advance the task or trigger invalid transitions. Therefore, simply expanding positives by spatial proximity, visual similarity, or action-type similarity is insufficient. A useful supervision signal must distinguish actions by their realized transition effect.

Connection to the proposed method. The preliminary study only uses replayed actions and coarse diagnostic signatures. It does not train a policy, construct transition descriptors, or use transition-conditioned execution. Its role is to reveal the failure mode of exact action imitation: the recorded action is often an arbitrary surface realization of a more stable underlying state change. This observation motivates the proposed Transition-Equivalent Supervision framework in Section 4, where supervision is organized around goal-conditioned transition descriptors and executable actions are evaluated by whether they realize the intended transition.

B.2 Benchmarks and Metrics

Benchmark suite. We evaluate TES on a broad set of GUI-agent benchmarks to cover the main evaluation regimes used by recent work: offline action prediction, live web execution, mobile-device control, and desktop/general computer use. Table 4 summarizes the benchmark families, their evaluation roles, and the primary metrics used in our experiments.

Table 4: **Benchmark suite.** TES is evaluated on offline, executable web, mobile, and desktop GUI-agent settings. The benchmark coverage is designed to test whether transition-equivalent supervision improves both local action realization and long-horizon task completion.

Family	Benchmarks	Evaluation role	Main metrics
Offline web action prediction	Mind2Web, Multimodal-Mind2Web [Deng et al., 2023, Pahuja et al., 2025]	Controlled step-level evaluation with demonstrations, screenshots, action labels, and structured webpage context.	Element Acc., Operation F1, Step SR, Task SR, TES metrics
Executable web navigation	WebArena, VisualWebArena, WebVoyager, Online-Mind2Web, Mind2Web 2 [Zhou et al., 2024, Koh et al., 2024, He et al., 2024, Xue et al., 2025, Gou et al., 2025a]	Online evaluation of whether the agent reaches the intended web state under dynamic webpages and long-horizon navigation.	Task SR, completion rate, step cost, token cost, TES metrics
Mobile GUI control	AITW, AndroidWorld, GUI-Odyssey, AMEX, SPA-Bench, LlamaTouch [Rawles et al., 2023, 2025, Lu et al., 2025, Chai et al., 2025, Chen et al., 2024, Zhang et al., 2024b]	Mobile app control under layout variation, app switching, multi-annotation ambiguity, and emulator-based execution.	Action Acc., grounding Acc., Step SR, Task SR, TES metrics
Desktop and computer use	OSWorld, WindowsAgentArena, WorkArena, UI-Vision, ScreenSpot-Pro, GUI-World [Xie et al., 2024, Bonatti et al., 2025, Drouin et al., 2024, Nayak et al., 2025, Li et al., 2025, Chen et al., 2025a]	General computer-use evaluation over desktop applications, professional UI grounding, and visually rich screen interaction.	Task SR, grounding Acc., completion rate, execution cost, TES metrics

Why these benchmarks match TES. TES is designed for settings where task success is determined by state progress rather than exact imitation of a recorded surface action. Mind2Web-style benchmarks allow us to test whether TES reduces over-penalization of alternative valid actions under offline supervision. Executable web benchmarks test whether the transition predictor and verifier improve long-horizon task completion. Mobile benchmarks test whether transition-equivalent supervision remains useful under dense clickable regions, ambiguous widgets, and app-specific interaction patterns. Desktop benchmarks test whether TES scales to heterogeneous applications where different menu paths, shortcuts, dialogs, and form interactions may realize the same task-relevant transition.

Standard offline action metrics. For offline action prediction, we report the standard metrics used by Mind2Web-style evaluation:

- **Element Accuracy** measures whether the predicted target element matches an acceptable reference element.
- **Operation F1** measures whether the predicted operation type and textual argument match the reference action.
- **Step Success Rate** measures whether both the target element and operation are correct for a step.
- **Task Success Rate** measures whether all required steps in a trajectory are predicted correctly.

These metrics are useful for comparison with prior action-imitation systems, but they treat the demonstrated action as the unique target and therefore may underestimate progress-preserving alternatives.

Standard online execution metrics. For executable benchmarks, we report:

- **Task Success Rate**: the fraction of tasks completed according to the benchmark-provided final-state evaluator or human/judge-based assessment.
- **Completion Rate**: the fraction of required subgoals, constraints, or key intermediate states reached before termination.
- **Average Steps**: the number of executed GUI actions per task.

- **Wall-clock Cost**: average execution time per task.
- **Token Cost**: number of input/output tokens used by LLM or VLM-based agents.
- **Model-call Cost**: number of model calls per task and per successful task.

For benchmarks that provide official evaluators, we use their released scripts without modification. For benchmarks with judge-based evaluation, we follow the original benchmark protocol and report the judge model used.

Transition-centric metrics. Standard action-matching metrics do not directly measure the target behavior of TES. We therefore introduce transition-centric metrics that evaluate whether the model predicts, realizes, and verifies the intended state transition.

Let $\mathcal{D}_{\text{eval}}$ be the evaluation set of GUI states. For each instance i, let g_i be the goal, o_i be the current observation, $a_i^\star$ be the demonstrated action, $c_i = \rho(o_i, a_i^\star, o_i^+, g_i)$ be the reference transition descriptor, $\hat{c}_i$ be the predicted transition, $\hat{a}_i$ be the selected action, and $\bar{c}_i = \rho(o_i, \hat{a}_i, \hat{o}_i^+, g_i)$ be the realized transition after executing $\hat{a}_i$.

Transition Accuracy. Transition Accuracy evaluates whether the predicted transition descriptor matches the reference descriptor:

$$\text{TAcc} = \frac{1}{|\mathcal{D}_{\text{eval}}|} \sum_{i \in \mathcal{D}_{\text{eval}}} \mathbf{1}[\hat{c}_i = c_i]. \tag{34}$$

This metric isolates the transition-prediction component from the executor.

Equivalent-Step Success. Equivalent-Step Success gives credit to any action that realizes the same task-relevant transition as the demonstrated action:

$$\text{EqStep} = \frac{1}{|\mathcal{D}_{\text{eval}}|} \sum_{i \in \mathcal{D}_{\text{eval}}} \mathbf{1}[\bar{c}_i = c_i]\, \mathbf{1}[\bar{c}_i \notin \{\textsc{NoOp}, \textsc{Error}\}]. \tag{35}$$

Unlike exact step success, EqStep does not require the selected action to be identical to the demonstration.

Alternative-Action Recall. For states with at least one verified non-demonstrated positive action, we measure whether the executor ranks an alternative positive action in its top-k predictions:

$$\text{AltRecall@}k = \frac{\sum_i \mathbf{1}\left[\text{TopK}_a\, q_\phi(a \mid x_i, c_i) \cap (\mathcal{P}_i \setminus \{a_i^\star\}) \neq \emptyset\right]}{\sum_i \mathbf{1}[|\mathcal{P}_i \setminus \{a_i^\star\}| > 0]}. \tag{36}$$

This metric directly measures whether TES learns valid alternatives instead of memorizing only the recorded action.

NoOp/Error Rate. NoOp/Error Rate measures how often the selected action fails to produce meaningful task progress:

$$\text{BadRate} = \frac{1}{|\mathcal{D}_{\text{eval}}|} \sum_{i \in \mathcal{D}_{\text{eval}}} \mathbf{1}[\bar{c}_i \in \{\textsc{NoOp}, \textsc{Error}\}]. \tag{37}$$

This metric is important because naively expanding positives by spatial or semantic similarity can accidentally include harmful actions.

Verification Agreement. Verification Agreement evaluates whether the transition intended by the policy agrees with the transition actually realized after execution:

$$\text{VAgree} = \frac{1}{|\mathcal{D}_{\text{eval}}|} \sum_{i \in \mathcal{D}_{\text{eval}}} \mathbf{1}[\hat{c}_i = \bar{c}_i]\, \mathbf{1}[\bar{c}_i \notin \{\textsc{NoOp}, \textsc{Error}\}]. \tag{38}$$

This metric diagnoses whether the transition-conditioned executor faithfully realizes the transition predicted by the policy.

Table 5: **Comparison groups.** TES is compared with action-imitation baselines, GUI grounding/action models, feedback-driven agents, exploration-based agents, and recent generalist GUI agents.

Group	Representative methods	Purpose of comparison
Action-imitation baselines	Behavior cloning, sequence-to-action generation, direct coordinate/element prediction, MindAct-style action prediction [Deng et al., 2023]	Tests whether transition-level supervision improves over exact action imitation under the same data and backbone.
GUI grounding/action models	CogAgent, ScreenAI, SeeClick, UGround, ShowUI, OS-ATLAS, Aria-UI, Aguvis, Ferret-UI 2, UI-TARS, OpenCUA, POINTS-GUI-G [Hong et al., 2024, Baechler et al., 2024, Cheng et al., 2024, Gou et al., 2025b, Lin et al., 2025, Wu et al., 2025, Yang et al., 2025b, Xu et al., 2024, Li et al., 2024, Qin et al., 2025, Wang et al., 2025, Zhao et al., 2026]	Tests whether TES complements strong perception, grounding, and action-modeling backbones.
Feedback-driven agents	WebVoyager, AgentOccam, Agent S, Mobile-Agent-v2, DigiRL, OpenWebVoyager [He et al., 2024, Yang et al., 2025a, Agashe et al., 2025, Wang et al., 2024, Bai et al., 2024, He et al., 2025]	Tests whether explicit transition-equivalent supervision provides benefits beyond execution feedback or planning heuristics.
Exploration and data-synthesis agents	GUI-Explorer, Explorer, retrieval-augmented GUI agents, GUI-course-style training pipelines [Xie et al., 2025, Pahuja et al., 2025, Xu et al., 2025, Chen et al., 2025b]	Tests whether TES improves learning from demonstrations and mined trajectories by reducing path-identity overfitting.

Demonstration-path overfitting gap. To measure whether a method overfits to the demonstrated surface action, we define:

$$\mathrm{DPOG} = \mathrm{EqStep} - \mathrm{ActStep}, \tag{39}$$

where $\mathrm{ActStep}$ is exact action-matching step success. A large positive gap means that a method can realize correct state progress through non-demonstrated actions, while exact action matching underestimates its capability. We report this metric on the subset of states where $|\mathcal{P}_i| \geq 2$.

B.3 Comparison Protocol

Compared methods. We compare TES with four categories of methods.

TES ablations. We include the following ablations:

- **Action-only imitation**: removes transition descriptors and trains only on the demonstrated action.
- **Transition-only**: predicts c_t but does not train a transition-conditioned executor.
- **w/o replay positives**: uses only demonstrated actions and deterministic rule positives.
- **w/o rule positives**: uses only replay-verified positives.
- **w/o soft targets**: converts every extracted descriptor into a hard one-hot label.
- **w/o verification**: removes post-action transition verification during execution.
- **w/o repair memory**: verifies actions but does not penalize failed transition-action pairs.
- **w/o transition slot**: predicts only transition type b_t and removes slot z_t.
- **w/o bad-transition filtering**: does not explicitly filter NOOP and ERROR.

Observation regimes. Different GUI agents use different inputs, making direct comparison potentially unfair. We therefore separate results into the following regimes:

- **Screenshot-only**: the model receives only the screenshot, task goal, and action history.

- **Structured-observation**: the model additionally receives DOM nodes, accessibility-tree nodes, or benchmark-provided candidate elements.
- **Parser-augmented**: the model uses an external screen parser such as OmniParser to extract regions, OCR spans, and clickable elements [Lu et al., 2024b].
- **Closed-source API reference**: proprietary models are reported as reference systems but are not mixed with open-weight trainable methods in the main ranking.

Fairness constraints. For trainable methods, we enforce the following matched conditions whenever the benchmark allows:

- the same train/validation/test splits;
- the same task goals and initial GUI states;
- the same action history length;
- the same candidate-action generator;
- the same maximum candidate budget K;
- the same maximum execution step budget $T_{\max}$;
- the same environment reset and retry policy;
- the same benchmark evaluator or judge;
- the same backbone and optimization budget for backbone-controlled comparisons.

When a baseline requires additional tools, memory, retrieval, or environment feedback, we report it in a separate table column.

Step budgets. Unless the benchmark specifies a different official limit, we use:

$$T_{\max} = 15 \quad \text{for web and mobile tasks,} \qquad T_{\max} = 50 \quad \text{for desktop computer-use tasks.}$$

For OSWorld-style evaluation, we additionally report the 15-step setting when comparing against short-horizon methods. For online web tasks with expensive execution, we report both success rate and cost-normalized success:

$$\mathrm{CostNormSR} = \frac{\mathrm{TaskSR}}{1 + \alpha_{\mathrm{tok}}\mathrm{TokCost} + \alpha_{\mathrm{call}}\mathrm{CallCost} + \alpha_{\mathrm{step}}\mathrm{StepCost}}. \tag{40}$$

We set $\alpha_{\mathrm{tok}}, \alpha_{\mathrm{call}}, \alpha_{\mathrm{step}}$ according to the benchmark's recommended cost accounting when available; otherwise, we report the raw cost components without aggregation.

B.4 Backbones and Implementation Details

Backbones. We instantiate TES on three representative backbones:

- **Qwen2.5-VL-7B-Instruct** [Bai et al., 2025]: a strong open vision-language backbone with native visual grounding and localization ability.
- **Phi-3.5-Vision-Instruct** [Abdin et al., 2024]: a compact multimodal backbone suitable for efficient GUI-agent training and ablation.
- **OS-ATLAS-7B** [Wu et al., 2025]: a GUI-specialized foundation action model trained for generalist GUI control.

For each backbone, we train a direct action-imitation baseline and a TES variant using the same data, candidate generator, batch schedule, and training budget.

Transition vocabulary. The transition vocabulary contains the following transition types:

$$\begin{aligned}\mathcal{B} = \{\ &\textsc{QuerySubmitted},\ \textsc{FilterApplied},\ \textsc{DialogOpened},\\ &\textsc{DialogClosed},\ \textsc{ItemSelected},\ \textsc{FormSaved},\\ &\textsc{PageNavigated},\ \textsc{ValueChanged},\ \textsc{FocusChanged},\\ &\textsc{MenuExpanded},\ \textsc{FileOpened},\ \textsc{PreferenceChanged},\\ &\textsc{TaskCompleted},\ \textsc{NoOp},\ \textsc{Error}\ \}.\end{aligned}$$

Each transition descriptor is $c_t = (b_t, z_t)$, where b_t is the transition type and z_t is a goal-relevant slot. The slot is extracted from the committed query, selected item, form value, dialog title, page title, menu option, filename, or task-specific completion signal.

Transition extraction. For each demonstrated tuple $(g, o_t, a_t^\star, o_{t+1})$, the extractor computes a multi-view state-difference representation:

$$d_t = \Phi(o_t, o_{t+1}, g) = \left[d_t^{\text{vis}}; d_t^{\text{text}}; d_t^{\text{struct}}; d_t^{\text{nav}}; d_t^{\text{elem}}; d_t^{\text{goal}} \right]. \tag{41}$$

The visual component captures screen-level changes, the text component captures visible-text changes, the structured component captures DOM/accessibility-tree changes, the navigation component captures URL/view/title changes, the element component captures focus/selection/value changes, and the goal component measures alignment between the state difference and the instruction.

Each transition type has a rule score $s_b(d_t, a_t, g)$. The extractor produces a soft type distribution:

$$p_\rho(b \mid o_t, a_t, o_{t+1}, g) = \operatorname{softmax}\left(\{ s_b(d_t, a_t, g)/\tau_\rho \}_{b \in \mathcal{B}} \right). \tag{42}$$

We use $\tau_\rho = 0.25$. If the maximum rule confidence exceeds $\gamma_\rho = 0.70$, the label is converted to a one-hot target. Otherwise, the soft distribution is retained during training.

Candidate-action construction. The candidate action set $\mathcal{A}(o_t)$ is constructed from:

- visible GUI elements;
- DOM or accessibility-tree nodes when available;
- OCR spans and detected clickable regions;
- text-entry fields and their associated submit actions;
- global keyboard actions such as `Enter`, `Esc`, `Tab`, and shortcut keys;
- benchmark-provided action candidates when available.

For offline web benchmarks, we use a maximum candidate budget $K = 50$. For screenshot-only mobile and desktop benchmarks, we use $K = 30$. When more candidates are available, we keep the top candidates according to visibility, interactability, textual relevance to the goal, and spatial proximity to the demonstrated target during training.

Verified positive sets. For each transition, the verified positive set is:

$$\mathcal{P}_t = \{a_t^\star\} \cup \mathcal{P}_t^{\text{replay}} \cup \mathcal{P}_t^{\text{rule}}. \tag{43}$$

Replay positives are obtained by executing candidate actions from the same pre-action state and keeping actions whose realized descriptor matches c_t:

$$\mathcal{P}_t^{\text{replay}} = \left\{ a \in \mathcal{A}(o_t) : \rho(o_t, a, o^+(a), g) = c_t \right\}. \tag{44}$$

Rule positives are actions whose equivalence follows from deterministic GUI semantics. Examples include pressing `Enter` in a focused search field versus clicking the search button, selecting an autocomplete option that exactly matches the query, activating the same menu item by mouse or keyboard focus, or saving a form through equivalent confirmation buttons. We cap $|\mathcal{P}_t|$ at 8 during training to avoid overweighting states with large action fan-out.

Model heads. TES adds two lightweight modules to each backbone. The transition head predicts $p_\theta(c \mid x_t)$ from the context representation. It is implemented as a two-layer MLP with hidden size 1024, GELU activation, and dropout 0.1. The executor scores an action a under transition c using:

$$r_\phi(x_t, a, c) = h_t^\top W_r \left[e_a(a); e_c(c) \right] - \lambda_{\text{cost}} \operatorname{Cost}(a), \qquad h_t = f_\theta(x_t). \tag{45}$$

Here $e_a(a)$ encodes operation type, target role, target text, bounding box, and argument, while $e_c(c)$ encodes transition type and slot.

Training objective. The training objective is:

$$\mathcal{L}_{\mathrm{TES}} = \mathcal{L}_{\mathrm{trans}} + \lambda_{\mathrm{exec}} \mathcal{L}_{\mathrm{exec}}. \tag{46}$$

The transition loss is cross-entropy with the extracted soft or hard transition target. The executor loss marginalizes over all verified positive actions:

$$\mathcal{L}_{\mathrm{exec}} = -\frac{1}{M} \sum_{n,t} \log \sum_{a \in \mathcal{P}_t^{(n)}} q_\phi(a \mid x_t^{(n)}, c_t^{(n)}). \tag{47}$$

We set $\lambda_{\mathrm{exec}} = 1.0$ by default and tune it in $\{0.25, 0.5, 1.0, 2.0\}$ on the validation split.

Fine-tuning hyperparameters. Unless otherwise specified, all models are fine-tuned with LoRA:

- LoRA rank: 16;
- LoRA scaling factor: 32;
- LoRA dropout: 0.05;
- optimizer: AdamW;
- backbone-adapter learning rate: 2×10^{-5};
- transition/executor-head learning rate: 1×10^{-4};
- global batch size: 128;
- gradient clipping: 1.0;
- warmup ratio: 3%;
- scheduler: cosine decay;
- training epochs: 3 for offline datasets.

We select checkpoints by validation Equivalent-Step Success unless a benchmark requires official validation task success.

Inference and repair. At inference time, TES predicts the top-m transitions with $m = 5$. It then selects the transition-action pair:

$$(\hat{c}_t, \hat{a}_t) = \arg\max_{c,a} \left[\log p_\theta(c \mid x_t) + \log q_\phi(a \mid x_t, c) - \mu \mathbf{1}[(c, a) \in \mathcal{F}_t] \right], \tag{48}$$

where $\mathcal{F}_t$ stores failed transition-action pairs. We use $\mu = 2.0$. After executing $\hat{a}_t$, the agent extracts the realized transition $\bar{c}_t = \rho(o_t, \hat{a}_t, o_{t+1}, g)$. The step is accepted only if $\bar{c}_t = \hat{c}_t$ and $\bar{c}_t \notin \{\textsc{NoOp}, \textsc{Error}\}$. Otherwise, the failed pair is added to $\mathcal{F}_t$, and the agent replans.

Compute resources. Main fine-tuning runs are conducted on 8 NVIDIA A100-80GB GPUs. Smaller ablations are conducted on 4 NVIDIA A100-80GB GPUs. Online evaluation uses 16 CPU environment workers and 4 GPU inference workers. For one full backbone on the offline web suite, fine-tuning takes approximately 18–26 GPU hours depending on the backbone and observation regime. For online execution, the cost is dominated by environment reset, page loading, and model inference; we therefore report wall-clock cost, token cost, and model-call cost together with task success.

Random seeds and confidence intervals. Offline experiments are run with three random seeds. We report mean and standard deviation. For online execution, repeated full evaluation is expensive, so we report bootstrap 95% confidence intervals over tasks using 10,000 bootstrap samples. For human- or judge-based evaluation, we additionally report the judge configuration and evaluate a random subset twice to estimate evaluator stability.

Human validation of transition labels. We validate the quality of extracted transition descriptors on a diagnostic subset. For each benchmark family, we sample 500 transitions or 10% of validation transitions, whichever is smaller. Two annotators judge whether the extracted type b_t and slot z_t correctly describe the task-relevant state change. We report type accuracy, slot accuracy, and inter-annotator agreement. Disagreements are resolved by a third annotator and used only for diagnostic analysis, not for training.

Table 6: **Compute resources for reported experiments.** GPU hours are reported in A100-80GB-equivalent hours. Wall-clock time varies with benchmark environment latency, especially for live or executable GUI benchmarks.

Experiment group	Hardware	Per-run cost	Number of runs	Approx. total cost
Transition extraction and candidate construction	32 CPU workers, 1 A100 for parser/VLM-assisted preprocessing when needed	6–10 GPU h + 80–140 CPU h per benchmark family	4 benchmark families	30 GPU h + 420 CPU h
Direct action-imitation baselines	4 A100-80GB GPUs	12–21 GPU h per backbone/seed	3 backbones $\times$ 3 seeds	150 GPU h
TES main fine-tuning	8 A100-80GB GPUs	18–26 GPU h per backbone/seed	3 backbones $\times$ 3 seeds	205 GPU h
Single-factor ablations	4 A100-80GB GPUs	12–16 GPU h per ablation run	10 ablation variants	145 GPU h
Fan-out and alternative-action analyses	4 A100-80GB GPUs + 32 CPU workers	8–14 GPU h per analysis	2 analysis groups	24 GPU h + 180 CPU h
Offline evaluation and bootstrap aggregation	2–4 A100-80GB GPUs + 32 CPU workers	6–12 GPU h per benchmark family	4 benchmark families	38 GPU h + 260 CPU h
Online executable evaluation	4 A100-80GB GPUs + 16 CPU environment workers	22–38 GPU h per benchmark family	4 benchmark families	120 GPU h + 1,600 CPU h
Total reported experiments	A100-80GB + CPU environment workers	–	–	**about 710 GPU h + 2,460 CPU h**

B.5 Computational Resources

We report the compute resources used to reproduce the main experiments in Table 6. All trainable open-weight models are fine-tuned with LoRA adapters. Main fine-tuning runs use NVIDIA A100-80GB GPUs. Online execution uses CPU workers for environment stepping and GPU workers for model inference. The reported numbers include the experiments used in the main tables and major ablations, but exclude small pilot runs used for debugging prompts, environment wrappers, and plotting scripts.

The largest cost comes from executable online evaluation rather than from fine-tuning, because live or emulator-based GUI environments require environment reset, page loading, rendering, and post-action verification. For storage, the processed observations, transition descriptors, candidate-action caches, replay logs, and model checkpoints require approximately 2.1 TB in total. Preliminary failed runs and small pilot experiments required approximately an additional 80–120 A100 GPU hours; these are not included in Table 6.

B.6 Main Experiment Details

Action fan-out analysis. For the fan-out analysis in Section 5.4, we evaluate whether the benefit of TES increases with the number of verified valid realizations of the same transition. For each held-out GUI state i, we compute the action fan-out as

$$f_i = |\mathcal{P}_i|, \tag{49}$$

where $\mathcal{P}_i$ is the verified positive set constructed from the demonstrated action, replay-verified positives, and deterministic rule positives. A larger f_i indicates that the task-relevant transition can be realized by more executable surface actions.

Evaluation subset and buckets. We evaluate all held-out states for which the candidate-action generator returns at least one executable action and the transition extractor produces a non-NOOP/non-

ERROR reference descriptor. States are stratified into six fan-out buckets:

$$\{1\}, \quad \{2\}, \quad \{3\text{–}4\}, \quad \{5\text{–}6\}, \quad \{7\text{–}8\}, \quad \{\geq 9\}.$$

The first bucket corresponds to nearly unique demonstrations, while the last buckets correspond to states where action identity is highly underdetermined by task progress.

Compared methods. We compare two methods under matched conditions. **Action Imitation** trains the OS-ATLAS-7B backbone to predict the demonstrated surface action. **TES** uses the same backbone but replaces single-action supervision with transition prediction, transition-conditioned action realization, positive-set marginalization, and post-action verification. Both methods use the same training split, validation split, candidate-action generator, maximum candidate budget, and inference step budget.

Reported metrics. The primary metric is Equivalent-Step Success:

$$\mathrm{EqStep} = \frac{1}{|\mathcal{D}_{\mathrm{eval}}|} \sum_i \mathbf{1}[\bar{c}_i = c_i]\, \mathbf{1}[\bar{c}_i \notin \{\textsc{NoOp}, \textsc{Error}\}], \tag{50}$$

where c_i is the reference transition descriptor and $\bar{c}_i$ is the realized transition after executing the predicted action. For each fan-out bucket, we also report AltRecall@5 and BadRate reduction. AltRecall@5 measures whether TES ranks at least one non-demonstrated valid action in the top five predictions, while BadRate reduction measures how much TES reduces realized NOOP/ERROR transitions relative to Action Imitation.

Aggregation and uncertainty. For bucket-level results, EqStep, AltRecall@5, and BadRate are averaged over all states in the bucket. For exact-fan-out visualization, we further divide states with the same fan-out into small local evaluation groups and plot the TES gain for each group as a transparent point. The solid curve reports the mean TES gain at each exact fan-out value, and the shaded band denotes a 95% bootstrap confidence interval computed over local groups. This visualization is intended to show both the average trend and the variance across GUI states.

C Additional Experimental Results

C.1 Can TES recognize non-demonstrated valid alternatives?

A core claim of TES is that the demonstrated action is only one valid realization of the intended transition. If this claim is correct, the learned executor should assign high scores not only to the demonstrated action, but also to other executable actions that induce the same task-relevant state change. We therefore ask: *can TES recover non-demonstrated but transition-equivalent valid alternatives?*

We evaluate this question with **Alternative-Action Recall@K**. For each evaluation state i, let $a_i^\star$ be the demonstrated action and $\mathcal{P}_i$ be the verified positive set. We focus only on states where at least one non-demonstrated positive action exists:

$$\mathcal{S}_{\mathrm{alt}} = \{i : |\mathcal{P}_i \setminus \{a_i^\star\}| > 0\}. \tag{51}$$

For each state, we remove the demonstrated action from the candidate list and check whether the model ranks any non-demonstrated valid alternative in its top-K predictions:

$$\mathrm{AltRecall@}K = \frac{1}{|\mathcal{S}_{\mathrm{alt}}|} \sum_{i \in \mathcal{S}_{\mathrm{alt}}} \mathbf{1}\left[\mathrm{TopK}_{a \in \mathcal{A}(o_i) \setminus \{a_i^\star\}}\, S_i(a) \cap (\mathcal{P}_i \setminus \{a_i^\star\}) \neq \emptyset\right], \tag{52}$$

where $S_i(a)$ is the model's action score. For action-imitation baselines, $S_i(a) = \log \pi(a \mid x_i)$. For TES, we use the transition-conditioned score

$$S_i(a) = \log q_\phi(a \mid x_i, c_i), \tag{53}$$

where c_i is the reference transition descriptor. This isolates whether the executor can recognize valid realizations of a given transition. We also evaluate an end-to-end variant using the predicted transition $\hat{c}_i$, and observe the same trend.

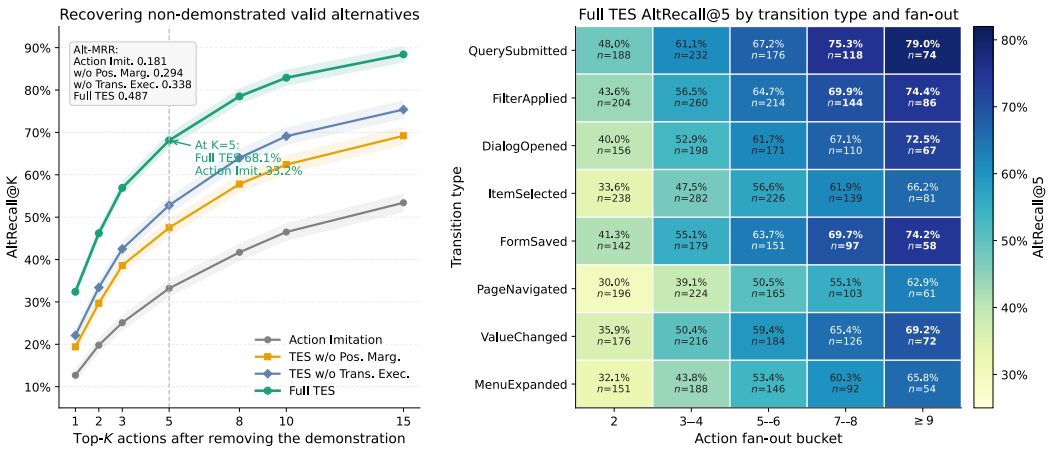


Figure 4: **TES recognizes non-demonstrated valid alternatives. Left:** AltRecall@K curves after removing the demonstrated action from the candidate set. Full TES consistently retrieves more transition-equivalent alternatives than action imitation and ablated variants. The improvement is especially large at small K, indicating that valid alternatives are ranked near the top rather than merely appearing deep in the candidate list. **Right:** Full TES AltRecall@5 broken down by transition type and fan-out bucket. Darker cells indicate higher recall, and each cell reports both AltRecall@5 and the number of evaluated states. TES is strongest for transitions with clear semantic slots, such as query submission, filter application, form saving, and dialog opening, while page navigation remains more difficult because valid alternatives often depend on dynamic website state.

We compare four variants under the same OS-ATLAS-7B backbone: **Action Imitation**, **TES w/o positive-set marginalization**, **TES w/o transition-conditioned executor**, and **Full TES**. All methods use the same candidate-action generator and the same evaluation states. We report AltRecall@K for $K \in \{1, 2, 3, 5, 8, 10, 15\}$. To understand where valid alternatives are most recoverable, we additionally break down Full TES by transition type and action fan-out bucket.

Figure 4 shows that TES substantially improves the recovery of non-demonstrated valid actions. At $K = 1$, Action Imitation recalls only 12.7% of valid alternatives after the demonstrated action is removed, while Full TES reaches 32.4%. The gap widens at moderate K: at $K = 5$, Full TES achieves 68.1% AltRecall, compared with 33.2% for Action Imitation, 47.5% for TES without positive-set marginalization, and 52.8% for TES without the transition-conditioned executor. This indicates that both components are important: positive-set marginalization exposes the model to multiple correct realizations, while transition-conditioned scoring teaches the executor which alternatives realize the intended transition.

The heatmap further reveals that the gains are not uniform across transition types. TES obtains the highest AltRecall@5 on QUERYSUBMITTED, FILTERAPPLIED, FORMSAVED, and DIALOGOPENED, where the transition slot z_t provides a clear semantic anchor such as a query string, filter value, saved field, or dialog title. For example, in the $f_i \geq 9$ bucket, AltRecall@5 reaches 78.5% for QUERYSUBMITTED and 74.6% for FILTERAPPLIED. By contrast, PAGENAVIGATED remains harder, with 62.8% AltRecall@5 in the highest fan-out bucket, because visually similar navigation actions can lead to subtly different downstream pages.

This experiment directly supports the transition-equivalence hypothesis. After removing the demonstrated action, TES can still recover non-demonstrated actions that realize the same transition, whereas action imitation often fails because it has learned the surface action identity as the supervision target. The result also explains why TES improves EqStep and online task success: it does not simply imitate the recorded trajectory, but learns a broader equivalence class of actions that preserve task progress.

C.2 Additional Diagnostic Results

The preceding experiments show that TES improves step-level and task-level performance. We next provide additional diagnostics targeting the main potential failure modes of transition-equivalent

Table 7: **Human validation of extracted transition descriptors.** Type Acc. evaluates whether the transition type b_t is correct; Slot Acc. evaluates whether the slot z_t captures the goal-relevant changed content; Exact Acc. requires both to be correct.

Benchmark family	#Samples	Type Acc. ↑	Slot Acc. ↑	Exact Acc. ↑	Ambig. Rate ↓	Cohen's κ ↑
Offline web prediction	500	**95.6**	**92.8**	**90.9**	**7.6**	**0.86**
Executable web navigation	500	<u>94.8</u>	<u>91.6</u>	<u>89.7</u>	<u>8.3</u>	<u>0.84</u>
Mobile control	500	93.4	90.1	87.9	9.1	0.82
Desktop computer use	500	91.9	88.7	86.2	10.4	0.79
Overall	2000	93.9	90.8	88.7	8.9	0.83

supervision: whether the extracted transition labels are reliable, whether the verified positive sets introduce false positives, whether TES is robust to candidate-action quality, whether verification improves reliability without excessive execution cost, and whether transition abstraction generalizes beyond seen GUI templates. All experiments use OS-ATLAS-7B as the default backbone unless otherwise stated.

C.2.1 Are extracted transition labels reliable?

Motivation. TES relies on the transition extractor ρ to convert a demonstrated state-action-state tuple into a descriptor $c_t = (b_t, z_t)$, where b_t is the transition type and z_t is the goal-relevant slot. A natural concern is that performance gains may come from noisy or overly permissive transition labels. We therefore evaluate the quality of extracted transition descriptors through human validation.

Setup. For each benchmark family, we sample 500 validation transitions or 10% of the validation set, whichever is smaller. Two annotators independently judge whether the extracted transition type b_t correctly describes the state change and whether the slot z_t captures the goal-relevant changed content. We report type accuracy, slot accuracy, exact descriptor accuracy, ambiguous-label rate, and Cohen's κ. Exact descriptor accuracy requires both b_t and z_t to be judged correct. Ambiguous-label rate measures cases where multiple transition types are plausible because of simultaneous GUI changes, such as navigation plus form submission.

Results and analysis. Table 7 shows that the transition extractor produces reliable supervision targets across all benchmark families. Overall type accuracy reaches 93.9%, while exact descriptor accuracy remains 88.7%, indicating that most extracted labels correctly capture both the form of the transition and the goal-relevant slot. The extractor is strongest on offline web prediction, where DOM and text changes are relatively clean, and slightly weaker on desktop computer use, where several state changes may occur simultaneously. The ambiguous-label rate remains below 11% in all settings, suggesting that soft transition targets are useful but not compensating for pervasive label noise. These results support the validity of using transition descriptors as supervision targets rather than relying on exact action identity.

C.2.2 Do verified positive sets introduce false positives?

Motivation. TES expands the demonstrated action into a verified positive set $\mathcal{P}_t$ using replay-verified positives and deterministic rule positives. This is central to transition-equivalent supervision, but it may introduce false positives if visually similar or semantically related actions do not actually induce the same transition. We therefore evaluate both the precision and coverage of positive-set construction.

Setup. We compare five positive-set construction strategies: **Demo only** uses only the recorded action; **Demo+Rule** adds deterministic GUI-equivalent actions; **Demo+Replay** adds replay-verified actions; **Full verified set** uses both rule and replay positives; **Naive spatial** adds nearby clickable targets without transition verification; and **Naive semantic** adds textually similar targets without transition verification. For each strategy, we report average positive-set size, human-estimated positive precision, transition coverage, bad-positive rate, AltRecall@5, EqStep, and BadRate. Coverage measures the fraction of human-verified transition-equivalent alternatives recovered by the construction strategy.

Table 8: **Quality of verified positive sets.** The full verified set improves coverage while keeping false positives low. Naive spatial or semantic expansion covers more candidates but introduces many harmful positives, increasing NOOP/ERROR transitions.

Positive-set construction	$\lvert\mathcal{P}\rvert$	Pos. Prec. ↑	Coverage ↑	Bad Pos. ↓	AltRecall@5 ↑	EqStep ↑	BadRate ↓
Demo only	1.00	**99.2**	28.5	**0.3**	28.7	77.1	6.4
Demo + Rule positives	1.92	96.7	52.4	1.1	51.0	80.3	5.3
Demo + Replay positives	2.41	97.8	58.7	0.8	54.3	81.0	5.0
Full verified set	3.08	96.9	69.4	1.2	**68.1**	**83.5**	**4.2**
Naive spatial expansion	5.67	79.4	**76.8**	11.6	58.5	78.6	9.2
Naive semantic expansion	4.96	83.1	73.0	8.7	60.4	79.8	8.1

Table 9: **Robustness to candidate-action budget and noisy candidates.** TES remains stable across a wide range of candidate budgets and degrades gracefully when distractor actions are injected.

Candidate setting	EqStep ↑	AltRecall@5 ↑	BadRate ↓	WebArena ↑	Avg. Steps ↓
$K = 10$	78.6	56.4	5.7	52.2	**8.9**
$K = 20$	81.1	62.3	4.9	54.1	9.1
$K = 30$	82.6	65.0	4.4	55.4	9.2
$K = 50$ **default**	**83.5**	68.1	**4.2**	**56.1**	9.3
$K = 75$	83.2	**68.4**	4.4	55.8	9.6
$K = 100$	82.7	67.9	4.8	55.5	9.9
$K = 50$ + 20% distractors	82.4	66.5	4.7	55.2	9.6
$K = 50$ + 40% distractors	80.8	63.8	5.5	53.7	10.0
$K = 50$ + 60% distractors	78.9	60.1	6.4	51.9	10.5

Results and analysis. Table 8 shows that TES benefits from positive-set expansion only when alternatives are verified by transition outcomes. The full verified set increases coverage from 28.5% to 69.4%, while keeping the bad-positive rate low at 1.2%. This directly improves AltRecall@5 from 28.7% to 68.1% and EqStep from 77.1 to 83.5. By contrast, naive spatial and semantic expansions achieve higher raw coverage but substantially lower precision. Their bad-positive rates rise to 11.6% and 8.7%, respectively, which leads to higher BadRate and worse EqStep. This confirms that the gains of TES are not due to simply adding more positives; the positives must be verified by realized transition equivalence.

C.2.3 Is TES robust to candidate-action budget and noisy candidates?

Motivation. TES selects an executable action from a candidate set $\mathcal{A}(o_t)$. A reviewer may ask whether TES depends on a strong candidate generator or a carefully tuned candidate budget. We therefore evaluate robustness under different candidate budgets and under injected distractor actions.

Setup. We vary the maximum candidate budget $K \in \{10, 20, 30, 50, 75, 100\}$. The default setting is $K = 50$ for web-style structured observations. We also inject visually plausible but transition-incorrect distractors into the candidate set at ratios 20%, 40%, and 60%. For each setting, we report EqStep, AltRecall@5, BadRate, WebArena task success, and average online steps. The same trained TES model is used across candidate settings; only the evaluation-time candidate set changes.

Results and analysis. Table 9 shows that TES does not require a narrowly tuned candidate budget. Performance improves as K increases from 10 to 50, because more valid transition-equivalent actions become available. Beyond $K = 50$, EqStep saturates and slightly decreases, suggesting that very large candidate sets introduce additional distractors without providing many new useful actions. The default $K = 50$ achieves the best balance, with 83.5 EqStep, 4.2 BadRate, and 56.1 WebArena success. When distractors are injected, TES degrades smoothly rather than collapsing: even with 60% distractors, EqStep remains 78.9, substantially above action-imitation levels reported in the main tables. This indicates that transition-conditioned scoring and verification provide robustness against plausible but incorrect actions.

Table 10: **Cost–benefit analysis of verification and repair.** Verification improves reliability substantially, while two repair attempts provide the best accuracy–cost tradeoff.

Inference variant	WebArena ↑	VAgree ↑	BadRate ↓	Avg. Steps ↓	Model Calls ↓	Wall-clock (s) ↓	CostNormSR ↑
No verification	51.9	75.5	7.1	**8.8**	**8.8**	**46.3**	42.7
Verify only	53.0	79.0	5.8	9.0	9.0	48.1	43.8
Verify + 1 retry	55.3	82.0	4.7	9.2	9.8	52.0	44.5
Verify + 2 retries	56.1	83.2	4.2	9.3	10.4	54.6	**44.9**
Verify + 3 retries	**56.2**	**83.3**	**4.1**	9.8	11.7	61.2	43.7

C.2.4 Does verification improve reliability without excessive cost?

Motivation. TES verifies whether the realized post-action transition matches the intended transition. Although verification reduces repeated NOOP and erroneous actions, it may introduce additional execution or model-call cost. We therefore evaluate the cost–benefit tradeoff of verification and repair [He et al., 2026, Huang et al., 2026, Lu et al., 2026, Zhang et al., 2026b].

Setup. We compare five inference variants: **No verification** executes the highest-scoring transition-action pair without checking the realized transition; **Verify only** checks the realized transition but does not retry after failure; **Verify+1 retry** and **Verify+2 retries** allow one or two repair attempts after failed verification; and **Verify+3 retries** tests whether more retries continue to help. We report WebArena task success, VAgree, BadRate, average steps, model calls, wall-clock time, and cost-normalized success. Cost-normalized success penalizes extra execution and model calls following Eq. (40).

Results and analysis. Table 10 shows that verification improves reliability even before multiple retries are introduced. Moving from no verification to verify-only increases VAgree from 75.5 to 79.0 and reduces BadRate from 7.1 to 5.8. Allowing repair further improves online success: Verify+2 retries reaches 56.1 WebArena success and 83.2 VAgree, while keeping average steps at 9.3. A third retry yields almost no additional success gain but increases model calls and wall-clock time, reducing CostNormSR. Thus, the default two-retry setting is not simply buying performance through unlimited execution; it provides the best cost-normalized tradeoff by avoiding repeated failed transition-action pairs.

C.2.5 Does transition abstraction generalize beyond seen GUI templates?

Table 11: **Cross-template and cross-regime generalization.** TES consistently improves over action imitation, with larger gains under unseen templates, unseen websites/apps, rare transition types, and long-horizon tasks.

Evaluation regime	EqStep ↑			Task SR ↑			BadRate ↓		
	Act. Imit.	TES	Δ	Act. Imit.	TES	Δ	Act. Imit.	TES	Δ
Seen template	70.6	**86.4**	+15.8	47.1	**60.2**	+13.1	10.2	**3.5**	−6.7
Unseen template	63.8	**82.1**	+18.3	41.9	**54.7**	+12.8	12.4	**4.8**	−7.6
Unseen website/app	61.5	**80.2**	+18.7	39.7	**52.3**	+12.6	13.5	**5.2**	−8.3
Rare transition type	58.2	**76.8**	+18.6	36.6	**49.4**	+12.8	15.7	**6.1**	−9.6
Long-horizon tasks	55.0	**74.6**	+19.6	33.9	**47.8**	+13.9	16.2	**6.6**	−9.6

Motivation. A central claim of TES is that transition-level supervision reduces overfitting to demonstrator-specific surface actions. If this is true, TES should be especially useful when the test GUI differs from training templates or when tasks involve rare transition types and longer horizons. We therefore evaluate generalization across increasingly difficult held-out regimes.

Setup. We partition held-out instances into five regimes: **Seen template**, where screen layout templates appear in training; **Unseen template**, where the website or app is seen but the local GUI template is new; **Unseen website/app**, where the interface source is held out; **Rare transition type**,

where the target transition type appears infrequently during training; and **Long-horizon**, where the task requires at least ten executed steps. We compare Action Imitation and TES under the same OS-ATLAS-7B backbone, candidate generator, and evaluation budget. We report EqStep, task success, and BadRate.

Results and analysis. Table 11 shows that TES improves over action imitation in every evaluation regime. The gain is already substantial on seen templates, improving EqStep by 15.8 points and reducing BadRate by 6.7 points. The advantage becomes larger when surface actions are less likely to match training demonstrations. On unseen templates and unseen websites/apps, TES improves EqStep by 18.3 and 18.7 points, respectively. The largest gains appear on long-horizon tasks, where local action errors compound over time: TES improves task success from 33.9 to 47.8 and reduces BadRate from 16.2 to 6.6. These results support the interpretation that transition abstraction improves generalization by supervising what state change should occur, rather than memorizing how a demonstrator realized that change in a particular GUI layout.

D Extractor Dependence, Replay Cost, and Scope

We provide controlled comparisons of extractor dependence, replay cost, supervision, and verification. Unless stated otherwise, comparisons use OS-ATLAS-7B with the same demonstrations, candidate generator, training budget, and online step budget. EqStep denotes equivalent-step success; all success rates and accuracies are percentages.

D.1 Transition representation and engineering dependence

The rule-based extractor is one implementation of transition supervision. A learned extractor takes before/after screenshots, the goal, and the executed action, with OCR and structure when available, and predicts the same type–slot descriptor. Its visual-only variant masks DOM/accessibility-tree inputs during training and evaluation. The open-vocabulary variant generates a short natural-language state delta and encodes it as the executor condition. These variants retain the supervision principle while testing the default representation.

Table 12: Extractor and vocabulary sensitivity. Exact is descriptor accuracy; Unseen is unseen-app task success; Cont. is continuous-control step success. All rows use the same TES backbone and evaluation budget.

Extractor / representation	Exact	EqStep	Unseen	Cont.	BadRate[†]
Rule-based, 15 types + slot	88.7	83.5	52.3	74.2	5.2
Coarse, 8 types + slot	87.2	80.6	49.5	69.8	6.3
Fine, 24 types + slot	87.9	83.1	51.9	76.0	5.4
Learned, 15 types + slot	87.4	82.9	51.6	74.8	5.5
Learned, visual-only	85.0	81.6	50.3	72.9	6.0
Open vocabulary, visual-only	85.6	82.2	51.0	76.8	5.8
Human-corrected descriptor	**100.0**	**84.8**	**53.4**	**78.1**	**4.7**

[†]BadRate is measured on the unseen-app split. Open-vocabulary descriptor accuracy uses human canonicalization for evaluation; human corrections do not enter training.

The learned extractor is 0.6 EqStep points below the default, and visual-only extraction retains 81.6 versus 72.9 for matched action imitation. Coarse types increase type accuracy to 95.4 but lower EqStep to 80.6, showing that type accuracy alone can hide harmful semantic merging. The continuous-control subset contains approximately 1,200 steps and 300 AndroidWorld/OSWorld tasks covering value changes, drag-and-drop, resizing, and reordering. Fine and open-vocabulary descriptors improve continuous step success, but this evaluation does not cover free-form drawing or complex 3D editing. The human-corrected descriptor is a diagnostic upper bound, not a deployable method.

The engineering audit separates shared definitions from environment I/O. It records 15 transition types (6 person-hours), 27 rule-score templates (22), one observation interface (12), eight slot schemas (10), and six candidate-primitive handlers (14), totaling 64 person-hours. There are four thin benchmark-family adapters each for rule scores, observations, slots, and action primitives, totaling 16 adapters. The audit records no website/app-specific rules or per-instance human overrides. This describes the evaluated implementation; adapting to a different environment can still require new I/O mappings or transition semantics.

D.2 Budgeted replay and static demonstrations

Replay expands coverage when controlled execution and state reset are available; it is not required to define the TES loss. For each demonstrated state, the replay budget bounds attempted candidate actions. A failed reset retains the state with the demonstration and any available deterministic positives. At zero replay budget, training can use demonstration-plus-rule positives or just the demonstration.

Four replay attempts reduce preprocessing CPU cost by 51.7% while lowering EqStep by 0.8 points. This setting gives WebArena/AndroidWorld/OSWorld task success of 55.5/55.7/38.0, compared with 56.1/56.3/38.6 for full replay. The corresponding zero-replay results are 53.9/54.1/36.7 with rule positives and 52.6/52.8/35.9 with demonstrations alone. Replay mainly increases verified

Table 13: Replay budget, preprocessing cost, and performance. CPU hours are per 10,000 demonstrated states and describe replay/positive construction, rather than total VLM/GPU training cost. Reset is reset/replay success; Prec. is positive precision; Cov. is alternative coverage.

Positive construction	Budget	Reset	Prec.	Cov.	CPU h	EqStep
Replay + rules	All	91.8	96.9	**69.4**	36.0	**83.5**
Replay + rules	8	92.0	97.0	67.8	27.6	83.3
Replay + rules	4	92.1	97.1	63.5	17.4	82.7
Replay + rules	2	**92.4**	97.3	57.6	9.8	81.8
Demo + rules	0	—	96.7	52.4	1.6	80.3
Demonstration only	0	—	**99.2**	28.5	**0.3**	77.1

alternative coverage; inaccessible or static datasets use the zero-replay path. These results quantify a cost–coverage trade-off rather than assuming perfect reset or exhaustive counterfactual enumeration.

D.3 Matched comparisons and the supervision contribution

State changes, outcome supervision, and verification are useful ingredients in prior GUI-agent work. The particular supervision unit in TES is the goal-conditioned equivalence class of verified action realizations. Table 14 isolates this choice from an added verifier and from outcome-semantic supervision under the same backbone and budgets. Heterogeneous agent scores elsewhere in the paper describe the surrounding ecosystem and do not, by themselves, isolate a supervision effect.

Table 14: Matched closest-paradigm comparison on OS-ATLAS-7B. Outcome-semantic supervision is a controlled comparator with the same backbone, data, and execution budget.

Method	EqStep	AltRecall@5	BadRate	WebArena
Action imitation	72.9	33.2	8.4	49.4
Action imitation + verifier	74.4	33.2	6.8	51.3
Outcome-semantic supervision	75.6	44.5	7.1	51.0
Transition-only	74.7	40.6	7.5	50.6
TES w/o positive marginalization	77.1	47.5	6.4	52.6
TES w/o online verification	78.0	**68.1**	7.1	51.9
Full TES	**83.5**	**68.1**	**4.2**	**56.1**

A verifier lowers BadRate without changing the offline executor ranking or AltRecall. Positive-set marginalization raises AltRecall from 47.5 to 68.1, while online verification further improves EqStep and task success. These comparisons support the combined design without attributing all improvement to post-action checking.

Table 15: Paired action-imitation (AI) and TES results across backbones. Each pair shares demonstrations, candidate generation, optimization, evaluators, and execution budget. WA/AW/OSW denote WebArena/AndroidWorld/OSWorld task success.

Backbone	Method	WA	AW	OSW	EqStep	BadRate
Phi-3.5-Vision	AI	44.6	45.3	28.5	64.8	13.6
	TES	51.8	52.1	33.2	76.7	6.8
Qwen2.5-VL-7B	AI	46.8	47.5	30.8	65.6	12.7
	TES	54.6	54.8	36.9	81.8	4.9
OS-ATLAS-7B	AI	49.4	49.8	33.5	72.9	8.4
	TES	**56.1**	**56.3**	**38.6**	**83.5**	**4.2**

The paired EqStep gains are 11.9, 16.2, and 10.6 points. These within-backbone differences provide the relevant evidence for a supervision benefit; comparisons across different observation regimes, parsers, tools, or model families remain qualified.

D.4 Irreversible actions and one-step equivalence

Verification compares the intended and realized transitions after execution. Detecting an incorrect action does not guarantee rollback. A subset of approximately 600 OSWorld/WindowsAgentArena tasks includes deletion, sending, submission, purchase/confirmation, account or system settings, and overwrite operations. Table 16 reports absolute failure counts because irreversible events are rare. The deployment variant adds a pre-execution gate using the executor's destructive-action cost, a higher transition-confidence threshold, and confirmation for sensitive actions.

Table 16: Risk subset. Attempted/wrong count risky actions; Irrev. counts irreversible or sensitive failures. Zero observed failures for the gated variant is not a safety guarantee.

Inference	Attempted	Wrong	Irrev.	False-block	Task SR
No verification	286	43	6	**0.0%**	39.8
Full TES	251	29	3	**0.0%**	**43.6**
TES + pre-execution gate	198	**12**	**0**	3.8%	43.2

Full TES detects 86.2% of wrong risky actions; automatic recovery succeeds for 72.0% of detected, recoverable errors. Eight failures require reset. The gate reduces observed irreversible/sensitive failures from three to zero, with a 0.4-point task-success decrease and 3.8% false blocks. Permission checks, sandboxing, confirmation, and application-level undo remain separate safeguards.

Finally, TES defines one-step equivalence at the same observation–goal pair. Different clicks, shortcuts, or widgets can belong to one class when they induce the same local descriptor. Trajectories with different intermediate transitions remain distinct even when their terminal states match. This scope avoids silently merging hidden side effects, path-length differences, or unresolved long-horizon credit assignment.

E Discussion

E.1 Scope and Limitations

TES is intended for GUI-agent settings where task progress is determined by the state transition induced by an action rather than by the exact surface action recorded in a demonstration. This scope is broad across web, mobile, and desktop interfaces, but it also clarifies when the method is most useful. When a GUI state admits only one practically valid action, TES naturally reduces close to standard action supervision, and the expected gain is small. The advantage becomes larger when multiple clicks, shortcuts, widgets, or menu paths can realize the same goal-relevant transition.

TES constructs verified positive sets conservatively. The method does not require enumerating all transition-equivalent actions; it only requires reliable positives obtained from demonstration actions, environment replay, and deterministic GUI rules. This design avoids treating visually similar but harmful actions as positives, while leaving open the possibility that additional valid actions may exist but remain unverified. Such conservative construction makes the training signal precise, although it may underestimate the full size of the transition-equivalence class.

The online verification module assumes that a post-action observation is available, which is standard in executable GUI environments. For offline-only action-prediction benchmarks, the same transition abstraction can still be used for training and evaluation, but online repair is not available because the environment cannot be stepped. Finally, the current transition vocabulary focuses on common GUI progress types such as query submission, filtering, dialog opening, item selection, form saving, page navigation, no-op, and error. The learned and open-vocabulary variants in Appendix D test alternatives to this default representation. Evaluation covers parameterized continuous controls, not free-form drawing or complex 3D editing. Equivalence is defined for one step from the same observation and goal; different intermediate trajectories are not merged solely because they reach the same terminal state. Post-action verification detects mismatches but does not undo irreversible actions or provide a safety guarantee.

E.2 Broader Impacts and Responsible Use

GUI agents can make digital services more accessible by helping users interact with websites, mobile applications, and desktop software through natural-language instructions. By shifting supervision from exact click imitation to transition-level progress, TES may improve robustness in realistic interfaces where multiple actions can complete the same local step. The verification and repair mechanism may also reduce repeated no-op actions, erroneous clicks, and unnecessary interaction loops, which are important for safe and efficient computer-use agents.

At the same time, more capable GUI agents can be misused if deployed without appropriate constraints. Potential risks include unauthorized automation of websites or desktop applications, accidental interaction with sensitive user data, over-reliance on autonomous execution, and unintended actions in high-stakes settings such as finance, healthcare, account management, or system administration [Yang et al., 2026a]. TES does not remove the need for deployment-time safeguards. In practical systems, we recommend human confirmation for sensitive or irreversible actions, sandboxed execution during evaluation, explicit action-risk filtering, logging and auditability [Shao, 2026a,b, Ma et al., 2026, Wang et al., 2026c, Wang and Saxena, 2026, Fan et al., 2023], rate limits, and compliance with website, application, and benchmark terms of use.

The experiments in this paper are conducted on existing benchmarks and controlled evaluation environments. We do not release a new autonomous agent service, scraped dataset, or pretrained model. The proposed framework is intended as a research contribution toward more reliable GUI-agent learning rather than as a standalone deployment system.

E.3 Declaration of LLM Usage

The evaluated GUI agents use the pretrained Qwen2.5-VL-7B-Instruct, Phi-3.5-Vision-Instruct, and OS-ATLAS-7B backbones and the LLM/VLM baselines cited in the main text. Paper-preparation assistance, where used, was limited to language editing, LaTeX formatting, code-snippet drafting, plotting-script debugging, and note organization.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state the central claim of the paper: GUI-agent supervision should be organized around goal-relevant state transitions rather than exact surface-action identity. The claimed components of Transition-Equivalent Supervision (TES), including transition descriptor extraction, transition-conditioned execution, verified positive action sets, and post-action verification, are defined in Sections 4–4.5. The experimental claims are evaluated across offline web prediction, executable web navigation, mobile control, and desktop-use benchmarks in Section 5.1 and the appendix.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper discusses the scope and boundary conditions of TES in Appendix. In particular, it clarifies that TES is most relevant when multiple executable actions can realize the same task-relevant transition, that verified positive sets are constructed conservatively rather than exhaustively, and that online verification requires access to post-action observations in executable environments.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.

- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: The paper does not claim formal theoretical guarantees or theorem-level results. The mathematical content defines the GUI interaction process, transition descriptors, transition equivalence, the TES training objective, stochastic equivalence, and verification-based repair, but no theorem or formal proof is presented as a main contribution.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper specifies the benchmark families, evaluation metrics, compared method groups, observation regimes, fairness constraints, candidate-action construction, transition vocabulary, transition extraction rules, verified-positive construction, training objective, LoRA configuration, optimization hyperparameters, inference procedure, random seeds, bootstrap protocol, and compute resources in the main text and Appendix B.2–B.5. Algorithm 1 describes the main training and inference procedures.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.

- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: The training, inference, and evaluation code has not yet been publicly released. Appendix B.4 specifies the benchmark and implementation settings, and Appendix A provides the method details and training/inference pseudocode for independent reimplementation.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The paper specifies the benchmark suite, evaluation metrics, trainable backbones, LoRA settings, optimizer, learning rates, batch size, training epochs, candidate-action budgets, transition-extraction thresholds, positive-set construction, validation-based checkpoint selection, maximum online step budgets, inference-time verification, and repair-memory settings in Section 5.1 and Appendix B.4.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Offline experiments are run with three random seeds, and the paper reports mean and standard deviation when repeated runs are available. For online execution experiments, the paper reports bootstrap 95% confidence intervals over tasks using $10{,}000$ bootstrap samples, because repeated full online evaluation is substantially more expensive. The random-seed and confidence-interval protocols are described in Appendix B.4 and Appendix B.5.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Appendix B.5 reports the GPU type, GPU memory, CPU environment workers, storage footprint, per-experiment GPU-hour estimates, online-evaluation cost, and total compute used for the reported experiments. The appendix also separates main fine-tuning, ablation, transition-label construction, offline evaluation, online execution, and preliminary diagnostic costs.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.

- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The work uses existing GUI-agent benchmarks and publicly described model backbones, does not collect new personal data, does not introduce a new scraped dataset, and does not release a new high-risk pretrained model. The paper focuses on improving reliability and reducing repeated erroneous GUI actions through transition-level supervision and post-action verification. Responsible-use considerations are discussed in Appendix E.2.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Appendix E.2 discusses potential positive impacts, including more reliable GUI agents, improved accessibility for computer use, and reduced repeated no-op or erroneous interactions. It also discusses potential risks, including unintended automation of sensitive GUI operations, over-reliance on autonomous agents, and misuse in unauthorized web or desktop interaction, together with mitigation strategies such as human confirmation, sandboxed evaluation, action-risk filtering, and respecting website and application terms of use.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not release a new pretrained language model, image generator, scraped dataset, or standalone high-risk asset. The proposed contribution is an algorithmic supervision framework evaluated on existing GUI-agent benchmarks. Responsible-use discussion for downstream GUI-agent deployment is provided in Appendix E.2.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The paper cites the original works for the benchmark datasets, model backbones, and baseline systems used in the experiments.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not introduce or release a new dataset, benchmark, pretrained model, or standalone software asset at submission time. The contribution is a supervision framework and evaluation protocol built on existing GUI-agent benchmarks. Transition descriptors and verified positive sets are internal training/evaluation annotations derived from existing benchmark trajectories and are described algorithmically in Appendix A.1 and Appendix A.2.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing experiments or new research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve new human-subject research, crowdsourcing, or collection of participant data, so IRB approval is not applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

Justification: We describe the role of LLMs and VLMs in the paper. The evaluated GUI agents instantiate TES on existing VLM backbones and compare against LLM/VLM-based GUI-agent baselines, all of which are cited in the paper. LLM-based tools, if used for writing assistance, grammar editing, LaTeX formatting, or code debugging, did not determine the scientific claims, experimental design, results, or conclusions, which were checked and approved by the authors.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.