SAVE: SUFFICIENCY-AWARE VISUAL EVIDENCE ACQUISITION FOR MULTIMODAL REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Visual tools allow multimodal large language models to acquire additional visual evidence during reasoning, but effective tool use requires judging whether the available evidence is sufficient. Existing models may miss useful inspection or continue interacting after sufficient evidence has already been obtained. We therefore formulate adaptive visual tool use as an evidence sufficiency problem with two decisions: whether to acquire additional evidence and when to stop. To learn these decisions from complete-trajectory feedback, we introduce **SAVE**, a **S**ufficiency-aware framework for **A**daptive **V**isual **E**vidence acquisition. For each question, SAVE samples multiple on-policy trajectories and constructs a hierarchical reward at two comparison scopes. Across direct-answer and tool-assisted strategies, it strengthens successful trajectories when observed success occurs in only one strategy; within successful tool-assisted trajectories, it favors the minimum observed successful acquisition depth. This design learns invocation and stopping preferences without estimating the marginal utility of individual tool calls or imposing a uniform per-call cost. Across three vision-language backbones and five multimodal reasoning benchmarks, SAVE improves Avg@3 over supervised fine-tuning by 8.33 percentage points on average while reducing the macro-average number of calls per tool-using trajectory by 8.6%.

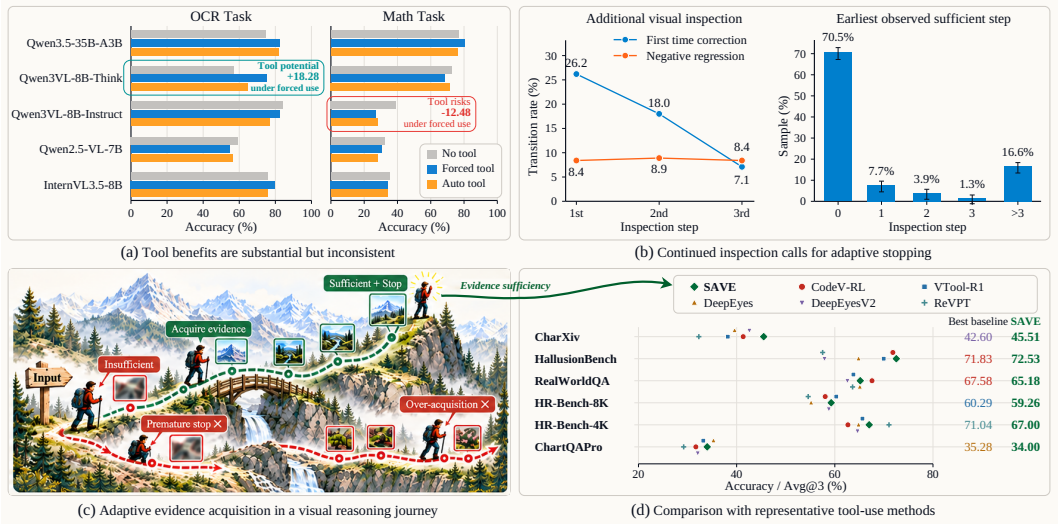
1 INTRODUCTION

Multimodal large language models (MLLMs) have achieved strong performance across visual question answering, document understanding, and chart reasoning Bai et al. (2025b); Wang et al. (2025). However, many challenging instances depend on fine-grained visual information that cannot be reliably resolved from a single global observation. Visual tool-augmented approaches address this limitation by allowing models to actively acquire additional evidence through operations such as cropping, grounding, and local enhancement Su et al. (2025); Zheng et al. (2026); Wu et al. (2026). Yet access to visual tools does not by itself determine how they should be used. The model must decide whether the currently available evidence is sufficient: if not, additional evidence should be acquired; once it is sufficient, further acquisition should stop. Effective visual tool use involves two coupled decisions: whether to acquire additional evidence and when to stop.

We first ask whether existing MLLMs can make these sufficiency decisions reliably. The answer is mixed in both directions. For tool invocation, Figure 1(a) shows that forced invocation can either improve or reduce accuracy depending on the model and task, while autonomous invocation does not consistently outperform the fixed settings. This suggests that current models do not reliably identify when additional evidence is needed. The stopping decision is similarly non-trivial. As shown in Figure 1(b), the rate of first-time corrections decreases over successive inspections, while negative regressions persist; meanwhile, the earliest observed sufficient step varies substantially across samples. Thus, useful acquisition depth is instance-dependent: some questions benefit from further inspection, whereas others should stop earlier to avoid redundant or harmful interaction.

These observations expose a learning challenge: how can a policy learn both invocation and stopping decisions from complete-trajectory feedback? Final-answer correctness provides only a coarse outcome signal. It can identify successful trajectories, but it does not distinguish whether a correct answer was obtained with or without additional visual evidence, nor whether a successful tool-assisted trajectory continued longer than another successful alternative. A natural remedy is to penalize every

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.



(a) Tool benefits are substantial but inconsistent

(b) Continued inspection calls for adaptive stopping

(c) Adaptive evidence acquisition in a visual reasoning journey

(d) Comparison with representative tool-use methods

Figure 1: Evidence sufficiency in adaptive visual tool use: motivation, formulation, and performance. (a) Accuracy of five vision-language models on OCRBench and MathVerse. No-tool uses only the original image, Forced-tool requires at least one visual crop, and Auto-tool allows autonomous tool invocation. The mixed effects of forced and autonomous inspection show that additional visual evidence is not uniformly beneficial. (b) First-time correction and negative regression rates across inspections (left), and the distribution of earliest observed sufficient step (right), motivating adaptive stopping. (c) Conceptual illustration of adaptive evidence acquisition: acquire additional evidence when the current observation is insufficient and stop once sufficient, avoiding premature stopping and redundant acquisition. (d) Comparison of SAVE with the strongest competing method on each benchmark among the representative visual tool-use baselines, using Qwen2.5-VL-7B.

tool call, but a uniform per-call cost introduces a different bias: among equally correct trajectories, it always prefers a direct trajectory over a tool-assisted one. A single observed direct success, however, is not sufficient to justify penalizing all successful tool-assisted trajectories for the same question. These two decisions therefore call for different comparison scopes: strategy-level comparison for whether additional evidence should be acquired, and within-tool comparison for how long acquisition should continue.

To instantiate this principle, we propose **SAVE**, a **S**ufficiency-aware framework for **A**daptive **V**isual **E**vidence acquisition. SAVE treats visual tool use as an evidence sufficiency problem and learns invocation and stopping preferences from complete-trajectory feedback. As illustrated in Figure 1(c), the desired behavior is to acquire additional evidence when the current observation is insufficient and to stop once sufficient evidence has been obtained. To realize this behavior, SAVE samples multiple on-policy trajectories for the same question and applies the two decisions at different comparison scopes. Across direct-answer and tool-assisted strategies, it strengthens successful trajectories when both strategies are sampled but observed success occurs in only one. Within successful tool-assisted trajectories, it instead favors the minimum observed successful acquisition depth, encouraging the policy to stop rather than continue along longer successful alternatives. SAVE combines these two comparisons with final-answer correctness in a hierarchical trajectory reward and trains the policy using REINFORCE Leave-One-Out.

Across three vision-language backbones and multimodal reasoning benchmarks, SAVE improves Avg@3 over supervised fine-tuning by 8.33 percentage points on average and reduces the macro-average number of calls per tool-using trajectory by 8.6%. As summarized in Figure 1(d), SAVE achieves the highest macro-average Avg@3 among the compared visual tool-use methods, ranking first on two benchmarks and second on the remaining four.

Our main contributions are summarized as follows:

- We formulate adaptive visual tool use as an evidence sufficiency problem with two instance-dependent decisions: whether additional evidence is needed and when acquisition should stop.
- We introduce SAVE, which constructs a hierarchical trajectory reward from on-policy samples for the same question, combining cross-strategy success comparison with a minimum-depth preference within successful tool-assisted trajectories.
- We evaluate SAVE across three vision-language backbones and five multimodal reasoning benchmarks, demonstrating broad improvements over supervised fine-tuning and providing ablation and controlled behavioral evidence for adaptive invocation and stopping.

2 Related Work

Visual Tool-Augmented Multimodal Reasoning. Visual tool-augmented models acquire additional evidence through grounding, cropping, retrieval, and program execution. DeepEyes, OpenThinkIMG, and VTool-R1 use reinforcement learning to extend visual reasoning beyond static perception Zheng et al. (2026); Su et al. (2025); Wu et al. (2026). ReVPT and ToolsRL explore broader visual operation spaces and structured tool-use trajectories Zhou et al. (2025); Dong et al. (2026). ACTIVE-O3 formulates active perception as reinforcement learning and uses GRPO for adaptive region selection Zhu et al. (2026b). Other approaches compose visual operations through generated programs or structured actions Song et al. (2026b); Hou et al. (2026).

Reinforcement Learning for Tool Use. Recent work structures tool-use optimization through staged training and specialized objectives. DeepEyesV2 introduces a cold-start stage before reinforcement learning, while ToolsRL separates tool capability acquisition from answer optimization Hong et al. (2026); Dong et al. (2026). AdaReasoner and SenseNova-MARS explore explicit objectives for tool orchestration Song et al. (2026a); Chng et al. (2026). SPyCE jointly evolves reusable visual skills and the policy during reinforcement learning, and studies call-count regularization to control unnecessarily long tool-use trajectories Zhang & Qiu (2026).

Adaptive Tool Invocation and Evidence Sufficiency. VisionThink learns to request higher-resolution images when needed, while CADER uses confidence estimates to guide additional tool-assisted evidence acquisition for long-video reasoning Yang et al. (2025; 2026). AdaTooler-V scales tool-use rewards using an offline Tool Benefit Score Wang et al. (2026a). Beacon conditions its adaptive reward on whether a rollout group contains a correct tool-free response, favoring direct reasoning when it succeeds Wang et al. (2026c). BEE penalizes ineffective dependence on implicit visual tools and encourages invocation only when internal model knowledge is insufficient Chen et al. (2026). Other approaches improve tool-use reliability through interaction-level credit assignment or evidence verification Feng et al. (2026); Hou et al. (2026); Wang et al. (2026b).

3 Method

3.1 Problem Setup

Let $x_i = (I_i, q_i)$ denote the i-th multimodal QA instance, where I_i is the input image, q_i is the question, and $y_i^\star$ is the reference answer. The actor policy π_θ, parameterized by the model parameters θ, may answer without tools or acquire additional visual evidence before answering. We use the superscripts D and T to distinguish direct-answer and tool-assisted strategies, respectively, and denote corresponding trajectories by τ_i^D and τ_i^T. Omitting intermediate reasoning, their schematic representations are

$$\tau_i^D = (\hat{y}_i^D), \qquad \tau_i^T = \left(\{(o_{i,t}, e_{i,t})\}_{t=1}^{K_i}, \hat{y}_i^T\right). \tag{1}$$

Here, $\hat{y}_i^D$ and $\hat{y}_i^T$ are the predicted final answers. In τ_i^T, $t = 1, \ldots, K_i$ indexes successive tool interactions, $o_{i,t}$ is the t-th tool operation, and $e_{i,t}$ is its returned visual evidence. The quantity $K_i > 0$ counts tool calls in this particular trajectory.

For multiple actor rollouts of the same instance, $\tau_{i,j}$ denotes the j-th trajectory sampled by π_θ for x_i; the rollout index j is suppressed in the schematic notation above. Let $c_{i,j}$ be its tool-call count, which is zero for direct-answer trajectories and positive for tool-assisted trajectories.

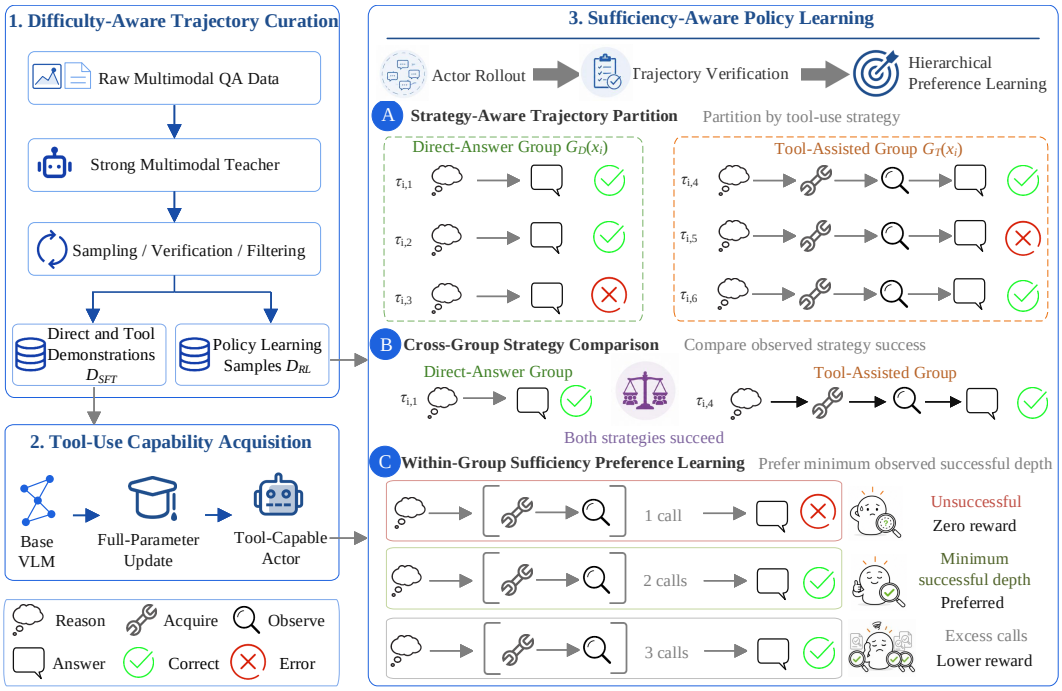


Figure 2: Overview of SAVE. Teacher-generated trajectories support difficulty-aware data curation and tool-use capability initialization. During on-policy learning, SAVE compares observed success across direct-answer and tool-assisted strategies and learns a within-tool preference against excess acquisition beyond the minimum observed successful tool depth. The policy is optimized with RLOO using the resulting trajectory rewards.

Let $z_{i,j} \in \{0, 1\}$ indicate trajectory success: $z_{i,j} = 1$ when the trajectory is valid, unmasked, and its final answer is judged correct against $y_i^\star$; otherwise, $z_{i,j} = 0$. These outcomes and counts provide complete-trajectory feedback, not labels of which individual calls were necessary. SAVE therefore uses alternative trajectories for the same question as empirical references for learning invocation and stopping preferences.

3.2 SAVE Training Initialization

Such comparisons require initial behaviors from which direct-answer and tool-assisted alternatives can be sampled. As shown in Figure 2, teacher-generated trajectories serve two preparatory roles: selecting candidate RL instances by empirical difficulty and initializing these behaviors. The comparison signals themselves are subsequently constructed from the actor's on-policy rollouts.

Difficulty-aware trajectory curation. To focus policy learning on questions not consistently solved by the teacher, we use repeated teacher outcomes as an empirical difficulty signal. A multimodal teacher generates $M = 3$ candidate trajectories per instance, with final predictions $\{\hat{y}_{i,k}\}_{k=1}^{M}$. We define the correctness indicator for each trajectory as

$$u_{i,k} = \mathbb{I}\left[J(I_i, q_i, y_i^\star, \hat{y}_{i,k}) = 1\right], \tag{2}$$

where $J(\cdot)$ is the answer-correctness evaluator, $\mathbb{I}[\cdot]$ is the indicator function, and k indexes teacher-sampled trajectories. The indicator $u_{i,k}$ equals one if the k-th teacher trajectory for x_i has a correct final answer, and zero otherwise. Let $n^+(x_i) \in \{0, \ldots, M\}$ denote the number of these trajectories with correct final answers. We assign the difficulty category

$$d(x_i) = \text{easy if } n^+(x_i) = M, \quad \text{medium if } 0 < n^+(x_i) < M, \quad \text{hard if } n^+(x_i) = 0. \tag{3}$$

Let $\mathcal{D}_{\text{easy}}$, $\mathcal{D}_{\text{medium}}$, and $\mathcal{D}_{\text{hard}}$ denote the corresponding instance sets. Easy instances are excluded. Hard instances are retained because on-policy exploration may discover successful behaviors absent from the teacher samples. After removing invalid instances, the dataset is $\mathcal{D}_{\text{RL}} = \mathcal{D}_{\text{medium}} \cup \mathcal{D}_{\text{hard}}$.

Tool-Use Capability Acquisition. To establish both strategies as alternatives for subsequent exploration, supervised fine-tuning initializes direct-answer and tool-assisted behaviors using verified teacher demonstrations of the types in Equation (1). Difficulty-based filtering applies to RL instances, not to these supervised demonstrations.

The supervised dataset contains both trajectory types, and the actor is initialized by minimizing the SFT loss:

$$\mathcal{D}_{\mathrm{SFT}} = \{(x_i, \tau_i) \mid \tau_i \in \{\tau_i^D, \tau_i^T\}\}, \quad \mathcal{L}_{\mathrm{SFT}}(\theta) = -\mathbb{E}\Big[\sum\nolimits_{\ell} \log \pi_\theta(\tau_\ell \mid x, \tau_{<\ell})\Big], \tag{4}$$

where $\mathbb{E}$ averages over input–demonstration pairs $(x, \tau) \sim \mathcal{D}_{\mathrm{SFT}}$, and π_θ is the actor policy with parameters θ. The sum runs over assistant-generated token positions ℓ, with τ_ℓ denoting the target token and $\tau_{<\ell}$ the preceding interaction context, including tool observations. This stage provides initial behaviors for subsequent on-policy comparison.

3.3 Sufficiency-Aware Policy Learning

Initialization provides behaviors to explore; the remaining objective is to learn question-specific invocation and stopping preferences. A uniform per-call penalty would favor a successful direct trajectory over every successful tool-assisted alternative, even though a single observed direct success does not establish reliable tool-free answering. SAVE therefore compares observed strategy success to guide invocation and successful tool-assisted depths to guide stopping. Appendix B provides a complementary cost–benefit interpretation of this separation.

Strategy-Aware Trajectory Partition. For each instance $x_i \in \mathcal{D}_{\mathrm{RL}}$, the actor samples $N = 16$ on-policy trajectories:

$$\mathcal{T}(x_i) = \{\tau_{i,j}\}_{j=1}^{N}. \tag{5}$$

Let $v_{i,j} \in \{0, 1\}$ indicate whether $\tau_{i,j}$ is eligible for reward comparison: $v_{i,j} = 1$ if the trajectory passes the implementation's validity checks and is not masked, and $v_{i,j} = 0$ otherwise. Eligibility is distinct from correctness: an eligible trajectory may have an incorrect final answer, whereas $v_{i,j} = 0$ implies $z_{i,j} = 0$. Eligible trajectories are partitioned by whether they invoke a visual tool:

$$\mathcal{G}_D(x_i) = \{\tau_{i,j} \in \mathcal{T}(x_i) \mid v_{i,j} = 1,\ c_{i,j} = 0\}, \mathcal{G}_T(x_i) = \{\tau_{i,j} \in \mathcal{T}(x_i) \mid v_{i,j} = 1,\ c_{i,j} > 0\}. \tag{6}$$

Here, $\mathcal{G}_D(x_i)$ and $\mathcal{G}_T(x_i)$ contain eligible direct-answer and tool-assisted trajectories, respectively, including those with incorrect final answers. Ineligible trajectories do not count toward the presence or success of either strategy.

Cross-Group Strategy Comparison. The invocation preference should depend on observed outcomes rather than tool presence alone. We therefore compare whether each sampled strategy demonstrates success for the same question. For each non-empty group, define its observed success indicator as

$$S_s(x_i) = \max_{\tau_{i,j} \in \mathcal{G}_s(x_i)} z_{i,j}, \qquad s \in \{D, T\}. \tag{7}$$

Thus, $S_s(x_i) = 1$ if at least one sampled trajectory from strategy s succeeds. This tests whether success has been demonstrated, rather than estimating the strategy's success probability. When both groups are non-empty, define the strategy-contrast indicator

$$\chi_i = \mathbb{I}[S_D(x_i) \neq S_T(x_i)]. \tag{8}$$

We set $\chi_i = 0$ when either group is empty, so an unobserved strategy is not treated as a failure. Otherwise, $\chi_i = 1$ only when exactly one strategy has observed success, and $z_{i,j}\chi_i$ rewards successful trajectories from that strategy. No cross-group bonus is applied when both or neither succeed. This contrast is a learning signal based on sampled outcomes, not a ground-truth label of tool necessity.

Within-Group Sufficiency Preference Learning. Strategy-level success does not distinguish successful tool-assisted trajectories with different acquisition depths. To learn a stopping preference, depth is therefore compared only among successful tool-assisted trajectories, so an unsuccessful short trajectory cannot serve as a stopping reference. Successful direct trajectories are excluded from this comparison: including them would produce a zero-call reference whenever direct answering succeeds, mixing cross-strategy selection with within-tool depth preference. Whenever a successful tool-assisted rollout exists, define

$$c_{\min}(x_i) = \min\{c_{i,j} \mid \tau_{i,j} \in \mathcal{G}_T(x_i),\ z_{i,j} = 1\}. \tag{9}$$

This is the minimum observed successful tool depth in the current sampled set, not a predefined call budget. For a successful tool-assisted trajectory, its excess call count is

$$\Delta c_{i,j} = c_{i,j} - c_{\min}(x_i), \qquad z_{i,j} = 1,\ c_{i,j} > 0. \tag{10}$$

We set $\Delta c_{i,j} = 0$ for all other trajectories. If no successful tool-assisted trajectory is sampled, no depth reference is formed and all $\Delta c_{i,j}$ are set to zero. To distinguish trajectories at the reference depth from those exceeding it, define the non-excess indicator

$$\eta_{i,j} = 1 - \mathbb{I}[\Delta c_{i,j} > 0]. \tag{11}$$

Among successful tool-assisted trajectories, $\eta_{i,j} = 1$ at $c_{\min}(x_i)$ and $\eta_{i,j} = 0$ at greater depths. This binary preference does not rank different non-minimal depths or require a tunable per-call penalty. The reference is empirical: a shorter successful trajectory need not be a prefix of a longer one, so excess depth does not identify individually unnecessary calls.

Hierarchical Trajectory Reward. To guide invocation and stopping without rewarding unsuccessful trajectories, SAVE combines task success with two success-conditioned comparison signals:

$$r_{i,j} = R\big(\tau_{i,j}, \mathcal{T}(x_i)\big) = \underbrace{z_{i,j}}_{\text{task success}} + \underbrace{z_{i,j}\chi_i}_{\text{strategy contrast}} + \underbrace{z_{i,j}\eta_{i,j}}_{\text{non-excess acquisition}}. \tag{12}$$

Here, R assigns trajectory reward $r_{i,j}$ using the sampled set $\mathcal{T}(x_i)$. The three binary components have equal unit weight, and unsuccessful trajectories receive zero reward. Successful direct trajectories receive the non-excess bonus by definition. When both strategies succeed, they tie with minimum-depth successful tool trajectories and receive higher rewards than longer successful ones. To convert these rewards into relative training signals, SAVE uses the REINFORCE Leave-One-Out estimator Ahmadian et al. (2024):

$$b_{i,j} = \frac{1}{N-1} \sum_{\substack{m=1 \\ m \neq j}}^{N} r_{i,m}, \qquad A_{i,j} = r_{i,j} - b_{i,j}. \tag{13}$$

Here, $b_{i,j}$ is the mean reward of the other rollouts for x_i, and $A_{i,j}$ is the advantage used for policy updates. For a fixed sampled set with $N > 1$, these advantages preserve within-instance reward ordering, including ties. Appendix B provides the derivation and scope. Appendix C provides the full component activation rules.

4 Experiments

Our experiments address three research questions:

RQ1: Does sufficiency-aware evidence acquisition improve multimodal reasoning compared with SFT and existing visual tool-use methods?

RQ2: What behavioral evidence supports SAVE's ability to decide whether to acquire additional evidence and when to stop?

RQ3: How do supervised initialization, difficulty-aware curation, and the two comparison-based reward components contribute to learning adaptive invocation and stopping?

4.1 Experimental Setup

Datasets. We construct SFT and RL data from the training split of ChartQA Masry et al. (2022), using Qwen3.5-397B-A17B Qwen Team (2026) as the multimodal teacher. Data construction and statistics are detailed in Appendix D. Evaluation covers six benchmarks: CharXiv Wang et al. (2024b), HallusionBench Guan et al. (2024), RealWorldQA xAI (2024), HR-Bench-4K/8K Wang et al. (2024a), and ChartQAPro Masry et al. (2025). The cross-backbone comparison uses five benchmarks, while the method comparison additionally includes HR-Bench-4K.

Backbones and Baselines. Qwen2.5-VL-7B-Instruct Bai et al. (2025b) is the primary backbone for method comparisons and ablations. We additionally train and evaluate SAVE on Qwen3-VL-8B-Instruct Bai et al. (2025a) and InternVL3.5-8B Wang et al. (2025). Qwen3-VL-8B-Thinking is

Table 1: Comparison of different vision-language backbones and training strategies. Direct denotes inference without tools; Base denotes the original backbone with tool access, before SAVE-specific SFT and RL. SFT denotes the supervised initialization, and SAVE denotes the checkpoint after subsequent sufficiency-aware policy learning.

Model	Method	CharXiv			HallusionBench			RealWorldQA			HR-Bench-8K			ChartQAPro		
		Acc	Rate	Calls	Acc	Rate	Calls	Acc	Rate	Calls	Acc	Rate	Calls	Acc	Rate	Calls
Qwen3-VL 8B-Think	Direct	49.80	–	–	73.57	–	–	67.32	–	–	65.67	–	–	49.38	–	–
	Base	45.17	44.00	1.696	69.30	24.03	1.240	66.30	36.76	1.482	66.04	39.68	1.667	49.75	0.86	1.260
Qwen3-VL 8B-Instruct	Direct	40.13	–	–	68.35	–	–	61.92	–	–	60.54	–	–	31.09	–	–
	Base	30.00	72.00	2.356	62.39	63.09	1.363	62.05	65.36	1.878	54.04	70.83	2.066	28.83	69.04	2.158
	SFT	40.77	24.00	1.388	71.14	15.67	1.181	60.83	45.78	1.159	67.17	41.83	1.089	41.66	28.75	1.377
	SAVE	**43.17**	**51.17**	**1.220**	**71.92**	**21.91**	**1.140**	**63.13**	**51.20**	**1.120**	**69.08**	**55.67**	**1.090**	**42.78**	**44.30**	**1.160**
InternVL 3.5-8B	Direct	31.87	–	–	65.16	–	–	53.16	–	–	41.58	–	–	26.27	–	–
	Base	20.47	95.00	2.976	57.17	80.62	2.163	33.73	95.42	3.332	23.17	93.75	3.396	23.36	85.56	2.656
	SFT	32.80	20.50	1.556	68.10	17.42	1.266	56.91	36.99	1.355	51.38	38.29	1.269	38.89	28.78	1.570
	SAVE	**45.90**	**40.93**	**1.480**	**78.23**	**24.92**	**1.230**	**62.66**	**52.29**	**1.310**	**60.29**	**50.29**	**1.380**	**43.69**	**45.07**	**1.500**
Qwen2.5 VL-7B	Direct	32.73	–	–	59.87	–	–	49.46	–	–	32.08	–	–	24.25	–	–
	Base	21.73	57.00	1.766	48.79	36.80	1.390	39.22	33.12	1.367	35.25	7.50	1.589	29.60	16.76	1.534
	SFT	35.67	23.30	1.474	62.67	18.40	1.248	40.92	31.98	1.546	31.29	8.38	1.776	32.17	24.20	1.633
	SAVE	**45.51**	**30.93**	**1.170**	**72.53**	**22.40**	**1.120**	**65.18**	**48.24**	**1.290**	**59.26**	**20.25**	**1.609**	**34.00**	**31.04**	**1.281**

included only as an inference comparison under the Direct and Base settings. External baselines comprise DeepEyes-7B Zheng et al. (2026), VTool-R1-7B Wu et al. (2026), ReVPT-7B Zhou et al. (2025), CodeV-RL Hou et al. (2026), AdaReasoner-7B Song et al. (2026a), and DeepEyesV2-7B Hong et al. (2026).

Evaluation Protocol. We evaluate all compared methods using their released checkpoints in our experimental environment; external baselines are not retrained or fine-tuned on SAVE's training data. We run inference and evaluation under a unified protocol. We sample three trajectories per evaluation instance, and all tool-enabled comparison models use the same visual crop interface. Accuracy (Avg@3) averages answer correctness across the three runs and then across instances. Tool usage rate (Rate) is the percentage of trajectories with at least one tool call. Calls/Used is the mean tool-call count among trajectories that use tools. Cross-benchmark summaries are macro-averaged. Base and SAVE are compared using the same answer evaluator and evaluation protocol. Thus, the comparison evaluates released models under a common inference environment. Training configurations are provided in Appendix E; prompts and the tool schema are provided in Appendix F.

4.2 Main Results (RQ1)

Accuracy and Tool-Use Behavior. Table 1 shows that SAVE improves Avg@3 over SFT on all 15 backbone–benchmark comparisons, with a mean gain of 8.33 percentage points. Paired 95% bootstrap confidence intervals for the gain exclude zero in 10 comparisons, as detailed in Appendix G.1. Tool-use rate increases in all 15 comparisons, while Calls/Used decreases in 13, lowering its macro-average from 1.392 to 1.273, an 8.6% relative reduction. Higher accuracy therefore accompanies more frequent invocation but shallower tool-assisted execution, not necessarily fewer calls across all trajectories.

Comparison with Existing Methods. In Table 2, SAVE achieves the highest macro-average Avg@3 among the compared methods across six benchmarks: 57.25%, versus 55.52% for CodeV-RL and 55.18% for VTool-R1-7B. It ranks first on CharXiv and HallusionBench and second on the remaining four benchmarks. Its macro-average tool-use rate and Calls/Used are 28.26% and 1.338, respectively.

4.3 Analysis of Evidence Acquisition Behavior (RQ2)

Higher accuracy alone does not reveal how evidence is acquired. We therefore examine training dynamics and matched trajectory outcomes to characterize invocation and stopping.

Training Dynamics. Figure 3 tracks invocation and acquisition depth during training. Tool-use rates generally decline while validation accuracy is maintained or improves, and Correct Without Tool Rate increases overall. Conditional depth decreases markedly for Qwen2.5-VL-7B and Qwen3-VL-8B,

Table 2: Comparison with representative vision-language tool-use reasoning methods using Qwen2.5-VL-7B as the backbone across six multimodal reasoning benchmarks. Best and second-best accuracy results are highlighted in **bold** and underline, respectively.

Model	CharXiv			HallusionBench			RealWorldQA			HR-Bench-8K			HR-Bench-4K			ChartQAPro		
	Acc	Rate	Calls	Acc	Rate	Calls	Acc	Rate	Calls	Acc	Rate	Calls	Acc	Rate	Calls	Acc	Rate	Calls
VTool-R1-7B	38.20	35.93	1.330	69.96	17.21	1.140	63.79	2.40	1.850	**60.29**	33.13	1.170	65.63	30.88	1.100	33.21	27.74	1.270
ReVPT-7B	32.30	50.80	1.220	57.52	42.90	1.420	63.62	49.32	1.280	54.58	40.58	1.930	**71.04**	37.96	1.490	29.21	39.70	1.420
CodeV-RL	41.33	66.07	1.780	71.83	0.49	2.000	**67.58**	0.39	1.560	58.04	0.58	1.710	62.67	0.83	1.950	31.69	21.89	1.930
AdaReasoner-7B	35.07	88.50	4.300	54.92	86.44	4.450	60.04	51.42	3.970	58.29	83.54	2.970	62.63	81.75	2.650	31.76	90.66	4.700
DeepEyesV2-7B	42.60	18.40	1.120	57.90	17.49	1.350	62.57	5.45	1.150	58.79	15.33	1.460	64.63	18.42	1.240	32.08	18.74	1.370
DeepEyes-7B	39.59	98.73	1.028	64.84	98.91	1.026	65.10	98.65	1.000	55.17	97.23	1.021	64.84	98.91	1.026	**35.28**	98.15	1.118
SAVE-7B	**45.51**	30.93	1.170	**72.53**	22.40	1.120	65.18	48.24	1.290	59.26	20.25	1.609	67.00	16.67	1.560	34.00	31.04	1.281



Figure 3: Training dynamics of SAVE across three vision-language backbones. **(a)** Validation Accuracy measures task performance during policy learning. **(b)** Tool Use Rate is the fraction of trajectories invoking at least one visual tool. **(c)** Tool Calls When Used is the mean tool-call count conditioned on tool use. **(d)** Correct Without Tool Rate is the fraction of all sampled trajectories that answer correctly without tool use.

whereas InternVL3.5-8B follows a less monotonic pattern. These within-training trends are distinct from the downstream SFT–SAVE comparisons in Table 1.

Post-hoc Trajectory Analysis. Panels c and d of Figure 4 analyze the HallusionBench trajectories from Table 1, using separate matched question sets for strategy success and acquisition depth. The Base–SAVE comparison includes both SFT and RL. Joint strategy success increases for Qwen2.5-VL and InternVL3.5 but decreases slightly for Qwen3-VL. In the depth-analysis sets, the share of questions with a successful one-call trajectory rises by 15.9, 11.3, and 1.1 percentage points for Qwen2.5-VL, InternVL3.5, and Qwen3-VL, respectively. Redundancy rates also decline, although only slightly for InternVL3.5, from 6.1% to 5.9%. The depth results are consistent with shallower successful acquisition, while strategy-success changes remain backbone-dependent. These findings are conditional on the matched subsets and do not establish ground-truth tool necessity or optimal stopping. Selection criteria and statistical definitions are provided in Appendix G.

4.4 Ablation Studies (RQ3)

Table 3 evaluates training initialization and reward design on Qwen2.5-VL-7B across six benchmarks. Full SAVE achieves the highest Avg@3 on all six. The cross-strategy and minimum-depth ablations each retain the other comparison component. We use `Only RL` for w/o Capability Acquisition, `accuracy_only` for Accuracy-Only Reward, `no_cross_strategy` for w/o Cross-Strategy Comparison, and `no_min_evidence` for w/o Minimum-Depth Preference.

Supervised Initialization and Data Curation. Full SAVE achieves the highest Avg@3 on all six benchmarks, with a macro-average of 57.25%. Removing capability acquisition lowers macro-average Avg@3 to 51.41% and nearly eliminates tool invocation, as shown by `Only RL` in Figure 4(a–b), supporting the role of supervised initialization in establishing tool-use behaviors. Removing difficulty-aware curation lowers macro-average Avg@3 to 39.90% despite shallower tool-assisted trajectories.

Effects of Reward Components. Accuracy-Only Reward lowers macro-average Avg@3 to 36.78% and increases macro-average Calls/Used relative to SAVE. Removing cross-strategy comparison

Table 3: Ablation studies of SAVE on Qwen2.5-VL-7B across six benchmarks. Acc denotes Avg@3 (%), Rate denotes tool-use rate (%), and Calls denotes Calls/Used.

Variant	CharXiv			HallusionBench			RealWorldQA			HR-Bench-8K			HR-Bench-4K			ChartQAPro		
	Acc	Rate	Calls	Acc	Rate	Calls	Acc	Rate	Calls	Acc	Rate	Calls	Acc	Rate	Calls	Acc	Rate	Calls
Training Initialization and Data Curation																		
w/o Capability Acquisition	36.20	0.00	–	62.32	0.00	–	61.92	0.00	–	57.04	0.00	–	61.21	0.00	–	29.77	0.05	1.000
w/o Difficulty-Aware Curation	28.70	27.73	1.053	58.68	21.21	1.021	45.62	40.61	1.079	38.54	17.54	1.138	40.38	13.88	1.129	27.45	28.10	1.076
Reward Design																		
Accuracy-Only Reward	29.90	48.23	1.530	56.82	32.74	1.316	39.65	44.71	1.590	25.21	8.54	1.556	36.08	13.00	1.663	33.03	47.31	1.580
w/o Cross-Strategy Comparison	34.60	70.13	1.087	56.61	39.89	1.040	53.07	68.41	1.100	54.25	36.29	1.175	59.46	35.54	1.203	31.30	61.24	1.108
w/o Minimum-Depth Preference	32.90	34.97	1.438	62.53	25.20	1.238	51.46	53.59	1.487	59.13	19.42	1.599	64.58	20.29	1.690	33.73	35.30	1.476
SAVE (Full Framework)	**45.51**	**30.93**	**1.170**	**72.53**	**22.40**	**1.120**	**65.18**	**48.24**	**1.290**	**59.26**	**20.25**	**1.609**	**67.00**	**16.67**	**1.560**	**34.00**	**31.04**	**1.281**

Variant definitions. w/o Capability Acquisition starts RL from the original backbone without SFT. w/o Difficulty-Aware Curation includes easy, medium, and hard instances in $\mathcal{D}_{\mathrm{RL}}$. For the reward in Equation 12, Accuracy-Only Reward uses $r_{i,j} = z_{i,j}$; w/o Cross-Strategy Comparison sets $\chi_i = 0$, yielding $r_{i,j} = z_{i,j}(1 + \eta_{i,j})$; w/o Minimum-Depth Preference sets $\eta_{i,j} = 1$, yielding $r_{i,j} = z_{i,j}(2 + \chi_i)$.

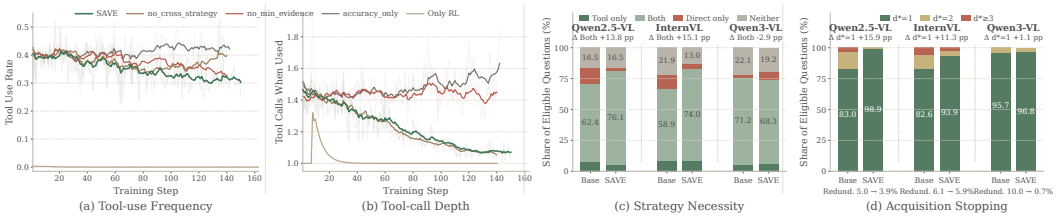


Figure 4: Ablation study and post-hoc analysis of SAVE. **(a)** Tool Use Rate and **(b)** Tool Calls When Used compare invocation frequency and conditional depth across ablations on Qwen2.5-VL. **(c)** Strategy Necessity reports observed strategy-success patterns on matched questions with both strategies sampled under each of Base and SAVE. **(d)** Acquisition Stopping reports minimum observed successful tool depth on matched questions with successful tool-assisted trajectories under both conditions. For each condition in (d), the redundancy rate is the pooled proportion of successful tool trajectories whose call count exceeds their question-specific minimum successful tool depth.

reduces macro-average Avg@3 to 48.22% despite more frequent but shallower tool use. Removing the minimum-depth preference lowers accuracy and increases macro-average Calls/Used from 1.338 to 1.488, with higher conditional depth on five of the six benchmarks. Thus, neither invocation frequency nor acquisition depth alone accounts for SAVE's performance. Appendix G.4 provides training-time diagnostics of invocation frequency, conditional depth, and redundant correct tool use.

Evidence on Uniform Call Penalties. A common way to control tool usage is to assign each trajectory τ the reward $\tilde{r}_\lambda(\tau) = z(\tau) - \lambda c(\tau)$, where $z(\tau) \in \{0, 1\}$ indicates final-answer correctness, $c(\tau)$ counts tool calls, and $\lambda > 0$ is a fixed per-call penalty. Recent evidence suggests that such unconditional penalties can introduce an undesirable accuracy–efficiency trade-off. In particular, SPyCE Zhang & Qiu (2026) reports that always penalizing tool calls reduces the average number of tool interactions but also substantially degrades task accuracy. This result suggests that uniformly assigning a fixed cost to every tool invocation may discourage useful tool interaction when additional evidence is required.

5 Conclusion

We introduced **SAVE**, a sufficiency-aware framework for adaptive visual tool use that learns when additional evidence is needed and when acquisition should stop. SAVE uses multiple on-policy trajectories as empirical references, combining strategy-level comparisons with depth preferences among successful tool trajectories. Across three vision-language backbones and multimodal reasoning benchmarks, SAVE improves Avg@3 over supervised fine-tuning by 8.33 percentage points on average while reducing macro-average Calls/Used by 8.6%. Ablations and behavioral analyses further support this training design for learning adaptive visual evidence acquisition from complete-trajectory comparisons.

ETHICS STATEMENT

This study uses publicly available multimodal reasoning benchmarks and curated visual question-answering datasets. It does not involve human-subject experiments or the collection of personally identifiable information.

The proposed method may inherit biases, annotation errors, and distributional limitations from the datasets and pretrained multimodal models used in the study. Tool-augmented reasoning may also amplify errors when an incorrect region is selected or when intermediate visual evidence is misinterpreted.

We encourage responsible use through clear dataset documentation, evaluation across diverse domains, appropriate access controls, and human oversight in applications where model errors could have significant consequences. Any release of code, processed data, and evaluation resources will follow the licenses and usage restrictions of the corresponding source datasets.

REPRODUCIBILITY STATEMENT

We provide the main components needed to reproduce SAVE, including the data construction procedure, supervised tool-use initialization, reward assignment, reinforcement-learning configuration, inference prompts, visual tool schema, and evaluation protocol.

The supplementary material describes the construction of supervision and policy-learning data, including trajectory generation, verification, and difficulty-aware curation. Training configurations are reported separately for the supervised and reinforcement-learning stages, together with backbone-specific distributed settings. The prompts used for tool interaction and semantic answer evaluation are also provided in full.

Anonymous source code and supplementary resources will be released to support independent verification and subsequent research.

AI USE STATEMENT

Generative AI assistants were used during the research and manuscript preparation process. AI-generated suggestions were treated as auxiliary input rather than authoritative results; experimental configurations, code, numerical results, citations, figures, and scientific claims were reviewed and verified by the authors.

Large multimodal models were also used as explicit components of the experimental pipeline, including teacher-guided trajectory generation, failure attribution, and semantic answer evaluation, as described in the corresponding sections of the paper. These model-based components constitute part of the reported methodology and are separately documented with their configurations and evaluation procedures.

The authors take full responsibility for the design of the study, the validity of the experiments and analyses, and the accuracy, originality, and final content of the paper.

REFERENCES

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting REINFORCE style optimization for learning from human feedback in LLMs. *arXiv preprint arXiv:2402.14740*, 2024. URL https://arxiv.org/abs/2402.14740.

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, 2025a. URL https://arxiv.org/abs/2511.21631.

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. *arXiv preprint arXiv:2502.13923*, 2025b. URL https://arxiv.org/abs/2502.13923.

Xiuwei Chen, Quanlin Chen, Wentao Hu, Zisheng Chen, Kun Xiang, Zehua Ma, Mingyang Zhang, Jianhua Han, Hanhui Li, Hang Xu, and Xiaodan Liang. Beyond the eye: Efficient multimodal reasoning via self-regulated implicit visual tools, 2026. URL https://arxiv.org/abs/2607.11106.

Yong Xien Chng, Tao Hu, Wenwen Tong, Xueheng Li, Jiandong Chen, Haojia Yu, Jiefan Lu, Hewei Guo, Hanming Deng, Chengjun Xie, Gao Huang, Dahua Lin, and Lewei Lu. Sensenova-mars: Empowering multimodal agentic reasoning and search via reinforcement learning, 2026. URL https://arxiv.org/abs/2512.24330.

Yalun Dai, Hao Li, Shulin Tian, Runmao Yao, Yuhao Dong, Fangzhou Hong, Zhaoxi Chen, Fangfu Liu, Tao Wang, Kim-Hui Yap, and Ziwei Liu. S-agent: Spatial tool-use elicits reasoning for spatial intelligence, 2026. URL https://arxiv.org/abs/2606.20515.

Qihua Dong, Gozde Sahin, Pei Wang, Zhaowei Cai, Robik Shrestha, Hao Yang, and Davide Modolo. Visual reasoning through tool-supervised reinforcement learning, 2026. URL https://arxiv.org/abs/2604.19945.

Mingkuan Feng, Jinyang Wu, Hao Gu, Fangrui Lv, Ruihan Jin, Chuyuan Zhang, Zhengqi Wen, and Jianhua Tao. Taco: Tool-augmented credit optimization for agentic tool use, 2026. URL https://arxiv.org/abs/2606.30251.

Zhenkun Gao, Yicheng Bao, Jinlong Peng, Xueheng Li, Theo Huang, Bangwei Liu, Kunquan Li, Zhenye Gan, Tao Hu, Chengjun Xie, Mingqian Yang, Xuanhua He, Zhizhong Zhang, Xin Tan, Chengjie Wang, and Yuan Xie. Videosearcher: Empowering video deep research with multi-tool agentic reasoning via reinforcement learning, 2026. URL https://arxiv.org/abs/2607.02927.

Amirhosein Ghasemabadi, Ruichen Chen, Bahador Rashidi, and Di Niu. Multi-head latent control: A unified interface for llm agent decision making, 2026. URL https://arxiv.org/abs/2607.14277.

Tianrui Guan, Fuxiao Liu, Xiyang Wu, Ruiqi Xian, Zongxia Li, Xiaoyu Liu, Xijun Wang, Lichang Chen, Furong Huang, Yaser Yacoob, Dinesh Manocha, and Tianyi Zhou. Hallusionbench: An advanced diagnostic suite for entangled language hallucination and visual illusion in large vision-language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14375–14385, 2024. URL https://arxiv.org/abs/2310.14566.

Nicholas Hay, Stuart Russell, David Tolpin, and Solomon Eyal Shimony. Selecting computations: Theory and applications. In *Proceedings of the Twenty-Eighth Conference on Uncertainty in Artificial Intelligence*, 2012. URL https://arxiv.org/abs/1207.5879.

Jack Hong, Chenxiao Zhao, ChengLin Zhu, Weiheng Lu, Guohai Xu, and Xing Yu. Deepeyesv2: Toward agentic multimodal model, 2026. URL https://arxiv.org/abs/2511.05271.

Xinhai Hou, Shaoyuan Xu, Manan Biyani, Moyan Li, Jia Liu, Todd C. Hollon, and Bryan Wang. Codev: Code with images for faithful visual reasoning via tool-aware policy optimization, 2026. URL https://arxiv.org/abs/2511.19661.

Sunil Kumar, Bowen Zhao, Leo Dirac, and Paulina Varshavskaya. Reinforcing vlms to use tools for detailed visual reasoning under resource constraints, 2025. URL https://arxiv.org/abs/2506.14821.

Changye Li, Meng Lu, Yi Wu, and Ligeng Zhu. Perceive, interact, reason: Building tool-augmented visual agents for spatial reasoning, 2026. URL https://arxiv.org/abs/2606.12830.

Jiaqi Liu, Kaiwen Xiong, Peng Xia, Yiyang Zhou, Haonian Ji, Lu Feng, Siwei Han, Mingyu Ding, and Huaxiu Yao. Agent0-vl: Exploring self-evolving agent for tool-integrated vision-language reasoning, 2025. URL https://arxiv.org/abs/2511.19900.

Yuliang Liu, Zhang Li, Mingxin Huang, Biao Yang, Wenwen Yu, Chunyuan Li, Xu-Cheng Yin, Cheng-Lin Liu, Lianwen Jin, and Xiang Bai. Ocrbench: On the hidden mystery of ocr in large multimodal models. *Science China Information Sciences*, 67(12):220102, 2024. doi: 10.1007/s11432-024-4235-6. URL https://doi.org/10.1007/s11432-024-4235-6.

Ahmed Masry, Do Xuan Long, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. ChartQA: A benchmark for question answering about charts with visual and logical reasoning. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), *Findings of the Association for Computational Linguistics: ACL 2022*, pp. 2263–2279, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-acl.177. URL https://aclanthology.org/2022.findings-acl.177/.

Ahmed Masry, Mohammed Saidul Islam, Mahir Ahmed, Aayush Bajaj, Firoz Kabir, et al. Chartqapro: A more diverse and challenging benchmark for chart question answering. *arXiv preprint arXiv:2504.05506*, 2025. URL https://arxiv.org/abs/2504.05506.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

Hajin Shim, Sung Ju Hwang, and Eunho Yang. Why pay more when you can pay less: A joint learning framework for active feature acquisition and classification. *arXiv preprint arXiv:1709.05964*, 2017. URL https://arxiv.org/abs/1709.05964.

Mingyang Song, Haoyu Sun, Jiawei Gu, Linjie Li, Luxin Xu, Ranjay Krishna, and Yu Cheng. Adareasoner: Dynamic tool orchestration for iterative visual reasoning, 2026a. URL https://arxiv.org/abs/2601.18631.

Qi Song, Honglin Li, Yingchen Yu, Haoyi Zhou, Lin Yang, Song Bai, Qi She, Zilong Huang, and Yunqing Zhao. Codedance: A dynamic tool-integrated mllm for executable visual reasoning, 2026b. URL https://arxiv.org/abs/2512.17312.

Zhaochen Su, Linjie Li, Mingyang Song, Yunzhuo Hao, Zhengyuan Yang, Jun Zhang, Guanjie Chen, Jiawei Gu, Juntao Li, Xiaoye Qu, and Yu Cheng. Openthinkimg: Learning to think with images via visual tool reinforcement learning, 2025. URL https://arxiv.org/abs/2505.08617.

Haoxiang Sun, Zhihang Yi, Langxuan Deng, Yuhao Zhou, Peiqi Jia, Jian Zhao, Li Yuan, Jiancheng Lv, and Tao Wang. V-zero: Answer-label-free on-policy distillation with contrastive evidence gating for fine-grained visual reasoning, 2026. URL https://arxiv.org/abs/2606.25319.

Chaoyang Wang, Kaituo Feng, Dongyang Chen, Zhongyu Wang, Zhixun Li, Sicheng Gao, Meng Meng, Xu Zhou, Manyuan Zhang, Yuzhang Shang, and Xiangyu Yue. Adatooler-v: Adaptive tool-use for images and videos, 2026a. URL https://arxiv.org/abs/2512.16918.

Haoqing Wang, Xingrun Xing, Wei Xia, Ziheng Li, and Yehui Tang. Faitheyes: Towards faithful tool use via multi-agent process-image verification, 2026b. URL https://arxiv.org/abs/2607.28225.

Qixun Wang, Yang Shi, Letian Cheng, Zhuoran Zhang, Yan He, Yuqi Tang, Qi Zhang, Xinlei Yu, Ruizhe Chen, Tianrun Xu, Yuanxing Zhang, Pengfei Wan, Haotian Wang, and Xianghua Ying. Beacon: Knowing when and how to perform agentic visual reasoning, 2026c. URL https://arxiv.org/abs/2607.28595.

Weiyun Wang, Zhangwei Gao, Lixin Gu, Hengjun Pu, Long Cui, Xingguang Wei, Zhaoyang Liu, Linglin Jing, Shenglong Ye, Jie Shao, et al. Internvl3.5: Advancing open-source multimodal models in versatility, reasoning, and efficiency. *arXiv preprint arXiv:2508.18265*, 2025. URL https://arxiv.org/abs/2508.18265.

Wenbin Wang, Liang Ding, Minyan Zeng, Xiabin Zhou, Li Shen, Yong Luo, and Dacheng Tao. Divide, conquer and combine: A training-free framework for high-resolution image perception in multimodal large language models, 2024a. URL https://arxiv.org/abs/2408.15556.

Zirui Wang, Mengzhou Xia, Luxi He, Howard Chen, Yitao Liu, Richard Zhu, Kaiqu Liang, Xindi Wu, Haotian Liu, Sadhika Malladi, Alexis Chevalier, Sanjeev Arora, and Danqi Chen. Charxiv: Charting gaps in realistic chart understanding in multimodal llms. *arXiv preprint arXiv:2406.18521*, 2024b. URL https://arxiv.org/abs/2406.18521.

Shouang Wei, Houcheng Min, Xinpeng Dong, Xin Lin, Sen Cui, Bo Jiang, Zhongxiang Dai, Kun Kuang, Guandong Xu, Fei Wu, and Min Zhang. Metaforge: A self-evolving multimodal agent that retrieves, adapts, and forges tools on demand, 2026. URL https://arxiv.org/abs/2606.01801.

Mingyuan Wu, Jingcheng Yang, Jize Jiang, Meitang Li, Kaizhuo Yan, Hanchao Yu, Minjia Zhang, Chengxiang Zhai, and Klara Nahrstedt. Vtool-r1: Vlms learn to think with images via reinforcement learning on multimodal tool use, 2026. URL https://arxiv.org/abs/2505.19255.

xAI. Realworldqa: A benchmark for real-world spatial understanding, 2024. URL https://huggingface.co/datasets/xai-org/RealworldQA.

Zhenghao Xing, Ruiyang Xu, Yuxuan Wang, Jinzheng He, Ziyang Ma, Qize Yang, Yunfei Chu, Jin Xu, Junyang Lin, Chi-Wing Fu, and Pheng-Ann Heng. Native active perception as reasoning for omni-modal understanding, 2026. URL https://arxiv.org/abs/2606.19341.

Jinlong Yang, Wenhao Zhang, Kuanwei Lin, and Sijie Cheng. Cader: Confidence-aware dynamic evidence reasoning for long-video understanding, 2026. URL https://arxiv.org/abs/2607.24582.

Senqiao Yang, Junyi Li, Xin Lai, Bei Yu, Hengshuang Zhao, and Jiaya Jia. Visionthink: Smart and efficient vision language model via reinforcement learning, 2025. URL https://arxiv.org/abs/2507.13348.

Yirong Zeng, Xiao Ding, Yufei Liu, Yuxian Wang, Qunyao Du, Yutai Hou, Wu Ning, Haonan Song, Duyu Tang, Dandan Tu, Bing Qin, and Ting Liu. Autotool: Automatic scaling of tool-use capabilities in rl via decoupled entropy constraints, 2026. URL https://arxiv.org/abs/2603.13348.

Haoji Zhang, Xin Gu, Jiawen Li, Chixiang Ma, Sule Bai, Chubin Zhang, Bowen Zhang, Zhichao Zhou, Dongliang He, and Yansong Tang. Thinking with videos: Multimodal tool-augmented reinforcement learning for long video reasoning, 2025. URL https://arxiv.org/abs/2508.04416.

Junkai Zhang, Yihe Deng, Kai-Wei Chang, and Wei Wang. Thinking with visual grounding, 2026. URL https://arxiv.org/abs/2606.16122.

Renrui Zhang, Dongzhi Jiang, Yichi Zhang, Haokun Lin, Ziyu Guo, Pengshuo Qiu, Aojun Zhou, Pan Lu, Kai-Wei Chang, Peng Gao, and Hongsheng Li. Mathverse: Does your multi-modal llm truly see the diagrams in visual math problems? *arXiv preprint arXiv:2403.14624*, 2024. URL https://arxiv.org/abs/2403.14624.

Ru Zhang and Weijie Qiu. SPyCE: Skill-policy co-evolution for multimodal agents, 2026. URL https://arxiv.org/abs/2607.13854.

Ziwei Zheng, Michael Yang, Jack Hong, Chenxiao Zhao, Guohai Xu, Le Yang, Chao Shen, and Xing Yu. Deepeyes: Incentivizing "thinking with images" via reinforcement learning, 2026. URL https://arxiv.org/abs/2505.14362.

Binglin Zhou, Peng Shi, Ryo Kamoi, Nan Zhang, and Rui Zhang. Toolsciver: Multimodal scientific claim verification with visual tool augmented reinforcement learning, 2026a. URL https://arxiv.org/abs/2607.16131.

Yang Zhou, Zixuan Huang, Sunzhu Li, Zhuo Yang, Chen Zhang, Shunian Chen, Caijun Yan, Jianyao Xu, Shunyu Liu, Weijie Fu, Peiliang Li, Xiaozhi Chen, and Yuxiang Cai. Spatialcli: Learning to reason with spatial tools, then without them, 2026b. URL https://arxiv.org/abs/2607.27703.

Yijin Zhou, Linqian Zeng, Xiaoya Lu, Wenyuan Xie, Dongrui Liu, Junchi Yan, and Jing Shao. Exploring agentic tool-calling decisions via uncertainty-aligned reinforcement learning, 2026c. URL https://arxiv.org/abs/2606.06976.

Zetong Zhou, Dongping Chen, Zixian Ma, Zhihan Hu, Mingyang Fu, Sinan Wang, Yao Wan, Zhou Zhao, and Ranjay Krishna. Reinforced visual perception with tools, 2025. URL https://arxiv.org/abs/2509.01656.

Chenming Zhu, Jingli Lin, Yilin Long, Peizhou Cao, Tai Wang, Jiangmiao Pang, and Xihui Liu. Thinking with imagination: Agentic visual spatial reasoning with world simulators, 2026a. URL https://arxiv.org/abs/2606.06476.

Muzhi Zhu, Hao Zhong, Canyu Zhao, Zongze Du, Mingyu Liu, Zheng Huang, Anzhou Li, Hao Chen, Cheng Zou, Jingdong Chen, Ming Yang, and Chunhua Shen. Active-o3: Empowering mllms with active perception via pure reinforcement learning, 2026b. URL https://arxiv.org/abs/2505.21457.

A MOTIVATION ANALYSIS

Visual tools allow vision-language models to acquire additional local evidence, but their benefit depends on whether that evidence is needed for the current question. We examine two complementary cases: whether additional visual inspection can recover errors caused by insufficient evidence, and whether unnecessary inspection can alter an otherwise correct prediction.

A.1 EXPERIMENTAL SETUP

For the model-level comparison in Figure 1(a), we evaluate five representative vision-language models: Qwen3.5-35B-A3B Qwen Team (2026), Qwen3-VL-8B-Thinking Bai et al. (2025a), Qwen3-VL-8B-Instruct Bai et al. (2025a), Qwen2.5-VL-7B-Instruct Bai et al. (2025b), and InternVL3.5-8B Wang et al. (2025).

The evaluation covers two multimodal reasoning tasks. For OCR understanding, we use OCR-Bench Liu et al. (2024) (echo840/OCRBench, test split, 1,000 samples). For visual mathematical reasoning, we use MathVerse Zhang et al. (2024) (AI4Math/MathVerse, testmini split, 3,940 samples).

A.2 TOOL-USE SETTINGS

We compare three inference settings: No-tool, Forced-tool, and Auto-tool. The two tool-enabled settings use the same visual crop operation and differ only in how tool invocation is determined.

A.2.1 NO-TOOL SETTING

In the No-tool setting, the model answers from the original image without additional visual interaction.

No-tool Instruction

You are an image-question answering agent.
Solve the question based only on the original image.
Do not call any external tools.
Provide the final answer directly.

This setting serves as the reference condition for reasoning without additional visual evidence.

A.2.2 FORCED-TOOL SETTING

In the Forced-tool setting, the model must invoke the visual tool at least once before producing the final answer. The tool interface is identical to that used in the Auto-tool setting; only the invocation requirement differs.

Forced Tool-use Instruction

You are an image-question agent that solves problems step by step.
Before answering the question, you must invoke the visual tool at least once.
If you call a tool, the environment will return exactly one new processed image inside a tool_response message.
After receiving a tool_response, prioritize the returned image as the latest evidence.
Tool policy:
Every local_enhance call must include:
bbox=[x1,y1,x2,y2].
The bbox represents a region by its relative position within the image.
Coordinates are normalized integers in the range [0,1000].
Actual tool call format:
<tool_call> { "name":"local_enhance", "arguments": { "operation":"crop", "bbox":[x1,y1,x2,y2] } } </tool_call>

This setting measures the effect of mandatory evidence acquisition, including both useful inspection and failures introduced by unnecessary tool use.

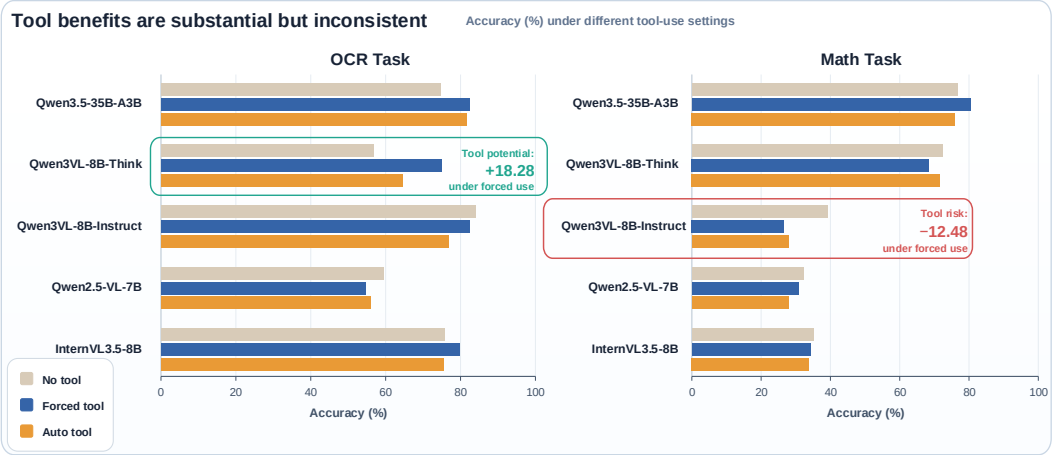


Figure 5: Comparison of tool-use reliability across five vision-language models on OCRBench and MathVerse. No-tool uses only the original image, Forced-tool requires at least one visual crop, and Auto-tool allows the model to decide whether additional evidence is needed. The effect of tool use varies across models and tasks: additional inspection can recover missing visual evidence, but unnecessary inspection can also reduce accuracy.

A.2.3 Auto-tool Setting

In the Auto-tool setting, the model receives the same tool capability but decides whether the current observation requires additional evidence.

Autonomous Tool-use Instruction

You are an image-question agent that solves problems step by step.
You can decide whether additional visual evidence is necessary before answering.
Use the visual tool only when the current observation is insufficient.
If you call a tool, the environment will return exactly one new processed image inside a tool_response message.
After receiving a tool_response, prioritize the returned image as the latest evidence.
Tool policy:
Every local_enhance call must include:
bbox=[x1,y1,x2,y2].
The bbox represents a region by its relative position within the image.
Coordinates are normalized integers in the range [0,1000].
Actual tool call format:

```
<tool_call> { "name":"local_enhance", "arguments": { "operation":"crop", "bbox":[x1,y1,x2,y2] } } </tool_call>
```

This setting evaluates whether an existing model can decide when additional visual inspection is warranted.

A.3 Tool-Use Reliability Analysis

The main paper summarizes the motivation study in Figure 1. Here, Figure 5 provides the model-level comparison of No-tool, Forced-tool, and Auto-tool inference on OCRBench and MathVerse.

The comparison shows that tool use is not uniformly beneficial. On OCRBench, Qwen3-VL-8B-Thinking improves from 56.68% under No-tool inference to 74.96% under Forced-tool inference, a gain of 18.28 percentage points. This suggests that local inspection can recover visual information missed when reasoning from the original image alone.

The opposite behavior appears in mathematical reasoning. On MathVerse, Qwen3-VL-8B-Instruct decreases from 39.12% under No-tool inference to 26.64% under Forced-tool inference, a drop of 12.48 percentage points. Mandatory inspection can therefore be harmful when the original observation

is already adequate or when the acquired local view does not provide additional evidence relevant to the question.

These results indicate that tool invocation should depend on the available evidence rather than follow a fixed direct- or tool-use strategy.

A.4 Visual Crop Tool

All tool-enabled settings use the same localized crop operation. The tool is implemented as `local_enhance` with the `crop` operation. Given a normalized bounding box $[x_1, y_1, x_2, y_2]$, it returns the corresponding image region as a new visual observation.

For the model-level comparison in Figure 1(a), we do not impose an explicit tool-call budget; interaction terminates when the model produces its final answer or reaches the maximum response length. The separate sequential-inspection diagnostic in Figure 1(b) uses a fixed budget of three additional crops, as detailed in Appendix A.5.

A.5 Sequential Visual Inspection

Figure 1(b) presents a separate inference-only diagnostic using the original Qwen3.5-35B-A3B model. For each question, we construct four visual evidence states, $E_{i,0}, \ldots, E_{i,3}$. The initial state contains only the original image. At each subsequent step, the model selects one additional crop from the original image, conditioned on the question and all currently available images. The original image and all acquired crops are retained. Inspection continues for three steps even when an earlier state already meets the sufficiency criterion.

At each evidence state, we generate three independent answers using fresh requests with the same tool-free answer prompt. Each request contains the question, the original image, and the accumulated crops, but no previous probe answers or judgments. The prompt requires only a final answer, without explanations, and thinking mode is disabled. Answer probes use temperature 0.7, top-$p = 1$, no top-k filtering, and a maximum of 4096 output tokens. Crop selection uses temperature 0, top-$p = 1$, and a maximum of 512 output tokens. Probe answers are evaluated against the reference answer; neither the answers nor their judgments are fed back into subsequent crop selection.

Let $a_{i,t,k} \in \{0, 1\}$ indicate whether the k-th probe at state $E_{i,t}$ is judged correct. A state is considered sufficient when at least two of its three valid probes are correct:

$$C_{i,t} = \mathbb{I}\left[\sum_{k=1}^{3} a_{i,t,k} \geq 2\right].$$

This is a repeated-answer criterion, not a human annotation of evidence completeness. Both panels use the same set $\mathcal{Q}$ of questions with three valid, evaluated probes at every state. Incomplete evidence chains are excluded from both analyses, rather than counted as insufficient.

For $t \in \{1, 2, 3\}$, first-time correction is conditioned on questions that have not met the sufficiency criterion at any earlier state, whereas negative regression is conditioned on questions that were sufficient at the immediately preceding state. Specifically, define

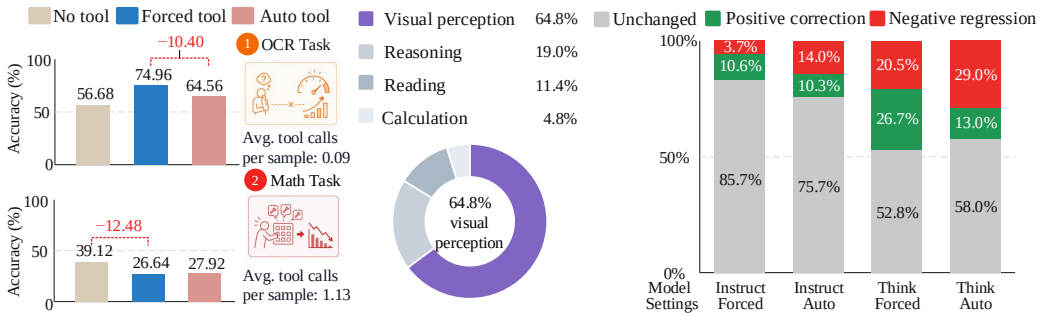
$$\mathcal{U}_{t-1} = \{i \in \mathcal{Q} : C_{i,0} = \cdots = C_{i,t-1} = 0\}, \qquad \mathcal{V}_{t-1} = \{i \in \mathcal{Q} : C_{i,t-1} = 1\}.$$

The reported rates are

$$\mathrm{FTC}_t = \frac{\sum_{i \in \mathcal{U}_{t-1}} C_{i,t}}{|\mathcal{U}_{t-1}|}, \qquad \mathrm{NR}_t = \frac{\sum_{i \in \mathcal{V}_{t-1}} (1 - C_{i,t})}{|\mathcal{V}_{t-1}|}.$$

A recovery after an earlier sufficient state and subsequent regression is not counted as first-time correction. The two rates therefore use different conditional denominators.

For each question, we record the first step $t \in \{0, 1, 2, 3\}$ at which $C_{i,t} = 1$. Step 0 denotes sufficiency with the original image alone. The > 3 category contains questions that never meet the criterion within the three-inspection budget; it does not indicate an observed success at a later step. A later regression does not change the recorded earliest sufficient step. All category proportions are normalized by $|\mathcal{Q}|$.



(a) Tool-use reliability evaluation (b) No-tool failure attribution (c) Prediction transition analysis

Figure 6: Motivation study on evidence sufficiency in tool-augmented multimodal reasoning. **(a)** Existing models do not consistently invoke tools when additional evidence is needed and may also invoke them unnecessarily. **(b)** Analysis of no-tool failures shows that insufficient visual perception accounts for a substantial fraction of errors. **(c)** Tool-assisted inspection can both correct incorrect predictions and introduce negative regressions.

The intermediate-state judgments are obtained from independent final-answer probes solely for this diagnostic. The analysis characterizes observed behavior under this protocol rather than establishing an optimal stopping rule or isolating the causal effect of additional visual evidence.

A.6 Teacher-Based Failure Attribution

To identify the primary causes of incorrect predictions, we use Qwen3.5-397B-A17B Qwen Team (2026) as an evaluator for failure attribution. The model is not involved in generating the prediction being evaluated. For each incorrect prediction, it receives the image, question, reference answer, and model output and assigns one primary failure category.

Figure 6(a) illustrates the contrasting effects of tool use: Forced-tool improves OCR accuracy over No-tool by 18.28 percentage points but reduces mathematical reasoning accuracy by 12.48 points, while Auto-tool does not consistently attain the higher accuracy of the two fixed settings. Figure 6(b) shows that visual perception is the primary attributed cause in 64.8% of the analyzed no-tool failures, suggesting that many no-tool failures are associated with insufficient visual evidence. However, Figure 6(c) shows that tool-assisted predictions exhibit both positive corrections and negative regressions. For Qwen3-VL-8B-Thinking under Auto-tool, negative regressions account for 29.0% of cases, exceeding the 13.0% positive corrections.

Teacher Evaluation Instruction

You are an expert evaluator for multimodal reasoning systems.
Given an image, question, ground-truth answer, and model prediction, identify the primary reason for the prediction failure.
Choose exactly one category:
Visual perception: The model fails to perceive, localize, or distinguish necessary non-textual visual evidence, excluding errors primarily caused by text recognition or reading.
Reasoning: The model perceives the information needed to answer the question but derives an incorrect conclusion.
Reading: The model fails to recognize or correctly interpret textual information visible in the image.
Calculation: The model identifies the required information and reasoning procedure but makes an arithmetic or numerical error.
Return the failure category and a brief explanation.

B Cost–Benefit Motivation for Hierarchical Evidence Acquisition

This appendix provides a sequential cost–benefit interpretation of the trajectory-level formulation in Section 3 and clarifies the binary preference and its ordering under RLOO. The intermediate success probabilities and utility functions are introduced only for analysis. SAVE itself computes training rewards from complete-trajectory outcomes and tool-call counts (Section 3.3).

B.1 Analytical View of Task Benefit and Acquisition Cost

Rational metareasoning balances improvements in decision quality against computational cost Hay et al. (2012). Active feature acquisition likewise considers predictive performance together with the cost of sequential observations Shim et al. (2017).

For a fixed instance x_i, consider one realized interaction path and omit its rollout index j. Let $h_{i,t}$ denote the interaction history after t tool calls, with $h_{i,0}$ containing the initial image and question. Let $p_\theta(h_{i,t})$ be the probability that, from this history and without further tool calls, the current policy produces a valid completion whose final answer is judged correct against $y_i^\star$. For $t \geq 1$, the change in task benefit between successive histories is

$$\Delta_{i,t} = p_\theta(h_{i,t}) - p_\theta(h_{i,t-1}). \tag{14}$$

Here, $p_\theta(h_{i,t})$ is a task-success probability rather than a self-reported confidence score. The increment $\Delta_{i,t}$ can be positive, zero, or negative. Because each history also contains intervening reasoning, this increment captures the change along the realized interaction path rather than the causal contribution of the returned visual crop alone.

Using call count as a cost proxy, define the net utility

$$U_\lambda(h_{i,t}) = p_\theta(h_{i,t}) - \lambda t, \qquad \lambda > 0. \tag{15}$$

The coefficient λ expresses the cost of one call on the same scale as task benefit. The utility change after one additional call is

$$U_\lambda(h_{i,t}) - U_\lambda(h_{i,t-1}) = \Delta_{i,t} - \lambda. \tag{16}$$

Under this utility, an additional call has positive immediate value when its task benefit exceeds its cost. This local comparison is not an optimal stopping rule, because a call with little immediate benefit may still enable useful later interactions. We therefore impose no monotonicity or diminishing-return assumption on $\Delta_{i,t}$.

B.2 Comparison with a Uniform Call-Cost Preference

To contrast SAVE with a uniform cost preference, suppose a successful trajectory receives unit benefit and each tool call incurs the same cost $\lambda > 0$. The resulting trajectory-level analytical score is

$$\widetilde{r}_\lambda(\tau_{i,j}) = z_{i,j} - \lambda c_{i,j}, \qquad \lambda > 0. \tag{17}$$

At the trajectory level, the score is the observed-outcome counterpart of Equation (15). Neither this score nor λ enters SAVE training; the construction is used only to examine the ordering induced by a uniform per-call cost, not to derive SAVE's training reward.

For a successful direct trajectory and a successful tool-assisted trajectory with $K_i > 0$ calls, subtracting their scores gives

$$\widetilde{r}_\lambda(\tau_i^D) - \widetilde{r}_\lambda(\tau_i^T) = 1 - (1 - \lambda K_i) = \lambda K_i > 0.$$

Thus, a uniform call-cost preference always ranks the successful direct trajectory above the successful tool-assisted trajectory in this comparison. The inequality describes only the ordering induced by the score and makes no empirical claim about optimization performance.

A related empirical result illustrates a potential limitation of unconditional call-count regularization. SPyCE reports that always penalizing tool calls reduces the average number of calls from 4.83 to 3.96, but decreases accuracy from 32.0 to 26.1, whereas its success-conditioned call penalty achieves a better accuracy–efficiency trade-off Zhang & Qiu (2026). This result suggests that uniformly

assigning a fixed cost to every tool invocation may discourage useful tool interaction when additional evidence is required.

In this related setting, uniformly suppressing tool invocation also discourages useful acquisition. The result is not, however, a controlled comparison between SAVE and a separately tuned linear call-cost objective under the same training setup.

SAVE instead separates the two comparison scopes. Observed strategy success determines the cross-group bonus, whereas acquisition depth is compared only among successful tool-assisted trajectories. When both strategies succeed, the cross-group bonus is inactive, while the within-tool preference still applies.

Consider a fixed sampled set $\mathcal{T}(x_i)$ containing at least one successful tool-assisted trajectory. For any such trajectory $\tau_{i,j}$, we have $z_{i,j} = 1$ and $c_{i,j} > 0$. Using the minimum observed successful tool depth $c_{\min}(x_i)$ from Equation (9), the analytical score gap relative to this reference is

$$[1 - \lambda c_{\min}(x_i)] - [1 - \lambda c_{i,j}] = \lambda \left[c_{i,j} - c_{\min}(x_i)\right] = \lambda \Delta c_{i,j}. \tag{18}$$

Here, $\Delta c_{i,j}$ is the excess call count defined in Equation (10). The gap expresses excess successful depth as a relative call-cost difference within the sampled set, without requiring a global target call count. The reference remains empirical: a shorter successful trajectory need not be a prefix of a longer one, and one observed correct answer does not establish reliable success under repeated sampling. Thus, $\Delta c_{i,j} > 0$ identifies a higher-cost successful realization in the sampled set without implying that its additional calls were causally unnecessary.

B.3 From Relative Cost to the Binary Training Preference

SAVE uses a binary excess-acquisition comparison rather than a penalty proportional to the number of additional calls. For a successful tool-assisted trajectory and $\lambda > 0$, the score gap in Equation (18) gives

$$\mathbb{I}[\lambda \Delta c_{i,j} > 0] = \mathbb{I}[\Delta c_{i,j} > 0] = 1 - \eta_{i,j}. \tag{19}$$

Here, $\eta_{i,j}$ is the non-excess indicator defined in Equation (11). The binary comparison depends only on whether the relative cost gap is positive, not on the value of λ. Within a sampled set, it favors minimum-depth successful tool trajectories and assigns the same lower preference to every longer successful tool trajectory. Consequently, it does not preserve the finer ordering among non-minimal depths induced by a linear call-cost penalty.

The analytical construction explains the comparison structure used by SAVE. Equation (12) implements that structure as a discrete training reward rather than optimizing the analytical utility directly. For a fixed sampled set with $N > 1$, let $\bar{r}_i = N^{-1} \sum_{j=1}^{N} r_{i,j}$. Rewriting Equation (13) yields

$$b_{i,j} = \frac{N\bar{r}_i - r_{i,j}}{N-1}, \qquad A_{i,j} = \frac{N}{N-1}\left(r_{i,j} - \bar{r}_i\right).$$

Since $N/(N-1) > 0$, the corresponding advantages preserve the within-instance reward ordering, including ties. The algebra preserves within-instance ordering only; it neither defines a globally optimal stopping rule nor establishes an unbiased policy-gradient estimator for group-dependent rewards.

Trajectory rewards are not globally normalized after reward assignment. Advantages are computed using a group-wise RLOO estimator, which subtracts a leave-one-out baseline without group-level or batch-level whitening or standard-deviation normalization. RLOO therefore removes group-wise additive shifts but preserves multiplicative differences in reward scale. Such differences may affect the optimization dynamics of the ablation variants; isolating this effect would require scale-matched retraining.

C Reward Design for Sufficiency-Aware Learning

Table 4 shows when the task-success ($z_{i,j}$), strategy-contrast ($z_{i,j}\chi_i$), and non-excess-acquisition ($z_{i,j}\eta_{i,j}$) components in Equation (12) are active. The total reward is their sum, and the table introduces no additional assignment rules.

Table 4: Complete component activation for the hierarchical reward in Equation (12). A checkmark indicates that the corresponding reward component is active. "Minimum-depth" and "longer" denote successful tool-assisted trajectories with $c_{i,j} = c_{\min}(x_i)$ and $c_{i,j} > c_{\min}(x_i)$, respectively.

Scenario	Trajectory condition	Task $z_{i,j}$	Contrast $z_{i,j}\chi_i$	Non-excess $z_{i,j}\eta_{i,j}$
Any scenario	Incorrect, invalid, or failed	–	–	–
Both sampled; only direct succeeds	Successful direct	✓	✓	✓
Both sampled; only tool succeeds	Minimum-depth successful tool	✓	✓	✓
Both sampled; only tool succeeds	Longer successful tool	✓	✓	–
Both sampled; both succeed	Successful direct	✓	–	✓
Both sampled; both succeed	Minimum-depth successful tool	✓	–	✓
Both sampled; both succeed	Longer successful tool	✓	–	–
Only direct sampled	Successful direct	✓	–	✓
Only tool sampled	Minimum-depth successful tool	✓	–	✓
Only tool sampled	Longer successful tool	✓	–	–

Successful direct trajectories and minimum-depth successful tool trajectories receive $2 + \chi_i$, whereas longer successful tool trajectories receive $1 + \chi_i$. All incorrect, invalid, or failed trajectories receive zero reward. For successful direct trajectories, the non-excess component is active because $\Delta c_{i,j} = 0$ by definition. Therefore, an inactive cross-group bonus does not imply equal rewards for all successful trajectories.

The reward variants use the following raw trajectory rewards:

$$\begin{aligned} r_{i,j}^{\text{full}} &= z_{i,j}(1 + \chi_i + \eta_{i,j}), \\ r_{i,j}^{\text{accuracy-only}} &= z_{i,j}, \\ r_{i,j}^{\text{no-cross}} &= z_{i,j}(1 + \eta_{i,j}), \\ r_{i,j}^{\text{no-min}} &= z_{i,j}(2 + \chi_i). \end{aligned} \tag{20}$$

`accuracy_only` retains only the unit correctness reward. `no_cross_strategy` removes the strategy-contrast bonus while retaining the within-tool depth preference. `no_min_evidence` removes the depth-dependent distinction by setting the non-excess indicator to one during reward assignment. As a result, successful trajectories retain the base reward of two and receive the strategy-contrast bonus whenever it is active.

`no_min_evidence` does not delete the entire $z_{i,j}\eta_{i,j}$ component. Instead, all successful trajectories receive $2 + \chi_i$, regardless of acquisition depth. All four variants assign zero reward when $z_{i,j} = 0$.

The possible raw reward values are $\{0, 1, 2, 3\}$ for full SAVE, $\{0, 1\}$ for `accuracy_only`, $\{0, 1, 2\}$ for `no_cross_strategy`, and $\{0, 2, 3\}$ for `no_min_evidence`. Reward processing, eligibility checks, and task-group rejection sampling are described in Appendix E.1.1.

D Detailed Data Construction

Both the supervised fine-tuning and reinforcement learning datasets are constructed from the training split of ChartQA Masry et al. (2022). After preprocessing, the source training split contains 28,299 question–image pairs.

ChartQA is used for training and ChartQAPro is used for evaluation, we audited potential cross-dataset overlap at both exact- and near-duplicate levels. Exact-match screening found no shared question–answer pairs, byte-identical image files, or pixel-identical decoded images. We then performed perceptual-hash candidate retrieval followed by structural-similarity verification.

For supervised tool-use initialization, Qwen3.5-397B-A17B is used to generate visual reasoning trajectories for the ChartQA question–image pairs. A trajectory may contain intermediate reasoning, optional local visual enhancement operations, visual evidence returned by the tool, and a final answer. Trajectory verification retains demonstrations with correct final responses, valid action formats, and

Table 5: Statistics of the training data used in SAVE.

Category	Data Split	Samples	Description
Source data	ChartQA training set	28,299	Original training samples after preprocessing
SFT	Trajectory set 1	6,179	Verified trajectories for tool-use supervision
SFT	Trajectory set 2	7,763	Verified trajectories for tool-use supervision
SFT	Trajectory set 3	4,455	Verified trajectories for tool-use supervision
SFT	**Merged SFT data**	**18,397**	**Unique trajectories after merging and deduplication**
RL	Medium	3,395	Medium-difficulty instances retained for policy optimization
RL	Hard	4,414	Hard-difficulty instances retained for policy optimization
RL	**Final RL data**	**7,809**	**Unique medium and hard instances after curation**

executable tool interactions when tools are invoked. SFT trajectories were generated with Qwen3.5-397B-A17B at temperature 0.6 and a maximum output length of 16,384 tokens. Top-p and top-k were not explicitly set, and the reference answer was not provided to the generator. We used two generation prompts. Under the auto-tool prompt, the teacher could either answer directly or invoke local visual enhancement; under the tool-forced prompt, one crop was required before the final answer. Each API invocation produced one trajectory, and multiple trajectories for the same question–image pair were obtained through independent generation passes. A generated trajectory was retained only if its action format and multimodal message structure were valid and, when a tool was invoked, its tool arguments, bounding box, and returned image were valid.

The three SFT sets are independently generated trajectory batches rather than difficulty-based partitions. Set 1 was generated with the auto-tool prompt, and Set 3 was generated with the tool-forced prompt. According to the available generation records, Set 2 comes from a separate auto-tool generation batch. The three sets contain 6,179, 7,763, and 4,455 trajectories, respectively. After merging, the exact-duplicate check removed no trajectories under the adopted criterion, yielding 18,397 trajectories in total. Multiple trajectories may therefore correspond to the same source question–image pair while differing in their reasoning or tool-use behavior. The SFT data are used to establish direct-answer and tool-assisted reasoning capabilities before reinforcement learning and are not partitioned by difficulty.

For reinforcement learning, we apply the difficulty-aware trajectory curation described in Section 3. Multiple candidate trajectories are generated for each instance and categorized according to their observed correctness outcomes. Easy instances are excluded from policy optimization, whereas medium and hard instances are retained. The reinforcement-learning dataset contains 7,809 unique instances, including 3,395 medium instances and 4,414 hard instances. These account for 43.48% and 56.52% of the dataset, respectively. No duplicate instances remain after filtering.

Table 5 summarizes the training data used in SAVE.

E Implementation Details

E.1 Training Configuration

SAVE uses supervised tool-use initialization followed by RLOO-based reinforcement learning. The supervised stage establishes direct-answer and tool-assisted behaviors, after which policy optimization is performed using group-wise RLOO advantages computed over the sampled trajectories. Reinforcement-learning rollouts are generated asynchronously with SGLang.

Table 6 reports the shared SFT configuration. During this stage, the visual encoder and multimodal projector are frozen, while the language model is optimized on verified tool-use demonstrations. Table 7 reports the reinforcement-learning configuration, including rollout sampling, sequence-length limits, tool-interaction budgets, and policy-optimization settings. Backbone-specific distributed training settings are provided in Table 8, including FSDP-based training and Megatron training with MBridge.

Table 6: Shared supervised fine-tuning configuration of SAVE. The visual encoder and multimodal projector are frozen during SFT, while the language model is fully optimized.

Category	Configuration	Description
Training stage	SFT	Supervised fine-tuning with tool-use demonstrations
Fine-tuning type	Full	Full-parameter optimization of the trainable language model
Vision encoder	Frozen	Visual backbone parameters are not updated
Multimodal projector	Frozen	Vision-language projection parameters are not updated
Language model	Trainable	Language model parameters are optimized during SFT
Distributed optimization	DeepSpeed ZeRO-3	Memory-efficient distributed parameter optimization
Training data	ChartQA-Train	Tool-use demonstration data used for supervised fine-tuning
Maximum sequence length	20,480 tokens	Maximum input sequence length during training
Maximum training samples	100,000	Upper bound on the number of training samples
Image resolution limit	262,144 pixels	Maximum number of input image pixels
Video resolution limit	16,384 pixels	Maximum number of input video pixels
Learning rate	1.0×10^{-5}	Initial learning rate for parameter optimization
Optimizer schedule	Cosine	Cosine learning-rate decay
Warm-up ratio	0.1	Fraction of training steps used for learning-rate warm-up
Training epochs	3	Number of passes over the training dataset
Per-device batch size	1	Number of samples processed per device per iteration
Gradient accumulation	1	Number of iterations accumulated before each optimizer update
Precision	bfloat16	Mixed-precision training format
Preprocessing workers	16	Number of parallel workers for dataset preprocessing
Dataloader workers	4	Number of workers used for training data loading

Table 7: Shared training configuration of SAVE.

Category	Configuration	Description
Training algorithm	RLOO	Group-relative reinforcement learning optimization
Advantage estimator	RLOO	Computes relative advantages from sampled trajectories
Optimizer	AdamW	Optimization method for policy updates
Learning rate	1×10^{-6}	Actor optimization learning rate
Weight decay	0.01	Parameter regularization coefficient
Warm-up steps	0	No learning-rate warm-up
Rollout samples per prompt	16	Number of sampled trajectories for each prompt
Prompt batch size	16	Number of prompts in each training batch
Total rollout samples per step	256	Total trajectories generated per optimization step
PPO mini-batch size	16	Mini-batch size for policy optimization
PPO micro-batch size	1/GPU	Memory-efficient gradient computation
Maximum prompt length	16,384 tokens	Maximum input context length
Maximum response length	16,384 tokens	Maximum generated response length
Maximum tool calls	4	Maximum visual tool invocation per trajectory
Maximum model calls	5	Maximum agent interaction turns
Rollout engine	Async SGLang	Asynchronous trajectory generation
Sampling temperature	Backbone-specific	Controls rollout sampling diversity
Top-p	Backbone-specific	Nucleus sampling parameter
Top-k	Backbone-specific	Candidate token filtering parameter
KL loss coefficient	0.001	KL regularization weight
Entropy coefficient	0.0	No additional entropy regularization
Clip ratio high	0.28	Upper clipping range for policy updates
Loss aggregation	seq-mean-token-sum	Token-level policy loss aggregation
Precision	bfloat16	Training numerical precision
Rejection sampling	Enabled	Rejects task groups with no eligible episodes or rewards
Invalid trajectory masking	Enabled	Masks failed tool interactions

E.1.1 Reward Processing and Trajectory Filtering

The reward-assignment stage uses the raw values specified in Equation (20). No min–max normalization, division by the maximum reward, or rescaling to a common interval is applied during reward assignment. The workflow initially emits binary correctness rewards at designated reward-token positions. The trainer preserves these positions and replaces the binary value at each position with the assigned trajectory reward. Reward assignment is performed before and separately from advantage estimation.

The direct and tool-assisted strategy groups are defined in Equation (6). At the implementation level, the eligibility indicator $v_{i,j}$ is set to one only when `is_valid=True` and `masked=False`; otherwise, it is set to zero. If an episode occupies multiple rows in the workflow batch, the trainer

Table 8: Backbone-specific training configurations.

Backbone	Training Framework	Parallel Strategy	Sampling Configuration
Qwen2.5-VL-7B	FSDP	Rollout TP=4	Temp=1.0, Top-p=1.0, Top-k=-1
Qwen3-VL-8B	Megatron + MBridge	Train TP=8, Rollout TP=8	Temp=0.7, Top-p=0.8, Top-k=20
InternVL3.5-8B	FSDP	Rollout TP=8	Temp=0.6, Top-p=0.95, Top-k=50

Table 9: Experimental environment.

Category	Configuration	Description
GPU	8 × NVIDIA A800-SXM4-80GB	Training accelerators
CPU	64 cores / 128 threads	Host computation resources
Operating System	Linux 4.19	Runtime environment
Python	3.12.13	Programming environment
CUDA	12.8/12.9	GPU computing platform
PyTorch	2.8.0–2.9.1	Deep learning framework
Distributed Training	FSDP / Megatron	Distributed optimization framework
Rollout Framework	SGLang	Asynchronous inference engine
RL Framework	verl	Reinforcement learning framework
Workflow Framework	rllm workflow	Agent trajectory management
Parallel Execution	Ray	Distributed task scheduling
Precision	bfloat16	Mixed precision training

constructs one episode-level record for reward comparison and broadcasts the assigned reward back to the corresponding rows.

Only eligible episodes are used to determine whether each strategy is represented, whether it has an observed success, and the minimum successful tool-call count. Eligible trajectories with incorrect answers remain in their corresponding strategy groups. Ineligible episodes are excluded from these group statistics and receive zero reward. They are not counted as unsuccessful direct or tool-assisted samples when strategy presence and observed strategy success are determined.

Group-wise rewards are assigned before rejection sampling. For each task, the rejection decision uses only the rewards of eligible episodes. A task group is discarded if it contains no eligible episodes, if all eligible rewards are zero, or if all eligible rewards are identical. All other task groups are retained. Retained groups are accumulated across candidate batches until the configured prompt batch size is reached.

Rejection sampling operates at the task-group level rather than replacing individual invalid rollouts within a retained task. All reward variants use the same rejection rule, applied to the rewards assigned by the corresponding variant. Because the assigned reward distributions differ across variants, the same candidate data and rejection rule can produce different sets of retained task groups.

Exclusion from reward-comparison statistics does not immediately remove the corresponding rows from the workflow batch. For advantage estimation, the trainer uses `step_ids` as the grouping key and `traj_rewards` when stepwise advantages are disabled. After reward assignment and task-group rejection sampling, padding rows are removed before `compute_advantage` is called. Rows with `is_valid=False` are explicitly removed after advantage computation and before the actor update. Thus, reward-comparison eligibility, advantage computation, and the final actor-update row filter are distinct stages of the training workflow.

E.2 Experimental Environment

All experiments are conducted on a single-node multi-GPU server. Table 9 summarizes the hardware and software environment.

F Prompts and Tool Schema

This section documents the prompts, visual tool interface, and answer-evaluation protocol used in SAVE. At inference time, the model may either answer from the current visual observation or request additional local evidence before producing its final answer.

F.1 System Prompt

The system prompt defines the interaction protocol and the normalized coordinate convention used for visual cropping.

System Prompt

You are an image-question agent that solves problems step by step.
Protocol:
- If you call a tool, the environment will return exactly one new processed image inside a tool response message.
- After receiving a tool response, prioritize the returned image as the latest evidence.
Tool policy:
- Every local_enhance call must include: bbox=[x1,y1,x2,y2].
- The model should provide only bbox.
About box:
- The bbox represents a region by its relative position within the image: [x1,y1,x2,y2].
- The coordinates are normalized integer values in the range [0,1000].
- They do not refer to absolute pixel positions.
- x=0 means the far left of the image, and x=1000 means the far right.
- y=0 means the top of the image, and y=1000 means the bottom.
Actual tool call format:
<tool_call> { "name":"local_enhance", "arguments": { "operation":"crop", "bbox":[x1,y1,x2,y2] } } </tool_call>
- If answering the question, output exactly one lowercase <answer>...</answer> block.
- Do not include a tool call and a final-answer block in the same turn.

F.2 Autonomous Tool-Use Instruction

Each inference instance contains the original image and question. The instruction asks the model to assess whether the available visual evidence is sufficient before deciding whether to invoke the crop tool.

User Instruction Template

<image>
Question:
{question}
Original image size: width={original_image_width}, height={original_image_height}
Requirements:
- First judge whether the current image provides enough information to answer the question reliably.
- If the current image is sufficient, do not call any tool.
- If the current image is insufficient and local enhancement is necessary, call the visual tool on the most relevant region.
- If a tool is called, only output the tool call in the current step.
- After receiving the processed image, continue reasoning and provide the final answer inside the answer field.

F.3 Tool-Call Schema

The visual crop operation is exposed through `local_enhance`. Its serialized tool-call format is shown below.

Tool Call Format

```
<tool_call> { "name":"local_enhance", "arguments": { "operation":"crop", "bbox":[x1,y1,x2,y2] } } </tool_call>
```

F.4 Tool-Response Format

A valid tool invocation returns one cropped image in the subsequent tool-response message.

Tool Response Format

<tool_response>
<image>
Output image returned by the visual crop tool.
The model should reason based on the returned image and generate the final answer.
</tool_response>

F.5 JUDGE-BASED ANSWER EVALUATION

Qwen3.5-397B-A17B is used as a semantic answer evaluator for model predictions and does not generate the predictions being evaluated. The complete equivalence criteria are given in the evaluation prompt below.

Model outputs were parsed using explicit XML-style action delimiters. A final-answer turn was required to contain exactly one lowercase <answer>...</answer> block, whereas a tool-use turn was required to contain exactly one <tool_call>...</tool_call> block. The two action types were mutually exclusive within a turn. Outputs containing both action types, neither action type, or multiple action blocks were treated as invalid rather than resolved heuristically. For a valid answer turn, surrounding whitespace was removed from the content inside the <answer> tags and the remaining text was used as the prediction. No free-text fallback was applied when the final <answer> field was absent. A valid tool call triggered tool execution and another model turn, while a trajectory that failed to produce a final answer within the configured interaction limit was treated as failed. Invalid or failed model trajectories were scored as incorrect.

The judge used temperature 0, a maximum generation length of 4,096 tokens, and disabled reasoning mode. Top-p was not explicitly specified by the evaluation client and therefore followed the inference-server setting. The judge returned a JSON object containing a Boolean `is_correct` field and a short textual reason. The parser accepted plain JSON as well as JSON embedded in code fences, auxiliary text, or reasoning tags, and selected a parsed object containing `is_correct`. If a valid Boolean judgment could not be obtained, the same prediction was re-evaluated for up to three judge attempts until a valid result was obtained. These repeated judgments operated on the same fixed prediction and did not constitute additional model rollouts.

Judge Evaluation Prompt

You are an answer equivalence evaluator.
Determine whether the predicted answer is equivalent to the gold answer for the given question. Judge semantic correctness rather than exact string matching.
Evaluation rules:
- Ignore harmless formatting differences such as capitalization, spaces, punctuation, LaTeX wrappers, brackets, and answer tags.
- For yes/no or true/false questions, treat semantically equivalent boolean expressions as the same answer. Interpret 0 or 1 as boolean values only when the question clearly uses binary encoding.
- For multiple-choice questions, accept either the correct option label or the corresponding option content.
- For numerical or mathematical answers, accept equivalent representations of the same value or expression, but do not accept a different value merely because it is close to the gold answer.
- A prediction may include additional explanation, but it must contain a clear answer consistent with the gold answer. Do not judge an answer correct based only on substring overlap.
If the prediction contradicts the gold answer, gives a different value, selects the wrong option, or does not provide enough information to establish equivalence, mark it incorrect.
Return JSON only:{"is_correct": true, "reason": "brief explanation"}
Question:{question}
Predicted answer:{pred}
Gold answer:{gold}

Table 10: Paired bootstrap confidence intervals for the main SFT–SAVE comparison across three vision-language backbones and five benchmarks. Values are Avg@3 percentages. Δ denotes SAVE minus SFT in percentage points.

Backbone	Benchmark	SFT Avg@3 [95% CI]	SAVE Avg@3 [95% CI]	Δ [95% CI]
Qwen2.5-VL-7B	CharXiv	35.67 [34.00, 37.30]	45.51 [43.70, 47.27]	+9.84 [7.47, 12.23]
	HallusionBench	62.67 [60.88, 64.42]	72.53 [70.91, 74.06]	+9.86 [7.36, 12.30]
	RealWorldQA	40.92 [39.00, 42.88]	65.18 [63.27, 67.19]	+24.26 [21.48, 27.06]
	HR-Bench-8K	31.29 [29.58, 33.00]	59.26 [57.25, 61.21]	+27.97 [25.38, 30.50]
	ChartQAPro	32.17 [30.97, 33.35]	34.00 [32.80, 35.23]	+1.83 [0.12, 3.52]
Qwen3-VL-8B	CharXiv	40.77 [39.07, 42.47]	43.17 [40.73, 45.67]	+2.40 [-0.60, 5.50]
	HallusionBench	71.14 [69.47, 72.84]	71.92 [69.33, 74.48]	+0.78 [-2.28, 3.82]
	RealWorldQA	60.83 [58.82, 62.83]	63.13 [61.13, 65.10]	+2.30 [-0.74, 5.27]
	HR-Bench-8K	67.17 [64.46, 69.83]	69.08 [66.33, 71.75]	+1.91 [-0.12, 3.96]
	ChartQAPro	41.66 [40.40, 42.90]	42.78 [41.50, 44.06]	+1.12 [-0.68, 2.94]
InternVL3.5-8B	CharXiv	32.80 [30.47, 35.17]	45.90 [43.37, 48.47]	+13.10 [10.90, 15.37]
	HallusionBench	68.10 [65.58, 70.63]	78.23 [75.92, 80.55]	+10.13 [8.03, 12.20]
	RealWorldQA	56.91 [54.03, 59.69]	62.66 [59.83, 65.45]	+5.75 [3.70, 7.80]
	HR-Bench-8K	51.38 [48.54, 54.13]	60.29 [57.54, 63.04]	+8.92 [6.54, 11.29]
	ChartQAPro	38.89 [37.03, 40.74]	43.69 [41.80, 45.55]	+4.79 [3.49, 6.14]

G Additional Experimental Analyses

G.1 Statistical Uncertainty of Main Results

To quantify the uncertainty of the main SFT–SAVE comparisons in Table 1, we perform paired bootstrap resampling at the question level. For each question, the three sampled responses are kept together as a single resampling unit rather than treated as independent observations. SFT and SAVE predictions are aligned by the same question index, and each bootstrap replicate uses the same resampled question indices for both conditions. The confidence interval for the performance difference is computed directly from the bootstrap distribution of $\mathrm{SAVE} - \mathrm{SFT}$, rather than by subtracting independently estimated confidence-interval endpoints. We use 10,000 bootstrap replicates with random seed 20260921 and report two-sided percentile 95% confidence intervals using NumPy's `linear` quantile interpolation. No duplicate keys, missing keys, incomplete three-sample groups, or additional question exclusions occur in these comparisons.

Table 10 reports the resulting intervals across the three backbones and five benchmarks. SAVE obtains a higher Avg@3 point estimate than SFT in all 15 backbone–benchmark comparisons. For Qwen2.5-VL-7B, the paired 95% confidence interval of the SAVE–SFT difference excludes zero on all five benchmarks. The same pattern holds for InternVL3.5-8B. For Qwen3-VL-8B-Instruct, all five point estimates remain positive, but the corresponding confidence intervals include zero. Overall, the SAVE–SFT difference has a 95% confidence interval excluding zero in 10 of the 15 individual comparisons.

G.2 Inference-Time Tool-Access and Stopping Diagnostics

We conduct two inference-only diagnostics. The first measures the effect of enabling the visual crop tool under a matched protocol, while the second examines the effect of one additional crop after an autonomous stop or continue decision. Neither analysis involves additional training or parameter updates.

G.2.1 Controlled Tool-Access Sensitivity

We evaluate Qwen2.5-VL-7B SFT and SAVE on HallusionBench and HR-Bench-8K under No-tool and Auto-tool inference. No-tool disables visual-tool invocation, whereas Auto-tool allows the model to invoke the crop tool autonomously. Each question is sampled three times under every model and tool-access condition. The runs were independently regenerated under a matched protocol with fixed image-processing and response budgets, deterministic per-task seeds, strict content-only action

Table 11: Tool-access sensitivity analysis with Qwen2.5-VL-7B. No-tool disables visual-tool invocation, whereas Auto-tool allows autonomous visual evidence acquisition. Gain denotes Auto-tool minus No-tool in percentage points. All confidence intervals are paired 95% bootstrap intervals.

Dataset	Checkpoint	No-tool Avg@3 [95% CI]	Auto-tool Avg@3 [95% CI]	Gain (pp) [95% CI]
HallusionBench	SFT	51.408 [48.984, 53.868]	64.884 [62.353, 67.380]	+13.476 [11.479, 15.437]
HallusionBench	SAVE	57.077 [54.617, 59.572]	67.201 [64.669, 69.697]	+10.125 [8.306, 11.907]
HR-Bench-8K	SFT	26.583 [24.500, 28.708]	29.417 [27.208, 31.667]	+2.833 [0.583, 5.042]
HR-Bench-8K	SAVE	28.333 [26.042, 30.667]	36.958 [34.583, 39.458]	+8.625 [6.542, 10.708]

Table 12: One-step stopping diagnostic on HR-Bench-8K with Qwen2.5-VL-7B. Histories are partitioned using an independently sampled autonomous decision from each checkpoint. Stop evaluates answering immediately from the frozen history, whereas Continue-one forces exactly one additional model-selected crop before answering. The final column reports Continue-one minus Stop in percentage points.

Check point	Decision group	Histories	Stop Avg@3 [95% CI]	Continue-one Avg@3 [95% CI]	Δ (pp) [95% CI]
SFT	stop	27	51.852 [34.568, 69.136]	44.444 [28.395, 60.494]	-7.407 [-12.346, -2.469]
SFT	continue	63	46.032 [35.979, 56.085]	46.561 [37.037, 56.085]	+0.529 [-7.407, 8.466]
SFT	invalid	8	16.667 [4.167, 33.333]	25.000 [8.333, 45.833]	+8.333 [-8.333, 25.000]
SAVE	stop	41	45.528 [31.707, 59.350]	37.398 [26.016, 49.593]	-8.130 [-15.447, -1.626]
SAVE	continue	46	43.478 [33.333, 53.623]	47.826 [36.957, 58.696]	+4.348 [-5.797, 14.493]
SAVE	invalid	11	36.364 [18.182, 54.545]	27.273 [12.121, 42.424]	-9.091 [-24.242, 9.091]

validation, protocol failures scored as incorrect, and a unified semantic evaluator. Their absolute scores are therefore treated as protocol-specific sensitivity results rather than exact reproductions of earlier evaluations.

We compute paired 95% confidence intervals using 10,000 question-level bootstrap replicates with random seed 20260918. The three samples and all model-by-condition cells for a question are kept together during resampling. On HallusionBench, 16 of 951 questions contain an unresolved evaluator outcome and are excluded as complete units, leaving 935 paired questions. All 800 HR-Bench-8K questions are retained. Table 11 shows positive Auto-tool-minus-No-tool Avg@3 differences for both models on both benchmarks, with all four confidence intervals excluding zero. The gains are 13.476 and 10.125 percentage points for SFT and SAVE on HallusionBench, and 2.833 and 8.625 points on HR-Bench-8K, respectively. These results establish positive tool-access effects within the matched protocol, but do not test whether the effect differs significantly between SFT and SAVE.

G.2.2 One-Step Stop/Continue Diagnostic

We further evaluate one-step acquisition decisions on HR-Bench-8K using a common pool of first-crop interaction histories. For each model, an independently sampled autonomous decision assigns each frozen history to the *stop*, *continue*, or *invalid* group.

Two branches are then evaluated from the same frozen state. The Stop branch answers immediately, whereas the Continue-one branch performs exactly one additional model-selected crop before answering. Each branch is sampled three times, independently of the decision used for group assignment. Of the 100 histories, two contain unresolved evaluation outcomes and are excluded as complete units, leaving 98 histories. Bootstrap resampling is clustered by question. Because SFT and SAVE assign histories to groups independently, comparisons between their model-specific strata are descriptive.

Table 12 shows that forcing another crop after a stop decision reduces Avg@3 by 7.407 percentage points for SFT 95% CI: [-12.346, -2.469] and 8.130 points for SAVE 95% CI: [-15.447, -1.626]. For histories assigned to the continue group, the Continue-one-minus-Stop estimates are positive for SFT $+0.529$ points and SAVE $+4.348$ points, but both confidence intervals include zero. The diagnostic therefore provides stronger evidence against forced continuation after a stop decision than

for a benefit of continuing after a continue decision. Invalid groups are reported only for completeness, and the results do not establish a globally optimal stopping policy.

G.3 Behavioral Statistics on HallusionBench

Figure 4(c–d) compares Base and SAVE on HallusionBench using the same answer evaluator and evaluation protocol. Base denotes the original backbone before SAVE-specific SFT and RL, whereas SAVE denotes the corresponding trained model. Qwen3-VL refers to Qwen3-VL-8B-Instruct in this analysis. The Base–SAVE comparison therefore reflects the complete SAVE training pipeline rather than isolating the effect of RL alone.

The analysis reuses the three HallusionBench evaluation trajectories per question and model condition reported in Table 1; no additional trajectories are generated. For each backbone, question IDs are matched between Base and SAVE, and only questions with three valid, evaluated trajectories under both conditions are retained. Valid trajectories with incorrect answers remain in the analysis. The two panels apply separate eligibility criteria to this shared complete-run set.

Observed strategy-success patterns. For Figure 4(c), a question is retained only when both direct-answer and tool-assisted trajectories are observed under each of Base and SAVE. For each condition, *Tool only* and *Direct only* denote observed success exclusively for the corresponding strategy, *Both* denotes at least one successful trajectory for each strategy, and *Neither* denotes no observed success for either strategy. The Base and SAVE distributions are computed on the same eligible questions and normalized by the size of that set. The eligible sets contain 109, 146, and 104 questions for Qwen2.5-VL-7B, InternVL3.5-8B, and Qwen3-VL-8B-Instruct, respectively.

Successful acquisition depth. For Figure 4(d), a question is retained when both Base and SAVE contain at least one successful tool-assisted trajectory. For each question and model condition, d^* is the minimum positive tool-call count among its successful tool-assisted trajectories. The depth distributions are normalized by the corresponding shared eligible sets, which contain 88, 115, and 94 questions for the three backbones in the same order.

The redundancy annotation uses a pooled trajectory-level denominator. For each model condition, it is the proportion of successful tool-assisted trajectories whose call count exceeds their question-specific d^* among all successful tool-assisted trajectories in the depth-analysis set. It is therefore not an average of per-question redundancy rates.

These results are conditional summaries over the panel-specific eligible subsets rather than population-level estimates over the full HallusionBench evaluation set. They describe observed strategy outcomes and acquisition depths, not ground-truth tool necessity or optimal stopping, and are distinct from the overall trajectory-level tool-use rates in Table 1.

G.4 Training-Time Reward Diagnostics

Figure 7 compares four reward variants that share the same SFT initialization, candidate training data, rollout configuration, optimization settings, and task-group rejection rule. The variants differ in reward assignment; because rejection sampling depends on the assigned rewards, they may retain different task groups.

Panels (a,d) summarize joint and exclusive strategy success at the task-group level, whereas panels (b,c) report successful-tool depth statistics at the episode level. The reduction in the redundant-correct-tool rate is largely preserved under `no_cross_strategy` but weakens under `no_min_evidence`. Together with the lower conditional tool-call depth in Figure 4(b), this pattern is consistent with the within-tool preference discouraging successful trajectories that continue beyond the minimum observed successful depth.

G.5 Direct/Tool Composition of On-Policy Rollout Groups

The composition metrics are logged on candidate rollout groups before task-group rejection. Consequently, a small *Neither* category may appear in the logged statistics, although such groups are subsequently discarded and do not enter policy optimization. Each training instance is associated



(a) Both Modes Succeed

(b) Minimum Successful Tool Rate

(c) Redundant Correct Tool Rate

(d) Cross-Strategy Contrast Rate

Figure 7: Training-time reward diagnostics for Qwen2.5-VL-7B. All variants share the same SFT initialization, candidate data, rollout and optimization settings, and task-group rejection rule, while differing in reward assignment. Because rejection is reward-dependent, the retained task groups may differ across variants. **(a)** Both Modes Succeed is the task-group-level rate at which both sampled strategies contain a successful trajectory. **(d)** Cross-Strategy Contrast is the task-group-level rate at which both strategies are sampled but exactly one contains a successful trajectory. **(b)** Minimum Successful Tool Rate and **(c)** Redundant Correct Tool Rate are episode-level rates of successful tool trajectories at the sampled group's minimum observed successful depth and strictly above that depth, respectively. Both are averaged over all logged episodes, and their sum equals the overall successful-tool episode rate.

with an on-policy group of nominally 16 rollouts. The policy does not impose a fixed split between direct and tool-assisted trajectories, so the realized composition varies across groups and training steps. We analyze the logged composition for Qwen3-VL-8B-Instruct, Qwen2.5-VL-7B-Instruct, and InternVL3.5-8B.

A trajectory is classified as *direct* if it makes no visual-tool call and as *tool-assisted* if it invokes the visual tool at least once. At optimization step t, the logger records the mean numbers of eligible direct and tool-assisted trajectories per group, $\bar{n}_t^D$ and $\bar{n}_t^T$, through `no_tool_episode_count` and `tool_episode_count`. Because these counts are computed after eligibility filtering, $\bar{n}_t^D + \bar{n}_t^T$ may be smaller than 16. The logger also records the proportions of groups containing at least one direct trajectory, at least one tool-assisted trajectory, and both strategies. We denote these quantities by s_t^D, s_t^T, and s_t^B, corresponding to `no_tool_sampled`, `tool_sampled`, and `both_modes_sampled`.

The mutually exclusive group categories are Both, Direct-only, Tool-only, and Neither. Their derived proportions are

$$p_t^{\text{D-only}} = s_t^D - s_t^B, \qquad p_t^{\text{T-only}} = s_t^T - s_t^B, \qquad p_t^{\text{neither}} = 1 - s_t^D - s_t^T + s_t^B. \tag{21}$$

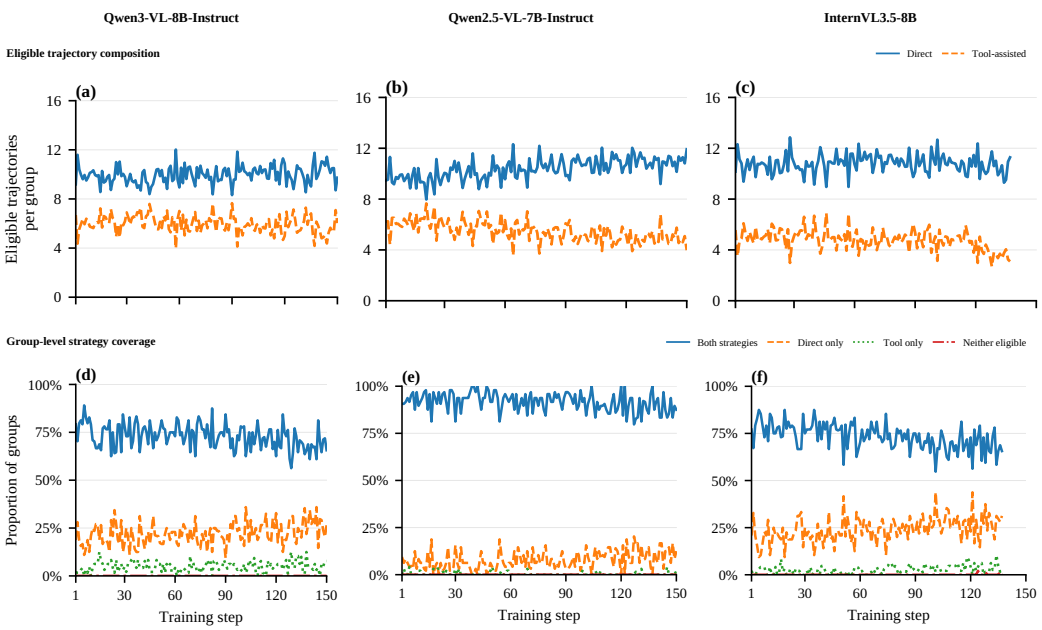


Figure 8: Realized Direct/Tool composition of on-policy rollout groups across three vision-language backbones. The top row shows the stepwise mean numbers of eligible direct and tool-assisted trajectories per group. The bottom row shows the proportions of groups containing both strategies, only direct trajectories, only tool-assisted trajectories, or neither eligible type. All curves use the raw logged values without smoothing.

Table 13: Step-averaged Direct/Tool composition of the logged on-policy groups. n_D and n_T denote the mean eligible direct and tool-assisted trajectory counts per group, and n_D^+ and n_T^+ denote their mean correct counts. "One" combines the Direct-only and Tool-only categories.

	Trajectory counts per group				Group composition (%)				
Backbone	n_D	n_T	n_D^+	n_T^+	Both	D-only	T-only	One	Neither
Qwen3-VL-8B-Instruct	9.96	5.97	4.39	1.99	73.00	22.10	4.90	27.00	0.00
Qwen2.5-VL-7B-Instruct	10.38	5.48	4.28	1.75	91.74	7.78	0.48	8.26	0.00
InternVL3.5-8B	10.79	4.78	4.91	1.67	73.32	24.06	2.58	26.64	0.04

For reporting, we additionally combine the two single-strategy categories as

$$p_t^{\text{one}} = p_t^{\text{D-only}} + p_t^{\text{T-only}} = s_t^D + s_t^T - 2s_t^B. \tag{22}$$

The Neither category contains groups for which no eligible trajectory remains after filtering.

Figure 8 shows that direct trajectories are more frequent than tool-assisted trajectories throughout the logged training windows for all three backbones. The realized groups therefore do not follow a fixed or approximately balanced $8/8$ allocation.

Both strategies appear in most groups, but single-strategy groups remain present and are predominantly direct-only. Cross-strategy reward comparison is consequently unavailable in a model-dependent subset of the on-policy groups.

Table 13 summarizes the eligible and correct trajectory counts together with strategy coverage. All entries are arithmetic means over optimization steps. The percentage columns are therefore step-averaged group proportions rather than pooled proportions over all logged groups.

Correct direct trajectories are also more numerous than correct tool-assisted trajectories in the logged groups. Because the two strategies are sampled at different frequencies, these raw counts describe the available successful training trajectories rather than a controlled comparison of strategy quality.

The exported histories contain step-level aggregate metrics but not the strategy assignment of every rollout in each group. They therefore do not support reconstruction of the full per-group joint distribution of direct and tool-assisted counts.

H Related Work

H.1 Visual Tool-Augmented Multimodal Reasoning

Modern multimodal large language models perform well on visual question answering, document understanding, chart interpretation, and related reasoning tasks, but most still rely on a fixed visual representation produced before language reasoning begins. This becomes limiting when decisive information is small, densely distributed, or difficult to resolve from the global image. Increasing image resolution can alleviate some of these failures, but applies the additional computation to every instance. Visual tool-augmented reasoning instead allows the model to request or manipulate visual evidence during inference.

DeepEyes learns active visual grounding and regional inspection through reinforcement learning Zheng et al. (2026), while OpenThinkIMG combines visual tool interfaces, trajectory generation, supervised initialization, and reinforcement learning in a unified framework Su et al. (2025). VTool-R1 similarly interleaves language reasoning with intermediate visual observations produced by image-processing operations Wu et al. (2026). ReVPT and ToolsRL extend this setting to broader visual operation spaces and reinforcement-learning objectives for tool acquisition Zhou et al. (2025); Dong et al. (2026). Related work also shows that selective zooming can improve detailed visual reasoning under constrained model capacity Kumar et al. (2025).

A broader class of methods uses executable programs or structured visual actions. CodeDance composes multiple visual operations through generated code Song et al. (2026b), whereas CodeV studies whether intermediate visual evidence actually supports the final prediction Hou et al. (2026). PERIA combines perception and interaction tools for spatial reasoning Li et al. (2026), and Thinking with Visual Grounding incorporates points and bounding boxes directly into the reasoning trajectory Zhang et al. (2026). SpatialCLI and S-Agent further extend tool-assisted reasoning to spatial and multi-view settings Zhou et al. (2026b); Dai et al. (2026).

Tool-augmented perception has also been applied beyond static images. Thinking With Videos uses selective visual interaction for long-video reasoning Zhang et al. (2025); OmniAgent formulates video understanding as iterative observation, reasoning, and action Xing et al. (2026); and VideoSearcher combines temporal localization, spatial focusing, and multimodal search Gao et al. (2026). SenseNova-MARS integrates image search, text search, and cropping for knowledge-intensive multimodal reasoning Chng et al. (2026), while DeepEyesV2 expands visual interaction with code execution and web search Hong et al. (2026).

These methods establish that visual interaction can recover information missed by static perception. However, the ability to execute a useful visual operation does not by itself determine whether that operation is necessary for a particular instance. Final-answer optimization can reinforce successful tool-containing trajectories even when some intermediate calls are redundant. This distinction between *tool capability* and *tool necessity* is central to SAVE.

H.2 Agentic Reinforcement Learning and Dynamic Tool Orchestration

Reinforcement learning is well suited to multimodal tool use because the space of possible interactions is difficult to cover with fixed supervised demonstrations. On-policy training allows a model to sample its own visual actions, observe the resulting evidence, and optimize subsequent behavior. OpenThinkIMG, DeepEyes, VTool-R1, ReVPT, and ToolsRL demonstrate that this process can elicit active visual perception beyond supervised imitation Su et al. (2025); Zheng et al. (2026); Wu et al. (2026); Zhou et al. (2025); Dong et al. (2026).

The main difficulty is credit assignment. A tool-assisted trajectory may include decisions about whether to invoke a tool, where to inspect, how to use the returned observation, and when to stop. A final-answer reward provides only one signal for all of these decisions. Correct trajectories can therefore reinforce unnecessary actions, while useful evidence acquisition may receive a negative signal when a later reasoning step fails.

Several methods address this issue through staged training or more structured optimization. DeepEyesV2 uses a cold-start stage before reinforcement learning Hong et al. (2026). ToolsRL separates acquisition of individual tool capabilities from answer-oriented optimization Dong et al. (2026). AdaReasoner introduces Tool-GRPO for multi-step tool orchestration Song et al. (2026a), SenseNova-MARS optimizes multi-tool interaction at the sequence level Chng et al. (2026), and VideoSearcher separates tool-invocation and answer objectives Gao et al. (2026).

Related approaches specialize agentic optimization to scientific, spatial, and video reasoning. ToolSciVer combines answer and tool-related objectives for scientific evidence verification Zhou et al. (2026a); PERIA trains perception and interaction behaviors jointly Li et al. (2026); Thinking with Imagination learns when to invoke a visual world simulator Zhu et al. (2026a); OmniAgent emphasizes important evidence-acquisition turns Xing et al. (2026); and Agent0-VL incorporates tools into both solving and verification Liu et al. (2025). V-Zero uses contrasting visual views to encourage grounding in task-relevant evidence Sun et al. (2026).

These advances improve how multimodal agents acquire and compose visual evidence. SAVE addresses a complementary question: before optimizing a long sequence of tool actions, the policy must determine whether external visual interaction is needed at all.

H.3 Adaptive, Efficient, and Faithful Tool Invocation

Recent work increasingly treats visual computation as an adaptive decision rather than a fixed inference step. VisionThink begins from a lower-resolution observation and requests additional visual detail when needed Yang et al. (2025). CADER follows a related idea for long videos, allowing confident instances to terminate without expensive evidence retrieval Yang et al. (2026). Multi-Head Latent Control studies a broader routing problem in which latent states are used to choose among reasoning, tool use, information requests, and other actions Ghasemabadi et al. (2026).

Other methods directly model the benefit or necessity of tool invocation. AdaTooler-V introduces a Tool Benefit Score for adaptive visual tool use Wang et al. (2026a). CodeDance balances tool exploration against excessive execution Song et al. (2026b), AutoTool regulates the distribution of short and long tool trajectories Zeng et al. (2026), and MetaForge jointly decides whether a tool should be invoked, retrieved, adapted, or created Wei et al. (2026). TRUST uses uncertainty signals to reduce both unsupported tool invocation and overconfident direct answers Zhou et al. (2026c). Beacon separates *Mode Adaptiveness* from *Tool Effect*, emphasizing that frequent tool use and effective tool use are not equivalent Wang et al. (2026c).

A related line of work asks whether individual calls deserve credit. TACO estimates the marginal contribution of tool interactions using answer probes Feng et al. (2026). VideoSearcher and ToolSciVer likewise separate or augment answer-level optimization with tool-specific signals Gao et al. (2026); Zhou et al. (2026a). CodeV and FaithEyes focus on faithfulness, showing that a correct final answer does not guarantee that the intermediate visual evidence is relevant or actually used Hou et al. (2026); Wang et al. (2026b).

These approaches address different parts of the adaptive-tool problem. Routing methods select an inference mode, benefit-aware methods estimate sample-level utility, credit-assignment methods evaluate individual interactions, and faithfulness methods verify intermediate evidence. SAVE instead derives its signal from the observed success relationship between complete direct and tool-assisted trajectories sampled for the same question.

H.4 Positioning of Our Work

SAVE formulates visual tool use as a within-question strategy-comparison problem. For each question, multiple on-policy trajectories are sampled and partitioned into direct-answer and tool-assisted groups.

The reward is constructed from their observed correctness patterns rather than from tool presence alone.

When only the tool-assisted strategy succeeds, successful tool use receives the stronger cross-group signal. When only the direct strategy succeeds, the preference is reversed. When both strategies succeed, SAVE introduces no additional cross-group preference solely because one strategy uses fewer tool calls. Within successful tool-assisted trajectories, the minimum observed successful acquisition depth is preferred over trajectories that continue with additional calls. When neither represented strategy succeeds, tool presence alone does not create a preference.

This differs from methods that estimate a separate tool-benefit score Wang et al. (2026a), predict an inference mode with an auxiliary router Ghasemabadi et al. (2026), estimate individual call contributions with answer probes Feng et al. (2026), or verify process images with an additional multimodal evaluator Hou et al. (2026); Wang et al. (2026b). CADER Yang et al. (2026) uses confidence-based inference-time routing, whereas Multi-Head Latent Control learns auxiliary control heads from latent traces. SAVE instead derives the relative signal from the empirical success relationship between naturally sampled direct and tool-assisted trajectories under the current policy.

The formulation is complementary to methods that focus on richer tool selection and long-horizon orchestration Song et al. (2026a); Chng et al. (2026); Gao et al. (2026). Those methods primarily address which operation to perform and how to compose multiple interactions. SAVE focuses on an earlier decision: whether additional visual evidence is needed, and when further acquisition should stop after successful evidence has already been obtained.

I LIMITATIONS

SAVE has several limitations. First, our experiments focus mainly on 7B–8B-scale vision-language models. Whether the same evidence-acquisition behavior transfers to substantially smaller or larger models remains to be studied.

Second, part of the evaluation relies on automatic metrics and model-based judges. Although representative cases are manually inspected, judge errors may affect semantic answer evaluation and failure attribution.

Third, the current implementation focuses on localized visual cropping. Incorrect region selection, crops that remove useful surrounding context, or low-quality local observations can limit the benefit of evidence acquisition. The conclusions therefore do not yet establish the same behavior for broader tool spaces.

Fourth, the training and evaluation data are drawn from existing multimodal benchmarks and curated visual reasoning datasets. Dataset bias and possible overlap with model pretraining corpora cannot be completely excluded.

Finally, tool-augmented multimodal systems require additional safeguards before use in high-stakes applications. Incorrect evidence acquisition or excessive reliance on intermediate tool outputs can propagate errors into the final decision. Future work should examine broader model scales, more diverse visual tools, and evaluation settings that include explicit human oversight.

J QUALITATIVE CASE STUDIES

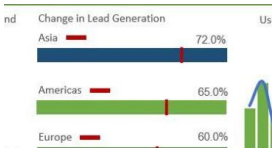
We present four illustrative cases covering unnecessary acquisition, necessary acquisition, timely stopping, and continued acquisition when the current evidence remains insufficient.

Case 1: Avoid Unnecessary Acquisition

Original Image



Unnecessary Acquisition (Contrast)



Crop 1: the "Change in Lead Generation" region.

Question: What is the difference in lead generation between the Americas and Europe?

Gold Answer: 5.

Initial Evidence: Sufficient

The original image clearly shows the required values in the "Change in Lead Generation" section:

Americas: 65.0%
Europe: 60.0%

Both values are directly readable from the original observation.

Sufficiency Assessment

The information required to solve the question is already available. The difference can be computed directly as

$$65.0 - 60.0 = 5.0.$$

No additional local inspection is necessary.

Avoid Acquisition

The displayed trajectory answers correctly from the original image without invoking the visual tool.

Tool Calls: 0

Final Prediction

The difference between the Americas and Europe is

5 percentage points.

The prediction matches the reference answer.

Contrastive Tool Trajectory

In another trajectory, the same information is already available in the original image, but the model performs an additional local crop:

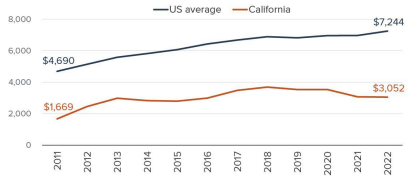
```
crop([700, 730, 900, 950])
```

The acquired view again shows 65.0% for the Americas and 60.0% for Europe. It introduces no new evidence and leads to the same semantic result, making the additional acquisition redundant.

Figure 9: Case study of direct answering from the original observation. The displayed direct-answer trajectory uses the Americas (65.0%) and Europe (60.0%) values to obtain the correct difference of 5 percentage points without invoking a tool. A contrasting one-crop trajectory reads the same two values and reaches the same answer. This example illustrates that a correct answer can be obtained without additional acquisition, even when a tool-assisted alternative also succeeds.

Case 2: Acquire Evidence When Necessary

Original Image



Acquired Evidence



Crop 1: the lower-left region containing the starting years.

Question: What is the value of US average in 2010?

Gold Answer: Unanswerable.

Initial Evidence: Ambiguous

The requested year is close to the beginning of the displayed time series. The first visible value is $4,690, but its association with the requested year is uncertain from the initial observation.

Decision: Acquire focused evidence to verify the starting year of the series.

Evidence Acquisition

The model inspects the lower-left portion of the chart containing the initial x-axis labels.

Tool Call 1: `crop([0, 800, 400, 1000])`

After Acquisition: Sufficient

The acquired view clearly shows that the x-axis begins with 2011, followed by 2012, 2013, and 2014. Therefore, the value $4,690 corresponds to the first displayed year rather than 2010.

Decision: The chart does not contain a data point for 2010.

Final Prediction

The displayed series starts in 2011, so no US average value for 2010 is provided.

Unanswerable.

The prediction matches the reference answer.

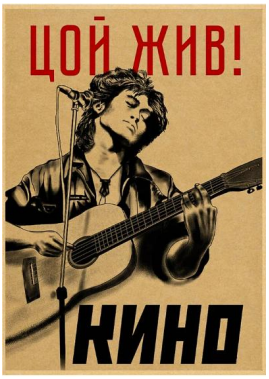
Contrastive No-Tool Trajectory

Without additional inspection, another trajectory assumes that the first displayed value corresponds to the requested year and predicts **$4,690**. This incorrectly assigns the 2011 value to 2010.

Figure 10: Case study of acquiring visual evidence to resolve a year-label ambiguity. The displayed tool-assisted trajectory crops the initial x-axis labels, reveals that the series begins in 2011, and correctly returns "Unanswerable" for 2010. A contrasting no-tool trajectory incorrectly assigns the first plotted value, $4,690, to 2010.

Case 3: Stop Once Evidence Is Sufficient

Original Image



Acquired Evidence



Crop 1: the lower region containing the band name.

Question: According to the lower text given in the image, is this poster for the DDT band?

Gold Answer: No.

Initial Evidence: Insufficient

The relevant evidence is the band name printed near the bottom of the poster. A local inspection is used to verify the lower text before answering the question.

Decision: Acquire local visual evidence.

After First Acquisition: Sufficient

The cropped region clearly reveals the lower text as "KINO". This provides the visual evidence needed to distinguish the poster from DDT.

Tool Call 1: `crop([0, 750, 1000, 1000])`

Decision: The acquired evidence is sufficient; no further visual evidence is needed.

Stop Acquisition

The displayed trajectory makes one crop call and then produces the correct final answer without further tool use.

Final Prediction

The lower text identifies the band as "KINO", rather than DDT.

No.

The prediction matches the reference answer.

Contrastive Trajectory

In another trajectory, the first crop already reveals the same "KINO" evidence, but the model repeats the same crop:

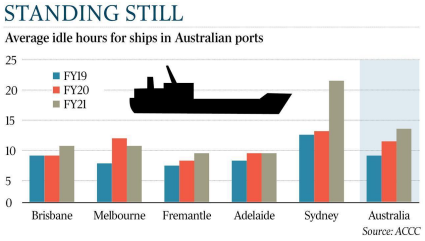
`crop([0, 750, 1000, 1000])`

The repeated acquisition provides no new visual evidence and the trajectory eventually produces the incorrect prediction **Yes**.

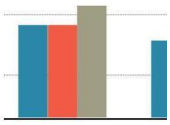
Figure 11: Case study of stopping after an informative visual acquisition. A local crop of the poster's lower text identifies the band as KINO rather than DDT. The displayed successful trajectory stops tool use after this single crop and correctly answers "No". A contrasting trajectory repeats the crop of the same region and ultimately answers "Yes" incorrectly. The comparison illustrates different continuation behaviors after inspecting the relevant visual content.

Case 4: Continue When Evidence Remains Insufficient

Original Image

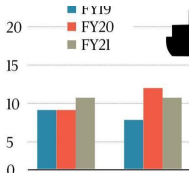


First Acquisition



Crop 1: a tight region around the Brisbane bars.

Second Acquisition



Crop 2: an expanded region including the y-axis and Brisbane bars.

Question: What is the average idle hours for ships in Brisbane over the three fiscal years?

Gold Answer: 9.67 hours.

Initial Evidence: Insufficient

The original chart contains multiple port groups and three fiscal-year bars per port. The Brisbane values are located in a small region on the left side of the chart, but the exact bar heights cannot be read reliably from the full image.

Decision: Acquire local visual evidence.

After First Acquisition: Still Insufficient

The first crop focuses on the Brisbane bars, but the region is too tight and does not preserve enough of the y-axis scale and surrounding chart context for reliable value estimation.

Tool Call 1: `crop([50, 500, 250, 850])`

Decision: The acquired evidence is still insufficient; continue evidence acquisition.

After Second Acquisition: Sufficient

The second crop expands the inspected region to include the y-axis scale and the Brisbane bar group. The three fiscal-year values can now be estimated from the chart, providing sufficient evidence to answer the question.

Tool Call 2: `crop([0, 300, 350, 850])`

Decision: Evidence is sufficient; stop acquiring additional visual evidence.

Final Prediction

The model estimates the Brisbane values at approximately 9, 9, and 11 hours and answers:

approximately 9.67 **hours**.

The reference answer is 9.67 hours, and the prediction is judged correct.

Figure 12: Case study of continuing acquisition to recover missing chart context. The first crop isolates the Brisbane bars but omits the labeled y-axis scale. A second, broader crop includes both the relevant bars and the scale, allowing the bar heights to be interpreted in context. The displayed trajectory then stops tool use and estimates an average of approximately 9.67 hours across the three fiscal years, matching the reference answer.