GUI-DROP: STATE TRANSITION-AWARE VISUAL TOKEN PRUNING FOR MULTIMODAL GUI AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision-language models enable GUI agents to interact with graphical interfaces by encoding high-resolution screenshots into long visual token sequences (e.g., 2,550 tokens for a 1080×2400 mobile screenshot). However, most visual tokens are found redundant or irrelevant to action generation, incurring high computational and latency overhead. While existing training-free pruning methods reduce visual tokens, they mostly focus on intra-frame token relevance and redundancy within individual static screenshots. They neglect the logical continuity across GUI state transitions and the complementary cues of UI elements. In particular, they treat localized action feedback identically to static elements, while fragmenting UI elements into disjointed patches that slice complete text and decouple icons from their accompanying labels. To address these limitations, we propose GUI-Drop, a training-free token pruning framework that integrates transition-aware token selection with local feature compensation. We assign higher retention quotas to dynamic regions identified by inter-frame patch similarity and guide token selection by fusing current instruction-to-vision attention with spatially aligned attention signals from prior action generation. To preserve fine-grained local UI evidence, we balance token selection with attention-weighted feature diversity and compensate discarded representations into spatially adjacent retained neighbors. Across diverse online and offline GUI benchmarks and multiple VLM backbones, GUI-Drop generally outperforms existing training-free pruning methods. On AndroidWorld with Qwen3-VL-8B, GUI-Drop outperforms the unpruned baseline at 40% visual token retention. With UI-Voyager, our approach achieves a 59.5% success rate at 20% retention, compared with 43.1% for the strongest baseline, while reducing LLM prefill FLOPs by 41.7% at 40% retention.

1 INTRODUCTION

Vision-language models (VLMs) enable GUI agents to interact with graphical interfaces through multi-turn cycles of perception, reasoning, and action (Cheng et al., 2024; Gou et al., 2025; Qin et al., 2025). Within each cycle, representative GUI agents process the single current screenshot alongside the task instruction and textual action history to predict the next action (Yan et al., 2025; Lin et al., 2025; Bai et al., 2025b). However, even this single screenshot generates thousands of visual tokens that dominate the sequence budget and inflate inference latency, yet most tokens are redundant for immediate action generation (Ouyang et al., 2026; Yang et al., 2025). Designed for general visual understanding, existing visual token pruning methods typically retain tokens based on attention-guided relevance (Chen et al., 2025; Zhang et al., 2025b; Xing et al., 2025), visual-semantic redundancy (Yang et al., 2025; Alvar et al., 2025; Dong et al., 2026; Zhang et al., 2025a), or spatial integrity (Huang et al., 2026; Sun et al., 2026), while recent embodied approaches leverage temporal action guidance (Liu et al., 2025). These approaches effectively accelerate inference and preserve task performance (Chen et al., 2025; Yang et al., 2025; Dong et al., 2026).

As illustrated in Figure 1, when applied to GUI tasks, these methods suffer substantial performance drops (Ouyang et al., 2026; Xu et al., 2026), as they overlook two characteristics fundamental to GUI interactions. *(1) Neglect of logical continuity across state transitions.* In sequential GUI interactions, the importance of visual tokens partly depends on how the interface evolves following the preceding action. General-purpose pruning methods, however, derive token importance primarily from the isolated visual input, neglecting how preceding actions induce state transitions across steps. Under

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.

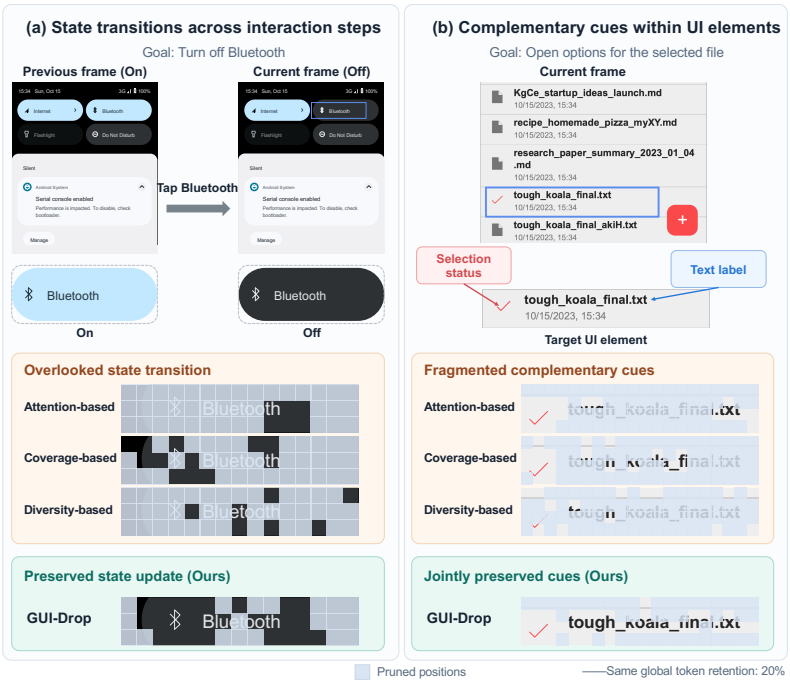


Figure 1: Comparison of visual token pruning behaviors in GUI environments at 20% retention. (a) Inter-action state transitions. Conventional baselines overlook the toggled Bluetooth update, while GUI-Drop preserves the active control. (b) Complementary UI cues. Baselines retain only fragmented tokens across the checkbox and text label, whereas GUI-Drop preserves the cohesive element. Light blue overlays denote pruned token positions.

such static visual evaluation, dominant interface backgrounds easily crowd out subtle interactive updates, such as toggled options (Figure 1a) or newly opened dialogs, leading to incorrect state perception. *(2) Disruption of local contextual cues within UI elements.* GUI elements typically span neighboring tokens that jointly convey discrete and complementary cues, such as text labels paired with visual icons. Existing methods often retain only fragmented subsets of tokens within the same element (Figure 1b). Consequently, essential text, icon, boundary, or contextual visual evidence are only partially preserved, impairing both spatial grounding and semantic understanding (Huang et al., 2026; Endo et al., 2025; Qian et al., 2024). While recent work in robotics (Liu et al., 2025) leverages past action attention, it assumes gradual visual changes across adjacent observations, failing to handle the discrete state transitions and structured layouts of GUI environments.

These limitations demonstrate the need for a token pruning paradigm tailored to GUI interactions, which must capture temporal dynamics across state transitions while preserving the semantic coherence of UI elements. Based on this insight, we introduce **GUI-Drop**, a training-free pruning framework for GUI agents that prevents critical transition oversights while mitigating UI representation fragmentation. To capture state transitions, we adaptively allocate retention budgets across dynamic and static regions identified by inter-frame patch similarity, evaluating token relevance using current instruction-to-vision attention and prior action decoding attention projected onto the current visual grid. Within the allocated quotas, we retain the top-ranking relevant tokens as anchors and fill the remaining slots with tokens ranked by relevance-weighted feature dissimilarity to the anchors, enhancing semantic and spatial coverage. To mitigate the loss of fine-grained details discarded during token pruning, we further compensate discarded token representations into their most similar spatially adjacent retained neighbors without expanding the visual sequence length.

We evaluate GUI-Drop across diverse online and offline GUI benchmarks, and further validate its effectiveness on multiple VLM backbones (Bai et al., 2025b;a). GUI-Drop generally outperforms existing training-free pruning methods, with particularly substantial gains under aggressive pruning budgets. Notably, when retaining only 20% of visual tokens, GUI-Drop reaches 59.5% on An-

droidWorld (Rawles et al., 2025) with UI-Voyager (Lin et al., 2026), substantially outperforming the leading competitor at 43.1%. Meanwhile, at 40% retention, our approach achieves 75.9% while reducing LLM prefill FLOPs by 41.7% relative to the unpruned model. Our contributions are summarized as follows:

- We identify two fundamental limitations of existing pruning methods based on static images for GUI agents, namely the neglect of logical continuity across state transitions and the loss of local contextual cues within UI elements.
- We propose GUI-Drop, a training-free token pruning framework tailored for GUI agents. It guides token selection across state transitions using inter-frame dynamics and historical action attention, while preserving fine-grained UI cues through feature diversity and spatial neighborhood aggregation.
- Our approach achieves competitive performance across visual budgets in both online and offline evaluation settings. With Qwen3-VL-8B, GUI-Drop attains 46.6% on AndroidWorld while retaining 40% of visual tokens, outperforming the unpruned baseline at 43.1%.

2 Related Work

Visual Token Pruning and Merging. Training-free visual token reduction accelerates multimodal inference by shortening visual sequences before or within transformer layers. Relevance-oriented methods prioritize task-relevant tokens via cross-modal attention (Chen et al., 2025; Xing et al., 2025; Zhang et al., 2025b), hidden state dynamics (Li et al., 2026a), zeroth-order sensitivity (Kim et al., 2026), or action decoding signals (Liu et al., 2025). Coverage techniques optimize feature diversity or multimodal alignment (Alvar et al., 2025; Dong et al., 2026), spatially aware methods preserve layout anchors or spatial integrity (Sun et al., 2026; Huang et al., 2026), and merging approaches cluster unselected features into retained tokens (Bolya et al., 2022; Dhouib et al., 2025; Yang et al., 2025).

Efficient Visual Processing for GUI Agents. Modern visual GUI agents operate directly on screenshots for perceptual grounding and action generation (Cheng et al., 2024; Gou et al., 2025; Yan et al., 2025; Qin et al., 2025), departing from earlier reliance on structured trees or auxiliary detectors (Shi et al., 2017; Wang et al., 2024). Repeatedly processing high-resolution screenshots throughout multi-step execution creates a severe computational bottleneck. To reduce visual overhead, FocusUI (Ouyang et al., 2026) preserves positional continuity during instruction-guided token selection, AQuaUI (Li et al., 2026c) allocates tokens via adaptive quadtree partitioning, and GUIPruner (Xu et al., 2026) combines structure-aware current-frame pruning with low-resolution history. Other works eliminate inter-frame redundancy across consecutive observations (Abaskohi et al., 2026), prune historical frames (Li et al., 2026b), or compress historical key-value states (Huang et al., 2025; Han et al., 2026). In contrast, GUI-Drop performs training-free pruning on the current observation, leveraging cross-step state transitions and local feature compensation to preserve the semantic integrity and grounding precision of UI elements.

3 Method

At each interaction step $t \in \{1, \ldots, T\}$, where T denotes the total number of steps required to complete a GUI task, the agent processes a task instruction I, an interaction history $\mathcal{H}_{t-1}$, and a high-resolution screenshot observation X_t to predict the next executable action a_t. The vision encoder transforms X_t into a sequence of N_t visual token embeddings $\mathbf{V}_t = (\mathbf{v}_1^t, \ldots, \mathbf{v}_{N_t}^t) \in \mathbb{R}^{N_t \times d}$, indexed by spatial positions on the visual grid $\mathcal{V}_t = \{1, \ldots, N_t\}$. These visual tokens are concatenated with the text tokens of I and $\mathcal{H}_{t-1}$ as input to the transformer decoder. Given a target visual retention ratio $R \in (0, 1]$, the visual token budget is set to $K_t = \mathrm{round}(RN_t)$, where $\mathrm{round}(\cdot)$ denotes the nearest-integer rounding function. The objective is to select an informative subset of K_t visual tokens from $\mathbf{V}_t$ to reduce sequence length and inference computation without impairing action generation capability.

We introduce GUI-Drop, a training-free visual token pruning framework tailored to sequential GUI interactions. The decoder has L layers indexed from 0 to $L - 1$. Operating after an intermediate

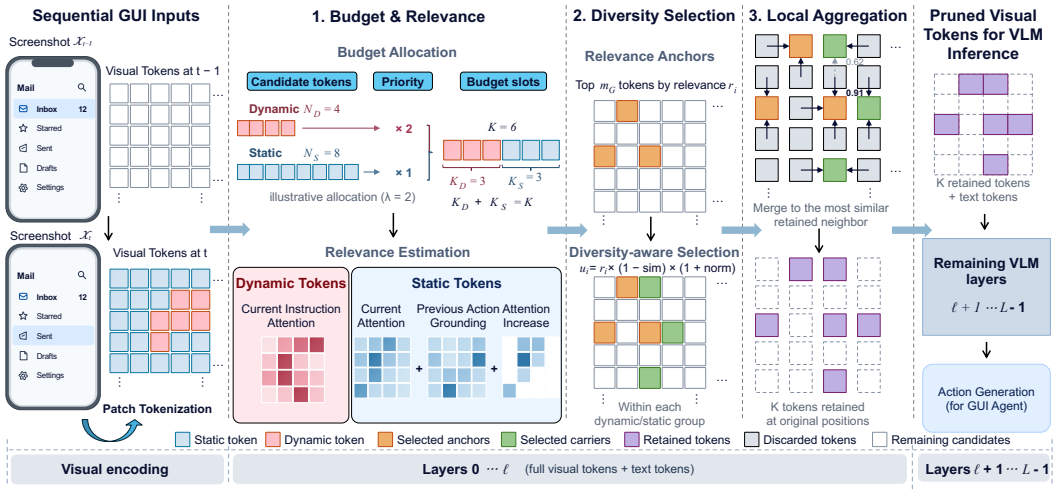


Figure 2: Overview of the GUI-Drop framework. (1) Budget and relevance estimation balances capacity and evaluates importance across dynamic and static tokens. (2) Diversity selection picks top relevance anchors followed by complementary context carriers within each group. (3) Local aggregation merges discarded tokens into their most similar retained spatial neighbors.

decoder layer $\ell \in \{0, \ldots, L-2\}$, GUI-Drop retains K_t informative visual tokens and updates their representations, allowing subsequent layers $\ell + 1$ to $L - 1$ to compute only the retained visual tokens alongside all text tokens. In addition to the current observation X_t, GUI-Drop incorporates the preceding screenshot X_{t-1} to capture visual state transitions and leverages the decoding attention recorded during action generation at step $t - 1$ to identify critical visual tokens guided by sequential interaction continuity. As illustrated in Figure 2, GUI-Drop realizes this pipeline through three coordinated stages: (1) budget allocation between dynamic and static regions with relevance estimated from current attention and historical decoding signals; (2) anchor-conditioned selection that balances high-relevance tokens with diverse context carriers; and (3) local feature aggregation that merges discarded neighboring representations into retained tokens. In the subsequent subsections, we describe each stage in detail.

3.1 Budget Allocation and Relevance across State Transitions

Budget allocation. Uniform budget allocation tends to waste token capacity on redundant static backgrounds while starving subtle dynamic updates. To eliminate this redundancy and prioritize active state changes, we partition visual tokens into dynamic and static subsets based on inter-frame patch similarity and allocate higher capacity to updated regions. Specifically, for each visual token position $i \in \mathcal{V}_t$, let $\mathbf{b}_i^t \in \mathbb{R}^{d_p}$ denote the unit-normalized concatenation of its constituent raw pixel patches. By evaluating the inter-frame patch cosine similarity $\langle \mathbf{b}_i^t, \mathbf{b}_i^{t-1} \rangle$ against a threshold $\tau \in [0, 1]$, we define the static token set $\mathcal{S}_t = \{i \in \mathcal{V}_t \mid \langle \mathbf{b}_i^t, \mathbf{b}_i^{t-1} \rangle \geq \tau\}$ and dynamic token set $\mathcal{D}_t = \mathcal{V}_t \setminus \mathcal{S}_t$, with cardinalities $N_{\mathcal{S}} = |\mathcal{S}_t|$ and $N_{\mathcal{D}} = |\mathcal{D}_t|$. Given the overall retention budget K_t, the dynamic quota $K_{\mathcal{D}}$ and static quota $K_{\mathcal{S}}$ are computed via

$$K_{\mathcal{D}} = \text{clip}\left(\text{round}\left(\frac{K_t \lambda N_{\mathcal{D}}}{\lambda N_{\mathcal{D}} + N_{\mathcal{S}}}\right), \max(0, K_t - N_{\mathcal{S}}), \min(N_{\mathcal{D}}, K_t)\right), \quad K_{\mathcal{S}} = K_t - K_{\mathcal{D}}. \quad (1)$$

Here, the scaling factor $\lambda > 1$ explicitly biases the allocation toward dynamic tokens, ensuring that visual updates receive sufficient capacity, while the clipping bounds guarantee that neither group quota exceeds its available token capacity. For the initial step ($t = 1$) without historical context, all tokens default to the dynamic set, which directly receives the entire step retention budget.

Relevance estimation for dynamic tokens. Dynamic tokens derive their task relevance solely from the current step, as inter-frame visual changes invalidate historical attention signals. Under causal self-attention, trailing prefill queries naturally aggregate the complete multimodal context,

and their attention to visual tokens at the pruning decision layer ℓ serves as an effective proxy for task relevance before token eviction. Let $A^t_{h,q,i}$ denote the attention weight from query q to visual token i in head h at this decision layer. We average this attention across all H heads within the trailing window $\mathcal{Q}_t$ of the last w prefill queries to obtain trailing prompt attention a^t_i,

$$a^t_i = \frac{1}{H|\mathcal{Q}_t|} \sum_{h=1}^{H} \sum_{q \in \mathcal{Q}_t} A^t_{h,q,i}.$$

The dynamic relevance score is directly set to $r^t_i = a^t_i$ for each dynamic token $i \in \mathcal{D}_t$.

Relevance estimation for static tokens. Although static tokens remain visually unchanged across steps, their task relevance shifts with interaction progression. Relying solely on current prompt attention overlooks the execution continuity of the prior action, while solely reusing past signals misses newly focused static elements. We therefore evaluate static tokens by fusing current attention, historical decoding attention, and positive attention share shifts. For the preceding step $t-1$, let $\mathcal{C}_{t-1}$ denote the set of grounding relevant semantic spans, covering reasoning fields, action types, and coordinate parameters while excluding structural formatting tokens. We average layer ℓ decoding attention across heads and queries within each span $c \in \mathcal{C}_{t-1}$ to obtain $\mathbf{d}^{t-1}_c$. Since layers 0 through ℓ preserve complete visual representations during decoding, $\mathbf{d}^{t-1}_c$ covers all original visual positions despite token pruning in deeper layers. For any vector $\mathbf{x}$ and index subset $\mathcal{G}$, we define the group normalization operator as $\mathcal{N}_{\mathcal{G}}(\mathbf{x})_i = x_i / \sum_{j \in \mathcal{G}} x_j$. Normalizing $\mathbf{d}^{t-1}_c$ over the current static set balances disparate attention magnitudes across semantic fields, ensuring that concise action coordinates are not overshadowed by extensive reasoning text. Let $\Delta^t_i = \left[\mathcal{N}_{\mathcal{S}_t}(\mathbf{a}^t)_i - \mathcal{N}_{\mathcal{S}_t}(\mathbf{a}^{t-1})_i\right]_+$ represent the positive increment in attention share, where $[x]_+ = \max(x, 0)$ is the rectified linear operator. For each static token $i \in \mathcal{S}_t$, the static relevance score r^t_i is formulated as

$$r^t_i = \frac{1}{3}\left(\mathcal{N}_{\mathcal{S}_t}(\mathbf{a}^t)_i + \frac{1}{|\mathcal{C}_{t-1}|} \sum_{c \in \mathcal{C}_{t-1}} \mathcal{N}_{\mathcal{S}_t}(\mathbf{d}^{t-1}_c)_i + \Delta^t_i\right). \tag{2}$$

Leaving Δ^t_i unnormalized scales its contribution directly with the magnitude of attention shift, ensuring that tokens receive an additional bonus only when experiencing a genuine surge in attention.

3.2 Anchor Conditioned Diversity Selection

While high attention indicates task relevance, highly attended tokens may nonetheless share redundant representations. To retain relevant content while capturing complementary context, we divide each group quota into relevance anchors and context carriers. For clarity, the step index t is omitted below. For each group $\mathcal{G} \in \{\mathcal{D}, \mathcal{S}\}$ with budget $K_{\mathcal{G}}$, we select the $m_{\mathcal{G}} = \mathrm{round}(\gamma K_{\mathcal{G}})$ tokens with the highest relevance scores r_i within $\mathcal{G}$ as relevance anchors $\mathcal{A}_{\mathcal{G}}$, where $\gamma \in (0, 1)$ is the anchor ratio. Let $\mathbf{h}_i$ denote the hidden state after layer ℓ, normalized as $\mathbf{z}_i = \mathbf{h}_i / \|\mathbf{h}_i\|_2$.

To select context carriers that provide complementary semantic information, we evaluate each remaining candidate token via a diversity-aware scoring function,

$$u_i = r_i \left(1 - \max_{j \in \mathcal{A}_{\mathcal{G}}} \mathbf{z}_i^\top \mathbf{z}_j\right)\left(1 + \min\left(\frac{\|\mathbf{h}_i\|_2}{\bar{n}}, 1\right)\right), \tag{3}$$

where $\bar{n} = \frac{1}{N}\sum_{j \in \mathcal{V}} \|\mathbf{h}_j\|_2$ denotes the average visual token norm. This scoring function balances base relevance r_i with directional dissimilarity to existing anchors and a bounded feature magnitude reward. Specifically, the dissimilarity term penalizes candidate tokens aligned with already selected anchors, while the magnitude term favors tokens with stronger feature activations to preserve localized interface elements. Saturating the magnitude reward at $\bar{n}$ prevents outlier activations from dominating the score. We then retain the $K_{\mathcal{G}} - m_{\mathcal{G}}$ highest-scoring candidate tokens as context carriers in a single pass without iterative updates. Both groups are processed independently, and their union forms the final retained set $\mathcal{K}$, exactly matching the target retention budget.

3.3 Spatial Neighborhood Feature Aggregation

Discarded tokens can still encode fine-grained details adjacent to the retained content. We aggregate these features locally to preserve spatial locality and avoid mixing unrelated features. For each

discarded token j, let $\mathcal{N}_j$ denote the set of retained tokens directly adjacent to j (above, below, left, or right) on the merged visual grid. We assign each discarded token j to its most similar retained neighbor $\pi(j) = \arg\max_{i \in \mathcal{N}_j} \mathbf{z}_j^\top \mathbf{z}_i$. Assignments may cross the static and dynamic boundary. Tokens lacking adjacent retained neighbors are dropped directly without aggregation. For each retained token $i \in \mathcal{K}$ with assigned neighbor set $\mathcal{M}_i = \{j \in \mathcal{V} \setminus \mathcal{K} \mid \pi(j) = i\}$, we update the hidden state via

$$\widetilde{\mathbf{h}}_i = \frac{\mathbf{h}_i + \sum_{j \in \mathcal{M}_i} \mathbf{h}_j}{1 + |\mathcal{M}_i|}. \tag{4}$$

Both anchors and carriers follow this uniform aggregation rule. Layers 0 through ℓ process the complete visual sequence and preserve its corresponding KV entries, allowing decode attention at layer ℓ to cover all original visual positions. From layer $\ell + 1$ onward, the model processes the aggregated visual tokens in $\mathcal{K}$ alongside all text tokens in their original order and with their original positional coordinates.

4 Experiments

4.1 Experimental Setup

Benchmarks and metrics. We evaluate GUI-Drop on representative benchmarks covering mobile and desktop platforms under online and offline settings. (1) **AndroidWorld** evaluates online mobile interaction and measures task success rate (SR) through live environment execution. (2) **AndroidControl** examines offline mobile trajectories, reporting step accuracy separately under high-level goal instructions (AC-High) and low-level step instructions (AC-Low). (3) **Android in the Wild (AITW)** assesses offline mobile interaction, measuring action accuracy across recorded trajectories. (4) **AgentNetBench** evaluates complex desktop workflows, reporting the mean continuous action scores over recorded trajectories following official protocols. Extended benchmark specifications and scoring protocols appear in Appendix C.

Models and baselines. Our primary evaluations employ two specialized GUI agents, Step-GUI-4B and UI-Voyager. To examine performance beyond specialized GUI agents, we further evaluate general vision language models on AndroidWorld and AITW, including Qwen3-VL at 2B, 4B, and 8B scales as well as Qwen2.5-VL at 7B and 32B scales. We compare GUI-Drop against representative visual token pruning baselines across distinct selection paradigms, including **FastV** (Chen et al., 2025) for attention selection, **MMTok** (Dong et al., 2026) for multimodal coverage, **DivPrune** (Alvar et al., 2025) for feature diversity, **ZOO-Prune** (Kim et al., 2026) for sensitivity guidance, and **VLA-Pruner** (Liu et al., 2025) for temporal action guidance. Recent GUI-specific pruning methods (Li et al., 2026c; Xu et al., 2026) are omitted from comparison due to lacking public implementations as of submission, with the unpruned Vanilla backbone serving as the full token reference.

Implementation details. To ensure a controlled comparison, all methods share the same observation setting, processing a single current screenshot and textual action history at each interaction step, evaluated under the native action format of each model with greedy decoding at temperature 0. By default, GUI-Drop uses decision layer $\ell = 3$ with zero-based indexing, with patch similarity threshold $\tau = 0.996$, dynamic priority $\lambda = 2.5$, trailing query window $w = 8$, and anchor ratio $\gamma = 0.75$. For the deeper 64-layer Qwen2.5-VL-32B backbone, the decision layer is adjusted to $\ell = 7$ to match its increased model depth. Regarding baseline implementations, FastV and VLA-Pruner are aligned to prune after the same decision layer ℓ, whereas MMTok, DivPrune, and ZOO-Prune prune before the language decoder. To capture state transitions across steps, GUI-Drop derives the preceding screenshot and decoding attention from prior execution during online interaction, and from the preceding step along recorded trajectories during offline evaluation. While main benchmark evaluations on AndroidControl report results under both high-level goal and low-level step instructions, subsequent evaluations default to high-level goal instructions, which require autonomous sub-goal planning and better reflect realistic GUI interaction.

4.2 Main Results and Backbone Generalization

Execution performance across visual retention regimes. As reported in Table 1, evaluating Step-GUI-4B and UI-Voyager across diverse visual retention budgets reveals consistent advantages

Table 1: Performance comparison across visual token retention budgets on Step-GUI-4B and UI-Voyager. All metrics are reported in percent (%). Bold marks the best result among pruning methods.

	Step-GUI-4B				UI-Voyager			
Method	AndroidWorld	AC-High	AC-Low	AITW	AndroidWorld	AC-High	AC-Low	AITW
Vanilla	60.3	56.2	68.4	48.1	80.2	33.2	43.6	33.1
Retain 10% of visual tokens								
FastV (ECCV'24)	5.2	33.2	42.8	24.6	12.1	19.4	26.8	14.9
ZOO-Prune (CVPR'26)	8.6	26.4	37.0	24.3	6.0	16.4	21.0	15.4
MMTok (ICLR'26)	6.9	35.6	48.6	20.9	9.5	23.6	31.0	16.5
DivPrune (CVPR'25)	6.9	29.6	38.0	21.9	8.6	23.6	21.2	13.7
VLA-Pruner (arXiv'25)	6.9	31.2	42.8	26.7	6.9	17.8	26.0	18.0
GUI-Drop	**14.7**	**37.6**	**51.8**	**28.4**	**29.3**	**26.6**	**32.6**	**19.1**
Retain 20% of visual tokens								
FastV (ECCV'24)	24.1	46.8	59.4	32.2	32.8	27.4	38.0	21.3
ZOO-Prune (CVPR'26)	15.5	34.6	46.4	31.0	12.9	17.6	25.8	19.3
MMTok (ICLR'26)	25.9	46.6	58.6	30.7	29.3	29.2	36.8	23.7
DivPrune (CVPR'25)	11.2	34.6	44.2	28.8	12.1	22.8	25.6	18.7
VLA-Pruner (arXiv'25)	12.9	38.8	50.4	33.0	43.1	22.6	30.8	23.1
GUI-Drop	**26.7**	**49.8**	**61.4**	**35.8**	**59.5**	**31.0**	**38.2**	**25.7**
Retain 40% of visual tokens								
FastV (ECCV'24)	46.6	53.4	**65.2**	42.0	71.6	**33.4**	41.8	29.3
ZOO-Prune (CVPR'26)	30.2	44.6	55.0	39.8	20.7	24.6	31.8	25.5
MMTok (ICLR'26)	50.0	54.0	65.0	40.3	54.3	31.4	41.4	30.4
DivPrune (CVPR'25)	25.0	44.4	53.4	37.8	25.9	25.6	33.0	25.7
VLA-Pruner (arXiv'25)	46.6	48.2	59.8	40.5	60.3	27.4	38.2	28.9
GUI-Drop	**53.4**	**55.0**	63.8	**44.3**	**75.9**	33.2	**43.2**	**30.6**

for GUI-Drop over competing pruning baselines. The performance divergence is particularly pronounced during online interactive execution on AndroidWorld, where the loss of task-critical visual cues causes errors to accumulate across steps, ultimately leading to task failure. At 20% retention on UI-Voyager, GUI-Drop achieves 59.5% task success, outperforming the strongest baseline VLA-Pruner by 16.4 percentage points. This gap stems from the discrete nature of GUI state transitions. Applying past action attention uniformly across the screenshot assumes smooth physical continuity, introducing misleading guidance when interface changes break the alignment between historical attention and current task relevance. Meanwhile, diversity-based selection in DivPrune disrupts the cohesive structure of UI elements, fragmenting widgets while expending quota on uninformative blank containers across sparse grids. As visual retention increases to 40%, GUI-Drop reaches within 4.3 and 6.9 percentage points of the uncompressed Vanilla baseline on UI-Voyager and Step-GUI-4B, respectively. In this regime, where performance across methods closely approaches Vanilla, FastV gains a slight edge on certain offline AndroidControl tasks by concentrating its entire visual quota on query attended regions, maximizing token density on target widgets. This difference arises because GUI-Drop performs local feature aggregation to compensate for discarded neighboring tokens, which is beneficial under constrained budgets but can slightly blend boundary features when target elements are already densely retained. Overall, GUI-Drop maintains the most stable task performance across retention budgets, confirming that coupling state transition awareness with local spatial compensation effectively safeguards execution reliability.

Performance across model families and scales. To evaluate the generalizability of pruning methods across architectures and scales, Figure 3 compares GUI-Drop against baselines on five backbones from the Qwen3-VL and Qwen2.5-VL families at 40% visual token retention. GUI-Drop achieves the highest average score relative to Vanilla on both benchmarks, demonstrating robust cross-model generalizability. On AndroidWorld, it consistently outperforms all pruning baselines across all five backbones, retaining 93.6% of Vanilla performance on average and leading the

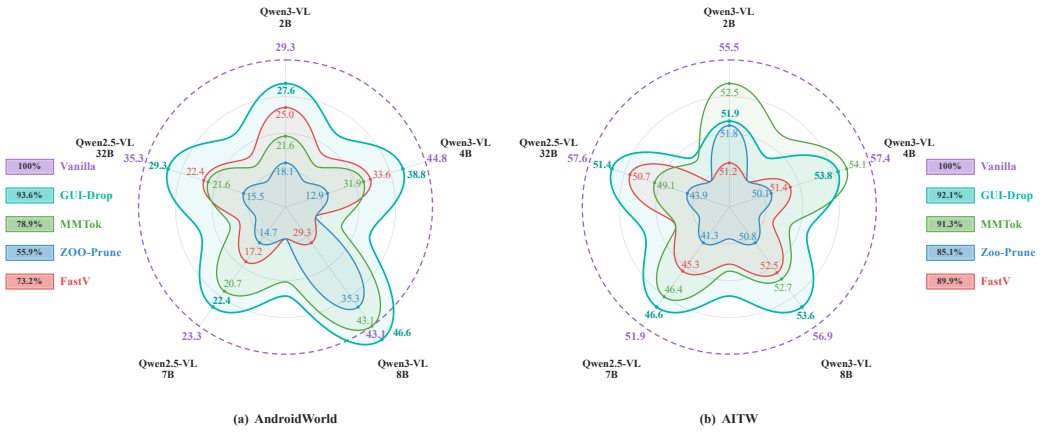


Figure 3: Performance across five backbones at 40% visual retention on (a) AndroidWorld and (b) AITW. Dashed circles denote unpruned Vanilla (100%). Legend values average relative performance over backbones. Axes use independent radial ranges for visual contrast, and radial distances do not reflect zero-based ratios.

Table 2: Action score (%) on AgentNetBench across retention budgets using Qwen2.5-VL-7B.

	Visual token retention				
Method	10%	20%	30%	40%	100%
Vanilla	—	—	—	—	50.5
FastV	14.8	23.1	31.2	35.8	—
ZOO-Prune	12.5	16.0	23.6	31.7	—
MMTok	14.4	17.4	26.3	37.5	—
GUI-Drop	**16.0**	**24.8**	**34.3**	**41.3**	—

Table 3: Component ablations on Step-GUI-4B at 20% retention on AndroidControl. Acc. denotes step accuracy (%).

	AndroidControl
Variant	Acc. (%) ↑
GUI-Drop	**49.8**
w/o cross-step relevance	48.4
w/o dynamic priority	48.8
w/o diversity selection	48.0
w/o local aggregation	47.6

strongest baseline by 14.7 percentage points. On Qwen3-VL-8B, GUI-Drop reaches 46.6% task success, exceeding the uncompressed Vanilla baseline at 43.1% by 3.5 percentage points, as filtering task-irrelevant background regions reduces redundancy and prevents attention dispersion on key widgets. In comparison, performance margins between pruning methods narrow on the offline AITW benchmark, where GUI-Drop retains 92.1% of Vanilla performance. On the smaller Qwen3-VL-2B and Qwen3-VL-4B models, MMTok achieves a slight edge over GUI-Drop. This occurs because the representative coverage of MMTok preserves the coarse global layout across the interface, which assists lightweight backbones with limited contextual reasoning in single-step grounding. As model scale increases to 8B and 32B, and particularly in multi-step online execution on AndroidWorld, the lack of cross-step temporal awareness degrades static baselines, allowing GUI-Drop to maintain obvious advantages across all scales.

Cross-platform transfer to desktop environments. To evaluate the applicability of token pruning to complex desktop interfaces, Table 2 presents action scores on AgentNetBench using Qwen2.5-VL-7B. Desktop interfaces exhibit dense multi-window hierarchies and compact widgets, posing severe challenges for visual token pruning. GUI-Drop maintains the highest action score across all evaluated retention budgets. At 40% retention, GUI-Drop reaches 41.3%, outperforming MMTok by 3.8 and FastV by 5.5 percentage points while approaching the uncompressed Vanilla baseline of 50.5%. Under tighter retention budgets of 20% and 10%, competing baselines degrade steeply as unstructured pruning frequently discards compact interactive elements. GUI-Drop addresses this challenge by selecting diverse context carriers alongside relevance anchors, preventing token over-concentration and preserving compact widgets through local feature aggregation. These results confirm that GUI-Drop generalizes reliably across complex desktop environments.

Table 4: Inference efficiency and task success using UI-Voyager on AndroidWorld at 40% visual retention. FLOPs and prefill time evaluate language decoder prefill excluding vision encoding. Parentheses report relative speedup over Vanilla. SR denotes success rate (%).

Models	Method	Vis. tokens	KV Cache (MiB)↓	FLOPs (T)↓	Prefill Time (ms)↓	Decode Latency (ms/token)↓	Total Time (s/step)↓	SR (%)↑
UI-Voyager	Vanilla	2550	526.0	35.46	693.6 (1.00×)	43.14 (1.00×)	6.306 (1.00×)	80.2
	FastV	1020	337.6	21.02	356.3 (1.95×)	43.51 (0.99×)	6.041 (1.04×)	71.6
	MMTok	1020	314.0	19.20	307.6 (2.25×)	42.52 (1.01×)	5.948 (1.06×)	54.3
	GUI-Drop	1020	333.0	20.68	361.1 (1.92×)	41.55 (1.04×)	5.779 (1.09×)	75.9

4.3 Ablation Studies

To isolate the contribution of each key design choice in GUI-Drop, we evaluate ablated configurations on Step-GUI-4B using AndroidControl at 20% visual retention in Table 3. The complete framework achieves 49.8% step accuracy, with local spatial compensation providing the primary performance safeguard. Disabling local spatial aggregation in Equation 4 incurs the steepest drop to 47.6% (a 2.2 percentage point decrease), confirming that compensating discarded neighbor representations into retained spatial tokens is essential for mitigating UI representation fragmentation. Reverting diversity selection in Equation 3 to relevance-only selection lowers accuracy to 48.0%, demonstrating the necessity of capturing complementary contextual cues within UI elements. On the temporal front, incorporating prior decoding signals in Equation 2 and dynamically prioritizing updated regions in Equation 1 improve accuracy by 1.4 and 1.0 percentage points, respectively. This validates that leveraging execution continuity across state transitions effectively guides visual token selection in sequential GUI interaction.

4.4 Comprehensive Inference Efficiency

Table 4 examines inference efficiency on UI-Voyager on AndroidWorld under 40% visual token retention. Because shallow layers constitute only a small fraction of the model depth, pruning after multimodal interaction at layer $\ell = 3$ allows GUI-Drop to bypass the majority of subsequent prefill computation, achieving a 1.92× prefill acceleration and the lowest overall per-step latency. Although pruning prior to the decoder in MMTok yields slightly lower computation, selecting tokens before attention layers fails to align with action intent, and pursuing global coverage over passive containers discards compact widgets targeted by the instruction, collapsing task success from 80.2% to 54.3%. GUI-Drop prevents this breakdown by deferring selection until layer $\ell = 3$, where shallow self-attention decisively concentrates relevance onto task-critical widgets, preserving 75.9% success. For controlled profiling across baselines without kernel fusion artifacts, all latency measurements evaluate eager attention with BF16 precision on NVIDIA A100 GPUs (Appendix E). Crucially, GUI-Drop natively supports FlashAttention 2 across all decoder layers, as the selector evaluates auxiliary query and key dot products at decision layer ℓ without interrupting the fused main attention pipeline, while subsequent layers execute directly on the compacted sequence of retained tokens.

5 Conclusion

In this paper, we presented GUI-Drop, a training-free visual token pruning framework designed for GUI navigation agents. To address the neglect of logical continuity and spatial representation fragmentation in existing pruning methods, GUI-Drop couples cross-step state transition awareness with anchor-conditioned diversity selection and local feature aggregation, preserving task-critical visual evidence for accurate action generation. Extensive evaluations across online interactive environments and offline trajectory benchmarks demonstrate that GUI-Drop consistently outperforms competing pruning baselines across diverse multimodal backbones, maintaining robust execution reliability under constrained visual budgets. Furthermore, efficiency profiling confirms that GUI-Drop substantially curtails decoder computation and memory footprints, translating theoretical FLOP savings into practical wall-clock speedups while safeguarding reliable closed-loop execution.

AI USE STATEMENT

We used generative AI tools to discuss methodological and experimental ideas, assist with code implementation, identify relevant literature, and draft and edit parts of the manuscript. The authors reviewed the AI-assisted work and take responsibility for the final content of the paper.

REPRODUCIBILITY STATEMENT

We provide anonymized source code and evaluation scripts as supplementary material. Implementation details and experimental settings are provided in the appendix. The code will be publicly released upon acceptance.

REFERENCES

Amirhossein Abaskohi, Yuhang He, Peter West, Giuseppe Carenini, Pranit Chawla, and Vibhav Vineet. ReVision: Scaling Computer-Use Agents via Temporal Visual Redundancy Reduction, June 2026. URL http://arxiv.org/abs/2605.11212. arXiv:2605.11212 [cs.CL] version: 3.

Saeed Ranjbar Alvar, Gursimran Singh, Mohammad Akbari, and Yong Zhang. DivPrune: Diversity-based Visual Token Pruning for Large Multimodal Models. pp. 9392–9401, 2025. URL https://openaccess.thecvf.com/content/CVPR2025/html/Alvar_DivPrune_Diversity-based_Visual_Token_Pruning_for_Large_Multimodal_Models_CVPR_2025_paper.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, November 2025a. URL http://arxiv.org/abs/2511.21631. arXiv:2511.21631 [cs.CV].

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-VL Technical Report, February 2025b. URL http://arxiv.org/abs/2502.13923. arXiv:2502.13923 [cs.CV].

Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token Merging: Your ViT But Faster. September 2022. URL https://openreview.net/forum?id=JroZRaRw7Eu.

Liang Chen, Haozhe Zhao, Tianyu Liu, Shuai Bai, Junyang Lin, Chang Zhou, and Baobao Chang. An Image is Worth 1/2 Tokens After Layer 2: Plug-and-Play Inference Acceleration for Large Vision-Language Models. In Aleš Leonardis, Elisa Ricci, Stefan Roth, Olga Russakovsky, Torsten Sattler, and Gül Varol (eds.), *Computer Vision – ECCV 2024*, pp. 19–35, Cham, 2025. Springer Nature Switzerland. ISBN 978-3-031-73004-7. doi: 10.1007/978-3-031-73004-7_2.

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 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)*, pp. 9313–9332, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.505. URL https://aclanthology.org/2024.acl-long.505/.

Mohamed Dhouib, Davide Buscaldi, Sonia Vanier, and Aymen Shabou. PACT: Pruning and Clustering-Based Token Reduction for Faster Visual Language Models. pp. 14582–14592, 2025. URL https://openaccess.thecvf.com/content/CVPR2025/html/Dhouib_PACT_Pruning_and_Clustering-Based_Token_Reduction_for_Faster_Visual_Language_CVPR_2025_paper.html.

Sixun Dong, Juhua Hu, Mian Zhang, Ming Yin, Yanjie Fu, and Qi Qian. MM-Tok: Multimodal Coverage Maximization for Efficient Inference of VLMs. *International Conference on Learning Representations*, 2026:48075–48099, April 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/4f8745c0ca5c951a92844ad7668cfca5-Abstract-Conference.html.

Mark Endo, Xiaohan Wang, and Serena Yeung-Levy. Feather the Throttle: Revisiting Visual Token Pruning for Vision-Language Model Acceleration. pp. 22826–22835, 2025. URL https://openaccess.thecvf.com/content/ICCV2025/html/Endo_Feather_the_Throttle_Revisiting_Visual_Token_Pruning_for_Vision-Language_Model_ICCV_2025_paper.html.

Boyu Gou, Demi Ruohan Wang, Boyuan Zheng, Yanan Xie, Cheng Chang, Yiheng Shu, Huan Sun, and Yu Su. Navigating the Digital World as Humans Do: Universal Visual Grounding for GUI Agents. *International Conference on Learning Representations*, 2025:30851–30883, May 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/4ca0e369689dadb25a5345ba9755ad6f-Abstract-Conference.html.

Yuhang Han, Wenzheng Yang, Yujie Chen, Xiangqi Jin, Yaojie Zhang, Siteng Huang, and Linfeng Zhang. STaR-KV: Spatio-Temporal Adaptive Re-weighting for KV Cache Compression in GUI Vision-Language Models, June 2026. URL http://arxiv.org/abs/2606.01790. arXiv:2606.01790 [cs.CV].

Kung-Hsiang Huang, Haoyi Qiu, Yutong Dai, Caiming Xiong, and Chien-Sheng Wu. GUI-KV: Efficient GUI agents via KV cache with spatio-temporal awareness, October 2025. URL http://arxiv.org/abs/2510.00536. arXiv:2510.00536 [cs].

Yihong Huang, Fei Ma, Yihua Shao, Jingcai Guo, Zitong Yu, Laizhong Cui, and Qi Tian. Nüwa: Mending the Spatial Integrity Torn by VLM Token Pruning. *International Conference on Learning Representations*, 2026:100825–100856, April 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/a46156bd3579c3b268108ea6aca71d13-Abstract-Conference.html.

Youngeun Kim, Youjia Zhang, Huiling Liu, Aecheon Jung, Sunwoo Lee, and Sungeun Hong. ZOO-Prune: Training-Free Token Pruning via Zeroth-Order Gradient Estimation in Vision-Language Models. pp. 39572–39582, 2026. URL https://openaccess.thecvf.com/content/CVPR2026/html/Kim_ZOO-Prune_Training-Free_Token_Pruning_via_Zeroth-Order_Gradient_Estimation_in_Vision-Language_CVPR_2026_paper.html.

Ao Li, Yuxiang Duan, Jinghui Zhang, Congbo Ma, Yutong Xie, Gustavo Carneiro, Mohammad Yaqub, and Hu Wang. TransPrune: Token Transition Pruning for Efficient Large Vision-Language Model. pp. 39529–39538, 2026a. URL https://openaccess.thecvf.com/content/CVPR2026/html/Li_TransPrune_Token_Transition_Pruning_for_Efficient_Large_Vision-Language_Model_CVPR_2026_paper.html.

Daiqiang Li, Zihao Pan, Zeyu Zhang, Xuyang Liu, Shijia Xu, Ronghao Chen, Huacan Wang, Honggang Chen, Linfeng Zhang, Zhangquan Chen, and Haiyun Jiang. Where and How to Prune: An Empirical Study of Visual Token Pruning for GUI Agent Navigation, August 2026b. URL http://arxiv.org/abs/2603.26041. arXiv:2603.26041 [cs.CV] version: 4.

Yuankai Li, Tinghui Zhu, Ha Min Son, Zhe Zhao, Xin Liu, and Muhao Chen. AQuaUI: Visual Token Reduction for GUI Agents with Adaptive Quadtrees, May 2026c. URL http://arxiv.org/abs/2605.19260. arXiv:2605.19260 [cs.AI].

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 *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 19498–19508, June 2025. doi: 10.1109/cvpr52734.2025.01816. URL https://ieeexplore.ieee.org/document/11093438/.

Zichuan Lin, Feiyu Liu, Yijun Yang, Jiafei Lyu, Yiming Gao, Yicheng Liu, Zhicong Lu, Yangbin Yu, Mingyu Yang, Junyou Li, Deheng Ye, and Jie Jiang. UI-Voyager: A Self-Evolving GUI Agent Learning via Failed Experience, March 2026. URL https://arxiv.org/abs/2603.24533. arXiv:2603.24533 [cs].

Ziyan Liu, Yeqiu Chen, Hongyi Cai, Tao Lin, Shuo Yang, Zheng Liu, and Bo Zhao. Bridging the Semantic-Action Gap in Visual Token Pruning for Efficient VLA Inference, November 2025. URL https://arxiv.org/abs/2511.16449v5.

Mingyu Ouyang, Kevin Qinghong Lin, Mike Zheng Shou, and Hwee Tou Ng. FocusUI: Efficient UI Grounding via Position-Preserving Visual Token Selection. pp. 20313–20323, 2026. URL https://openaccess.thecvf.com/content/CVPR2026/html/Ouyang_FocusUI_Efficient_UI_Grounding_via_Position-Preserving_Visual_Token_Selection_CVPR_2026_paper.html.

Yijun Qian, Yujie Lu, Alexander Hauptmann, and Oriana Riva. Visual Grounding for User Interfaces. In Yi Yang, Aida Davani, Avi Sil, and Anoop Kumar (eds.), *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 6: Industry Track)*, pp. 97–107, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-industry.9. URL https://aclanthology.org/2024.naacl-industry.9/.

Yujia Qin, Yining Ye, Junjie Fang, Haoming Wang, Shihao Liang, Shizuo Tian, Junda Zhang, Jiahao Li, Yunxin Li, Shijue Huang, Wanjun Zhong, Kuanye Li, Jiale Yang, Yu Miao, Woyu Lin, Longxiang Liu, Xu Jiang, Qianli Ma, Jingyu Li, Xiaojun Xiao, Kai Cai, Chuang Li, Yaowei Zheng, Chaolin Jin, Chen Li, Xiao Zhou, Minchao Wang, Haoli Chen, Zhaojian Li, Haihua Yang, Haifeng Liu, Feng Lin, Tao Peng, Xin Liu, and Guang Shi. UI-TARS: Pioneering Automated GUI Interaction with Native Agents, January 2025. URL http://arxiv.org/abs/2501.12326. arXiv:2501.12326 [cs].

Chris Rawles, Sarah Clinckemaillie, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth Fair, Alice Li, William Bishop, Wei Li, Folawiyo Campbell-Ajala, Daniel Toyama, Robert Berry, Divya Tyamagundlu, Timothy Lillicrap, and Oriana Riva. AndroidWorld: A Dynamic Benchmarking Environment for Autonomous Agents. *International Conference on Learning Representations*, 2025:406–441, May 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/01a83bc2f2732a58e6aa731e659e7101-Abstract-Conference.html.

Tianlin Shi, Andrej Karpathy, Linxi Fan, Jonathan Hernandez, and Percy Liang. World of Bits: An Open-Domain Platform for Web-Based Agents. In *Proceedings of the 34th International Conference on Machine Learning*, pp. 3135–3144. PMLR, July 2017. URL https://proceedings.mlr.press/v70/shi17a.html.

Zhichao Sun, Yidong Ma, Gang Liu, Yibo Chen, Xu Tang, Yao Hu, and Yongchao Xu. IVC-Prune: Revealing the Implicit Visual Coordinates in LVLMs for Vision Token Pruning. *International Conference on Learning Representations*, 2026:111471–111492, April 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/b51b3565a6775c557093cb99457fc7fc-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, April 2024. URL http://arxiv.org/abs/2401.16158. arXiv:2401.16158 [cs.CL].

Long Xing, Qidong Huang, Xiaoyi Dong, Jiajie Lu, Pan Zhang, Yuhang Zang, Yuhang Cao, Conghui He, Jiaqi Wang, Feng Wu, and Dahua Lin. PyramidDrop: Accelerating Your Large Vision-Language Models via Pyramid Visual Redundancy Reduction, February 2025. URL http://arxiv.org/abs/2410.17247. arXiv:2410.17247 [cs.CV].

Zhou Xu, Bowen Zhou, Qi Wang, Shuwen Feng, and Jingyu Xiao. Spatio-Temporal Token Pruning for Efficient High-Resolution GUI Agents, February 2026. URL https://arxiv.org/abs/2602.23235v1.

Haolong Yan, Jia Wang, Xin Huang, Yeqing Shen, Ziyang Meng, Zhimin Fan, Kaijun Tan, Jin Gao, Lieyu Shi, Mi Yang, Shiliang Yang, Zhirui Wang, Brian Li, Kang An, Chenyang Li, Lei Lei, Mengmeng Duan, Danxun Liang, Guodong Liu, Hang Cheng, Hao Wu, Jie Dong, Junhao Huang, Mei Chen, Renjie Yu, Shunshan Li, Xu Zhou, Yiting Dai, Yineng Deng, Yingdan Liang, Zelin Chen, Wen Sun, Chengxu Yan, Chunqin Xu, Dong Li, Fengqiong Xiao, Guanghao Fan, Guopeng Li, Guozhen Peng, Hongbing Li, Hang Li, Hongming Chen, Jingjing Xie, Jianyong Li, Jingyang Zhang, Jiaju Ren, Jiayu Yuan, Jianpeng Yin, Kai Cao, Liang Zhao, Liguo Tan, Liying Shi, Mengqiang Ren, Min Xu, Manjiao Liu, Mao Luo, Mingxin Wan, Na Wang, Nan Wu, Ning Wang, Peiyao Ma, Qingzhou Zhang, Qiao Wang, Qinlin Zeng, Qiong Gao, Qiongyao Li, Shangwu Zhong, Shuli Gao, Shaofan Liu, Shisi Gao, Shuang Luo, Xingbin Liu, Xiaojia Liu, Xiaojie Hou, Xin Liu, Xuanti Feng, Xuedan Cai, Xuan Wen, Xianwei Zhu, Xin Liang, Xin Liu, Xin Zhou, Yifan Sui, Yingxiu Zhao, Yukang Shi, Yunfang Xu, Yuqing Zeng, Yixun Zhang, Zejia Weng, Zhonghao Yan, Zhiguo Huang, Zhuoyu Wang, Zihan Yan, Zheng Ge, Jing Li, Yibo Zhu, Binxing Jiao, Xiangyu Zhang, and Daxin Jiang. Step-GUI Technical Report, December 2025. URL http://arxiv.org/abs/2512.15431. arXiv:2512.15431 [cs.CV].

Senqiao Yang, Yukang Chen, Zhuotao Tian, Chengyao Wang, Jingyao Li, Bei Yu, and Jiaya Jia. VisionZip: Longer is Better but Not Necessary in Vision Language Models. pp. 19792–19802, 2025. URL https://openaccess.thecvf.com/content/CVPR2025/html/Yang_VisionZip_Longer_is_Better_but_Not_Necessary_in_Vision_Language_CVPR_2025_paper.html.

Qizhe Zhang, Mengzhen Liu, Lichen Li, Ming Lu, Yuan Zhang, Junwen Pan, Qi She, and Shanghang Zhang. Beyond Attention or Similarity: Maximizing Conditional Diversity for Token Pruning in MLLMs. In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 25438–25468. Curran Associates, Inc., 2025a. doi: 10.52202/085713-0855. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/2433fec2144ccf5fea1c9c5ebdbc3924-Abstract-Conference.html.

Yuan Zhang, Chun-Kai Fan, Junpeng Ma, Wenzhao Zheng, Tao Huang, Kuan Cheng, Denis A. Gudovskiy, Tomoyuki Okuno, Yohei Nakata, Kurt Keutzer, and Shanghang Zhang. SparseVLM: Visual Token Sparsification for Efficient Vision-Language Model Inference. In *Proceedings of the 42nd International Conference on Machine Learning*, pp. 74840–74857. PMLR, July 2025b. URL https://proceedings.mlr.press/v267/zhang25s.html.

A Implementation and Reproducibility Details

A.1 Backbones and Visual Inputs

GUI-Drop operates on pretrained backbones without updating model parameters or introducing learned selector modules. UI-Voyager and Step-GUI-4B build upon the Qwen3-VL architecture, evaluated alongside general Qwen3-VL models across 2B, 4B, and 8B parameter scales. For the Qwen2.5-VL family evaluated at 7B and 32B scales, GUI-Drop applies identical selection and local aggregation mechanisms while preserving their native architectural layouts, including the visual encoder, spatial patch merging, and positional encodings. Model inference executes in bfloat16 precision, whereas attention reductions, feature normalization, and selection scores are accumulated in float32.

Each interaction step processes a single current screenshot observation. Input images follow the canonical vision processing pipeline of each backbone, with benchmark-specific resizing detailed in Appendix C. For an input image partitioned into $H_p \times W_p$ raw patches with spatial merging factor s, the resulting visual grid comprises H_p/s rows and W_p/s columns, yielding $N = H_p W_p / s^2$ visual tokens. Inter-frame patch representations $\mathbf{b}_i^t$ are constructed by grouping raw patches according to this spatial layout. In the UI-Voyager AndroidWorld setting, a screenshot with width 1080 and height 2400 pixels is resized to width 1088 and height 2400 pixels. With a patch size of 16 pixels and $s = 2$, the merged grid has 75 rows and 34 columns, giving $N_t = 2550$. The token retention budget is determined dynamically from the actual sequence length N at each step.

The action format remains native to each agent. UI-Voyager and general Qwen3-VL models output structured tool calls, Step-GUI employs its native tab-delimited action syntax, and Qwen2.5-VL outputs tool calls with screenshot pixel coordinates. Environment executors and benchmark evaluators parse these predictions into their respective target formats. Screenshot and textual history remain accessible to all baseline methods through the standard context prompt, whereas GUI-Drop maintains its auxiliary cross-step attention state independently of the prompt sequence.

A.2 Reference Configuration and Parameter Selection

Table 5 summarizes the reference hyperparameter configuration for GUI-Drop introduced in Section 3. Hyperparameters were empirically selected based on preliminary exploration across representative everyday Android tasks spanning diverse applications such as Calendar, Messaging, Notes, Files, and Expense management. Once determined, this reference configuration is kept strictly fixed across all benchmarks and model backbones throughout all experiments without per-benchmark fine-tuning. The visual retention ratio R specifies the budget evaluated in each experiment, while the remaining parameters govern transition scoring and diversity selection.

Table 5: Reference GUI-Drop configuration. Layer indices are zero based. The anchor ratio applies within each dynamic or static quota.

Parameter	Symbol	Setting
Decision layer	ℓ	3 (7 for Qwen2.5-VL-32B)
Static patch similarity threshold	τ	0.996
Dynamic priority weight	λ	2.5
Trailing prefill query window	w	8
Relevance anchor ratio	γ	0.75

Inference across all benchmarks adopts greedy decoding with temperature set to zero, top-p to one, and a maximum generation limit of 1024 tokens. Each interaction step constitutes an independent forward evaluation. Cross-step state carries only the auxiliary signals necessary for transition scoring, including preceding patch representations, spatial grid dimensions, prefill attention distributions, semantic decoding attention summaries, and the parsed action category. It retains no key-value caches or hidden representations from previous steps.

A.3 Prompts and Action Protocols

Each backbone preserves its native instruction format and response structure. Within each benchmark and model pairing, all pruning approaches share identical prompt templates, action parsers, and coordinate transformations. An input instance comprises the overall task objective, textual interaction history, the single current screenshot observation, and an instruction prompting the next executable action. AndroidControl Low additionally supplies the low-level step directive.

Under the mobile Qwen setting, the system message specifies the mobile interaction tool schema. The user prompt provides the user query, interaction progress, and current screenshot blocks sequentially, concluding with a prompt requesting the next interaction step. Step-GUI formats its native Chinese task descriptions and action guidelines in the user context alongside history, the observation, and response specifications. AgentNetBench employs the desktop environment tool schema parameterized by current display dimensions.

Table 6: Native response formats and coordinate conventions. Pixel coordinates refer to the screenshot supplied to the model before its internal image processor. Benchmark adapters convert parsed coordinates to the execution or scoring convention.

Backbone and setting	Response format	Spatial parameters
UI-Voyager and Qwen3-VL on mobile	Thought and Action text, followed by a tagged `mobile_use` JSON call	Normalized screen coordinates
Qwen2.5-VL on mobile	Thought and Action text, followed by a tagged `mobile_use` JSON call	Screenshot pixels
Step-GUI-4B on mobile	Native action fields separated by tabs, with optional thinking content	Coordinates normalized to $[0, 1000]$
Qwen2.5-VL on AgentNetBench	Thought text and tagged `computer_use` JSON calls	Screenshot pixels; signed wheel steps for scrolling

Action space. The mobile tool supports `click` and `long_press` with a coordinate, `swipe` with start and end coordinates, `type` and `answer` with text, `system_button` with a button name, `wait` with a duration, and `terminate` with a status. Step-GUI expresses the corresponding operations through `CLICK`, `LONGPRESS`, `SLIDE`, `TYPE`, `WAIT`, `HOME`, and `BACK`; it also supports `AWAKE` with an application name, `INFO` with a question, `COMPLETE` with a return value, and `ABORT`. The desktop tool exposes mouse movement, left and right clicks, double clicks, dragging, text entry, key presses, scrolling, waiting, and termination. Desktop predictions may contain an ordered list of tool calls, which is passed to the benchmark scorer without discarding extra actions.

Figure 4 shows compact response skeletons. Angle bracket placeholders denote variable content. They specify serialization, not additional instructions or recorded predictions. GUI-Drop extracts the supported reasoning, action, and parameter spans from these native responses without requesting an additional response for pruning.

Mobile tool response

```
Thought: <concise reasoning>
Action: <short action description>
<tool_call>
{"name": "mobile_use", "arguments": {
  "action": "click", "coordinate": [<x>, <y>]}}
</tool_call>
```

Step-GUI response Here `\t` denotes a tab.

```
<THINK><reasoning></THINK>
explain:<reason>\taction:CLICK\tpoint:<x>,<y>\tsummary:<text>
```

Figure 4: Representative native response skeletons. Other actions replace the action name and its required parameters. In the desktop protocol, the tool name is `computer_use` and the action and argument schema follow the desktop action space.

B Algorithmic Details

B.1 Complete Selection Procedure

Algorithm 1 summarizes token selection and aggregation at layer ℓ. Its inputs are the visual hidden states $\mathbf{h}_i$, grouped patch vectors $\mathbf{b}_i^t$, trailing attention $\mathbf{a}^t$, and preceding state $\mathcal{E}_{t-1}$. The state stores preceding patch vectors, grid dimensions, prefill attention, semantic decoding summaries, and action type. Valid checks grid compatibility and the required attention summaries. $\mathrm{Allocate}$, $\mathrm{StaticScore}$, and $\mathrm{CarrierScore}$ denote Equations 1, 2, and 3, respectively. TopK returns original visual indices, with $\mathrm{TopK}(\cdot, 0) = \emptyset$.

Algorithm 1 GUI-Drop token selection and aggregation

Require: $\{\mathbf{h}_i, \mathbf{b}_i^t\}_{i\in\mathcal{V}_t}$, $\mathbf{a}^t$, $\mathcal{E}_{t-1}$, $(R, \tau, \lambda, \gamma)$
Ensure: Retained indices $\mathcal{K}_t$ and features $\{\widetilde{\mathbf{h}}_i\}_{i\in\mathcal{K}_t}$
// Transition partition and relevance
$K_t \leftarrow \mathrm{round}(RN_t)$; $\mathcal{S}_t \leftarrow \emptyset$
if $\mathrm{Valid}(\mathcal{E}_{t-1})$ **then**
  $\mathcal{S}_t \leftarrow \{i \in \mathcal{V}_t \mid \langle \mathbf{b}_i^t, \mathbf{b}_i^{t-1} \rangle \geq \tau\}$
end if
$\mathcal{D}_t \leftarrow \mathcal{V}_t \setminus \mathcal{S}_t$
$(K_\mathcal{D}, K_\mathcal{S}) \leftarrow \mathrm{Allocate}(K_t, |\mathcal{D}_t|, |\mathcal{S}_t|, \lambda)$
$r_i^t \leftarrow a_i^t \ (i \in \mathcal{D}_t)$; $\{r_i^t\}_{i\in\mathcal{S}_t} \leftarrow \mathrm{StaticScore}(\mathbf{a}^t, \mathcal{E}_{t-1}, \mathcal{S}_t)$
// Anchor conditioned diversity selection
$\mathbf{z}_i \leftarrow \mathbf{h}_i / \|\mathbf{h}_i\|_2$; $\bar{n} \leftarrow N_t^{-1} \sum_{i\in\mathcal{V}_t} \|\mathbf{h}_i\|_2$
$\mathcal{I}_\mathcal{G} \leftarrow \mathcal{G}$ if $K_\mathcal{G} = |\mathcal{G}|$, else $\emptyset$, for $\mathcal{G} \in \{\mathcal{D}_t, \mathcal{S}_t\}$
for $\mathcal{G} \in \{\mathcal{D}_t, \mathcal{S}_t\}$ with $0 < K_\mathcal{G} < |\mathcal{G}|$ **do**
  $m_\mathcal{G} \leftarrow \min(K_\mathcal{G}, \max(1, \mathrm{round}(\gamma K_\mathcal{G})))$
  $\mathcal{A}_\mathcal{G} \leftarrow \mathrm{TopK}(\{r_i^t\}_{i\in\mathcal{G}}, m_\mathcal{G})$
  $u_i \leftarrow \mathrm{CarrierScore}(r_i^t, \mathbf{h}_i, \{\mathbf{z}_j\}_{j\in\mathcal{A}_\mathcal{G}}, \bar{n}), \quad i \in \mathcal{G} \setminus \mathcal{A}_\mathcal{G}$
  $\mathcal{C}_\mathcal{G} \leftarrow \mathrm{TopK}(\{u_i\}_{i\in\mathcal{G}\setminus\mathcal{A}_\mathcal{G}}, K_\mathcal{G} - m_\mathcal{G})$
  $\mathcal{I}_\mathcal{G} \leftarrow \mathcal{A}_\mathcal{G} \cup \mathcal{C}_\mathcal{G}$
end for
$\mathcal{K}_t \leftarrow \mathrm{Sort}(\mathcal{I}_{\mathcal{D}_t} \cup \mathcal{I}_{\mathcal{S}_t})$
// Spatial neighborhood aggregation
$\pi(j) \leftarrow \arg\max_{i\in\mathcal{N}_j} \mathbf{z}_j^\top \mathbf{z}_i, \quad j \in \mathcal{V}_t \setminus \mathcal{K}_t, \ \mathcal{N}_j \neq \emptyset$
$\mathcal{M}_i \leftarrow \{j \in \mathcal{V}_t \setminus \mathcal{K}_t \mid \mathcal{N}_j \neq \emptyset, \ \pi(j) = i\}, \quad i \in \mathcal{K}_t$
$\widetilde{\mathbf{h}}_i \leftarrow (\mathbf{h}_i + \sum_{j\in\mathcal{M}_i} \mathbf{h}_j)/(1 + |\mathcal{M}_i|), \quad i \in \mathcal{K}_t$
return $\mathcal{K}_t, \{\widetilde{\mathbf{h}}_i\}_{i\in\mathcal{K}_t}$

Visual encoding and layers 0 through ℓ precede the algorithm. The hidden states are collected after the decision layer's visual feature injection, and $\mathbf{a}^t$ averages attention over the trailing w queries and all heads. An empty static set skips static scoring. Carrier scores are evaluated once against the fixed anchor set. Sorting restores original visual order. $\mathcal{N}_j$ contains only retained positions immediately above, below, left, or right of j; discarded tokens without retained neighbors are omitted from aggregation.

Subsequent layers process the aggregated visual features alongside all text tokens at their original positions. During action generation, layer ℓ decode attention covers all original visual positions and is reduced into semantic summaries. After action parsing, valid summaries and the current patch vectors, grid dimensions, prefill attention, and action type form $\mathcal{E}_t$; otherwise the state is cleared. Decoder KV states remain local to the current request. Budget rounding, score ties, and zero relevance are specified below.

B.2 Budget and Selection Conventions

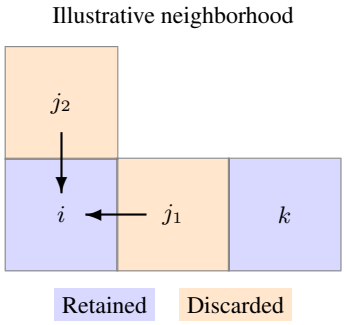
Equation 1 already constrains both group quotas by their available capacity and preserves $K_\mathcal{D} + K_\mathcal{S} = K$. An empty group receives zero tokens, while the other receives the full budget. Rounding follows nearest integer rounding with ties to even. Equal selection scores are resolved in favor of the smaller original visual index. If all relevance scores within a candidate group are zero, their multiplicative relevance weights are set to one for carrier scoring. The feature novelty and norm

terms then determine the carrier ranking. Carrier scores are evaluated against the anchors only, without updating the anchor set after selecting a carrier.

The norm reference $\bar{n}$ is the mean over all visual tokens in the current image. Local aggregation uses hidden states from the same decision point as diversity selection. A discarded token may be assigned across the dynamic and static group boundary, but only to an immediately adjacent retained grid position. Equal cosine similarities are resolved by the smaller visual index. Every source contributes to at most one target, and a source without a retained neighbor is omitted. All assignments are computed before updating any target representation.

B.3 Local Aggregation Illustration

Figure 5 illustrates the assignment and update in Equation 4. A discarded token considers only retained positions immediately above, below, to its left, or to its right on the merged grid. When several are available, cosine similarity of their hidden states determines its target. Both relevance anchors and context carriers can receive discarded representations. The target retains its original index and positional coordinates, so aggregation changes its feature without changing the selected set or its size.



Assignment

If $\langle \mathbf{z}_{j_1}, \mathbf{z}_i \rangle > \langle \mathbf{z}_{j_1}, \mathbf{z}_k \rangle$, then j_1 is assigned to i. Token j_2 has only i as a retained neighbor. Hence $\mathcal{M}_i = \{j_1, j_2\}$ and $\mathcal{M}_k = \emptyset$.

Equal weight update

$$\mathbf{h}'_i = \frac{\mathbf{h}_i + \mathbf{h}_{j_1} + \mathbf{h}_{j_2}}{3}, \qquad \mathbf{h}'_k = \mathbf{h}_k.$$

The illustrated set $\mathcal{K} = \{i, k\}$ remains unchanged. Both output tokens keep their original grid positions.

Figure 5: Schematic of local feature aggregation. Blue positions are retained targets, orange positions are discarded sources, and arrows indicate assignments under the stated similarity condition. This illustration explains the operator; it is not an empirical feature visualization.

B.4 Attention Collection and State Updates

For static relevance, the current and preceding prefill vectors are each normalized over the *current* static set. Their positive difference is left unnormalized. Preceding decode attention is reduced separately over the reasoning content, action content, and spatial action parameters exposed by the agent protocol. Each available semantic component is normalized over the current static set before the components are averaged. Spatial parameter evidence is used for click, long press, and swipe actions. Structural delimiters are excluded. This gives each semantic field equal weight independently of its response length. A zero attention mass produces a zero normalized vector.

For standard attention implementations that materialize the full attention matrix, the pruning selector directly retrieves query-key attention probabilities during the forward pass. Under fused kernels such as FlashAttention-2, which avoid materializing attention weights in memory, the pruning layer evaluates auxiliary query-key dot products with rotary positional embeddings and causal masking, normalizes the attention probabilities across the complete key sequence through softmax, and extracts the attention weights corresponding to visual token positions. This auxiliary evaluation produces the exact attention distributions required for token selection without computing value-weighted outputs or interfering with the primary model execution path.

Layers 0 through ℓ retain their complete visual KV entries during the current request. Decode attention at the decision layer therefore remains defined at all original visual positions. Only tokens actually forwarded during generation contribute attention evidence; no extra pass is added to obtain a final response token's attention. A missing history, incompatible visual grid, or invalid required semantic summary invokes the same fixed budget selector with an entirely dynamic group. New

tasks start with empty temporal state. The default policy always applies its configured retention budget, including at initialization.

C Benchmarks and Evaluation Protocols

Table 7: Evaluation units and temporal context. All protocols supply a single current screenshot to each model request. Reference history denotes recorded actions, rather than predictions inserted into later prompts.

Benchmark	Evaluation unit	Textual context	GUI-Drop state
AndroidWorld	116 tasks; task success	Goal and executed history	Within each task
AndroidControl	500 targets per track; step accuracy	Goal and reference history; step instruction for Low	One predecessor and target
AITW	4,663 steps in 585 episodes; action accuracy	Goal and recent reference actions	Within each episode
AgentNetBench	1,208 steps in 100 trajectories; average action score	Goal and reference instructions and actions	Within each trajectory

C.1 AndroidWorld

Online evaluation is conducted on AndroidWorld (Rawles et al., 2025), an interactive mobile environment simulating real-world user interactions across 20 Android applications, including system utilities and daily apps such as Contacts, Calendar, Settings, Files, and messaging. We evaluate on the full benchmark suite of 116 realistic tasks, using the fixed task seed 20260614 to ensure deterministic environment initialization and exact reproducibility. Each task starts from an independent session with a clean environment state. The agent interacts with the operating system through executable actions dispatched via ADB, with subsequent screenshots capturing the resulting interface state transitions. The maximum interaction limit is set to $\lfloor 10c \rfloor$ steps for a task with complexity score c, defaulting to 10 steps when complexity is unspecified. A task is deemed successful only when accepted by the environment evaluator via automated programmatic checks on device databases, file systems, or UI component states. Overall success rate is defined as the number of successfully completed tasks divided by 116. GUI-Drop maintains its auxiliary temporal representations across steps within each task session, clearing all temporal state between distinct tasks.

C.2 AndroidControl

We evaluate on the standardized AndroidControl benchmark subset established by UGround (Gou et al., 2025) and hosted on Hugging Face,[1] without resampling. AndroidControl comprises extensive human demonstrations across diverse Android applications. Following the evaluation protocol in UGround and GUI-KV, we evaluate on the fixed 500-step test instances across two distinct instruction settings, namely goal-level instructions (High) providing the overall task objective and reference action history, and step-level instructions (Low) which additionally supply the low-level step directive. Predictions are scored offline without execution, with step accuracy reported over all 500 targets per track.

The evaluation protocol matches click and long press actions against the smallest accessibility hierarchy element bounding the ground-truth target coordinate. A predicted action is deemed correct if its coordinate falls within this annotated bounding box. Accessibility information serves exclusively for offline evaluation and remains unavailable during model inference. Text entry requires an exact string match. Non-spatial actions are validated against reference parameters, allowing functional equivalences such as opening an application or navigating back through designated clickable controls. Malformed or unrecognized predictions are treated as incorrect.

[1] https://huggingface.co/datasets/demisama/UGround-Offline-Evaluation

Temporal methods use an isolated session for each target. GUI-Drop processes one predecessor, when available, to initialize attention state. VLA-Pruner processes up to three available predecessors with its history policy described in Appendix D.2. Each initialization request receives its own reference screenshot and textual context. The target still receives its reference textual history. Only the target is scored, after which the session is destroyed. Initial episode steps start without history, and stateless methods process only the target. Initialization requests and their costs are recorded separately from scored targets.

C.3 Android in the Wild

We evaluate on the widely adopted `HongxinLi/AITW_test` benchmark split hosted on Hugging Face,[2] which extracts a standardized evaluation suite from Android in the Wild spanning all five canonical task domains, namely General, GoogleApps, Install, Single, and WebShopping. The benchmark comprises 4663 interaction steps across 585 recorded episodes. Each input instance supplies the high-level goal, the current reference screenshot observation, and up to six recent historical reference actions after filtering adjacent duplicate actions. Episodes are evaluated sequentially with isolated internal state. Model predictions update the auxiliary attention history of GUI-Drop, while subsequent steps proceed along the ground-truth recorded trajectory.

Accuracy pools correct actions across the five application groups and divides by 4,663. The official matcher compares action types for nonpoint actions, including TYPE. It distinguishes taps from swipes using touch and release distance. Taps match within a common augmented annotation box or a normalized coordinate distance of at most 0.14; swipes match by principal axis. Invalid predictions are incorrect. Resuming an episode replays its completed prefix to reconstruct state without repeating scores.

C.4 AgentNetBench

We evaluate on the official `xlangai/AgentNetBench` benchmark suite hosted on Hugging Face,[3] containing 100 desktop workflow trajectories and 1208 steps. Qwen2.5-VL-7B receives the current screenshot observation and overall workflow goal. The four most recent interaction steps provide reference actions and low-level step instructions, whereas preceding steps supply instructions only. Current and future actions are excluded from the context. Spatial coordinates are formatted relative to the respective image resolutions, and the tool specification reflects current display dimensions.

Preprocessing aligns image dimensions to multiples of 28, constrains overall image area to $[56^2, 14^2 \cdot 4 \cdot 2560]$ pixels, and limits maximum edge length to 8192 pixels. Desktop action predictions output screen pixel coordinates, normalized during evaluation. Scroll operations specify signed wheel increments. Trajectories are evaluated sequentially with isolated state.

The standardized OpenCUA evaluation suite[4] computes a continuous similarity score in $[0, 1]$ for each step based on mouse Euclidean distance, text character edit distance, and scroll magnitude agreement. The evaluator considers multiple valid reference annotations and applies standard penalties for redundant actions. Unparseable actions receive zero reward. Overall performance corresponds to the average score scaled to a percentage over all 1208 evaluation steps.

D Baseline and Ablation Configurations

D.1 Shared Inference Setting

The pruning policies are integrated into a shared inference runtime. Comparisons within a model use its native prompt template, action format, and benchmark scorer. All policies preserve text tokens and operate on the current screenshot's visual tokens. The budget $K = \mathrm{round}(RN)$ is defined after spatial merging. FastV, VLA-Pruner, and GUI-Drop shorten the sequence after the decision layer; MMTok, DivPrune, and ZOO-Prune select before the language decoder. These locations are

[2] `https://huggingface.co/datasets/HongxinLi/AITW_test`
[3] `https://huggingface.co/datasets/xlangai/AgentNetBench`
[4] `https://github.com/xlang-ai/OpenCUA`

explicit because equal retained visual token counts do not imply identical propagation lengths across decoder layers.

For Qwen3-VL, selection is also applied to the corresponding DeepStack visual features when they remain to be injected. Selected tokens retain their original visual order and positional coordinates. The reference baseline settings are listed in Table 8; R denotes the budget variable shared with the main comparisons.

Table 8: Reference baseline configurations. Before decoder denotes selection after visual encoding and before the first language layer. The decision layer ℓ follows Table 5.

Method	Pruning location	Selection configuration
FastV	After layer ℓ	Trailing query attention; query window 1; retain K visual tokens
MMTok	Before decoder	Text to vision temperature 0.01; vision to vision temperature 0.2; visual coverage weight 0.5
DivPrune	Before decoder	Cosine feature distance; greedy maximum minimum diversity; retain K tokens
ZOO-Prune	Before decoder	64 perturbation directions; scale 0.01; seed 0; sensitivity aware diversity selection
VLA-Pruner	After layer ℓ	Current semantic and historical action Top-K union; cosine maximum minimum filtering; three action histories with decay 0.8; action attention at layer 15

MMTok constructs coverage scores from text and visual embeddings. Its coverage text excludes serialized historical actions, while the full textual history remains in the language model prompt. The selector uses the stated text and vision temperatures and greedily fills the visual budget. DivPrune selects a visually diverse subset using pairwise cosine distance. ZOO-Prune estimates token sensitivity by perturbing the inputs to the visual merger and combines normalized sensitivity with feature diversity. All three methods gather the selected embeddings before language decoding. Their scoring and selection computations are part of the model serving path.

D.2 VLA-Pruner Adaptation to GUI Actions

We adapt VLA-Pruner (Liu et al., 2025) to the backbone's native GUI action output. Current semantic scores average attention over prefill queries and heads at layer ℓ. Historical action scores are collected at layer 15 from the parsed action and parameter spans. The three most recent valid action summaries receive weights 1, 0.8, and 0.64, from newest to oldest, normalized by the sum of available weights. The selector takes the union of the semantic Top-K and historical action Top-K sets. When the union exceeds K, cosine maximum minimum filtering on the original visual embeddings reduces it to K tokens. Selected positions are then restored to their original sequence order. With no action history, selection uses the current semantic Top-K alone.

AndroidWorld and AITW accumulate three valid action summaries within the current task or episode before enabling pruning. Requests during this initialization period retain all visual tokens and collect action attention; later requests use the experiment's retention ratio. AndroidControl uses up to three predecessor requests and prunes with the history actually available, including semantic selection when no predecessor is available. The retention ratio is therefore independent of the benchmark's initialization policy.

Historical action summaries come from generated predictions with valid parsing and attention, without filtering by reference correctness. An invalid prediction is still scored according to the benchmark protocol but is not appended to the action history. Attention collected after pruning is mapped back to the original visual grid, assigning zero to removed positions. Temporal state is cleared between tasks, episodes, or isolated AndroidControl targets.

D.3 Definitions of the Component Ablations

Table 9 specifies the interventions behind Table 3. Each variant uses Step-GUI-4B, AndroidControl High, and a 20% visual retention budget. The complete method retains its dynamic and static parti-

tion, three static relevance components, anchor conditioned diversity, and four neighbor aggregation. The variants change the component indicated below.

Table 9: Configuration changes for the component ablations in the main paper. Each variant retains the same 20% visual token budget.

Variant	Change relative to GUI-Drop
Without cross step relevance	Disable both preceding decode attention and the positive prefill attention share change. Static relevance uses current attention; patch based partition and dynamic priority remain active.
Without dynamic priority	Set $\lambda = 1$, retaining the dynamic and static groups and their relevance definitions.
Without diversity selection	Set $\gamma = 1$, filling both quotas entirely with relevance anchors. Local aggregation remains enabled.
Without local aggregation	Pass the selected hidden states forward without the update in Equation 4. Selection and budget remain unchanged.

E Efficiency Measurement

E.1 Measurement Scope and Aggregation

Table 4 profiles UI-Voyager on NVIDIA A100-SXM4-80GB GPUs with two independent replicas, BF16 inference, and eager attention. Times are means over model requests completed without request errors; success rates follow the main benchmark results. Speedups divide the Vanilla mean by the method mean. Operation counts and KV capacity are analytical estimates.

Prefill time. Dedicated CUDA events bracket the decoder layer loop, including attention, feedforward computation, DeepStack injections, pruning, and aggregation. Vision encoding, final normalization, and the output projection lie outside this interval. The table uses this decoder measurement rather than a generic total prefill field.

Decode latency. For each request, autoregressive model time is divided by the number of decode forward steps, and these ratios are subsequently averaged across requests. The initial generated token is emitted during the prefill phase and is excluded from the decode forward count.

Total time. Total Time measures the server model call for one action prediction, including input preparation, visual encoding, prefill, generation, and method state handling. It excludes client communication and environment action execution, and is distinct from the wall clock time for a task collection.

E.2 Decoder Operation Count

We estimate dense matrix operations using the actual query and key lengths at each decoder layer. Let q and k denote these lengths, d the hidden width, a the total query head width, d_{KV} the grouped KV width, and m the intermediate feedforward width. Counting a multiplication and an addition as two operations gives

$$F(q,k) = 2\left(2qda + 2qdd_{\mathrm{KV}} + 2qka + 3qdm\right). \tag{5}$$

The terms account for query and output projections, key and value projections, the two attention products, and the three gated feedforward projections. UI-Voyager has $d = 2560$, $a = 4096$, $d_{\mathrm{KV}} = 1024$, $m = 9728$, and $L = 36$. We sum $F(q_j, k_j)$ over layers per request, then average within each method. Relative reductions use these method means.

Vanilla processes the complete prompt at every layer. GUI-Drop and FastV retain full lengths through layer ℓ and use recorded compact query and key lengths thereafter, preserving all text tokens. The estimate covers decoder prefill matrix operations. Visual encoding, selection, aggregation, output generation, the output head, and elementwise operations remain outside this analytical scope, while their executed costs belong to the applicable measured intervals.

E.3 KV Cache Capacity

KV Cache denotes the tensor payload immediately after prefill, reconstructed from all decoder key lengths. For element size b bytes,

$$M_{\mathrm{KV}} = \frac{2b\, d_{\mathrm{KV}}}{2^{20}} \sum_{j=0}^{L-1} k_j \quad \text{MiB}. \tag{6}$$

The factor two accounts for keys and values. UI-Voyager uses $b = 2$ and $d_{\mathrm{KV}} = 8 \times 128$. This includes text and retained visual entries, with full visual KV in layers 0 through ℓ. Model weights, later generated tokens, and auxiliary temporal state are excluded. The table averages this capacity per request; it does not measure process peak memory.

F Additional Results

F.1 AndroidWorld Results across Backbones

Table 10 provides the absolute AndroidWorld scores underlying Figure 3, using 40% visual retention and 116 tasks per backbone. GUI-Drop leads the listed pruning methods on all five backbones. Vanilla provides the unpruned reference.

Table 10: AndroidWorld success rate (%) at 40% visual retention. Bold marks the best pruning result within each backbone. Vanilla retains all visual tokens.

	Qwen3-VL			Qwen2.5-VL	
Method	2B	4B	8B	7B	32B
Vanilla	29.3	44.8	43.1	23.3	35.3
FastV	25.0	33.6	29.3	17.2	22.4
ZOO-Prune	18.1	12.9	35.3	14.7	15.5
MMTok	21.6	31.9	43.1	20.7	21.6
GUI-Drop	**27.6**	**38.8**	**46.6**	**22.4**	**29.3**

F.2 AITW Results across Backbones

Table 11 reports the absolute action accuracies underlying Figure 3(b), using the AITW evaluation protocol in Appendix C. GUI-Drop achieves the highest average score relative to Vanilla among the listed pruning methods. This average gives equal weight to each backbone and is computed as

$$\mathrm{RelativeMean}(p) = \frac{100}{5} \sum_{m=1}^{5} \frac{\mathrm{Score}(p, m)}{\mathrm{Score}(\mathrm{Vanilla}, m)}, \tag{7}$$

where p denotes a pruning method and m a backbone. The same definition applies to both panels of Figure 3. We compute ratios before rounding the displayed scores.

Table 11: AITW action accuracy (%) at 40% visual retention. The last column is the mean score relative to Vanilla, defined in Equation 7. Bold marks the best pruning result in each column.

	Qwen3-VL			Qwen2.5-VL		Relative
Method	2B	4B	8B	7B	32B	mean (%)
Vanilla	55.5	57.4	56.9	51.9	57.6	100.0
FastV	51.2	51.4	52.5	45.3	50.7	89.9
ZOO-Prune	51.8	50.1	50.8	41.3	43.9	85.1
MMTok	**52.5**	**54.1**	52.7	46.4	49.1	91.3
GUI-Drop	51.9	53.8	**53.6**	**46.6**	**51.4**	**92.1**

G CASE STUDIES

We present four recorded GUI interactions covering complementary cues, calendar grounding, state transition aware allocation, and action feedback. The expense, calendar, and messaging cases show the complete current screenshot, its recorded GUI-Drop selection, and the complete next screenshot. The file case instead follows the preceding action into the current observation, its dynamic and static partition, and its token allocation. All cases preserve the full interface and its spatial context.

All four cases use UI-Voyager with the complete GUI-Drop configuration at 20% visual retention, keeping 510 of 2,550 tokens. Every screenshot panel displays its full 1080×2400 source image, with a common scale within each figure. In the selection panels, red denotes dynamic positions and blue denotes static positions. Dark, medium, and pale shades respectively denote anchors, carriers, and pruned positions. These overlays identify selected positions before local aggregation. Observation panels contain the original screenshots without overlays.

G.1 EXPENSE IDENTITY AND A SMALL DELETION CONTROL

The task requests deletion of the Accessories expense. The current interface combines a weekly expense overview with an expanded detail card at the bottom. That card contains the item name, amount, date, note, and a small trash icon, so the action must associate the requested identity with its control within the broader page.

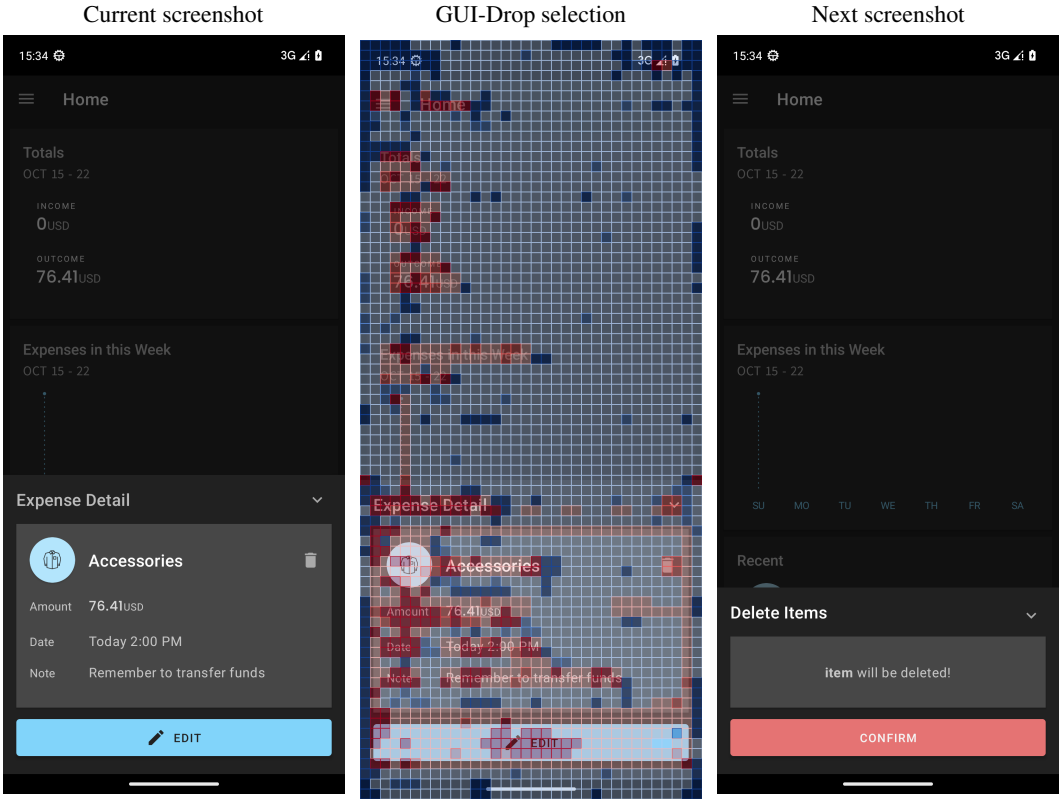


Figure 6: Expense deletion at 20% retention. The complete screen shows the target detail card below the weekly overview. The recorded click on its trash icon opens the deletion confirmation panel shown on the right.

Complementary identity and control cues. Anchors and carriers jointly cover positions in the Accessories name and amount regions. The deletion icon is covered by two carriers, while the annotated icon region contains no anchors. The full selection map places these retained positions in the context of the surrounding card and the rest of the dashboard. This provides a concrete example of carriers supplementing the spatial coverage of the relevance anchors at an actionable control.

Recorded action and subsequent state. The model clicks the trash icon beside Accessories. The next screenshot replaces the detail card with a Delete Items panel and a CONFIRM button. The following recorded action confirms the deletion, and the environment evaluator accepts the completed task. The two screenshots expose both the object associated with the click and the interface transition that follows it.

G.2 Calendar Title, Time, and Event Marker

The task requests deletion of Appointment for Project X on October 18, 2023 at 17:00. The agenda displays multiple competing entries. Identifying the requested event therefore requires resolving its title and scheduled time alongside the visual grouping that separates its card from neighboring events.

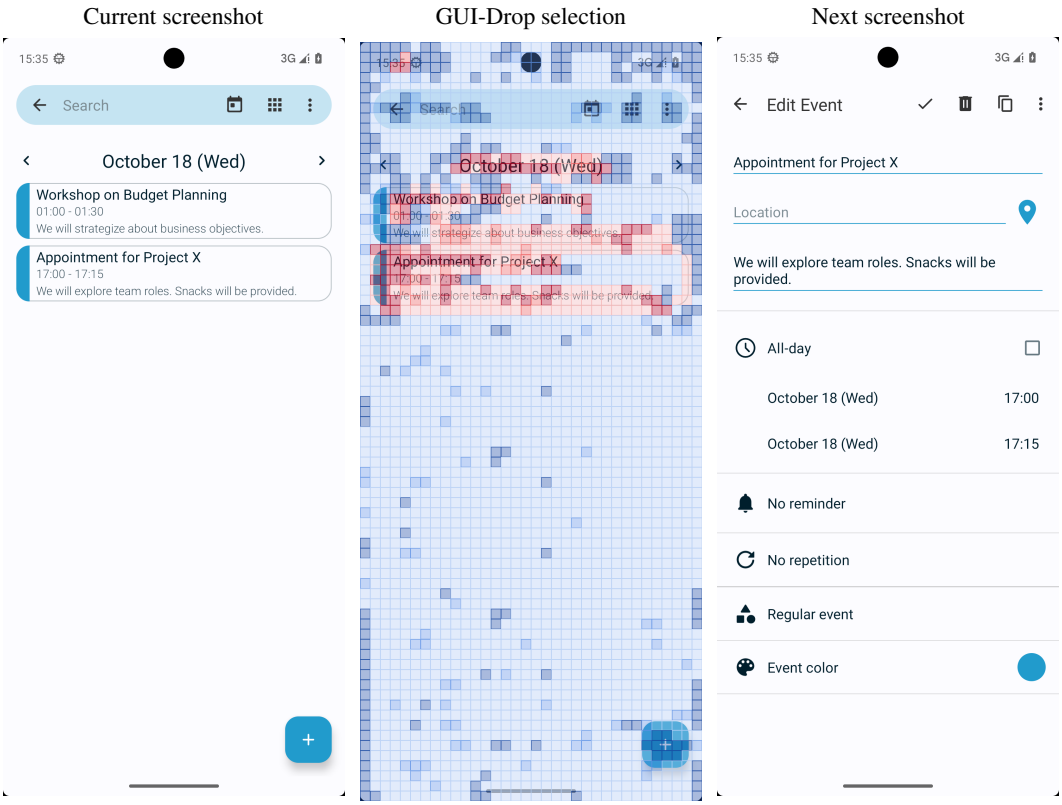


Figure 7: Selecting the requested calendar event. The full agenda shows the date and neighboring entry. Clicking Appointment for Project X opens the Edit Event screen, which displays the matching title, date, and time.

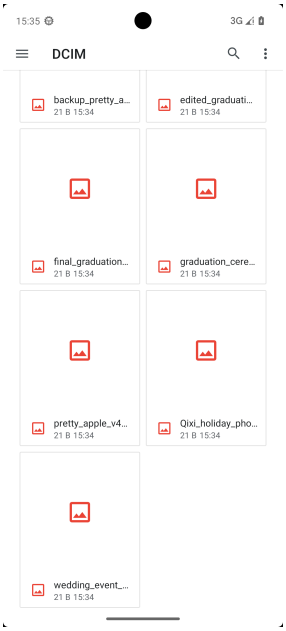
Multiple cues within one event. The retained positions cover the target title, its 17:00 to 17:15 interval, and the distinctive blue event marker. Across the corresponding interface regions, GUI-Drop retains 23, 10, and 8 token centers, whereas FastV retains 15, 8, and 5 tokens on the identical screenshot under the same 510 token budget. Both methods identify the relevant visual elements, while GUI-Drop preserves denser token coverage within each functional region. The complete screenshot illustrates the spatial positioning of these cues relative to the calendar header and adjacent event entries.

Recorded action and subsequent state. The model clicks the requested event. The next observation displays its title and scheduled interval in the editing form, together with the deletion control in the top toolbar. This transition connects the grouped cues in the agenda to the selected event's editable attributes. The recorded trajectory then deletes the event and completes successfully.

G.3 State Transition and Token Allocation

The task requests deletion of `pretty_apple_v4RL.jpg` from DCIM. Figure 8 follows the long press that selects the file into the next pruning decision.

Previous observation X_{t-1}



Recorded action a_{t-1}

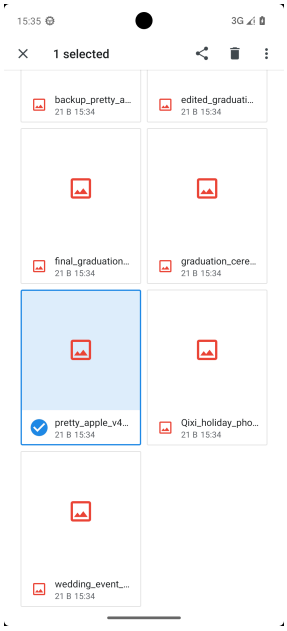
Long press the requested file

`pretty_apple_v4RL.jpg`

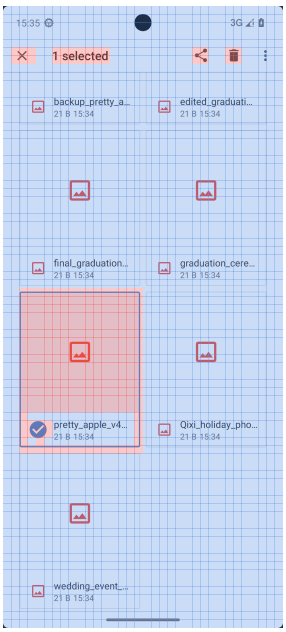
$$X_{t-1} \longrightarrow X_t$$

The selected card gains a checkmark and highlighting. The top toolbar changes to selection mode and exposes a deletion control.

Current observation X_t



Dynamic and static partition



Recorded token allocation

Group	N	K	K/N
Dynamic	283	121	42.8%
Static	2,267	389	17.2%

Dynamic positions occupy 11.1% of the grid and receive 23.7% of the retained budget under $\lambda = 2.5$.

The 389 retained static positions use the relevance rule in Equation 2. Both groups then select anchors and carriers.

$$K_{\mathcal{D}} + K_{\mathcal{S}} = 510$$

Current token selection

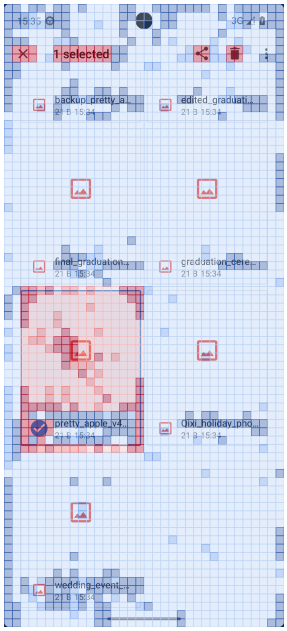


Figure 8: A recorded state transition and its allocation at 20% retention. All four panels preserve the complete screen. The partition uses red for dynamic positions and blue for static positions; the final selection additionally distinguishes anchors, carriers, and pruned tokens by shade. Counts and quotas come from the current step's trace. The top row shows the state transition used to compute the partition and selection in the bottom row.

G.4 From a Composed Message to a Sent Message

The task asks the agent to reply to the specified contact with “A quick brown fox.” The current screen shows the conversation, the composed reply, the Send control, and the keyboard. Completing the step requires associating the text in the composer with the adjacent action control.

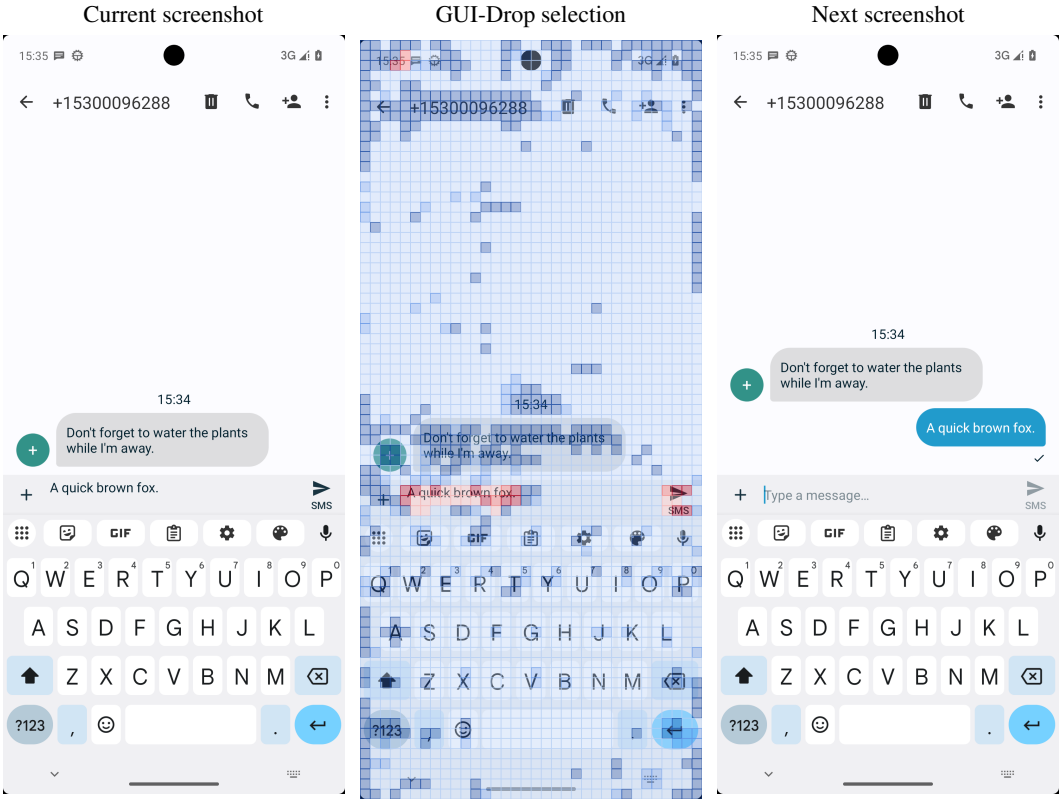


Figure 9: Sending the requested reply. The complete screens preserve the conversation and keyboard context. After the recorded click on Send, the reply appears as an outgoing bubble with a checkmark, and the composer becomes empty.

Content and an adjacent action control. Retained positions cover parts of the composed message region and the paper plane Send control. The complete selection map also exposes their location relative to the earlier incoming message and the keyboard. These are distinct regions with different roles in the interaction. The composed text supplies the action content, while the adjacent control triggers its submission.

Recorded action and subsequent state. The model clicks Send. The next observation shows the same reply in an outgoing message bubble, a checkmark below it, and the emptied composer. The agent then terminates successfully. This transition makes the change from editable input to submitted content visible within the same conversation, while retaining the full interface context used for the decision.