LEARN HOW BEFORE WHEN: DECOUPLING CAPABILITY AND ROUTING FOR VISUAL TOOL USE

Anonymous authors
Paper under double-blind review

ABSTRACT

Tool-augmented reasoning requires learning to answer in both the direct and tool modes and to route each question appropriately. Existing reinforcement learning with verifiable rewards (RLVR) methods often train mode capability and routing jointly and can exhibit phenomena like tool overuse or tool collapse, yet the connection between these behaviors and joint training remains unclear. We analyze GRPO advantages and their effects on one-step changes in mode capability and routing. The analysis shows how a mode can lose sampling probability while its capability is still improving, and why reward incentives can favor tool use without a corresponding accuracy benefit. Training diagnostics under representative reward settings further illustrate sharp declines in mode frequency and persistent tool calls under sustained balancing. Motivated by these findings, we propose a two-stage framework that separates mode capability learning from routing. We first use GRPO to train a shared expert under explicit mode instructions, then compare its performance across modes to guide an autonomous student through on-policy self-distillation. This design transfers mode capabilities and question-specific routing without additional routing rewards or large-scale supervised tool-use trajectories. Experiments show accuracy gains of up to 14.13 percentage points over Qwen2.5-VL-7B on high-resolution perception benchmarks. Distillation process further substantially reducing tool use rate, supporting more effective tool-use decisions.

1 INTRODUCTION

Recent advances in vision-language models (VLMs) have expanded their ability to interpret visual content and reason about complex questions (Liu et al., 2023; Li et al., 2025). Building on these models, visual agents incorporate external operations into multimodal chains of thought, acquiring or manipulating evidence through visual search, cropping, and image editing (Wu & Xie, 2024; Hu et al., 2024; Fu et al., 2025). An agent can inspect a relevant region, incorporate the returned observation, and continue reasoning with details that were difficult to resolve in the original image. This interaction makes visual reasoning an active process in which intermediate reasoning steps guide the acquisition of further evidence.

Reinforcement learning with verifiable rewards (RLVR) has advanced this paradigm by improving reasoning and tool use through feedback on self-generated trajectories (Su et al., 2025; Zheng et al., 2026; Lai et al., 2026). Much of this progress has focused on developing the ability to execute tools and use their outputs to answer questions. We use *mode capability* to mean the ability to answer correctly within the direct mode or the tool mode, and *routing* to mean the choice between them. Efficient agents also need to decide when those operations are warranted. A model may reliably crop and interpret an image region yet repeatedly do so on questions it could already answer directly, incurring additional computation without improving accuracy. Selective tool use therefore requires assessing a question's demands relative to the model's mode capabilities: answering directly when the direct mode suffices and seeking tool assistance when further visual evidence is needed.

Recent RL-based methods encourage this behavior through efficiency penalties, adaptive incentives, and rewards informed by estimated tool benefits (Yang et al., 2025; Song et al., 2026; Ma et al., 2026; Wang et al., 2026). These designs have improved inference efficiency and encouraged more selective tool use. In many such approaches, however, mode capability and routing continue to evolve within

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.

the same RL process. Current mode capabilities determine the benefit of tool use, while routing determines the opportunities to improve each mode. Prior observations that tool use can collapse even as accuracy improves, and that increasing call frequency yields limited additional gains, make this interaction particularly consequential (Kim et al., 2026). How reward-driven updates coordinate mode capability and routing therefore remains a central question beyond the design of efficiency incentives.

We study this question through a question-level decomposition of GRPO advantages into inter-mode competition and within-mode learning. Our analysis, supported by training diagnostics under representative reward settings, shows that improving mode capability need not lead to appropriate routing. These findings motivate separating capability development from learning when to use tools.

Motivated by this analysis, we propose a two-stage framework that separates mode capability learning from routing. We first train a shared expert under explicit instructions for the direct and tool modes, computing advantages separately within each instruction-conditioned rollout group so that both receive dedicated training opportunities. We then compare the frozen expert's performance across modes for each question and transfer the preferred behavior to an autonomous student through on-policy self-distillation (Agarwal et al., 2024; Zhao et al., 2026). The student learns on its own generated trajectories, integrating mode capability and routing into a single inference policy. This arrangement reduces reliance on additional supervised tool-use demonstrations and learns routing without introducing another routing reward into RL.

Experiments show that our final 7B model improves accuracy over its Qwen2.5-VL-7B backbone by 14.13, 7.50, and 6.75 percentage points on V^*Bench, HRBench-4K, and HRBench-8K, respectively. Compared with the expert under autonomous inference, the distilled student gains 8.38 and 2.00 percentage points in accuracy on the two HRBench variants while reducing tool-call rates by 66.0% and 70.5%, supporting more effective tool-use decisions.

Our contributions are threefold:

- We connect a question-level decomposition of GRPO advantages to one-step changes in mode capability and routing. Our analysis and training diagnostics reveal that capability improvement need not yield appropriate mode selection.
- We propose a two-stage framework that first trains a shared dual-mode expert and then learns autonomous routing through expert-guided on-policy self-distillation, without additional routing rewards.
- Our method improves accuracy over its backbone on high-resolution perception benchmarks and outperforms learned and score-based routing baselines. Distillation yields higher accuracy with fewer tool calls than the expert under autonomous inference.

2 Related Work

Visual Tool-augmented Reasoning. V^*, Visual Sketchpad, and ReFocus augment visual reasoning through guided search, drawing, and image editing (Wu & Xie, 2024; Hu et al., 2024; Fu et al., 2025). Pixel Reasoner and Mini-o3 combine supervised initialization with RL (Su et al., 2025; Lai et al., 2026), while DeepEyes, VTool-R1, and Thyme develop active perception, visual editing, and code-based image manipulation (Zheng et al., 2026; Wu et al., 2026; Zhang et al., 2026). These approaches primarily develop the tool-mode capabilities on which selective visual tool use builds.

Adaptive Tool Use. VisionThink penalizes requests for higher-resolution images, while CodeDance combines adaptive tool rewards with execution-level feedback (Yang et al., 2025; Song et al., 2026). AutoTool uses mode-specific rewards and adaptive frequency balancing, relaxing the balancing constraint later in training (Ma et al., 2026). AdaTooler-V makes rewards question-dependent using a Tool Benefit Score estimated from a reference model's performance with and without tools (Wang et al., 2026). Kim et al. (2026) document tool-use collapse and show that higher invocation frequency alone does not reliably improve performance. Many such methods still update mode capability and routing jointly, even with externally estimated tool benefits. We examine their one-step dynamics and separate capability learning from routing through expert-guided distillation.

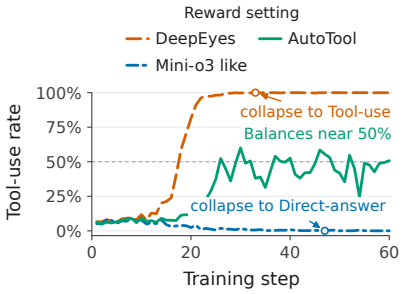

Figure 1: Tool-use frequency during joint RL training under three representative reward settings. Correctness-and-format rewards and additional tool incentives can favor opposite modes, while adaptive balancing maintains rollouts from both modes.

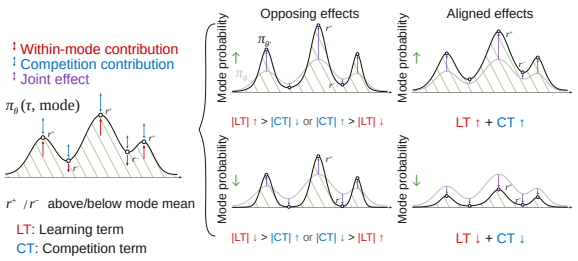

Figure 2: Schematic effects of competition and within-mode learning on mode probability: opposing (left) or aligned (right), with a net increase (top) or decrease (bottom). Curves depict joint trajectory-and-mode probability; r^+ and r^- denote rewards above and below the mode mean. Arrows reflect both advantages and trajectory gradients.

3 Understanding Joint Learning of Mode Capability and Routing

3.1 Problem Formulation

Mode capability and routing. Given an image–question pair $x \sim \mathcal{D}$, a visual agent generates a trajectory τ consisting of reasoning steps, optional tool interactions, and a final answer. Tool outputs, such as enlarged image regions, are appended to the interaction history and provide evidence for subsequent reasoning. We write $\pi_\theta(\tau \mid x, s)$ for the trajectory distribution under instruction s, and omit $s = s_{\mathrm{A}}$ when discussing autonomous tool use. A trajectory belongs to the *direct mode*, $M(\tau) = \mathrm{D}$, if it answers without external tool calls, and to the *tool mode*, $M(\tau) = \mathrm{T}$, if it invokes a visual tool at least once. Both modes may include intermediate reasoning. Let $C(x, \tau) \in \{0, 1\}$ indicate whether the final answer is correct. We define *mode capability* as the conditional accuracy $p_m(x) = \Pr_\theta(C = 1 \mid x, M = m)$, for $m \in \{\mathrm{D}, \mathrm{T}\}$. We describe *routing* by the tool-mode probability $q(x) = \Pr_\theta(M = \mathrm{T} \mid x)$; the direct-mode probability is $1 - q(x)$. Both quantities depend on the shared policy parameters θ, which we omit from scalar subscripts for clarity.

Joint optimization objective. RLVR trains the policy to maximize the expected trajectory reward $J(\theta) = \mathbb{E}_{x\sim\mathcal{D},\tau\sim\pi_\theta(\cdot|x)}[R(x, \tau)]$. Let $\bar{r}_m(x) = \mathbb{E}_\theta[R(x, \tau) \mid x, M = m]$ denote the conditional reward of mode m. The objective can be written as

$$J(\theta) = \mathbb{E}_{x\sim\mathcal{D}}[q(x)\bar{r}_{\mathrm{T}}(x) + [1 - q(x)]\bar{r}_{\mathrm{D}}(x)]\,. \tag{1}$$

The overall accuracy on question x is

$$\mathrm{Acc}(x) = q(x)p_{\mathrm{T}}(x) + [1 - q(x)]p_{\mathrm{D}}(x). \tag{2}$$

Ideally, joint optimization would fully develop both mode capabilities while adapting question-specific routing to their evolving relative strengths. Correctness-and-format rewards are compatible with this goal when format requirements are satisfied and their reward contribution becomes constant. We next examine whether practical joint RL training follows this idealized learning process.

3.2 An Empirical Puzzle: Unstable Routing During Joint Training

We consider three reward settings: a Mini-o3-like correctness-and-format reward $R = C+0.1F$ (Lai et al., 2026), the original DeepEyes reward with a successful-tool bonus (Zheng et al., 2026), and the AutoTool reward with adaptive mode balancing retained throughout training (Ma et al., 2026). Detailed reward definitions are provided in Appendix G. As shown in Figure 1, correctness-and-format rewards can make tool-mode trajectories scarce or absent, whereas a tool bonus can make them dominate. Adaptive balancing keeps both modes represented.

However, mode frequency alone does not reveal whether these shifts reflect appropriate question-specific routing or imbalanced learning. As one mode becomes rare, it contributes fewer trajectories

to subsequent on-policy updates, potentially limiting its capability development. We therefore examine the learning signals received by the two modes.

3.3 Why Does the Imbalance Arise?

We connect GRPO advantages to changes in mode capability and routing. For a group of G rollouts containing both modes, let r_i denote trajectory τ_i's reward, μ and σ the group reward mean and standard deviation, and μ_m the mean within mode m. The advantage of a trajectory in mode m decomposes as

$$A_i = \frac{r_i - \mu}{\sigma + \delta} = \underbrace{\frac{\mu_m - \mu}{\sigma + \delta}}_{\text{inter-mode competition}} + \underbrace{\frac{r_i - \mu_m}{\sigma + \delta}}_{\text{within-mode learning}}, \tag{3}$$

where $\delta > 0$ stabilizes normalization. Competition shifts all advantages in a mode according to its relative reward, while within-mode learning distinguishes better and worse attempts. When $\mu_m < \mu$, the competition term lowers advantages even for the mode's better attempts; the resulting probability changes also depend on trajectory gradients.

Following the local learning-dynamics perspective of Ren & Sutherland (2025), consider a small GRPO gradient-ascent step $\theta^+ = \theta + \eta g$ at the rollout parameters, with the sampled group and advantages fixed, clipping inactive, and no regularization or optimizer preconditioning. Define $u(\tau) = \langle \nabla_\theta \log \pi_\theta(\tau \mid x), g \rangle$ as the trajectory's relative probability growth per unit step, so that $\Delta \log \pi(\tau \mid x) \approx \eta u(\tau)$. Both advantage terms contribute to g and can affect mode capability and routing through the shared parameters. Let $\bar{u}_m$ denote the mean growth rate within mode m, and $\bar{u}_{m,c}$ its mean among trajectories with correctness c; both averages are weighted by the pre-update policy. Suppressing x, the first-order changes for $0 < q, p_m < 1$ are

$$\Delta q \approx \eta q(1-q) \underbrace{\left(\bar{u}_{\mathrm{T}} - \bar{u}_{\mathrm{D}}\right)}_{\text{tool vs. direct}}, \qquad \Delta p_m \approx \eta p_m (1 - p_m) \underbrace{\left(\bar{u}_{m,1} - \bar{u}_{m,0}\right)}_{\text{correct vs. incorrect}}. \tag{4}$$

Here Δ denotes the post-update change. Routing compares the two modes as a whole, whereas mode capability compares correct and incorrect trajectories within the same mode. Appendix F specifies g and gives the derivation.

These comparisons can disagree: correct tool trajectories may grow faster than incorrect ones ($\Delta p_{\mathrm{T}} > 0$), while the tool mode grows more slowly overall than the direct mode ($\Delta q < 0$). With the updated tool probability q^+, the next group of G independent rollouts contains Gq^+ tool trajectories in expectation and none with probability $(1 - q^+)^G$. Repeated losses of sampling probability can therefore curtail a mode's training opportunities even as its capability improves. Figure 2 illustrates how competition and within-mode learning can jointly increase or decrease mode probability.

From one-step dynamics to sustained training. Across updates, the expected number of correct tool trajectories Gqp_{T} can decrease even as p_{T} improves if q falls sufficiently quickly, limiting opportunities for further reinforcement. We track predictive concentration through mean token entropy alongside mode frequency (Wang et al., 2025a), grouping trajectories by mode and answer correctness to separate trends along successful attempts from changes in the rollout mixture. Correct answers need not have positive advantages. For a nonempty group with mode m and correctness c, the statistic is

$$\widehat{H}_{m,c} = \frac{1}{|\mathcal{G}_{m,c}|} \sum_{i \in \mathcal{G}_{m,c}} \frac{1}{T_i} \sum_{t=1}^{T_i} H(\pi_\theta(\cdot \mid h_t)), \tag{5}$$

where H is Shannon entropy over the full next-token distribution, $\mathcal{G}_{m,c}$ indexes the corresponding trajectories, T_i counts agent-generated tokens in τ_i, and h_t is their preceding history with the autonomous instruction left implicit. Tool observations are excluded from the average. The curves describe predictive concentration along changing sampled histories, providing a qualitative diagnostic rather than a direct measure of mode capability or update magnitude.

Figure 3 presents these diagnostics under the three reward settings in Section 3.2. Each panel follows 50 training steps on a fixed question, with 128 fresh rollouts per step; panels (b) and (c) use the same question. For these diagnostics, mode labels record whether a valid tool execution occurred, as detailed in Appendix G.

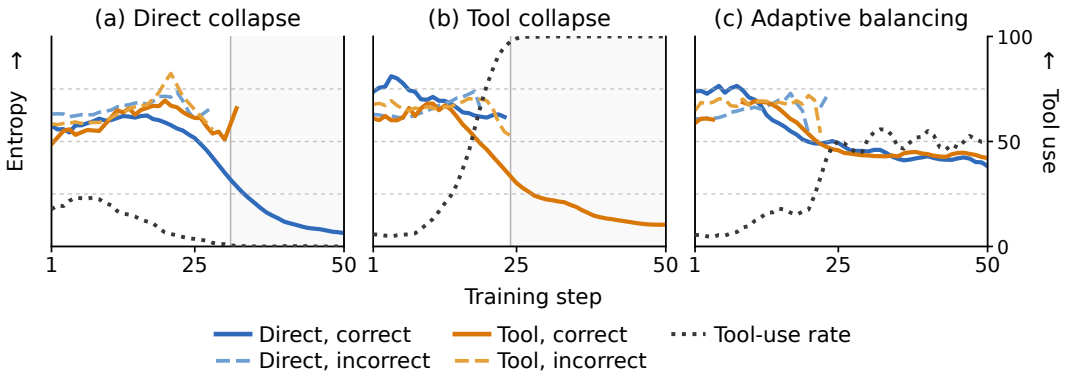

Figure 3: Mean token entropy and tool-use rate during single-question training under (a) correctness-and-format rewards, (b) a successful-tool bonus, and (c) sustained adaptive balancing. Blue/orange denote direct/tool trajectories; solid/dashed lines denote correct/incorrect answers. The dotted black line shows tool-use rate on the right axis (0–100% in every panel), while the entropy axes are independently scaled and omit numeric ticks. gaps in entropy curves indicate absent samples, not zero entropy.

(a)–(b) Concentration toward opposite modes. Under correctness-and-format rewards, tool trajectories become scarce or absent in (a); a successful-tool bonus instead drives tool-use frequency toward 100% in (b). In both cases, entropy along correct trajectories declines persistently in the dominant mode, while the other mode shows no comparable sustained decline before becoming rare. These patterns are consistent with reduced opportunities to reinforce the rarer mode, without establishing that its potential for improvement is exhausted.

(c) Sustained balancing with persistent tool use. On the same question as in (b), adaptive balancing keeps both modes represented, with decreasing entropy along correct trajectories in both. Yet tool-use frequency still fluctuates around 50% when late-stage sampled answers are correct in both modes. In this example, preserving training opportunities does not by itself eliminate tool calls with no observed correctness advantage.

Together, these cases illustrate two obstacles to the ideal coordination in Section 3.1: concentrated sampling can restrict continued mode capability learning, while sustaining both modes does not by itself align routing with their relative capabilities. This motivates first developing both mode capabilities, then using their question-specific performance to guide autonomous routing.

4 Method

4.1 Overview: Decoupling Mode Capability and Routing

Motivated by Section 3, we separate mode capability learning from routing (Figure 4). Stage I trains a shared expert under direct and tool instructions, with advantages normalized separately within each instructed group. Stage II freezes the expert, compares its two modes on a disjoint training set, and transfers the preferred mode's behavior to an autonomous student through on-policy self-distillation, without an additional routing reward. The two modes share parameters; the separation lies in how training opportunities and supervision are constructed.

4.2 Stage I: Learning Mode Capabilities

For each Stage-I question x, we sample trajectories under both prompt s_{D}, which requests answering from the original image without tools, and s_{T}, which requires visual tool use before answering. The tool crops a requested region, whose output is returned to the reasoning context. Both instructions train the same policy, providing training opportunities independently of autonomous routing.

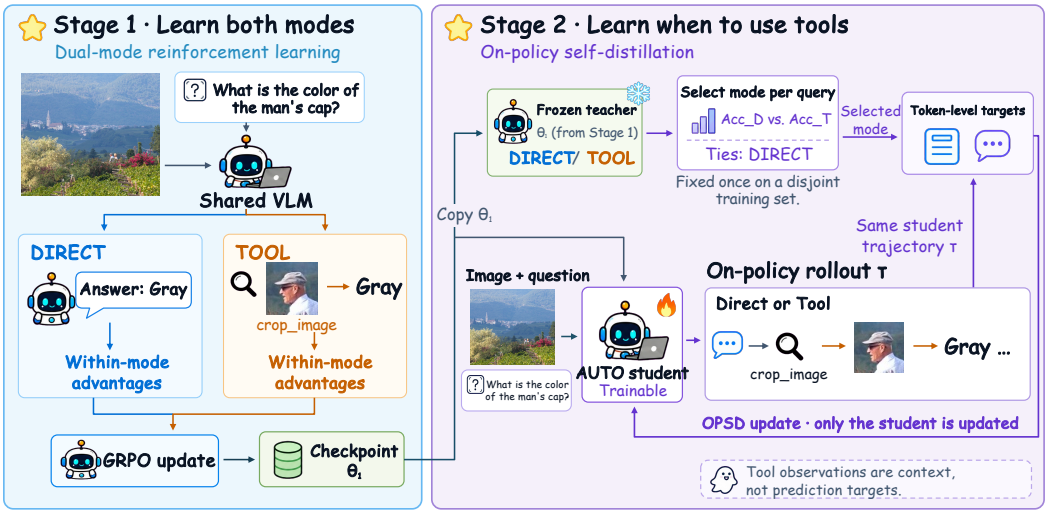

Figure 4: Overview of the two-stage training pipeline. Stage I trains a shared policy under direct and tool instructions using GRPO with advantages normalized within each instructed group. Stage II freezes this policy, selects a mode for each question on a disjoint training set, and distills its mode-conditioned token distributions into an autonomous student along the student's own on-policy trajectories.

We train with answer correctness and format rewards:

$$R(x, \tau) = C(x, \tau) + \lambda_{\mathrm{F}} F(x, \tau). \tag{6}$$

Here C is the correctness indicator defined in Section 3.1, and $F \in \{0, 1\}$ checks the response format and applicable tool-call syntax, with weight $\lambda_{\mathrm{F}} \geq 0$. There is no additional bonus or penalty for choosing either mode.

For a question and a fixed instruction $s \in \{s_{\mathrm{D}}, s_{\mathrm{T}}\}$, we sample a group of G trajectories $\{\tau_i\}_{i=1}^{G}$ from the rollout policy $\pi_{\mathrm{old}}(\cdot \mid x, s)$ and evaluate their rewards $r_i = R(x, \tau_i)$. The group-relative advantage is

$$A_i = \frac{r_i - \mu}{\sigma + \delta}, \qquad \mu = \frac{1}{G} \sum_{j=1}^{G} r_j, \tag{7}$$

where σ is the reward standard deviation within this group and $\delta > 0$ stabilizes normalization. Each instruction-conditioned group has its own reward statistics, so one mode's rewards do not enter the other mode's advantage baseline.

We update the policy with GRPO (Guo et al., 2025). Within each trajectory, let a_t denote an agent-generated token and h_t its preceding history, including the shared system message, image–question input, and any tool observations, but excluding the mode instruction s. The token probability ratio is $\rho_t = \pi_\theta(a_t \mid h_t, s)/\pi_{\mathrm{old}}(a_t \mid h_t, s)$. Writing T_i for the number of agent-generated tokens in τ_i, the training loss is

$$\mathcal{L}_{\mathrm{RL}}(\theta) = -\mathbb{E}\left[\frac{1}{G} \sum_{i=1}^{G} \frac{1}{T_i} \sum_{t=1}^{T_i} \min\{\rho_t A_i,\ \mathrm{clip}(\rho_t, 1-\epsilon, 1+\epsilon)\, A_i\}\right]. \tag{8}$$

The expectation averages over questions, both mode instructions with equal weight, and their rollout groups; the tokens and histories in each inner sum belong to τ_i. Here ϵ is the clipping threshold. Tool observations provide context but are not prediction targets. After training, we freeze the shared policy as the dual-mode expert π_{E}.

4.3 Stage II: Expert-Guided Routing via On-Policy Self-Distillation

On a disjoint training set unseen during Stage I, we sample K trajectories $\tau_k^s \sim \pi_{\mathrm{E}}(\cdot \mid x, s)$ for each question x under each instruction $s \in \{s_{\mathrm{D}}, s_{\mathrm{T}}\}$ and choose the instruction with more correct answers:

$$s^*(x) = \arg\max_{s \in \{s_{\mathrm{D}}, s_{\mathrm{T}}\}} \sum_{k=1}^{K} C(x, \tau_k^s). \tag{9}$$

Both instructions use the same K, ties favor s_{D}, and assignments are computed once. This evaluates the instructed expert's accuracy, rather than the autonomous policy's conditional accuracy p_m in Section 3.1.

We initialize the student π_θ from π_{E} and apply on-policy self-distillation (Agarwal et al., 2024; Zhao et al., 2026). The student generates complete trajectories under the autonomous instruction s_{A}, including any tool calls and returned observations. For expert scoring, we replace s_{A} with $s^*(x)$ in the original user message while retaining the shared system message and actual student history, even when the student deviates from the assigned mode.

With the mode instruction kept separate from h_t, the target token-level forward-KL objective is

$$\mathcal{L}_{\mathrm{SD}}(\theta) = \mathbb{E}_{x,\tau}\left[\frac{1}{T}\sum_{t=1}^{T} D_{\mathrm{KL}}(\pi_{\mathrm{E}}(\cdot \mid h_t, s^*(x)) \,\|\, \pi_\theta(\cdot \mid h_t, s_{\mathrm{A}}))\right], \tag{10}$$

where x comes from the Stage-II training set, $\tau \sim \pi_\theta(\cdot \mid x, s_{\mathrm{A}})$, and T counts its agent-generated tokens. We implement this objective with a sampled-token `low_var_kl` surrogate, whose connection to the forward-KL update is detailed in Appendix H. Expert weights and mode assignments remain fixed, while token targets are reevaluated on fresh trajectories from the updated student. Each update treats the expert targets and sampled histories as constants, without backpropagating through trajectory sampling.

The loss covers reasoning, tool calls, and final answers; tool observations provide context but receive no prediction loss. The selected expert thus supervises both mode capability and routing, without a separate routing objective. At inference, the student uses only the autonomous instruction to decide when to invoke the visual tool and produce its answer. All prompts are provided in Appendix E.

5 Experiments

5.1 Experimental Setup

Baselines and Benchmarks. We compare our method with three categories of baselines: (1) proprietary models, including GPT-4o (OpenAI, 2024) and GPT-4.1 (OpenAI, 2025); (2) open-source general-purpose VLMs, including InternVL3-8B and InternVL3-38B (Zhu et al., 2025), as well as Qwen2.5-VL-7B-Instruct and Qwen2.5-VL-32B-Instruct (Bai et al., 2025); and (3) models designed for visual tool use or adaptive visual acquisition, including DeepEyes (Zheng et al., 2026), Pixel Reasoner (Su et al., 2025), ERGO-7B (Lee et al., 2026), VisionThink-Efficient (Yang et al., 2025), and AdaptVision-7B (Lin et al., 2026). Our primary evaluation focuses on V^*Bench (Wu & Xie, 2024) and HRBench (Wang et al., 2025b), which require identifying visual details in high-resolution images. We report supplementary referring-expression grounding results on RefCOCO, RefCOCO+, and RefCOCOg in Appendix C.1. We additionally evaluate on MMBench (English dev) (Liu et al., 2024) for multimodal perception and reasoning, POPE (Li et al., 2023) for object hallucination for perception; results are provided in Appendix C. Training and implementation details are provided in Appendix H.

5.2 Main Results

Perception benchmarks. Our autonomous 7B model achieves 76.38%, 71.25%, and 90.05% accuracy on HRBench-4K, HRBench-8K, and V^*Bench, respectively, improving over Qwen2.5-VL-7B by 7.50, 6.75, and 14.13 percentage points (Table 1). It outperforms all compared visual tool-use and adaptive-perception baselines on all three benchmarks, matching InternVL3-38B on HRBench-4K and exceeding it on HRBench-8K and V^*Bench. Relative to the backbone, gains are particularly

Table 1: Accuracy (%) on high-resolution perception benchmarks. Higher is better. All Qwen2.5-VL baselines use Instruct checkpoints. Gray rows denote our method and its difference from Qwen2.5-VL-7B in percentage points. Bold indicates the best accuracy in each column across all compared models. "–" denotes unavailable baseline scores or pending results for our method.

Model	Size	HRBench-4K			HRBench-8K			V^*Bench		
		FSP	FCP	Overall	FSP	FCP	Overall	Attribute	Spatial	Overall
Proprietary General VLMs										
GPT-4o	–	64.25	60.50	62.38	62.25	57.25	59.75	67.83	56.58	63.35
GPT-4.1	–	66.75	60.75	63.75	60.50	59.50	60.00	69.57	57.89	64.92
Open-Source General VLMs										
Qwen2.5-VL-7B	7B	82.75	55.00	68.88	76.50	52.50	64.50	77.39	73.68	75.92
Qwen2.5-VL-32B	32B	86.00	54.50	70.25	80.00	53.25	66.63	76.52	84.21	79.58
InternVL3-8B	8B	79.00	61.50	70.25	78.00	58.75	68.38	69.57	71.05	70.16
InternVL3-38B	38B	84.00	**68.75**	**76.38**	69.00	**60.25**	64.63	76.52	82.89	79.06
Visual Tool Use and Adaptive Perception Models										
DeepEyes	7B	–	–	73.20	–	–	69.50	–	–	83.30
Pixel Reasoner	7B	86.75	60.00	73.38	79.50	57.25	68.38	86.96	76.32	82.72
ERGO-7B	7B	83.75	57.00	70.38	79.50	55.25	67.38	84.35	81.58	83.25
VisionThink-Efficient	7B	80.50	62.50	71.50	78.75	56.00	67.38	62.61	73.68	67.02
AdaptVision-7B	7B	83.75	61.00	72.38	82.75	52.25	67.50	66.96	73.68	69.63
Ours STAGE I										
Ours (direct mode)	7B	90.25	55.00	72.62	84.25	56.50	70.38	87.83	93.42	90.05
Ours (tool mode)	7B	89.25	58.25	73.75	85.00	58.75	**71.88**	87.83	**94.74**	**90.58**
Ours STAGE II										
Ours (auto mode)	7B	**93.75**	59.00	**76.38**	**88.25**	54.25	71.25	**90.43**	89.47	90.05
Δ vs. Qwen2.5-VL-7B	–	+11.0	+4.00	+7.50	+11.75	+1.75	+6.75	+13.04	+15.79	+14.13

large in single-instance perception on HRBench-4K/8K (+11.00/+11.75 points) and spatial reasoning on V^*Bench (+15.79 points). On HRBench-4K, the autonomous student also surpasses both instructed Stage-I modes, demonstrating an advantage over consistently using either direct or tool-assisted inference.

Referring-expression grounding. On RefCOCO, RefCOCO+, and RefCOCOg, our model achieves a macro-average Precision@1 of 85.65% and an average IoU of 0.772, the highest among the compared models (Table 4, Appendix C.1).

General multimodal benchmarks. Our model achieves 87.26% accuracy on MMBench and 87.36% overall F1 on POPE, the best overall results among the compared models (Table 5). Full breakdowns appear in Appendix C.

5.3 Routing Baselines and an Empirical Oracle Ceiling

Table 2 compares autonomous policies with a learned MLP, four image-based heuristics, and an LLM-based router that select between the fixed Stage-I checkpoint's direct and tool modes. The five score-based routers match the Stage-II tool-call rate. The empirical oracle selects the mode with higher accuracy estimated from eight Stage-I rollouts per mode and question; it is a sampling-based reference rather than a strict upper bound for Stage II. Implementation details are in Appendix D.

Stage II improves over Stage-I autonomous routing by 8.38 and 2.00 percentage points on HRBench-4K and HRBench-8K, respectively, outperforming the compared routers while remaining 3.20 and 1.80 points below the empirical oracle.

5.4 Evaluation Wall-Clock Time

We compare the wall-clock time required to evaluate our autonomous policy with DeepEyes-7B and Pixel Reasoner (Su et al., 2025). All three methods run on eight NVIDIA H200 GPUs, a sampling

Table 2: HRBench accuracy and tool-call rate (%) under autonomous inference or routing between the fixed Stage-I modes. The five score-based routers match Stage II's tool-call rate. Bold indicates the best accuracy; "–" denotes an unreported oracle tool-call rate.

Model / source	Inference or routing rule	HRBench-4K		HRBench-8K	
		Accuracy	Tool-call rate	Accuracy	Tool-call rate
Stage I	Autonomous (auto)	68.00	44.50%	69.25	47.88%
Stage II	Autonomous (auto)	76.38	15.12%	71.25	14.12%
Stage I	Estimated oracle routing	**79.58**	19%-81%	**73.05**	21%-77%
Stage-I modes	Two-layer MLP router	73.56	20.25%	65.55	18.75%
Stage-I modes	Image information entropy	74.64	15.12%	67.86	14.12%
Stage-I modes	Image resolution	73.53	15.12%	66.89	14.12%
Stage-I modes	Laplacian variance	74.33	15.12%	66.19	14.12%
Stage-I modes	Canny edge density	74.80	15.12%	67.22	14.12%
Stage-I modes	LLM tool-necessity score	74.81	15.12%	67.19	14.12%

temperature of zero, and a limit of four tool-interaction turns. For all methods, timing excludes model loading and online-judge evaluation. Table 3 reports the total time to process each benchmark under these settings; the HRBench measurement combines its 4K and 8K variants.

Table 3: Evaluation wall-clock time on V^*Bench and the combined HRBench variants. Speedup is each baseline's runtime divided by ours; time saved compares our runtime with that baseline's runtime. Within each benchmark, bold marks the lowest time and the largest speedup and time saving.

Benchmark	Method	Questions	Time (s)	Speedup	Time saved (s; %)
V^*Bench	Ours	191	**57.87**	–	–
	Pixel Reasoner	191	71.89	1.24×	14.02 (19.5%)
	DeepEyes-7B	191	568.45	9.82×	510.58 (89.8%)
HRBench (4K+8K)	Ours	1,600	**399.56**	–	–
	Pixel Reasoner	1,600	681.73	1.71×	282.17 (41.4%)
	DeepEyes-7B	1,600	1,828.19	4.58×	1,428.63 (78.1%)

Our method is 1.24× and 1.71× faster than Pixel Reasoner on V^*Bench and the combined HRBench variants, respectively; the corresponding speedups over DeepEyes-7B are 9.82× and 4.58×.

6 Conclusion

We studied the joint learning of mode capability and routing in visual agents. Our question-level analysis links GRPO advantages to one-step probability changes, showing how mode capability can improve while mode probability declines and how rewards can favor tools without an accuracy benefit. Together with training diagnostics, these findings motivate a two-stage framework that develops mode capabilities before using a frozen expert to guide routing through on-policy self-distillation. By comparing the expert's performance across modes for each question, the framework provides supervision along complete student-generated trajectories without additional routing rewards. Our study highlights the value of separating mode capability learning from routing when training visual agents to use tools according to their mode capabilities and the demands of individual questions.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/5be69a584901a26c521c2b51e40a4c20-Abstract-Conference.html.

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

Xingyu Fu, Minqian Liu, Zhengyuan Yang, John Richard Corring, Yijuan Lu, Jianwei Yang, Dan Roth, Dinei Florencio, and Cha Zhang. ReFocus: Visual editing as a chain of thought for structured image understanding. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 17783–17805. PMLR, 2025. URL https://proceedings.mlr.press/v267/fu25d.html.

Daya Guo, Dejian Yang, Haowei Zhang, et al. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645:633–638, 2025. doi: 10.1038/s41586-025-09422-z. URL https://www.nature.com/articles/s41586-025-09422-z.

Yushi Hu, Weijia Shi, Xingyu Fu, Dan Roth, Mari Ostendorf, Luke Zettlemoyer, Noah A. Smith, and Ranjay Krishna. Visual sketchpad: Sketching as a visual chain of thought for multimodal language models. In *Advances in Neural Information Processing Systems*, volume 37, pp. 139348–139379, 2024. doi: 10.52202/079017-4423. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/fb82011040977c7712409fbdb5456647-Abstract-Conference.html.

Dong-Hee Kim, Reuben Tan, and Donghyun Kim. Diversity over frequency: Rethinking tool use in visual chain-of-thought agents. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. URL https://icml.cc/virtual/2026/poster/61111.

Xin Lai, Junyi Li, Wei Li, Tao Liu, Tianjian Li, and Hengshuang Zhao. Mini-o3: Scaling up reasoning patterns and interaction turns for visual search. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/7c40c5050bd029a3ea7ff8b01412f735-Abstract-Conference.html.

Jewon Lee, Wooksu Shin, Seungmin Yang, Ki-Ung Song, DongUk Lim, Jaeyeon Kim, Tae-Ho Kim, and Bo-Kyeong Kim. ERGO: Efficient high-resolution visual understanding for vision-language models. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=ehKzPoOReW.

Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, Feng Li, Hao Zhang, Kaichen Zhang, Peiyuan Zhang, Yanwei Li, Ziwei Liu, and Chunyuan Li. LLaVA-OneVision: Easy visual task transfer. *Transactions on Machine Learning Research*, 2025. URL https://openreview.net/forum?id=zKv8qULV6n.

Yifan Li, Yifan Du, Kun Zhou, Jinpeng Wang, Xin Zhao, and Ji-Rong Wen. Evaluating object hallucination in large vision-language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 292–305. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.emnlp-main.20. URL https://aclanthology.org/2023.emnlp-main.20/.

Zichuan Lin, Yicheng Liu, Yang Yang, Lvfang Tao, and Deheng Ye. AdaptVision: Efficient vision-language models via adaptive visual acquisition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11923–11932, 2026. URL https://openaccess.thecvf.com/content/CVPR2026/html/Lin_AdaptVision_Efficient_Vision-Language_Models_via_Adaptive_Visual_Acquisition_CVPR_2026_paper.html.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://arxiv.org/abs/2304.08485.

Yuan Liu, Haodong Duan, Yuanhan Zhang, Bo Li, Songyang Zhang, Wangbo Zhao, Yike Yuan, Jiaqi Wang, Conghui He, Ziwei Liu, Kai Chen, and Dahua Lin. MMBench: Is your multi-modal model an all-around player? In *Computer Vision – ECCV 2024*, pp. 216–233. Springer, 2024. URL https://www.ecva.net/papers/eccv_2024/papers_ECCV/papers/00959.pdf.

Qinghe Ma, Zhen Zhao, Yiming Wu, Jian Zhang, Lei Bai, and Yinghuan Shi. Are tools always beneficial? learning to invoke tools adaptively for dual-mode multimodal LLM reasoning. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2605.19852.

OpenAI. GPT-4o system card. Technical report, 2024. URL https://openai.com/index/gpt-4o-system-card/.

OpenAI. Introducing GPT-4.1 in the API. Model release announcement, 2025. URL https://openai.com/index/gpt-4-1/.

Yi Ren and Danica J. Sutherland. Learning dynamics of LLM finetuning. In *International Conference on Learning Representations*, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/afe1aa79e5eea7955f553c61a307273e-Abstract-Conference.html.

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. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 19186–19195, 2026. URL https://openaccess.thecvf.com/content/CVPR2026/html/Song_CodeDance_A_Dynamic_Tool-integrated_MLLM_for_Executable_Visual_Reasoning_CVPR_2026_paper.html.

Alex Su, Haozhe Wang, Weiming Ren, Fangzhen Lin, and Wenhu Chen. Pixel Reasoner: Incentivizing pixel space reasoning via curiosity-driven reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/0c38f54740062529aa4117a04b583f3c-Abstract-Conference.html.

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. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 18058–18072, San Diego, California, United States, July 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.findings-acl.898. URL https://aclanthology.org/2026.findings-acl.898/.

Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xiong-Hui Chen, Jianxin Yang, Zhenru Zhang, Yuqiong Liu, An Yang, Andrew Zhao, Yang Yue, Shiji Song, Bowen Yu, Gao Huang, and Junyang Lin. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for LLM reasoning. In *Advances in Neural Information Processing Systems*, volume 38, 2025a. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/a797c2d2e0c1fdabf4d1ab8cd0b465c6-Abstract-Conference.html.

Wenbin Wang, Liang Ding, Minyan Zeng, Xiabin Zhou, Li Shen, Yong Luo, Wei Yu, and Dacheng Tao. Divide, conquer and combine: A training-free framework for high-resolution image perception in multimodal large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2025b. URL https://ojs.aaai.org/index.php/AAAI/article/view/32852.

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. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/7e6f445a74cdb71931aac64f1e3f49c9-Abstract-Conference.html.

Penghao Wu and Saining Xie. V*: Guided visual search as a core mechanism in multimodal LLMs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13084–13094, 2024. URL https://openaccess.thecvf.com/content/CVPR2024/html/Wu_V_Guided_Visual_Search_as_a_Core_Mechanism_in_Multimodal_CVPR_2024_paper.html.

Senqiao Yang, Junyi Li, Xin Lai, Jinming Wu, Wei Li, Zejun Ma, Bei Yu, Hengshuang Zhao, and Jiaya Jia. VisionThink: Smart and efficient vision language model via reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/88be023075a5a3ff3dc3b5d26623fa22-Abstract-Conference.html.

YiFan Zhang, Xingyu Lu, Shukang Yin, Chaoyou Fu, Wei Chen, Xiao Hu, Bin Wen, Kaiyu Jiang, Changyi Liu, Tianke Zhang, Haonan Fan, Kaibing Chen, Jiankang Chen, Haojie Ding, Kaiyu Tang, Zhang Zhang, Liang Wang, Fan Yang, Tingting Gao, and Guorui Zhou. Thyme: Think beyond images. In *International Conference on Learning Representations*, pp. 79536–79572, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/8080912b919ad0344c0d34e68983dfcb-Abstract-Conference.html.

Siyan Zhao, Zhihui Xie, Mengchen Liu, Jing Huang, Guan Pang, Feiyu Chen, and Aditya Grover. Self-Distilled Reasoner: On-policy self-distillation for large language models. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. URL https://icml.cc/virtual/2026/poster/64784.

Ziwei Zheng, Minghao Yang, Jack Hong, Chenxiao Zhao, Guohai Xu, Le Yang, Chao Shen, and Xing Yu. DeepEyes: Incentivizing "thinking with images" via reinforcement learning. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/cdb347d7516d52a9280ea9d6708911f6-Abstract-Conference.html.

Jinguo Zhu, Weiyun Wang, Zhe Chen, Zhaoyang Liu, Shenglong Ye, Lixin Gu, Yuchen Duan, Hao Tian, Weijie Su, Jie Shao, Zhangwei Gao, Erfei Cui, Yue Cao, Yangzhou Liu, Weiye Xu, Hao Li, Jiahao Wang, Han Lv, Dengnian Chen, Songze Li, Yinan He, Tan Jiang, Jiapeng Luo, Yi Wang, Conghui He, Botian Shi, Xingcheng Zhang, Wenqi Shao, Junjun He, Yingtong Xiong, Wenwen Qu, Peng Sun, Penglong Jiao, Lijun Wu, Kaipeng Zhang, Huipeng Deng, Jiaye Ge, Kai Chen, Limin Wang, Min Dou, Lewei Lu, Xizhou Zhu, Tong Lu, Dahua Lin, Yu Qiao, Jifeng Dai, and Wenhai Wang. InternVL3: Exploring advanced training and test-time recipes for open-source multimodal models. *arXiv preprint arXiv:2504.10479*, 2025. URL https://arxiv.org/abs/2504.10479.

A LIMITATIONS

Sampling uncertainty. Our framework estimates the frozen expert's accuracy in each mode from a finite number of sampled trajectories. These estimates are subject to sampling error, particularly when the two modes have similar accuracies on a question. Consequently, the selected mode may not maximize the expert's true accuracy, introducing noise into the distillation supervision.

Scope of analysis. Our analysis focuses on the interaction between the direct and tool modes in a setting with a single visual tool type. Although this setting permits repeated tool calls, it does not capture the richer decisions involved in selecting and composing multiple tools. Whether the observed dynamics and the proposed training framework extend to these more complex settings remains to be investigated.

B AI USE DISCLOSURE

Generative AI tools were used to assist with software implementation, code completion, polish writing and debugging. All AI-generated or AI-modified code used in this work was manually reviewed and tested by the authors for correctness before being incorporated into the experimental pipeline. The authors take full responsibility for the implementation, experimental results, and final content of this work.

C SUPPLEMENTARY BENCHMARK RESULTS

C.1 REFERRING-EXPRESSION GROUNDING

Table 4 reports supplementary referring-expression grounding results on RefCOCO, RefCOCO+, and RefCOCOg. All models are evaluated with greedy decoding. We report Precision@1 (%), counting a prediction as correct when the intersection-over-union (IoU) between the predicted and ground-truth boxes is at least 0.5. The average is an equally weighted macro average over the eight evaluation splits. Our model achieves the highest reported macro average (85.65%) and average IoU (0.772) among the models compared here.

Table 4: Referring-expression grounding results. Split-level scores and the macro average are Precision@1 (%); Avg. IoU is reported separately. All models use greedy decoding, and the macro average weights the eight splits equally. Bold indicates the best result in each column.

Model	RefCOCO			RefCOCO+			RefCOCOg		Macro avg.	Avg. IoU
	val	A	B	val	A	B	val	test		
Qwen2.5-VL-7B (Bai et al., 2025)	**89.45**	92.13	85.06	82.03	87.36	74.15	85.48	85.75	85.18	0.766
AdaptVision-7B (Lin et al., 2026)	82.48	85.65	75.43	75.05	80.35	66.66	75.43	76.10	77.14	0.638
ERGO-7B (Lee et al., 2026)	88.72	91.67	84.87	81.33	86.33	73.51	83.99	85.15	84.45	0.760
Ours	89.00	**92.54**	**85.20**	**83.23**	**88.60**	**75.05**	**85.60**	**86.02**	**85.65**	**0.772**

C.2 MMBENCH AND POPE

Table 5 reports results on MMBench (English dev) and POPE. We report accuracy on MMBench and F1 on the three POPE splits and overall.

D ROUTING BASELINE DETAILS

Section 5.3 compares autonomous routing with alternatives that select between the Stage-I checkpoint's direct and tool modes on HRBench-4K and HRBench-8K. To estimate the performance attainable by a question-wise oracle for this fixed checkpoint, we sample eight rollouts in each mode for every question x. Let $\hat{p}_m(x)$ be the fraction of correct rollouts under mode $m \in \{\mathrm{D}, \mathrm{T}\}$. The

Table 5: Supplementary results on MMBench (English dev) and POPE (%). POPE Overall is the combined F1 across its three splits. All Qwen2.5-VL baselines use Instruct checkpoints. Bold indicates the best result in each column; gray rows denote our final model.

(a) MMBench: accuracy

Model	Perception			Reasoning			Overall
	Coarse	Cross-instance	Instance-level	Attribute	Logic	Relation	
Qwen2.5-VL-7B	**88.51**	79.02	87.03	86.43	74.58	78.26	84.19
Qwen2.5-VL-32B	86.15	**83.22**	89.42	87.44	77.12	**86.96**	86.00
DeepEyes	81.76	65.73	80.20	75.88	48.31	64.35	73.28
ERGO-7B	85.81	72.73	83.96	78.89	70.34	78.26	80.24
Ours	86.89	82.91	**94.15**	**89.88**	**78.26**	84.46	**87.26**

(b) POPE: F1

Model	Adversarial	Popular	Random	Overall
Qwen2.5-VL-7B	85.10	85.95	87.16	86.06
Qwen2.5-VL-32B	83.82	84.71	85.53	84.68
DeepEyes	83.87	84.46	88.16	85.47
ERGO-7B	83.51	85.99	**90.94**	86.70
Ours	**86.44**	**87.34**	88.32	**87.36**

estimated oracle accuracy is

$$\hat{A}_{\text{oracle}} = \frac{1}{|\mathcal{D}|} \sum_{x \in \mathcal{D}} \max\{\hat{p}_{\text{D}}(x), \hat{p}_{\text{T}}(x)\}, \qquad \hat{p}_m(x) = \frac{1}{8} \sum_{k=1}^{8} C(x, \tau_k^m). \tag{11}$$

This quantity assumes that every question is routed to the mode with the higher estimated success rate. It is an empirical ceiling for routing the Stage-I checkpoint between these two modes under this sampling protocol, not an upper bound on what a separately trained model could achieve.

For comparison, we train a lightweight binary router on the Stage-II training set: a frozen Qwen3-VL-Embedding-8B model encodes each image–question pair into a 4,096-dimensional representation. A two-layer MLP maps these features through a 256-dimensional hidden layer to a single output, followed by a sigmoid. We route to the tool mode when the resulting probability is at least 0.5 and to the direct mode otherwise. The other routing rules rank questions by estimated difficulty using image information entropy, image resolution, Laplacian variance, or Canny edge density. We also obtain a tool-necessity score from Qwen3.5-27B by presenting it with the image and question and asking for a 0–100 confidence score indicating whether a tool is needed to solve the question. For each of these five score-based rules, we route the highest-ranked fraction of questions to the tool mode and the rest to the direct mode; that fraction matches the Stage-II model's tool-call rate: 15.12% on HRBench-4K and 14.12% on HRBench-8K.

E Prompt Templates for the Three Modes

All three modes share a system message specifying the `crop_image` tool and its invocation syntax. Mode-specific instructions are appended to the user message following each image–question input. The direct and tool instructions condition the Stage-I expert, whereas the auto instruction is used by the Stage-II student. We provide the exact prompt text below for reproducibility.

Shared system message (auto, direct, and tool).

```
You are a helpful assistant.

# Tools
```

```
You may call one or more functions to assist with the user query.

You are provided with function signatures within <tools></tools> XML
tags:
<tools>
{"type": "function", "function": {"name": "crop_image",
"description": "Zoom in on the image based on the bounding box
coordinates.", "parameters": {"type": "object", "properties":
{"bbox_2d": {"type": "array", "description": "coordinates [x1, y1,
x2, y2] for the region to zoom in. minimum 0, maximum is the
width/height of the image.", "items": {"type": "number"}},
"target_image": {"type": "number", "description": "The index of the
image to crop, from 1 to the number of images. Choose 1 for the
original image."}}, "required": ["bbox_2d", "target_image"]}}}
</tools>

For each function call, return a json object with function name and
arguments within <tool_call></tool_call> XML tags:
<tool_call>
{"name": <function-name>, "arguments": <args-json-object>}
</tool_call>
```

Auto mode (autonomous student); user-message instruction.

```
You have access to the crop_image tool, but whether to use it is YOUR
choice. First decide: can you answer correctly from the image as you
currently see it? If the relevant region is too small or unclear,
call the crop_image tool to zoom into it (you may call it more than
once); otherwise answer directly without any tool call.
Respond in exactly one of these two formats.
If you answer directly:
<think> your reasoning </think>
<answer> your final answer </answer>
If you use the tool, at each turn:
<think> your reasoning </think>
<tool_call> {"name": "crop_image", "arguments": {...}} </tool_call>
[after the tool observation, repeat <think> + <tool_call> as needed;
when ready to answer:]
<think> your reasoning </think>
<answer> your final answer </answer>
```

Direct mode (instructed expert); user-message instruction.

```
Answer the question using ONLY the image shown. Do NOT use any tool.
Put your step-by-step reasoning inside <think>...</think>, then give
your final answer inside <answer>...</answer>. Respond in exactly
this format:
<think> your reasoning </think>
<answer> your final answer </answer>
```

Tool mode (instructed expert); user-message instruction.

```
You MUST use the crop_image tool to help you answer the question:
zoom into the relevant region to read the visual details. At each
turn, first inspect the currently available visual information and
put your reasoning inside <think>...</think>. Then either call the
crop_image tool if more visual detail is needed, or give the final
answer if the information is sufficient.
```

```
Respond in exactly this format:
<think> your reasoning </think>
<tool_call> {"name": "crop_image", "arguments": {...}} </tool_call>
[after the tool observation, repeat the following as needed:]
<think> your reasoning </think>
<tool_call> {"name": "crop_image", "arguments": {...}} </tool_call>
[when ready to answer:]
<think> your reasoning </think>
<answer> your final answer </answer>
```

F Derivation of the One-Step Dynamics

Fix a question x and a sampled group of G trajectories. Holding the group rewards, advantages, and trajectory lengths constant, consider

$$\theta^+ = \theta + \eta g, \qquad g = \frac{1}{G}\sum_{i=1}^{G} \frac{A_i}{T_i} \nabla_\theta \log \pi_\theta(\tau_i \mid x), \tag{12}$$

where η is the step size and T_i counts agent-generated tokens, matching the length normalization in our GRPO loss. We assume a twice continuously differentiable policy with locally fixed support, a bounded interaction horizon, and tool observations whose distribution has no direct dependence on θ. The trajectory score therefore sums the scores of agent-generated tokens, including termination; tool observations receive no prediction gradient. Equation 12 is the local gradient-ascent direction of the length-normalized GRPO objective at the rollout parameters when clipping is inactive and regularization is omitted. It isolates this direction rather than modeling the full AdamW update or subsequent optimization epochs.

Write $\pi(\tau) = \pi_\theta(\tau \mid x)$ and $\pi^+(\tau) = \pi_{\theta+\eta g}(\tau \mid x)$. With $u(\tau) = \langle \nabla_\theta \log \pi(\tau), g \rangle$, a first-order expansion gives

$$\pi^+(\tau) = \pi(\tau)\big[1 + \eta u(\tau)\big] + O(\eta^2), \qquad \mathbb{E}[u] = 0. \tag{13}$$

The zero-mean identity follows from normalization: $\mathbb{E}[\nabla_\theta \log \pi(\tau)] = 0$. All expectations below use the pre-update policy; the finite group determines g but does not replace these population expectations. The factor $1/T_i$ belongs to g, not to the score in u, because u describes the change in the probability of a complete trajectory. Define the policy-weighted mean growth rates

$$\bar{u}_m = \mathbb{E}[u(\tau) \mid M = m], \qquad \bar{u}_{m,c} = \mathbb{E}[u(\tau) \mid M = m, C = c].$$

For routing, summing Equation 13 over the tool mode gives $\Delta q = \eta q \bar{u}_{\mathrm{T}} + O(\eta^2)$. Using $q\bar{u}_{\mathrm{T}} + (1-q)\bar{u}_{\mathrm{D}} = 0$ yields

$$\Delta q = \eta q(1-q)\big(\bar{u}_{\mathrm{T}} - \bar{u}_{\mathrm{D}}\big) + O(\eta^2). \tag{14}$$

For mode capability, differentiate the ratio $p_m = \Pr(C = 1, M = m)/\Pr(M = m)$. The changes in its numerator and denominator give

$$\begin{aligned} \Delta p_m &= \eta p_m\big(\bar{u}_{m,1} - \bar{u}_m\big) + O(\eta^2) \\ &= \eta p_m(1-p_m)\big(\bar{u}_{m,1} - \bar{u}_{m,0}\big) + O(\eta^2). \end{aligned} \tag{15}$$

The difference forms assume that the conditioned events have positive probability. Equivalently, the changes can be written as $\Delta q = \eta \operatorname{Cov}(\mathbb{I}\{M = \mathrm{T}\}, u) + O(\eta^2)$ and $\Delta p_m = \eta \operatorname{Cov}(C, u \mid M = m) + O(\eta^2)$, with the second identity requiring only $\Pr(M = m) > 0$. These relations describe local changes, not a guarantee of eventual collapse or convergence: subsequent groups, rewards, and gradients evolve with the policy.

G DeepEyes and AutoTool Rewards

We use binary correctness C, format compliance F, and an indicator V for at least one valid tool execution. The diagnostic tool/direct categories correspond to $V = 1/0$.

DeepEyes. The original reward (Zheng et al., 2026) is

$$R_{\text{bonus}} = 0.8C - 0.2(1 - F) + 1.2CV. \tag{16}$$

The tool bonus requires both a correct answer and a valid tool execution.

AutoTool. The original mode-specific reward (Ma et al., 2026) is

$$R_{\text{bal}} = 0.8C - 0.2(1 - F) + \lambda_m(f)B, \tag{17}$$

where $B = 1$ for a correct answer, $B = -0.5$ for an incorrect tool-mode answer, and $B = 0$ otherwise. Here f is the observed tool-mode fraction in the current rollout batch, and

$$\lambda_{\text{T}}(f) = 1.2 + 0.5 - f, \qquad \lambda_{\text{D}}(f) = 1.2 - 0.5 + f. \tag{18}$$

We retain these adaptive coefficients throughout training, omitting AutoTool's final switch to fixed coefficients.

H Training and Implementation Details

Training data. We train on the `v012` subset of DeepEyes' released 47K-example dataset (Zheng et al., 2026), which combines fine-grained visual search questions from V^*, chart questions from ArxivQA, and multimodal reasoning questions from ThinkLite-VL. We reserve 6,000 questions for Stage-II expert assignment and self-distillation, using the remainder for Stage I. Auxiliary mathematical reasoning examples use the same correctness-and-format reward.

Optimization and rollout settings. We initialize our model from Qwen2.5-VL-7B-Instruct and train both mode capabilities with GRPO for 50 optimization steps on eight NVIDIA H200 GPUs. We use Qwen3.5-27B as our judge model. Table 6 summarizes the main hyperparameters for both stages. We train all parameters of the policy being optimized, including the vision encoder, using FSDP2 across eight GPUs; the Stage-II expert remains frozen. Stage I uses one PPO optimization epoch per rollout batch, with a clipping threshold of 0.2 and zero reference-KL and entropy-regularization coefficients. Its format-violation penalty is 0.2, equivalent to $\lambda_{\text{F}} = 0.2$ in Equation 6 up to an additive constant that cancels in group-relative advantages. The maximum Stage-I response length is 20,480 tokens. Stage II uses AdamW with $(\beta_1, \beta_2) = (0.9, 0.999)$ and 10 warmup updates followed by a constant learning rate.

Table 6: Training hyperparameters. Rollouts per prompt refer to training samples; the separate expert-assignment budget is $K = 4$ per mode.

Setting	Stage I	Stage II
Training objective	GRPO	On-policy self-distillation
Optimizer	AdamW	AdamW
Learning rate	1×10^{-6}	1×10^{-6}
Weight decay	0.01	0.01
Gradient clipping (maximum norm)	1.0	1.0
Training updates	50	50
Prompts per training batch	64	64
Rollouts per prompt	16	8
Actor/PPO mini-batch size	32	32
Micro-batch size per GPU	4	4
Rollout temperature	1.0	1.0
Rollout top-p	1.0	1.0
Maximum tool-interaction rounds	4	4
Maximum tokens per action	2,048	2,048
Data-loader seed	1	1

Constructing expert assignments. For each question in the disjoint Stage-II training set, we sample $K = 4$ rollouts under each of the frozen expert's direct and tool instructions, giving eight expert rollouts in total. We compare their empirical accuracies, assign the more accurate mode, and break ties in favor of the direct mode, following Equation 9. Assignments are computed once and remain fixed during distillation. This evaluation budget is separate from the eight fresh student rollouts per prompt used in each Stage-II training batch.

Distillation implementation. We set PPO advantages to zero and use the frozen teacher's log-probabilities on student trajectories as the actor's reference log-probabilities. The distillation loss is Verl's `low_var_kl` (k_3) with coefficient 1.0. Writing $\Delta_t = \log \pi_{\mathrm{E}}(a_t \mid h_t, s^*(x)) - \log \pi_\theta(a_t \mid h_t, s_{\mathrm{A}})$, its unclipped token penalty is $\ell_t = \exp(\Delta_t) - \Delta_t - 1$. We differentiate k_3 directly, without the k_3^+ straight-through modification; the implementation applies numerical clipping to both Δ_t and the resulting penalty, then averages over valid response tokens. Tool observations provide context but are excluded from this loss. For the unclipped estimator, direct differentiation with fixed token histories and samples recovers the forward-KL token gradient in expectation under exact on-policy sampling. The correspondence to Equation 10 concerns this expected update; numerical clipping makes it approximate.

I Training Accuracy of the Two Modes

Figure 5 compares the training accuracy of the tool and direct modes over a full 150-step run. The two curves remain close overall, with the tool mode retaining a small advantage toward the end: over steps 101–150, its mean per-step accuracy is 89.52%, compared with 89.07% for the direct mode.



Figure 5: Training accuracy of the direct (blue) and tool (orange) modes over all 50 steps. Faint lines show raw per-step accuracies; bold lines show centered five-step moving averages, using the available steps at the endpoints.

Neither mode exhibits an overwhelming accuracy advantage that would justify exclusively using it across questions. Collapsing to always-tool or always-direct behavior would discard an alternative with comparable observed accuracy and prevent question-dependent mode selection. Always using tools overlooks the strong direct-mode baseline, while always answering directly forgoes the potential benefit of tool-assisted reasoning indicated by its modest late-stage advantage. These trends support interpreting mode collapse as an undesirable loss of routing flexibility, rather than evidence that the remaining mode is universally preferable. The curves summarize aggregate training accuracy; they do not by themselves establish which mode is better for individual questions.