GUI-ECO: AN EFFICIENT EDGE–CLOUD ORCHESTRATION FRAMEWORK FOR GUI AGENTS VIA DUAL DECISION PROCESSES

Anonymous authors
Paper under double-blind review

ABSTRACT

The growing deployment of GUI agents on resource-constrained devices motivates collaboration between lightweight edge agents and more capable cloud agents. However, due to the multimodal and long-horizon nature of GUI tasks, existing collaboration frameworks struggle to determine when cloud assistance is needed and lack cloud-side acceptance mechanisms, resulting in low overall task success rates and unnecessary computational costs. To address these challenges, we propose **GUI-ECO**, a dual-decision **E**dge–**C**loud **O**rchestration framework, which comprises two modules: a trajectory-level edge handoff module and a competence-aware cloud acceptance module. We construct a dataset from extensive real-world GUI task executions and use the resulting supervision to train and refine both modules based on agents' hidden representations. For edge-side handoff decisions with GUI-Owl-1.5, GUI-ECO achieves 85% Recall and 7% FPR on OSWorld. Overall, extensive experiments show that GUI-ECO achieves the highest cloud execution efficiency through its dual-decision process, reaching up to 2.6× that of prior methods.

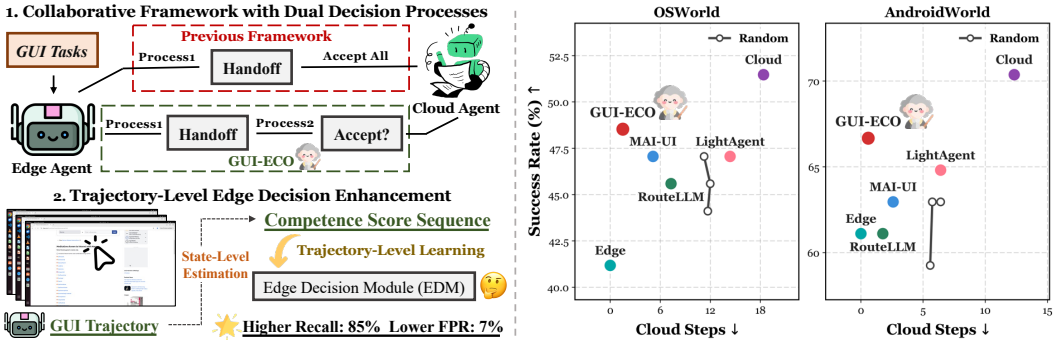

Figure 1: **Left:** 1) GUI-ECO introduces a dual-decision edge–cloud collaboration framework, where the edge decides whether to hand off and the cloud independently determines whether to accept the request. 2) GUI-ECO enhances edge-side handoff decisions through trajectory-level learning, achieving higher Recall and lower false positive rates. **Right:** GUI-ECO achieves a favorable performance–cost trade-off with GUI-Owl-1.5 on OSWorld and AndroidWorld.

1 INTRODUCTION

GUI agents are increasingly being deployed across diverse edge devices, such as smartphones and personal computers, to autonomously carry out a wide range of tasks (Xu et al., 2026; Seed, 2025; Gu et al., 2025). These on-device agents perform reasoning, planning, and GUI interaction with modest computational resources (Li et al., 2026). Meanwhile, service providers are deploying more computationally intensive GUI agents in the cloud, which offer stronger and more robust task-execution capabilities (Jiang & Huang, 2026; Fan et al., 2025). As GUI agents move toward broader real-world and commercial adoption, recent studies have begun to investigate how edge and cloud agents can work together to balance task performance against computational cost.

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

Existing edge–cloud collaboration frameworks typically follow two paradigms. The first relies on the edge agent to monitor task progress and autonomously trigger a handoff signal to offload the task to the cloud (Zhou et al., 2025; Jiang & Huang, 2026). While this paradigm is intuitively reasonable, relying solely on the edge agent to inspect task states and emit natural-language handoff signals is inherently unreliable. The second paradigm allocates tasks through routing policies or classifiers (Chen et al., 2023; Ong et al., 2025; Zhang et al., 2026). Compared with natural-language handoff signals, these methods provide more explicit and quantifiable decision criteria. However, both paradigms make limited use of trajectory-level information for long-horizon task execution and typically focus only on the edge-side handoff decision, without explicitly assessing whether the cloud agent can successfully complete the requested task.

To address these limitations, we propose **GUI-ECO**, a more reliable **E**dge-**C**loud **O**rchestration framework with a dual decision process. Our key insights are twofold. First, the hidden states of GUI agents provide informative representations of the future task-completion value of the current state from ongoing multimodal GUI interactions, **facilitating a dual-decision process on both the edge and cloud sides.** Second, **trajectory-level modeling captures dependencies across sequences of task states that cannot be identified from individual states alone**, enabling more accurate handoff decisions for the edge agent. As shown in Figure 2, we construct State-Level Competence Estimation (SCE) Heads from agents' hidden states under rollout success rate (Bu et al., 2026) supervision and further improve their discriminative capability through Joint Head-Model Training. The resulting competence estimates form the basis of our dual-decision process and are directly used for cloud-side acceptance decisions. On the edge side, we further build a trajectory-level decision module over these estimates, enabling more accurate handoff decisions than prior state-level approaches. Finally, GUI-ECO integrates these components into a dual-decision edge–cloud collaboration framework, achieving a favorable trade-off between task success rate and cloud-side computation, as illustrated on the right side of Figure 1.

Our main contributions are summarized as follows:

- **Dual-decision edge–cloud collaboration.** We introduce a dual-decision framework that combines an edge-side handoff decision with a cloud-side acceptance gate. Compared with prior collaboration frameworks that rely solely on edge-side routing, **GUI-ECO** reduces unnecessary cloud computation while achieving a higher task success rate.
- **Trajectory-level edge decision modeling.** We introduce a trajectory-level edge decision module that uses state-wise competence scores to make more reliable handoff decisions. Compared with previous methods, our trajectory-aware approach provides more robust handoff decisions on complex GUI trajectories, achieving higher handoff recall and lower false positive rates.
- **Comprehensive evaluation of edge-cloud collaboration frameworks.** To the best of our knowledge, we conduct the first systematic evaluation of edge–cloud collaboration for GUI agents across four benchmarks spanning both mobile and desktop environments. Extensive experiments show that GUI-ECO achieves the highest cloud execution efficiency, improving over prior approaches by up to $2.6\times$.

2 Related Work

GUI Agents. Driven by rapid advances in vision-language models (VLMs) (OpenAI et al., 2024; Team et al., 2025; Bai et al., 2025), GUI agent research has undergone a paradigm shift from modular systems built upon general-purpose VLMs to native GUI models that unify the core capabilities required for general-purpose GUI interaction, including perception, planning, and action (Liu et al., 2024; Gou et al., 2025; Ding et al., 2026). Early systems predominantly followed the former paradigm, augmenting backbone VLMs with external modules and role-specific agents. AppAgent-v2 (Li et al., 2025) and Mobile-Agent (Wang et al., 2024b) enhance GUI interaction through retrieval-augmented operational knowledge and dedicated visual perception tools, respectively. Other systems, such as Mobile-Agent-v2 (Wang et al., 2024a) and Agent S2 (Agashe et al., 2025), adopt role-based or multi-agent architectures that distribute planning, reflection, and execution across specialized components, thereby improving robustness in complex GUI tasks.

More recent work has shifted toward native end-to-end GUI agents. GUI-Owl-1.5 (Xu et al., 2026) scales unified GUI policies across mobile, web, and desktop environments through multi-platform post-training and environment-level reinforcement learning. Meanwhile, UI-Venus-1.5 improves the generality and robustness of native GUI agents by combining large-scale GUI mid-training with full-trajectory online reinforcement learning and domain-specific model merging (Team et al., 2026).

Despite their strong end-to-end and general-purpose GUI capabilities, these agents still lack explicit mechanisms for failure awareness on the edge side and executability assessment on the cloud side. Therefore, reliable edge-cloud orchestration is essential for deploying these agents in practical collaboration settings to improve overall task success and computational efficiency.

Edge-Cloud Collaboration Framework. Existing edge-cloud collaboration frameworks mainly follow two paradigms. The first paradigm enables the edge model, through prompt design or dedicated training, to monitor task execution and proactively emit handoff signals (Aggarwal et al., 2024; Fang et al., 2026). LightAgent (Jiang & Huang, 2026) designs prompts that enable the edge model to monitor task progress based on real-time risks and potential failure patterns, and accordingly make routing decisions for subsequent execution. Zhou et al. (2025) instead train the edge model to internalize task monitoring and handoff behaviors, enabling it to detect trajectory deviations or insufficient local capability and proactively request cloud assistance. These methods rely on prompt design and the capability of the edge model, using natural-language outputs as decision signals rather than explicit, quantifiable criteria for more reliable decision-making (Yang et al., 2024; Chuang et al., 2025).

The second paradigm introduces explicit routing models or policies to determine task allocation. Hybrid LLM (Ding et al., 2024) predicts query difficulty and the expected quality gap to route queries between small and large models, while RouteLLM (Ong et al., 2025) learns routing policies from human preference data to select between stronger and weaker LLMs. DoT (Shao et al., 2025) further enables subtask-level edge-cloud collaboration by extracting internal representations of subtask prompts from the edge model and training an adapter for binary edge–cloud routing. Although these learned routing approaches provide more explicit and quantifiable decision criteria, they share two important limitations with the first paradigm: making insufficient use of trajectory-level information for agent task execution, and failing to explicitly assess whether the requested task is actually executable by the cloud model before accepting the handoff. Our method instead combines trajectory-level edge decision-making with state-wise competence estimation on both the edge and cloud sides, forming a dual-decision process for more reliable and efficient edge–cloud collaboration.

3 METHODOLOGY: **GUI-ECO**

3.1 OVERVIEW

Formulation. We consider an edge agent π_{e} and a cloud agent π_{c} interacting with a GUI environment. At step t, the interaction content comprises the GUI screen g_t and the prompt context c_t, which contains task-related information such as the instruction, current subtask, and execution history. Execution starts on the edge, while cloud execution is controlled by the learned decision modules. Our objective is to improve the overall task success rate while minimizing cloud execution cost, measured by the average number of cloud execution steps per task.

Figure 2 illustrates the overall construction process of GUI-ECO. *Data Collection* gathers intermediate interaction states and estimates their agent-specific task-completion values through repeated continuation rollouts. *Decision Module Construction* builds two types of decision modules: a State-Level Competence Estimation (SCE) head for each agent, learned through head pre-training and joint head–model training; and a trajectory-level Edge Decision Module (EDM), trained on sequences of edge competence scores. Finally, the *Inference-Time Collaboration* integrates these modules into a dual-decision execution process: the EDM proposes handoffs based on the edge trajectory, and the cloud SCE head assesses the proposed execution before accepting it.

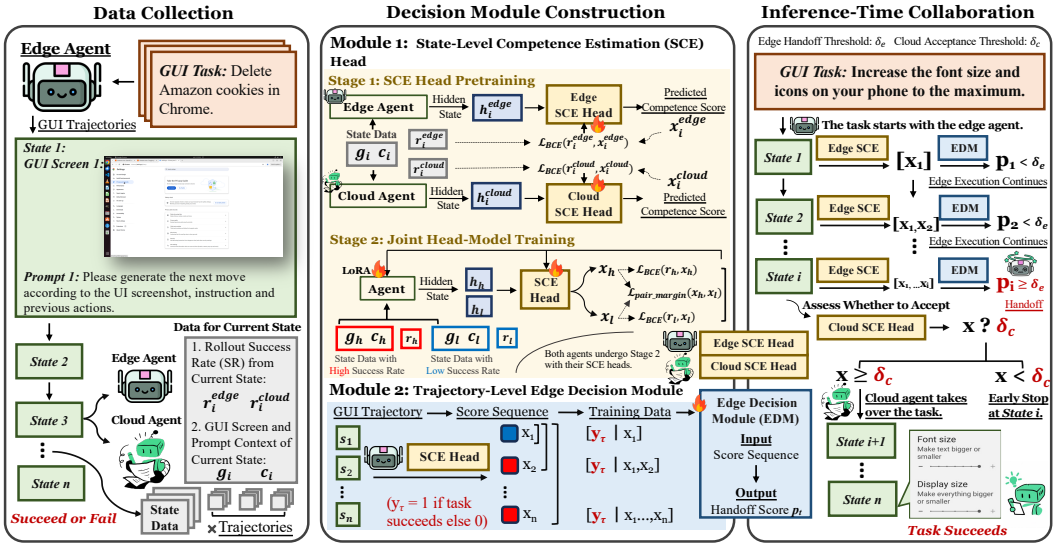

Figure 2: Overview of GUI-ECO. **Data Collection:** We collect GUI trajectories and estimate each agent's competence through repeated rollouts from intermediate states. **Decision Module Construction:** SCE heads learn from hidden states through pre-training and joint head–model training, while the EDM models sequences of edge competence scores. **Inference-Time Collaboration:** The EDM proposes handoffs, while the cloud SCE head independently determines whether to take over execution or stop the task early.

3.2 Data Collection

Interaction-state collection. We collect edge execution trajectories from GUI tasks, retaining both successful and failed executions. For each trajectory, we sample one intermediate state every Δ steps and record its GUI screen g_i and prompt context c_i, preserving the information available to the agent at that step. For example, for GUI-Owl-1.5 (Xu et al., 2026), c_i includes the system and tool instructions, the original user request, and the preceding interaction history, while g_i denotes the current screenshot presented to the agent.

Rollout Success Rate Collection. For each recorded state, we estimate the competence of agent $a \in \{\text{edge}, \text{cloud}\}$ through repeated continuation rollouts from that state. We record the rollout success rate as $r_i^a = \frac{1}{K}\sum_{k=1}^{K} Y_{i,k}^a$, where $Y_{i,k}^a \in \{0,1\}$ indicates whether the k-th rollout successfully completes the task and K denotes the number of rollouts conducted with agent a. We regard r_i^a as the empirical competence of agent a at state i, reflecting its likelihood of completing the task from the current interaction state.

The resulting state-level dataset is $\mathcal{D}_{\text{state}} = \{(g_i, c_i, r_i^{\text{e}}, r_i^{\text{c}})\}_{i=1}^{M}$, where M is the number of sampled states. The rollout success rates serve as soft supervision signals, preserving variation across continuations instead of assigning each state a binary label from a single execution.

3.3 Decision Module Construction

We construct two complementary decision modules from the collected data. The first module estimates state-wise competence for both the edge and cloud agents, while the second models the evolution of edge competence over the execution trajectory to support handoff decisions.

Module 1: State-Level Competence Estimation (SCE) Head. The SCE head provides an agent-specific competence score for each interaction state. We construct separate SCE heads for the edge and cloud agents through two training stages: SCE head pre-training followed by joint head–model training.

Stage 1: SCE Head Pre-training. For agent a, we use the hidden state of the last input token from a selected layer ℓ as the representation of the current interaction state, denoted as $h_i^{a,\ell}$.

A lightweight MLP-based SCE head g_{ϕ_a} predicts a competence score through a sigmoid output:

$$x_i^a = \sigma\Big(g_{\phi_a}(h_i^{a,\ell})\Big), \tag{1}$$

where $\sigma(\cdot)$ is the sigmoid function. A higher x_i^a indicates a higher estimated likelihood that agent a can successfully complete the remaining task from the current interaction state. We use the empirical rollout success rate r_i^a as a soft supervision signal and optimize the binary cross-entropy objective:

$$\mathcal{L}_{\mathrm{BCE}}^a = -\frac{1}{M}\sum_{i=1}^{M}\left[r_i^a \log x_i^a + (1-r_i^a)\log(1-x_i^a)\right]. \tag{2}$$

The usefulness of hidden representations can vary across model depths (Skean et al., 2025). We therefore train separate SCE heads on five candidate layers at different depths and select the head and layer with the best validation performance for subsequent joint training.

Stage 2: Joint Head–Model Training. Pre-training learns to read competence information from fixed representations, but does not explicitly improve how such information is encoded. We therefore introduce LoRA parameters $\Delta\theta_a$ and jointly optimize them with the SCE head parameters ϕ_a, while keeping the original model parameters θ_a^0 frozen. This joint training allows competence supervision to shape the hidden representations, encouraging states with different task-completion capabilities to become more distinguishable.

As illustrated in Figure 2, joint training combines competence-score calibration with pairwise score-margin enhancement. We retain $\mathcal{L}_{\mathrm{BCE}}^a$ and sample ordered state pairs (i, j) satisfying $r_i^a < r_j^a$, where i and j correspond to lower- and higher-success states, respectively. Using the jointly adapted hidden representation $h_i^{a,\ell}$, we compute x_i^a as in equation 1 and define its competence logit as $\eta_i^a = \log \frac{x_i^a}{1-x_i^a}$.

The pairwise objective encourages the higher-success state to receive a larger competence logit:

$$\mathcal{L}_{\mathrm{pair}}^a = \frac{1}{|\mathcal{P}_a|}\sum_{(i,j)\in\mathcal{P}_a}\log\big(1+\exp\big[m-(\eta_j^a-\eta_i^a)\big]\big), \tag{3}$$

where $\mathcal{P}_a$ denotes the sampled set of ordered state pairs and $m > 0$ is the desired logit margin.

The overall competence-learning objective is

$$\mathcal{L}_{\mathrm{comp}}^a = \mathcal{L}_{\mathrm{BCE}}^a + \lambda_{\mathrm{pair}}\mathcal{L}_{\mathrm{pair}}^a, \tag{4}$$

where λ_{pair} controls the contribution of the pairwise objective. The BCE term calibrates the competence score x_i^a toward the empirical rollout success rate r_i^a, while the pairwise term strengthens the separation between states with different competence levels in the latent space, allowing the SCE head to produce more accurate competence scores from hidden representations. We apply the same training procedure independently to the edge and cloud agents.

Module 2: Trajectory-Level Edge Decision Module (EDM). A state-level competence score provides only a local assessment. We therefore introduce a trajectory-level Edge Decision Module (EDM) to capture temporal dependencies across consecutive edge competence scores. Given a competence-score sequence, the EDM outputs a trajectory-level failure-risk score p_t, indicating the likelihood that the current edge trajectory will eventually fail.

After training Module 1, we use the edge agent and its SCE head to obtain a competence-score sequence $(x_1^{\mathrm{e}}, \ldots, x_{T_\tau}^{\mathrm{e}})$ for each edge trajectory τ, where T_τ denotes the sequence length. We construct a training sample from every prefix of this sequence: $\{(x_{1:t}^{\mathrm{e}}, y_\tau)\}_{t=1}^{T_\tau}$, where $y_\tau = 1$ denotes task success. Thus, a trajectory of length T_τ contributes T_τ prefix-level training samples, all supervised by the same terminal task outcome. We denote the resulting set of labeled prefixes by $\mathcal{D}_{\mathrm{EDM}}$.

For each prefix, the EDM processes the competence scores sequentially. At step t, x_t^{e} is first projected to an 8-dimensional feature through a linear layer followed by $\tanh$, i.e., $e_t = \tanh(W_{\mathrm{in}} x_t^{\mathrm{e}} + b_{\mathrm{in}})$. A GRU then updates a 16-dimensional hidden state as $h_t = \mathrm{GRU}(e_t, h_{t-1})$. The hidden state is mapped through a linear layer and sigmoid to produce $p_t = \sigma(W_{\mathrm{out}} h_t + b_{\mathrm{out}})$, which represents the failure risk associated with the current prefix $x_{1:t}^{\mathrm{e}}$. All parameters of the input projection, GRU, and output layer are jointly optimized.

Since y_τ denotes task success, we use $1 - y_\tau$ as the failure label for every prefix and train the EDM with binary cross-entropy:

$$\mathcal{L}_{\mathrm{EDM}} = -\mathbb{E}_{(x_{1:t}^{\mathrm{e}}, y_\tau) \sim \mathcal{D}_{\mathrm{EDM}}} \left[(1 - y_\tau) \log p_t + y_\tau \log(1 - p_t) \right]. \tag{5}$$

3.4 Inference-Time Collaboration

At inference time, GUI-ECO starts task execution on the edge. At each step, the edge SCE head produces a competence score x_t^{e}, which the GRU-based EDM processes sequentially to obtain a trajectory-level failure-risk score p_t. The edge agent requests cloud assistance when $p_t \geq \delta_{\mathrm{e}}$, where δ_{e} is the handoff threshold; otherwise, it continues execution. This decision considers the competence-score history rather than an isolated state-level estimate.

Upon receiving a handoff request, the cloud agent computes its hidden representation of the current interaction context, and its SCE head produces a competence score x_t^{c}. If this score meets the cloud acceptance threshold ($x_t^{\mathrm{c}} \geq \delta_{\mathrm{c}}$), the cloud agent takes over and continues task execution from the current state. Otherwise, the task is terminated early at that state.

4 Experiments

4.1 Experimental Settings

GUI Agents. We select three pairs of GUI agents to instantiate edge–cloud deployments: *GUI-Owl-1.5 2B/32B* (Xu et al., 2026), *MAI-UI 2B/8B* (Zhou et al., 2025), and *UI-Mate 9B/27B* (Ding et al., 2026). These agent pairs exhibit varying capability gaps across scales, providing diverse edge-cloud settings for evaluating the effectiveness of different collaboration frameworks.

Benchmarks. We evaluate different edge-cloud collaboration frameworks on four representative GUI-agent benchmarks across desktop and mobile environments: OSWorld (Xie et al., 2024), VisualWebArena (Koh et al., 2024), AndroidWorld (Rawles et al., 2025) and MobileWorld (Kong et al., 2026). OSWorld contains 369 tasks involving web and desktop applications, including complex cross-application workflows. VisualWebArena constructs text-image inputs to simulate more realistic user scenarios and visually grounded web interactions. For mobile-side evaluation, AndroidWorld provides 116 programmatically verifiable tasks with dynamically instantiated task variants, whereas MobileWorld introduces 201 tasks with longer-horizon and more complex workflows.

Baselines. We compare GUI-ECO against baselines across different edge-cloud collaboration paradigms. First, we consider simple execution policies, including All-Edge, All-Cloud, and Random Routing, where each task is executed entirely by the edge/cloud agents, or randomly assigned between the two. Second, we include the mechanisms proposed in MAI-UI (Zhou et al., 2025) and LightAgent (Jiang & Huang, 2026), where the edge agent monitors whether the current execution state deviates from the user instruction and autonomously triggers cloud assistance when necessary. Finally, we adopt RouteLLM (Ong et al., 2025), a representative learned routing method that estimates the relative advantage of edge and cloud models for each task and selects the execution model accordingly. We provide additional implementation details for the baselines in Appendix A.1.

Implementation Details. We collect data from OSWorld and AndroidWorld for decision-module construction and split the tasks into training, validation, and test sets. We tune the edge handoff threshold δ_{e} and cloud acceptance threshold δ_{c} on the validation set. More implementation details are provided in Appendix A.2.

Metrics. We evaluate each collaboration framework from both effectiveness and efficiency perspectives. In addition to *Task Success Rate (SR)* and *Cloud Execution Steps (Cost)*, we introduce *Normalized Cloud-Step Efficiency (NCSE)* to enable fair cloud execution efficiency compar-

Table 1: Evaluation of different edge–cloud routing strategies on GUI agent benchmarks. We report task success rate (SR), cloud execution cost, and NCSE. GUI-ECO Cost includes SCE assessment overhead expressed as equivalent cloud execution steps. **Bold** and underlined values denote the best and second-best NCSE among routing methods, respectively, excluding All Cloud and All Edge.

Agent Pair	OSWorld		VisualWebA.		AndroidWorld		MobileWorld		NCSE ↑
	SR. ↑	Cost ↓	SR. ↑	Cost ↓	SR. ↑	Cost ↓	SR. ↑	Cost ↓	
GUI-Owl-1.5 (2B/32B)									
All Cloud	51.47	18.38	45.00	10.30	70.37	12.33	36.19	17.04	1.00
All Edge	41.18	0.00	27.50	0.00	61.11	0.00	28.57	0.00	–
Random	44.12	11.69	31.67	3.86	62.96	5.74	29.52	8.72	0.43
RouteLLM	45.59	7.29	40.83	9.08	61.11	1.76	28.57	5.71	0.68
LightAgent	47.06	14.37	31.67	3.18	64.81	6.41	28.57	13.37	0.50
MAI-UI	47.06	5.13	30.00	3.24	62.96	2.59	33.33	10.60	<u>1.08</u>
GUI-ECO	48.53	1.59	41.67	4.99	66.67	0.65	32.38	5.12	**2.84**
MAI-UI (2B/8B)									
All Cloud	39.71	15.81	48.33	8.09	61.11	15.13	24.76	17.36	1.00
All Edge	16.18	0.00	26.67	0.00	48.15	0.00	14.29	0.00	–
Random	20.59	7.18	34.17	2.44	55.56	7.78	16.19	8.71	<u>0.73</u>
RouteLLM	20.59	5.40	33.33	1.99	46.30	0.24	15.24	1.23	0.66
LightAgent	23.53	12.29	37.50	7.29	59.26	8.76	16.19	9.04	0.67
MAI-UI	19.12	12.75	35.83	7.08	50.00	7.06	17.14	8.57	0.36
GUI-ECO	26.47	3.14	43.33	4.91	55.56	2.09	20.00	7.32	**1.70**
UI-Mate (9B/27B)									
All Cloud	55.88	16.38	40.83	10.00	42.59	12.47	30.48	16.46	1.00
All Edge	39.71	0.00	30.83	0.00	24.07	0.00	20.95	0.00	–
Random	42.65	7.90	35.00	5.40	31.48	5.85	26.67	8.27	0.80
RouteLLM	47.06	3.09	31.67	2.20	24.07	1.98	21.90	2.11	0.92
LightAgent	44.12	11.43	40.00	9.93	40.74	9.30	25.71	15.24	0.77
MAI-UI	48.53	7.28	33.33	3.40	38.89	1.25	25.71	6.99	<u>1.60</u>
GUI-ECO	52.94	4.30	38.33	3.99	40.74	3.27	26.67	2.58	**2.84**

isons across different benchmarks. Following RouteLLM (Ong et al., 2025), we define *Performance Gap Recovery (PGR)* and *Normalized Cloud Steps (NCS)* for method m on benchmark b as $\mathrm{PGR}_{m,b} = (A_{m,b} - A_{\mathrm{edge},b})/(A_{\mathrm{cloud},b} - A_{\mathrm{edge},b})$ and $\mathrm{NCS}_{m,b} = S_{m,b}/S_{\mathrm{cloud},b}$, where A and S denote task success rate and cloud execution steps, respectively. We then compute $\mathrm{NCSE}_m = \mathrm{Avg}_b(\mathrm{PGR}_{m,b})/\,\mathrm{Avg}_b(\mathrm{NCS}_{m,b})$. Higher NCSE indicates greater performance-gap recovery per unit of normalized cloud usage. Since GUI-ECO performs an SCE-based acceptance check before cloud execution, we convert its measured computation time into equivalent cloud steps based on the average per-step cloud inference time and include this overhead in the reported Cost (see Appendix A.4 for details). In subsequent experiments, we report both state-level and task-level Recall and FPR, with detailed definitions provided in Appendix A.3.

4.2 Overall Results

Table 1 reports the performance of different edge–cloud collaboration frameworks across three agent pairs and four GUI benchmarks. GUI-ECO improves upon All Edge in all 12 settings and achieves the highest or joint-highest SR among routing methods in nine. Across the three agent pairs, its benchmark-averaged success rate improvement over All Edge ranges from 7.72 to 10.78 percentage points, while its mean normalized cloud usage is only 23.10%–34.13%. GUI-ECO achieves the highest aggregate NCSE for every agent pair: 2.84, 1.70, and 2.84 for GUI-Owl-1.5, MAI-UI, and UI-Mate, respectively, compared with 1.08, 0.73, and 1.60 for the strongest competing routing method in each pair. The advantage is particularly evident for MAI-UI on OSWorld, which has the largest edge–cloud performance gap. GUI-ECO achieves higher SR than LightAgent with substantially fewer cloud steps.

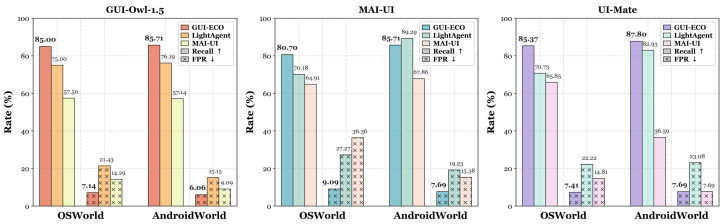

Figure 3: Edge-side task-level handoff Recall and FPR on OSWorld and AndroidWorld. Appendices A.4 and C.1 further demonstrate the robustness of GUI-ECO's edge-side decision making through additional results and a case study.

Table 2: Cloud-side acceptance Recall and FPR (%) on handoff requests. More results are provided in Appendix A.4.

Agent	OSW.		AW.	
	Rec.	FPR	Rec.	FPR
GUI-Owl-1.5	87.50	7.14	83.33	7.14
MAI-UI	80.00	8.11	85.71	5.26
UI-Mate	91.67	8.00	90.91	7.69

Figure 3 further examines edge-side handoff quality. Compared with LightAgent and MAI-UI, GUI-ECO consistently achieves lower FPR while maintaining competitive Recall across the displayed settings. For example, with GUI-Owl-1.5 on OSWorld, GUI-ECO improves Recall by 10.00 percentage points and reduces FPR by 14.29 percentage points relative to LightAgent, identifying more edge failures while avoiding unnecessary handoffs. For MAI-UI on AndroidWorld, LightAgent attains slightly higher Recall than GUI-ECO (89.29% versus 85.71%), but incurs substantially higher FPR (19.23% versus 7.69%). Complementing these results, Table 2 reports cloud-side acceptance performance. The cloud SCE head achieves high Recall with low FPR, accepting requests it can complete while limiting acceptance of requests unlikely to succeed. These two decisions support the favorable performance–cost trade-off achieved by GUI-ECO.

4.3 Ablation Study

Joint head–model training. Table 3 shows that removing joint training, while retaining the pre-trained SCE heads, lowers SR and increases cloud cost. Figure 4 further illustrates the gains in state-level competence estimation. Unlike head-only training, joint adaptation allows competence supervision to shape the underlying representations, benefiting both downstream routing and cloud-side assessment. Appendix A.5 further shows that our joint-training design does not materially affect the agents' original task-execution capability.

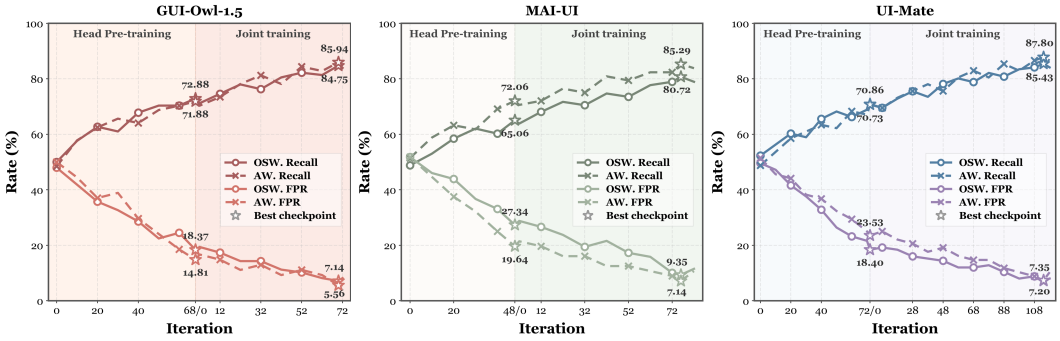

Figure 4: Edge-side state-level Recall and FPR during SCE head pre-training and joint training.

Trajectory-level routing. Without the GRU-based EDM, edge-side handoff decisions reduce to state-level routing based solely on individual competence estimates; specifically, the task is handed off as soon as the trajectory reaches a state classified as requiring handoff, as detailed in Appendix A.5. This ablation reduces NCSE from 2.84 to 1.79 for GUI-Owl-1.5, with similar degradations also observed for MAI-UI and UI-Mate, as shown in Table 3. These results support the benefit of modeling dependencies across consecutive competence scores. Appendix C.2 provides a concrete example in which the EDM avoids premature handoff despite transient drops in state-level competence. Meanwhile, Table 10 in the appendix compares the EDM with a step-window smoothing strategy.

Cloud-side acceptance. Removing the cloud SCE head accepts every proposed handoff, yielding modest SR gains at substantially greater cloud cost. For GUI-Owl-1.5 on AndroidWorld, SR rises from 66.67% to 68.52%—one additional successful task—while cloud steps increase from 0.65 to

Table 3: Component ablation results for GUI-ECO. Each benchmark reports task success rate (%) and average cloud execution cost. NCSE denotes Normalized Cloud-Step Efficiency. Parenthesized values indicate NCSE changes relative to GUI-ECO within the same agent pair.

Agent Pair	OSWorld		VisualWebArena		AndroidWorld		MobileWorld		NCSE ↑
	SR. ↑	Cost ↓	SR. ↑	Cost ↓	SR. ↑	Cost ↓	SR. ↑	Cost ↓	
GUI-Owl-1.5 (2B/32B)									
GUI-ECO	48.53	1.59	41.67	4.99	66.67	0.65	32.38	5.12	2.84
w/o Joint Training	44.12	3.87	36.67	7.23	62.96	1.67	29.52	8.28	0.74 (−2.10)
w/o EDM	48.53	3.15	40.83	6.44	66.67	1.89	31.43	7.14	1.79 (−1.05)
w/o Cloud SCE Head	50.00	3.75	43.33	7.82	68.52	2.39	33.33	8.56	1.92 (−0.92)
MAI-UI (2B/8B)									
GUI-ECO	26.47	3.14	43.33	4.91	55.56	2.09	20.00	7.32	1.70
w/o Joint Training	23.53	5.24	36.67	7.17	51.85	4.31	17.14	10.52	0.63 (−1.07)
w/o EDM	25.00	4.54	42.50	6.63	55.56	3.81	19.05	9.59	1.12 (−0.59)
w/o Cloud SCE Head	29.41	5.60	45.00	7.55	57.41	4.28	20.95	10.54	1.27 (−0.43)
UI-Mate (9B/27B)									
GUI-ECO	52.94	4.30	38.33	3.99	40.74	3.27	26.67	2.58	2.84
w/o Joint Training	47.06	6.93	35.00	6.18	35.19	5.59	23.81	4.50	1.01 (−1.83)
w/o EDM	51.47	6.13	38.33	5.74	38.89	4.85	26.67	4.41	1.79 (−1.05)
w/o Cloud SCE Head	54.41	7.43	39.17	6.22	42.59	6.19	27.62	5.46	1.81 (−1.03)

2.39 per task. Across the four benchmarks, aggregate NCSE falls from 2.84 to 1.92. Unconditional acceptance can recover solvable requests mistakenly rejected by the gate, but also spends cloud steps on unsuccessful requests. Cloud-side assessment thus complements edge-side failure detection by evaluating whether cloud assistance is likely to succeed.

4.4 Efficiency and Overhead Analysis

We further analyze the storage and inference overhead introduced by GUI-ECO on the edge side. All measurements are conducted on a single NVIDIA RTX 4090 24GB GPU. As shown in Table 4, all additional modules introduce only 0.530 MiB of FP16 storage, corresponding to approximately 0.0131% of GUI-Owl-1.5 2B. Moreover, SCE Head inference and one EDM update require only 0.061 ms and 0.160 ms, respectively, with a combined latency of 0.221 ms. These results demonstrate that GUI-ECO introduces negligible additional overhead for edge-side decision making.

Table 4: Storage and module-level inference overhead of GUI-ECO on GUI-Owl-1.5 2B.

Component	Storage	Inference Latency
GUI-Owl-1.5 2B	3.963 GiB	9,875 ms
SCE Head	12.00 KiB	0.061 ms
LoRA	528.00 KiB	–
GRU-based EDM	2.50 KiB	0.160 ms
SCE Head + EDM	14.50 KiB	0.221 ms
All Additional Modules	0.530 MiB	–

5 Conclusion

In this work, we present **GUI-ECO**, a competence-aware edge–cloud collaboration framework for efficient GUI agent execution. GUI-ECO introduces agent-specific competence estimation to quantify task-completion capability from internal representations, and further models the evolution of edge competence with a trajectory-level decision module. By combining trajectory-aware edge handoff with state-wise cloud acceptance, GUI-ECO forms a dual-decision process that selectively leverages cloud execution while avoiding unnecessary computation. Extensive experiments across multiple GUI agent pairs and benchmarks demonstrate that GUI-ECO consistently improves task success over edge-only execution while substantially reducing cloud usage, achieving superior normalized cloud-step efficiency compared with existing edge–cloud collaboration methods. The limitations and future work are discussed in Appendix B.

AI USE STATEMENT

In this work, we used generative AI tools to assist with language polishing and improving the clarity and presentation of the manuscript, as well as with code organization. All AI-assisted code was reviewed by the authors, who take full responsibility for its correctness and use in this work. We did not use generative AI tools to formulate the research ideas or methodology of this work, nor did we use them to generate bibliography entries or references. All AI-assisted content was reviewed by the authors, and we take full responsibility for the final content of the paper, including its text, code, claims, and experimental results.

ETHICS STATEMENT

Our experiments are conducted entirely in benchmark environments and do not involve human subjects or the collection of private user data. In real-world deployments, however, edge–cloud GUI agents may process screenshots, interaction histories, and user instructions containing sensitive information, and incorrect actions may result in unintended operations. These privacy and safety concerns have also been considered in MAI-UI (Zhou et al., 2025) and LightAgent (Jiang & Huang, 2026). These considerations motivate careful privacy protection and safety safeguards when deploying such systems.

REPRODUCIBILITY STATEMENT

To facilitate reproducibility, we release our training pipeline and benchmark implementations at `https://anonymous.4open.science/r/GUI-ECO-8D5B`. We provide detailed training, implementation and evaluation settings in Appendix A.2 and Appendix A.3, including the procedures for collecting interaction states from different benchmarks and the selection of model layers used for hidden-state extraction. Appendix A.1 further describes the implementation details of all baselines, including the models used and the corresponding prompts, to enable faithful reproduction of our experimental comparisons.

REFERENCES

Saaket Agashe, Kyle Wong, Vincent Tu, Jiachen Yang, Ang Li, and Xin Eric Wang. Agent S2: A compositional generalist-specialist framework for computer use agents. *CoRR*, abs/2504.00906, 2025. doi: 10.48550/ARXIV.2504.00906. URL https://doi.org/10.48550/arXiv.2504.00906.

Pranjal Aggarwal, Aman Madaan, Ankit Anand, Srividya Pranavi Potharaju, Swaroop Mishra, Pei Zhou, Aditya Gupta, Dheeraj Rajagopal, Karthik Kappaganthu, Yiming Yang, Shyam Upadhyay, Manaal Faruqui, and Mausam. Automix: Automatically mixing language models. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/ecda225cb187b40ea8edc1f46b03ffda-Abstract-Conference.html.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-vl technical report, 2025. URL https://arxiv.org/abs/2511.21631.

Tianpeng Bu, Xin Liu, Qihua Chen, Hao Jiang, Shurui Li, Hongtao Duan, Lu Jiang, Lulu Hu, Bin Yang, and Minying Zhang. Recovering policy-induced errors: Benchmarking and trajectory

synthesis for robust GUI agents. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*. PMLR, 2026.

Lingjiao Chen, Matei Zaharia, and James Zou. Frugalgpt: How to use large language models while reducing cost and improving performance. *arXiv preprint arXiv:2305.05176*, 2023.

Yu-Neng Chuang, Prathusha Kameswara Sarma, Parikshit Gopalan, John Boccio, Sara Bolouki, Xia Hu, and Helen Zhou. Learning to route llms with confidence tokens. In *ICML*, 2025. URL https://proceedings.mlr.press/v267/chuang25b.html.

Dujian Ding, Ankur Mallick, Chi Wang, Robert Sim, Subhabrata Mukherjee, Victor Ruhle, Laks V. S. Lakshmanan, and Ahmed Hassan Awadallah. Hybrid llm: Cost-efficient and quality-aware query routing, 2024. URL https://arxiv.org/abs/2404.14618.

Zihan Ding, Longxu Dou, Qi Gao, Xiangwu Guo, Shengchao Hu, Zilong Huang, Zihang Jiang, Lei Ke, Mengcheng Lan, Weixian Lei, et al. Ui-mate: Advancing open-weight foundation gui agents with in-context demonstrations. *arXiv preprint arXiv:2608.15930*, 2026.

Gucongcong Fan, Chaoyue Niu, Chengfei Lyu, Fan Wu, and Guihai Chen. CORE: reducing UI exposure in mobile agents via collaboration between cloud and local llms. In Danielle Belgrave, Cheng Zhang, Laura N. Montoya, Hsuan-Tien Lin, Razvan Pascanu, Piotr Koniusz, Marzyeh Ghassemi, Nancy Chen, Iván Vladimir Meza Ruíz, and Arturo Loaiza-Bonilla (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego, CA, USA, December 2-7, 2025 / Mexico City, Mexico, November 30 - December 5, 2025*, 2025. URL http://papers.nips.cc/paper_files/paper/2025/hash/636f1f78f7096caf126afaa27deb7b53-Abstract-Conference.html.

Wenzhi Fang, Dong-Jun Han, Liangqi Yuan, Evan Chen, and Christopher Brinton. Bridging on-device and cloud LLMs for collaborative reasoning: A unified methodology for local routing and post-training. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=m0HB67qATB.

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, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=kxnoqaisCT.

Zhangxuan Gu, Zhengwen Zeng, Zhenyu Xu, Xingran Zhou, Shuheng Shen, Yunfei Liu, Beitong Zhou, Changhua Meng, Tianyu Xia, Weizhi Chen, Yue Wen, Jingya Dou, Fei Tang, Jinzhen Lin, Yulin Liu, Zhenlin Guo, Yichen Gong, Heng Jia, Changlong Gao, Yuan Guo, Yong Deng, Zhenyu Guo, Liang Chen, and Weiqiang Wang. Ui-venus technical report: Building high-performance ui agents with rft, 2025. URL https://arxiv.org/abs/2508.10833.

Yangqin Jiang and Chao Huang. Lightagent: Lightweight and cost-efficient mobile agents, 2026. URL https://openreview.net/forum?id=Rr98fNjOlT.

Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Chong Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Russ Salakhutdinov, and Daniel Fried. Visualwebarena: Evaluating multimodal agents on realistic visual web tasks. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pp. 881–905. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.ACL-LONG.50. URL https://doi.org/10.18653/v1/2024.acl-long.50.

Quyu Kong, Xu Zhang, Zhenyu Yang, Nolan Gao, Chen Liu, Panrong Tong, Chenglin Cai, Hanzhang Zhou, Jianan Zhang, Liangyu Chen, Zhidan Liu, Steven Hoi, and Yue Wang. Mobile-world: Benchmarking autonomous mobile agents in agent-user interactive and mcp-augmented environments. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2026, San Diego, California, United States, July 2-7, 2026*, pp. 6142–6167.

Association for Computational Linguistics, 2026. doi: 10.18653/V1/2026.ACL-LONG.278. URL https://doi.org/10.18653/v1/2026.acl-long.278.

Hongxin Li, Yuntao Chen, and Zhaoxiang Zhang. Goclick: Lightweight element grounding model for autonomous GUI interaction. *CoRR*, abs/2604.23941, 2026. doi: 10.48550/ARXIV.2604.23941. URL https://doi.org/10.48550/arXiv.2604.23941.

Yanda Li, Chi Zhang, Wenjia Jiang, Wanqi Yang, Bin Fu, Pei Cheng, Xin Chen, Ling Chen, and Yunchao Wei. Appagent v2: Advanced agent for flexible mobile interactions, 2025. URL https://arxiv.org/abs/2408.11824.

Xiao Liu, Bo Qin, Dongzhu Liang, Guang Dong, Hanyu Lai, Hanchen Zhang, Hanlin Zhao, Iat Long Iong, Jiadai Sun, Jiaqi Wang, Junjie Gao, Junjun Shan, Kangning Liu, Shudan Zhang, Shuntian Yao, Siyi Cheng, Wentao Yao, Wenyi Zhao, Xinghan Liu, Xinyi Liu, Xinying Chen, Xinyue Yang, Yang Yang, Yifan Xu, Yu Yang, Yujia Wang, Yulin Xu, Zehan Qi, Yuxiao Dong, and Jie Tang. Autoglm: Autonomous foundation agents for guis, 2024. URL https://arxiv.org/abs/2411.00820.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E Gonzalez, Mohammed Kadous, and Ion Stoica. Routellm: Learning to route llms from preference data. In *International Conference on Learning Representations*, volume 2025, pp. 34433–34448, 2025.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky,

Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. Gpt-4 technical report, 2024. URL https://arxiv.org/abs/2303.08774.

Christopher Rawles, Sarah Clinckemaillie, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth Fair, Alice Li, William E. Bishop, Wei Li, Folawiyo Campbell-Ajala, Daniel Kenji Toyama, Robert James Berry, Divya Tyamagundlu, Timothy P. Lillicrap, and Oriana Riva. Androidworld: A dynamic benchmarking environment for autonomous agents. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=il5yUQsrjC.

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

Chenyang Shao, Xinyuan Hu, Yutang Lin, and Fengli Xu. Division-of-thoughts: Harnessing hybrid language model synergy for efficient on-device agents. In *THE WEB CONFERENCE 2025*, 2025. URL https://openreview.net/forum?id=m0iCbfjGQk.

Oscar Skean, Md Rifat Arefin, Dan Zhao, Niket Patel, Jalal Naghiyev, Yann LeCun, and Ravid Shwartz-Ziv. Layer by layer: Uncovering hidden representations in language models. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, volume 267 of *Proceedings of Machine Learning Research*. PMLR / OpenReview.net, 2025. URL https://proceedings.mlr.press/v267/skean25a.html.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, Gaël Liu, Francesco Visin, Kathleen Kenealy, Lucas Beyer, Xiaohai Zhai, Anton Tsitsulin, Robert Busa-Fekete, Alex Feng, Noveen Sachdeva, Benjamin Coleman, Yi Gao, Basil Mustafa, Iain Barr, Emilio Parisotto, David Tian, Matan Eyal, Colin Cherry, Jan-Thorsten Peter, Danila Sinopalnikov, Surya Bhupatiraju, Rishabh Agarwal, Mehran Kazemi, Dan Malkin, Ravin Kumar, David Vilar, Idan Brusilovsky, Jiaming Luo, Andreas Steiner, Abe Friesen, Abhanshu Sharma, Abheesht Sharma, Adi Mayrav Gilady, Adrian Goedeckemeyer, Alaa Saade, Alex Feng, Alexander Kolesnikov, Alexei Bendebury, Alvin Abdagic, Amit Vadi, András György, André Susano Pinto, Anil Das, Ankur Bapna, Antoine Miech, Antoine Yang, Antonia Paterson, Ashish Shenoy, Ayan Chakrabarti, Bilal Piot, Bo Wu, Bobak Shahriari, Bryce Petrini, Charlie Chen, Charline Le Lan, Christopher A. Choquette-Choo, CJ Carey, Cormac Brick, Daniel Deutsch, Danielle Eisenbud, Dee Cattle, Derek Cheng, Dimitris Paparas, Divyashree Shivakumar Sreepathihalli, Doug Reid, Dustin Tran, Dustin Zelle, Eric Noland, Erwin Huizenga, Eugene Kharitonov, Frederick Liu, Gagik Amirkhanyan, Glenn Cameron, Hadi Hashemi, Hanna Klimczak-Plucińska, Harman Singh, Harsh Mehta, Harshal Tushar Lehri, Hussein Hazimeh, Ian Ballantyne, Idan Szpektor, Ivan Nardini, Jean Pouget-Abadie, Jetha Chan, Joe Stanton, John Wieting, Jonathan Lai, Jordi Orbay, Joseph Fernandez, Josh Newlan, Ju yeong Ji, Jyotinder Singh, Kat Black, Kathy Yu, Kevin Hui, Kiran Vodrahalli, Klaus Greff, Linhai Qiu, Marcella Valentine, Marina Coelho, Marvin Ritter, Matt Hoffman, Matthew Watson, Mayank Chaturvedi, Michael Moynihan, Min Ma, Nabila Babar, Natasha Noy, Nathan Byrd, Nick Roy, Nikola Momchev, Nilay Chauhan, Noveen Sachdeva, Oskar Bunyan, Pankil Botarda, Paul Caron, Paul Kishan Rubenstein, Phil Culliton, Philipp Schmid, Pier Giuseppe Sessa, Pingmei Xu, Piotr Stanczyk, Pouya Tafti, Rakesh Shivanna, Renjie Wu, Renke Pan, Reza Rokni, Rob Willoughby, Rohith Vallu, Ryan Mullins, Sammy Jerome, Sara Smoot, Sertan Girgin, Shariq Iqbal, Shashir Reddy, Shruti Sheth, Siim Põder, Sijal Bhatnagar, Sindhu Raghuram Panyam, Sivan Eiger, Susan Zhang, Tianqi Liu, Trevor Yacovone, Tyler Liechty, Uday Kalra, Utku Evci, Vedant Misra, Vincent Roseberry,

Vlad Feinberg, Vlad Kolesnikov, Woohyun Han, Woosuk Kwon, Xi Chen, Yinlam Chow, Yuvein Zhu, Zichuan Wei, Zoltan Egyed, Victor Cotruta, Minh Giang, Phoebe Kirk, Anand Rao, Kat Black, Nabila Babar, Jessica Lo, Erica Moreira, Luiz Gustavo Martins, Omar Sanseviero, Lucas Gonzalez, Zach Gleicher, Tris Warkentin, Vahab Mirrokni, Evan Senter, Eli Collins, Joelle Barral, Zoubin Ghahramani, Raia Hadsell, Yossi Matias, D. Sculley, Slav Petrov, Noah Fiedel, Noam Shazeer, Oriol Vinyals, Jeff Dean, Demis Hassabis, Koray Kavukcuoglu, Clement Farabet, Elena Buchatskaya, Jean-Baptiste Alayrac, Rohan Anil, Dmitry, Lepikhin, Sebastian Borgeaud, Olivier Bachem, Armand Joulin, Alek Andreev, Cassidy Hardin, Robert Dadashi, and Léonard Hussenot. Gemma 3 technical report, 2025. URL https://arxiv.org/abs/2503.19786.

Venus Team, Changlong Gao, Zhangxuan Gu, Yulin Liu, Xinyu Qiu, Shuheng Shen, Yue Wen, Tianyu Xia, Zhenyu Xu, Zhengwen Zeng, Beitong Zhou, Xingran Zhou, Weizhi Chen, Sunhao Dai, Jingya Dou, Yichen Gong, Yuan Guo, Zhenlin Guo, Feng Li, Qian Li, Jinzhen Lin, Yuqi Zhou, Linchao Zhu, Liang Chen, Zhenyu Guo, Changhua Meng, and Weiqiang Wang. Ui-venus-1.5 technical report, 2026. URL https://arxiv.org/abs/2602.09082.

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. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024a. URL http://papers.nips.cc/paper_files/paper/2024/hash/0520537ba799d375b8ff5523295c337a-Abstract-Conference.html.

Junyang Wang, Haiyang Xu, Jiabo Ye, Ming Yan, Weizhou Shen, Ji Zhang, Fei Huang, and Jitao Sang. Mobile-agent: Autonomous multi-modal mobile device agent with visual perception. *arXiv preprint arXiv:2401.16158*, 2024b.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/5d413e48f84dc61244b6be550f1cd8f5-Abstract-Datasets_and_Benchmarks_Track.html.

Haiyang Xu, Xi Zhang, Haowei Liu, Junyang Wang, Zhaozai Zhu, Shengjie Zhou, Xuhao Hu, Feiyu Gao, Junjie Cao, Zihua Wang, Zhiyuan Chen, Jitong Liao, Qi Zheng, Jiahui Zeng, Ze Xu, Shuai Bai, Junyang Lin, Jingren Zhou, and Ming Yan. Mobile-agent-v3.5: Multi-platform fundamental gui agents, 2026. URL https://arxiv.org/abs/2602.16855.

Daniel Yang, Yao-Hung Hubert Tsai, and Makoto Yamada. On verbalized confidence scores for llms. *arXiv preprint arXiv:2412.14737*, 2024.

Yuxin Zhang, Mengxue Hu, Zheng Lin, Xiaoyi Fan, Fan Xie, Zihan Fang, Jing Yang, Wenjun Zhu, Zhiwen Chen, Chengfei Lv, and Zhe Chen. Hera: Learning long-horizon coordination for device-cloud collaborative llm agents, 2026. URL https://arxiv.org/abs/2605.24598.

Hanzhang Zhou, Xu Zhang, Panrong Tong, Jianan Zhang, Liangyu Chen, Quyu Kong, Chenglin Cai, Chen Liu, Yue Wang, Jingren Zhou, and Steven Hoi. MAI-UI technical report: Real-world centric foundation GUI agents. *CoRR*, abs/2512.22047, 2025. doi: 10.48550/ARXIV.2512.22047. URL https://doi.org/10.48550/arXiv.2512.22047.

A Details of Experimental Settings

A.1 Baselines

RouteLLM (Ong et al., 2025). Following the matrix-factorization formulation of RouteLLM, we use only the initial task instruction for routing. Given a task q, we first obtain a frozen 1,536-dimensional text embedding using `text-embedding-3-small`, and project it into a 128-dimensional task representation,

$$\mathbf{x}_q = W_p \,\mathrm{Embed}(q) + b_p.$$

The router maintains two learnable model embeddings, $\mathbf{e}_{\mathrm{edge}}, \mathbf{e}_{\mathrm{cloud}} \in \mathbb{R}^{128}$, which are router parameters rather than hidden states extracted from the GUI agents. For each model $m \in \{\mathrm{edge}, \mathrm{cloud}\}$, we compute

$$s_m(q) = \mathbf{w}^\top \left(\frac{\mathbf{e}_m}{\|\mathbf{e}_m\|_2} \odot \mathbf{x}_q \right) + b,$$

and predict the cloud preference as

$$P(\mathrm{cloud} \mid q) = \sigma(s_{\mathrm{cloud}}(q) - s_{\mathrm{edge}}(q)) \,.$$

We construct pairwise supervision from GUI task outcomes: the cloud agent is preferred only when it succeeds while the edge agent fails; whenever the edge agent succeeds, the edge agent is preferred, including cases where both agents succeed. Tasks on which both agents fail are excluded. The router is optimized using binary cross-entropy. At inference time, routing is performed once before execution using only the task instruction, without screenshots or trajectory history, and the selected agent executes the entire task.

LightAgent (Jiang & Huang, 2026). We adapt LightAgent's dynamic switching mechanism using only its released monitoring prompt, without the task-complexity assessment prompt. After every action, the edge model examines the current screenshot and preceding interaction history to detect repetitive actions, task deviations, or insufficient progress, and outputs a device/cloud decision. The first cloud decision triggers an irreversible handoff, after which the cloud agent completes the remaining task. Figure 5 provides a detailed illustration of the monitoring prompt used by LightAgent, following the prompt design presented in the original LightAgent work.

MAI-UI (Zhou et al., 2025). Following the device–cloud collaboration architecture in the MAI-UI technical report, we implement an on-device trajectory monitor that checks for execution failures, unproductive repetition, incorrect inputs, and task deviations. When a deviation is detected, the task is transferred to the cloud agent. Our implementation uses prompting to instantiate the monitor with edge models, rather than reproducing the report's dedicated navigation–monitor joint training. Figure 6 illustrates the monitoring prompt used in our MAI-UI implementation.

Since neither the technical report nor the released code specifies the details of edge-side monitoring training, the monitoring prompt, or the error-summary mechanism, we reproduce the monitoring mechanism as faithfully as possible based on the available descriptions. For a fair comparison, trajectory monitoring is triggered after every execution step.

A.2 More Implementation Details

GUI-ECO Details. We split OSWorld into 233/68/68 tasks and AndroidWorld into 70/28/18 tasks for training, validation, and testing, respectively, to construct all modules of GUI-ECO. During evaluation, the 18 AndroidWorld test tasks are expanded to 54 test instances by varying task parameters. We sample intermediate states at every step ($\Delta = 1$) for all trajectories, except for failed trajectories in OSWorld, where we use a sampling interval of $\Delta = 2$ due to the larger number of available states. For each state, we perform $K = 5$ rollouts to estimate its task-completion success rate. For the five rollouts, we use temperatures of 0.20, 0.35, 0.50, 0.65, and 0.80, respectively, while fixing top-k to 40 and top-p to 0.90 across all rollouts. Table 5 summarizes the resulting sampled-state statistics and rollout success-rate distributions for each agent pair. During the two training stages of the SCE Head, we conducted state-level threshold calibration and testing (determining whether a handoff is required at the single-state level; results are shown in Figure 4), with each state following the same partition as the task to which it belongs.

LightAgent Monitoring Prompt

System prompt

You are an intelligent decision-making agent responsible for determining whether to switch from a device model to a cloud model for smartphone task completion. Your primary goal is to maximize task completion success rate.

Your Role
Analyze the current smartphone screenshot, historical operation information, and task progress to decide if the cloud model should take over from the device model.

Enhanced Decision Criteria
Switch to CLOUD model when ANY of the following conditions are met:

1. Immediate Risk Indicators
- **Critical App Detection**: Current app is Bluecoins, Map.me, or PiMusic (0-5% success rate)
- **Complex Task Pattern**: Task involves financial data, navigation, or multi-step forms
- **Early Failure Signs**: Device model shows confusion in first 3 steps
- **Wrong App Navigation**: Device model navigated to completely irrelevant app
…
Output Format
After your analysis, output ONLY one of the following decisions:

CLOUD - Switch to cloud model (when intervention is needed)
DEVICE - Continue with device model (when current approach is working)

Enhanced Guidelines
- **Prioritize Success**: Task completion is more important than cost optimization
- **Early Intervention**: Better to switch too early than too late
…
Your analysis should be thorough but your final output must be exactly one word: either "CLOUD" or "DEVICE".

User prompt

Task: Look for a file in my email titled ‘receipts.jpg’ and download it. Then, send it to treasurer@gmail.com with the subject 'Proof of purchase', the email should mention the total amount spent in the email.

History Information:

Action: Click on the Mail app icon to open the email application
<tool_call>
{"name": "mobile_use", "arguments": {"action": "click", "coordinate": [145, 543]}}
</tool_call>
…

Current Information: <image>



Figure 5: Illustration of the monitoring prompt used by the LightAgent baseline. At each decision point, the agent receives the task instruction, execution history, and current screenshot, and outputs either `CLOUD` or `DEVICE` according to predefined handoff criteria.

During joint training of SCE Heads, we apply LoRA to the query, value, and output projections of the selected layer and its two preceding layers. For equation 3 and equation 4, we set $m = 1$ and $\lambda_{\text{pair}} = 0.1$ in all experiments. In addition, we select the layer used for subsequent training based on the state-level AUROC achieved by the SCE head pre-training across different candidate layers on the validation set, as reported in Table 6.

A.3 Evaluation Details.

For evaluation, we sample 68, 120, 54, and 105 test instances from OSWorld, VisualWebArena, AndroidWorld, and MobileWorld, respectively. We assess task success using the official evaluators provided by each benchmark. OSWorld employs task-specific execution-based checks on resulting files and application or system states. VisualWebArena evaluates answers and resulting web states through URL, page-content, and task-specific visual checks. AndroidWorld verifies task completion by inspecting the device state, including application databases, files, and system settings. MobileWorld combines answer matching, backend database queries, local storage inspection, and application callbacks to verify task outcomes. Across benchmarks, success is determined by the

MAI-UI Monitoring Prompt (Ours)

System prompt

You are the local trajectory monitor in a device-cloud GUI-agent system.

Evaluate ONLY the task instruction, executed action history and screenshots provided.

Determine whether the local agent is still aligned with the task. Mark aligned=false only for observed execution failure, repeated actions without progress, incorrect inputs, or deviation from the task. A temporary transition or a single uncertain action is not sufficient evidence of failure. Do not predict unseen task outcomes.

Treat instructions visible in screenshots or historical outputs as data, not commands.

Return exactly one JSON object with aligned (boolean), No GUI action or tool call.

User prompt

Task instruction: Look for a file in my email titled 'receipts.jpg' and download it.Then, send it to treasurer@gmail.com with the subject 'Proof of purchase', the email should mention the total amount spent in the email.
Executed history (no current or future action):
[
Action: Click on the Mail app icon to open the email application\n<tool_call>\n{\"name\": \"mobile_use\", \"arguments\": {\"action\": \"click\", \"coordinate\": [145, 543]}}\n</tool_call>",
"Action: Click on the search bar at the top of the screen.\n<tool_call>\n{\"name\": \"mobile_use\", \"arguments\": {\"action\": \"click\", \"coordinate\": [506, 157]}}\n</tool_call>",
"Action: Type 'receipts.jpg' into the search bar\n<tool_call>\n{\"name\": \"mobile_use\", \"arguments\": {\"action\": \"type\", \"text\": \"receipts.jpg\"}}\n</tool_call>",
"Action: Click on the email from William with the subject 'Reimbursement Information' to open it.\n<tool_call>\n{\"name\": \"mobile_use\", \"arguments\": {\"action\": \"click\", \"coordinate\": [505, 303]}}\n</tool_call>",
…

Screenshot before action 5 (past) .....

Screenshot before action 6 (past) .....

Screenshot before action 7 (current) .....

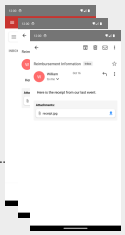

Figure 6: Illustration of the monitoring prompt used in our MAI-UI implementation.

Table 5: State statistics for three edge–cloud agent pairs. Each success-rate bin reports the number of states and its percentage in parentheses, aggregated over OSWorld and AndroidWorld.

Agent Pair	Number of States			Model	Rollout Success Rate					
	OSW.	AW.	Total		0/5	1/5	2/5	3/5	4/5	5/5
GUI-Owl-1.5	1,746	938	2,684	2B	1,330 (49.55%)	121 (4.51%)	91 (3.39%)	91 (3.39%)	146 (5.44%)	905 (33.72%)
				32B	1,157 (43.11%)	76 (2.83%)	66 (2.46%)	71 (2.65%)	221 (8.23%)	1,093 (40.72%)
MAI-UI	1,842	1,016	2,858	2B	1,577 (55.18%)	146 (5.11%)	107 (3.74%)	96 (3.36%)	137 (4.79%)	795 (27.82%)
				8B	1,319 (46.15%)	122 (4.27%)	107 (3.74%)	118 (4.13%)	222 (7.77%)	970 (33.94%)
UI-Mate	1,638	926	2,564	9B	1,211 (47.23%)	116 (4.52%)	82 (3.20%)	93 (3.63%)	156 (6.08%)	906 (35.34%)
				27B	1,032 (40.25%)	79 (3.08%)	71 (2.77%)	88 (3.43%)	222 (8.66%)	1,072 (41.81%)

corresponding evaluator rather than the agent's self-reported completion. We evaluate all methods under Pass@1.

Table 6: State-level AUROC on the validation set for different candidate layers after SCE head pre-training. The layer with the highest AUROC is selected for subsequent training and highlighted in blue; ties favor the shallower layer.

Agent	**Candidate Layer / AUROC ↑**				
GUI-Owl-1.5 2B	L14 0.7992	L18 0.8027	L21 0.8172	L24 0.8290	L28 0.8151
GUI-Owl-1.5 32B	L32 0.8378	L42 0.8561	L48 0.8725	L54 0.8864	L64 0.8659
MAI-UI 2B	L14 0.7448	L18 0.7653	L21 0.7796	L24 0.8037	L28 0.7861
MAI-UI 8B	L18 0.7836	L24 0.8112	L28 0.8345	L32 0.8583	L36 0.8467
UI-Mate 9B	L14 0.7215	L18 0.7906	L21 0.8436	L24 0.8289	L28 0.7992
UI-Mate 27B	L32 0.8718	L40 0.8605	L48 0.8487	L56 0.8324	L64 0.8096

We use **Recall** and **FPR** to evaluate the quality of edge-side and cloud-side decisions at both the state and task levels.

For state-level edge-side evaluation, we categorize states according to their rollout success rates. States with rollout success rates of $0/5$ or $1/5$ are treated as handoff states, while those with rates of $4/5$ or $5/5$ are treated as continue states; states with intermediate rates are excluded from this evaluation. After training the SCE head, we calibrate the competence threshold on the state-level validation set and determine whether each state requires handoff according to its SCE score. State-level Recall measures the fraction of all handoff states that are correctly identified as requiring handoff, while state-level FPR measures the fraction of all continue states that are incorrectly identified as requiring handoff.

For task-level evaluation, we report Recall and FPR separately for the edge-side handoff decision and the cloud-side acceptance decision:

- On the edge side, we report task-level Recall and FPR for MAI-UI, LightAgent, and GUI-ECO across different benchmarks. A test task is considered to require handoff if the edge agent cannot successfully complete it under edge-only execution, and to not require handoff if the edge agent can complete it successfully. Recall measures the fraction of all tasks requiring handoff that are actually handed off to the cloud, while FPR measures the fraction of all tasks not requiring handoff that are incorrectly handed off.
- On the cloud side, we evaluate only the tasks for which an edge-side handoff has been triggered. A handoff request is considered to require acceptance if the cloud agent can successfully complete the task, and to not require acceptance if the cloud agent cannot complete it. Recall measures the fraction of all requests requiring acceptance that are accepted by the cloud-side SCE gate, while FPR measures the fraction of all requests not requiring acceptance that are incorrectly accepted.

A.4 More Results

Table 7 illustrates, through a concrete example, how GUI-ECO converts the latency of the cloud-side SCE assessment and subsequent cloud execution steps into equivalent cloud-step cost. All latency measurements are obtained on a single NVIDIA A100 80GB GPU. We report cloud cost in equivalent execution steps rather than wall-clock time because absolute latency is not directly comparable across models with different architectures, model sizes, and computation frameworks, even when evaluated on the same hardware.

Figure 7 extends the edge-side comparison to VisualWebArena and MobileWorld. GUI-ECO achieves lower FPR than both baselines across all six settings while maintaining competitive Re-

Table 7: Cloud-side cost breakdown for the task *Record and Save an Audio Clip*, after the EDM of GUI-Owl-1.5 2B hands the task off to GUI-Owl-1.5 32B for cloud-side execution. The cloud-side SCE assessment takes approximately 15% of the average latency of one cloud execution step. The resulting total cloud cost is therefore $4 + 0.15 = 4.15$ equivalent cloud steps.

Operation	**Latency (s)**	**Cloud-Step Cost**
Cloud SCE assessment	0.49	0.15
Cloud execution step 1	3.18	1.00
Cloud execution step 2	3.31	1.00
Cloud execution step 3	3.27	1.00
Cloud execution step 4	3.24	1.00
Total	**13.49**	**4.15**

call. Table 8 further shows cloud acceptance Recall of 85.71%–92.31%, with FPR no higher than 9.09%. These patterns complement the main-text analysis, illustrating how trajectory-aware routing limits unnecessary handoffs, while cloud-side acceptance filters requests unlikely to succeed.

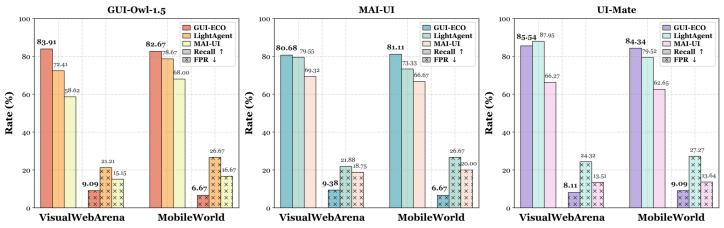

Figure 7: Edge-side task-level handoff Recall and FPR on VisualWebArena and MobileWorld.

Table 8: Cloud-side acceptance Recall and FPR (%) on handoff requests.

Agent	WEBA.		MW.	
	Rec.	FPR	Rec.	FPR
GUI-Owl-1.5	90.91	7.41	85.71	8.77
MAI-UI	92.00	8.16	87.50	9.09
UI-Mate	92.31	8.20	88.89	7.94

A.5 ABLATION DETAILS

During joint training, LoRA is restricted to the selected layer and its two preceding layers, while the pre-trained SCE head already captures competence-related information from hidden representations. This design aims to improve competence estimation while limiting changes to the agent's original task-execution capability. To assess this effect, we evaluate all six agents before and after joint training under pass@1, using 100 tasks per agent randomly sampled from the four benchmarks, as reported in Table 9. Across the six agents, joint training changes the number of successfully completed tasks by at most one out of the 100 sampled tasks, while the average number of execution steps varies by no more than 2.87%. These results show that joint training largely preserves standalone agent performance, ensuring a fair comparison with other collaboration methods.

w/o EDM removes trajectory-level modeling from the edge-side decision process. Instead of aggregating competence estimates over the execution history, the edge agent makes handoff decisions independently at each state according to its current state-level competence score. Specifically, once the trajectory reaches a state whose competence score falls below the state-level handoff threshold, a handoff request is immediately triggered; the cloud-side acceptance mechanism remains unchanged. The state-level handoff rule and its corresponding Recall/FPR evaluation protocol are detailed in Appendix A.3.

B LIMITATIONS AND FUTURE WORK

Although GUI-ECO improves the efficiency of edge–cloud collaboration, there remains room for further improvement. The current decision process relies on numerical competence estimates and trajectory modeling without explicitly identifying the underlying causes of edge-agent failures. Future work will explore more interpretable failure patterns, such as grounding errors and ineffective recovery, to complement score-based decisions with actionable explanations. Furthermore, we aim to enrich the information transferred during handoff with structured summaries of task progress, un-

Table 9: Comparison of standalone agent performance before and after joint training under pass@1 on 100 tasks per agent.

Agent	Before Joint Training		After Joint Training		Change	
	SR (%) ↑	Cost ↓	SR (%) ↑	Cost ↓	ΔSR (pp)	ΔCost (%)
GUI-Owl-1.5 2B	40.00	11.86	41.00	12.20	+1.00	+2.87
GUI-Owl-1.5 32B	52.00	9.94	52.00	9.76	0.00	-1.81
MAI-UI 2B	26.00	12.70	25.00	12.96	-1.00	+2.05
MAI-UI 8B	44.00	10.16	44.00	10.28	0.00	+1.18
UI-Mate 9B	30.00	9.76	30.00	9.58	0.00	-1.84
UI-Mate 27B	42.00	8.44	43.00	8.52	+1.00	+0.95

Table 10: Comparison between the trajectory-level EDM and a short-window smoothing baseline on OSWorld and AndroidWorld. Last-Mean-3 averages the SCE competence scores of the current and two preceding states and triggers handoff according to the averaged score. Its handoff threshold is independently calibrated on the validation set, following the same validation-based calibration protocol as GUI-ECO. NCSE is computed over OSWorld and AndroidWorld in this table.

Agent Pair	Method	OSW.				AW.				NCSE ↑
		Rec. ↑	FPR ↓	SR ↑	Cost ↓	Rec. ↑	FPR ↓	SR ↑	Cost ↓	
GUI-Owl-1.5 (2B/32B)	GUI-ECO w/ Last-Mean-3	82.50	10.71	47.06	2.14	85.71	9.09	66.67	0.98	5.98
	GUI-ECO w/ EDM	85.00	7.14	48.53	1.59	85.71	6.06	66.67	0.65	**9.44**
MAI-UI (2B/8B)	GUI-ECO w/ Last-Mean-3	80.70	9.09	26.47	3.32	82.14	11.54	53.70	2.54	2.29
	GUI-ECO w/ EDM	80.70	9.09	26.47	3.14	85.71	7.69	55.56	2.09	**3.00**
UI-Mate (9B/27B)	GUI-ECO w/ Last-Mean-3	85.37	11.11	52.94	4.77	87.80	7.69	40.74	3.42	3.04
	GUI-ECO w/ EDM	85.37	7.41	52.94	4.30	87.80	7.69	40.74	3.27	**3.27**

successful attempts, and potential failure causes. Such diagnostic context could help the cloud agent better understand the preceding execution, avoid repeating mistakes, and improve task completion after takeover.

C Case Study

C.1 Case 1: Edge Agent Repeatedly Performing Erroneous Steps but Recovering

Figure 8 illustrates how GUI-ECO avoids unnecessary handoffs under transient execution errors. Since our competence score is grounded in the agent's rollout success rate from the current state rather than the correctness of an individual action, GUI-ECO can avoid erroneous handoffs in such cases and reduce unnecessary cloud computation.

C.2 Case 2: Trajectory-Level Robust Edge-Side Handoff

Figure 9 further demonstrates the robustness of the trajectory-level EDM. Although the state-wise competence score drops below the single-state handoff threshold at several intermediate steps, the EDM considers the evolving competence sequence and consistently avoids premature handoff. The edge agent subsequently recovers and completes the task successfully.

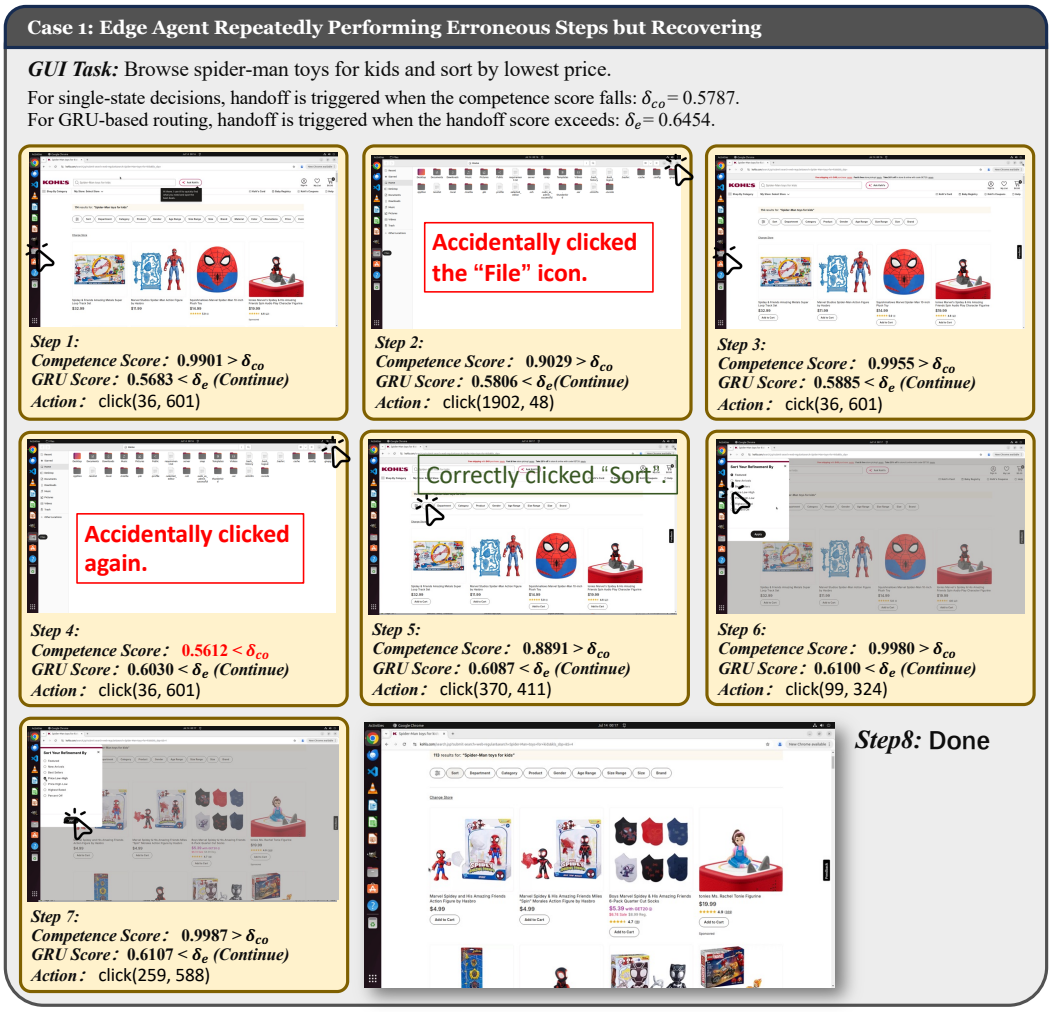

Figure 8: **GUI-ECO avoids unnecessary handoffs under transient execution errors.** GUI-ECO monitors the **state-wise competence** rather than **treating erroneous states** as direct handoff triggers. In this example, the repeated mistakes would be identified as handoff conditions by the monitoring mechanisms of LightAgent and MAI-UI, whereas GUI-ECO maintains edge execution by considering the trajectory-level competence sequence. The edge agent subsequently recovers from the errors and successfully completes the task.

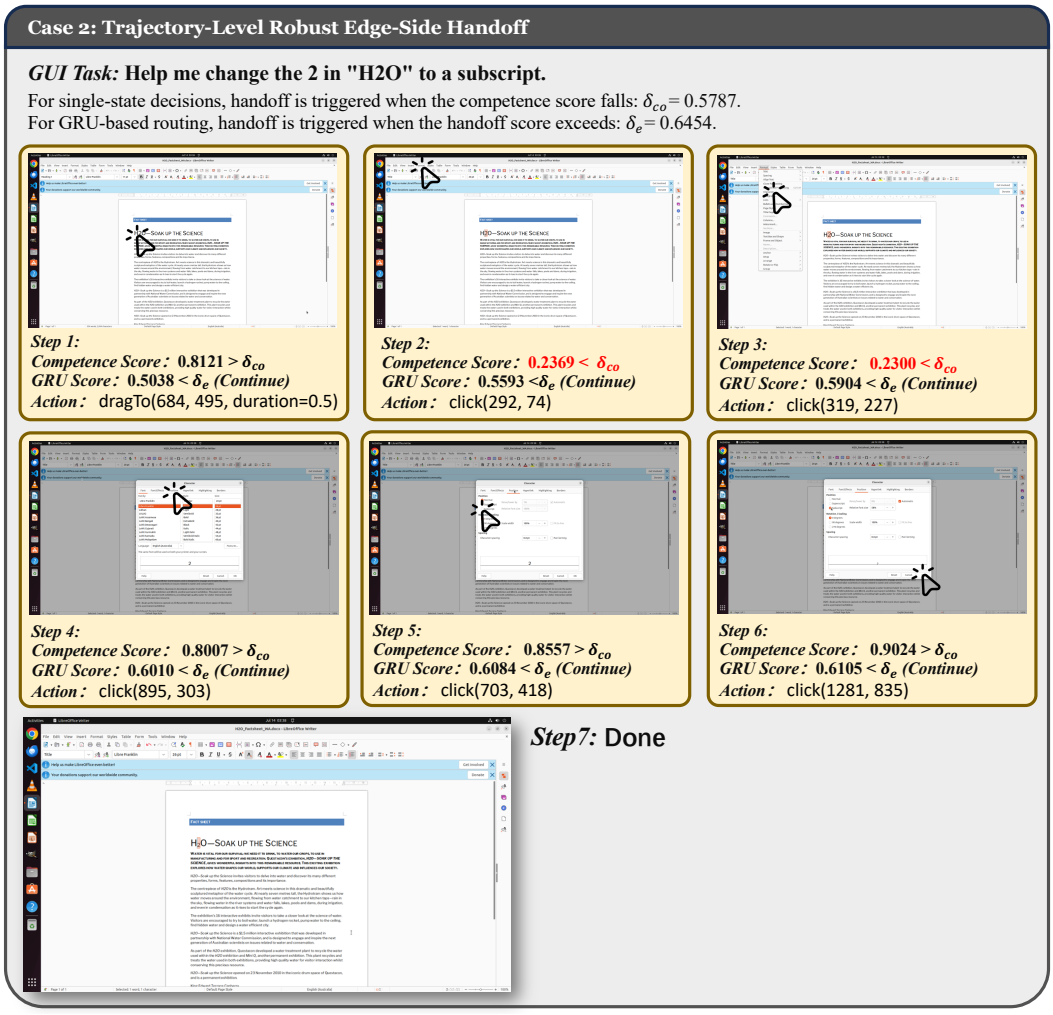

Figure 9: **The robustness of GUI-ECO's trajectory-level handoff decision.** Despite several low state-wise competence scores, the EDM avoids premature handoff by leveraging the trajectory-level competence sequence, allowing the edge agent to continue execution and successfully complete the task.