EASY TO CALL, HARD TO USE EFFECTIVELY: LEARNING HOW AND WHEN TO USE VISUAL TOOLS IN SMALL MLLM AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Multimodal large language models (MLLMs) trained with reinforcement learning with verifiable rewards on tool-using trajectories (Tool-RLVR) have become agents that call visual tools such as zooming and Python code. Small MLLM agents are in growing demand for on-device applications, but turning small MLLMs into such agents remains difficult, as they struggle to use tools effectively. They fail to turn tool results into correct answers and keep calling tools even when the question does not need them, revealing limitations in *how* and *when* to use tools. We show that these two limitations hinder Tool-RLVR from training small MLLMs into effective agents, for two reasons. First, Tool-RLVR penalizes a proper tool-call decision whenever a later stage of the trajectory fails, so small MLLM agents decide not to use tools. Second, it ignores whether the tool was necessary for the question, so the call rate moves as a whole rather than following necessity. To address them, we propose **A**gentic **N**ecessity-Aware **P**olicy **O**ptimization (ANPO), which rewards the decision, execution, and answer stages separately and judges the tool-call decision by the necessity of the tool for each question rather than by the outcome of each trajectory. Extensive experiments on fifteen benchmarks show that ANPO achieves the highest accuracy among Tool-RLVR methods while calling tools mainly where they are necessary.

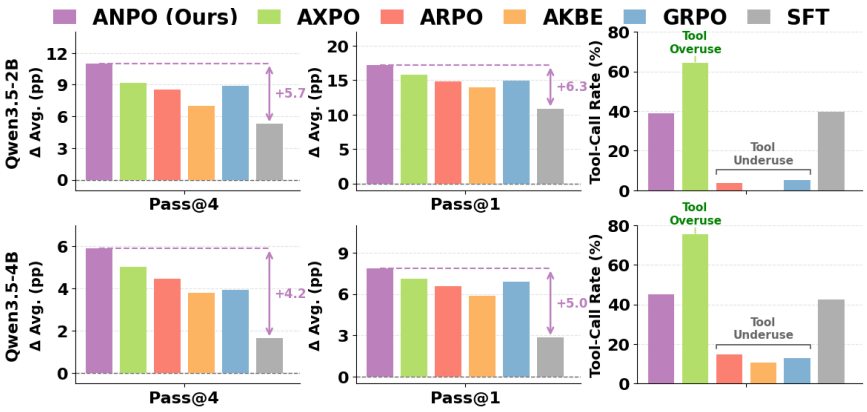


Figure 1: (Left & Middle) Average Pass@4 and Pass@1 gains over the Base model. (Right) Average tool-call rate for Qwen3.5-2B (top) and Qwen3.5-4B (bottom) under each Tool-RLVR method. ANPO (Ours) achieves the best performance while avoiding both tool underuse and tool overuse.

1 INTRODUCTION

Multimodal large language models (MLLMs) extend large language models (LLMs) to visual inputs, enabling visual recognition such as identifying objects (Liu et al., 2023; Bai et al., 2023; Wang et al., 2025e; Bai et al., 2025). Reinforcement learning with verifiable rewards (RLVR) has further extended them to complex visual reasoning and fine-grained perception, such as locating small details in an image (Liu et al., 2025c; Zhang et al., 2025a). Tool-RLVR turns MLLMs into agents that can call tools, such as zooming into an image region or executing Python code (Zheng et al., 2026; Wang et al., 2025c; Lai et al., 2025).

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.

At the same time, the demand for small MLLM agents has been steadily growing, as on-device applications such as robotics (Wen et al., 2025) and personal assistants (Yang et al., 2026a) need agents that can run under limited compute and memory. However, turning small MLLMs into such agents remains difficult, as they inherently struggle to use tools effectively (Wang et al., 2025a; 2026b; Kang et al., 2026). Effective tool use entails judging when a tool is needed beyond the model's own visual perception, and knowing how to execute it appropriately to reach the correct answer.

In Figure 2(top), we examine *how* effectively small MLLMs utilize visual tools. We compare the accuracy of MLLMs with and without tool access on fine-grained perception questions, where visual tools help find the answer.[1] Whereas larger MLLMs gain from tool access, small MLLMs perform worse with tools than without them. Small MLLMs thus fail to turn the tool result into a correct answer, even where the tool is expected to help.

We next examine whether small MLLMs can tell *when* a visual tool is needed. On HR-Bench (Wang et al., 2024b), we create three levels of visual difficulty by resizing the target region while keeping the question and answer fixed. We ensure that even small MLLMs answer most questions correctly at the easy level without a tool. As shown in Figure 2(bottom), larger MLLMs call the tool less as the questions become easier, whereas small MLLMs keep a high tool-call rate at all three levels. Together, these observations reveal two limitations of small MLLMs: ***how* to utilize visual tools effectively and *when* to use them.**[1]

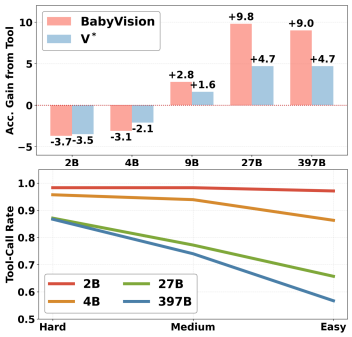


Figure 2: (Top) Accuracy gain from tool access and (Bottom) tool-call rate under controlled visual difficulty on HR-Bench (Wang et al., 2024b), across Qwen3.5 (Qwen Team, 2026) model sizes.

We hypothesize that these two limitations also hinder Tool-RLVR from teaching small MLLMs to use tools effectively. Small MLLMs produce (1) many tool-using trajectories that end in a wrong answer, which we call *failed* trajectories, and (2) many tool calls on questions where the tool is not necessary. In Section 3, we show that existing Tool-RLVR trained on such trajectories does not remedy these limitations and identify two root causes behind it. First, Tool-RLVR does not distinguish which step is responsible for a failed trajectory: deciding whether to call the tool, executing the call, or answering from the returned result. Thus, a proper tool-call decision can be penalized for a failure that occurs later, leading small MLLM agents to decide not to use tools, known as *tool collapse* (Kang et al., 2026; Kim et al., 2026a). We refer to this mismatch as ***misaligned reward assignment***. Second, Tool-RLVR teaches the model to call tools without considering whether the tool was necessary for the question, which we term ***necessity-agnostic tool learning***. As a result, the call rate moves up or down as a whole rather than following necessity.

Based on this analysis, we propose **A**gentic **N**ecessity-Aware **P**olicy **O**ptimization (ANPO), which enables small MLLM agents to learn *how* and *when* to use visual tools through two components, as illustrated in Figure 3. **Stage-wise Reward Assignment (SRA)** separates a tool-using trajectory into three *stages*—decision, execution, and answer—and assigns each stage a separate learning signal, so that the decision is not directly penalized for a later failure. **Necessity-Aware Decision Reward (NDR)** strengthens the decision to call on questions where the tool is necessary. Since necessity depends on the model's current ability, which changes during training, NDR estimates it on the fly by comparing how well the model solves the same question with and without the tool.

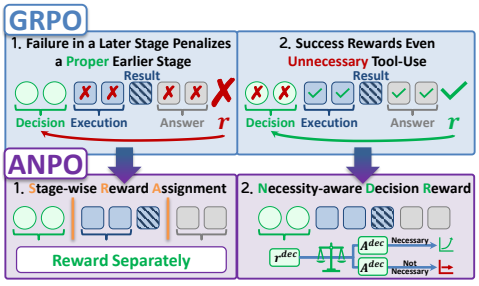


Figure 3: Comparison of GRPO and ANPO, which consists of two components.

Through extensive experiments, we show that ANPO achieves the highest accuracy across various tasks while calling tools selectively, as shown in Figure 1. Further analyses show that ANPO learns both *when* and *how* to use tools: it adjusts its tool calls to the difficulty and necessity of each question, relies on the tool where direct reasoning falls short, and makes fewer redundant tool calls.

Our contributions are as follows. **1)** We show that small MLLMs have two limitations, in *how* and *when* to use visual tools, and identify two root causes that keep existing Tool-RLVR from remedying

[1] In both experiments, MLLMs can call Zoom-In and Python. The construction of the difficulty levels is detailed in Appendix A.1.

them. **2)** We propose ANPO, which rewards each stage of tool use separately and strengthens the decision to call where the tool is necessary. **3)** ANPO achieves the highest accuracy across various tasks while calling tools mainly where they are necessary.

2 PRELIMINARY

RLVR with GRPO. RLVR trains a model on questions whose final answer can be verified and uses its correctness as the reward. For each question q, the model produces G trajectories $\{o_1, \ldots, o_G\}$, each containing the reasoning and the final answer. Each trajectory receives a reward $r_i \in \{0, 1\}$, and GRPO (Shao et al., 2024) normalizes the rewards within the group to obtain the advantage $A_i = (r_i - \bar{r})/\operatorname{std}(\{r_j\}_{j=1}^{G})$ with the group mean baseline $\bar{r}$, which serves as the learning signal. This advantage A_i is assigned to every token that the model generated in o_i.

Tool-RLVR for MLLM agents. Tool-RLVR trains an MLLM agent that can call tools such as zooming or Python code. Since the agent decides whether to call a tool, a group mixes trajectories that call a tool and trajectories that answer without a tool (Kang et al., 2026; Chen et al., 2026a). Every trajectory receives the same advantage as in GRPO (Feng et al., 2025a). Each trajectory that calls a tool has three stages: the *decision* to call a tool, the *execution* that writes the call, and the *answer* that reaches the final answer from the tool result. ARPO (Dong et al., 2025b) branches additional trajectories after a tool call when the token entropy rises, and AXPO (Kang et al., 2026) resamples continuations from the tool call in groups where every tool-using trajectory fails, and rewards the tool call by whether a continuation succeeds.

3 WHY IS TOOL USE DIFFICULT TO LEARN FOR SMALL MLLM AGENTS?

In this section, we show that existing Tool-RLVR does not remedy the limitations in *how* and *when* to use visual tools identified in Section 1, and identify two root causes in the learning signal of the tool-call decision. Section 3.1 shows that ***misaligned reward assignment*** penalizes a proper tool-call decision for a failure in a later stage. In Section 3.2, we find that under ***necessity-agnostic tool learning***, the call rate moves up or down as a whole rather than following necessity. All experiments in this section use Qwen3.5-2B. Experiment details are given in Appendix A.2.

3.1 WHY DO SMALL MLLM AGENTS STOP USING TOOLS DURING TOOL-RLVR?

We first show that small MLLM agents stop using tools during Tool-RLVR training, and then analyze which stage of the tool-using trajectories causes this. We find that most tool-call decisions in failed trajectories are proper, yet they are penalized for failures in later stages, and this ***misaligned reward assignment*** drives *tool collapse*.

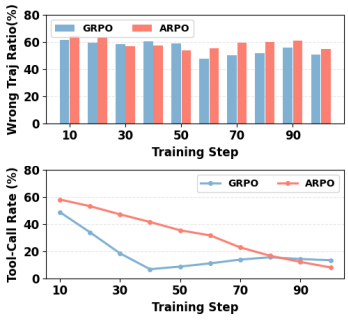


Figure 4: (Top) Ratio of trajectories receiving negative learning signals. (Bottom) Tool-call rate during Tool-RLVR training.

Tool Collapse under Tool-RLVR. Figure 4 tracks the fraction of tool-using trajectories that receive a negative learning signal and the tool-call rate during Tool-RLVR training on questions where visual tools are expected to help. Even on these questions, most tool-using trajectories receive a negative learning signal, and the tool-call rate decreases as training proceeds. This suggests that training suppresses tool calls rather than improving how small MLLM agents use tools, which is the *tool collapse* reported in prior work (Kang et al., 2026; Kim et al., 2026a).

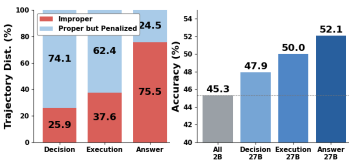


Figure 5: (Left) Ratio of failed trajectories judged proper by an LLM. (Right) Accuracy after replacing each stage with a larger model.

A Failed Answer Does Not Imply a Failed Decision. In Figure 5(left), an LLM judge assesses whether each stage of the failed tool-using trajectories is proper on its own. A decision is proper if the tool call brings in answer-relevant evidence that is hard to see in the image. An execution is proper if the call is well-formed and returns a result, and an answer stage is proper if its conclusion follows from the tool results. Most decisions and executions in failed trajectories are actually proper, and improper stages concentrate in the answer stage, yet the reward of the whole trajectory penalizes the decision and execution as well. In Figure 5(right), we rewrite one stage of these trajectories with Qwen3.5-27B, while the 2B model

writes the rest with the same prompt and tools. The accuracy gain from each stage follows how often the judge finds it improper, which also confirms this.

Penalizing Proper Decisions Suppresses Tool Use. To examine which stage's penalty drives *tool collapse*, we fix the failed trajectories of GRPO and train the model while penalizing only one stage at a time in Figure 6. Penalizing the decision alone reduces tool use even more than penalizing the whole trajectory, whereas penalizing the execution or the answer stage leaves it unchanged. We then penalize only the decisions that the judge of Figure 5(left) finds proper, or only those it finds improper. Penalizing only the proper decisions collapses tool use, while penalizing only the improper ones barely changes it. *Tool collapse* therefore comes not from penalizing the decision itself but from ***misaligned reward assignment***, which penalizes a proper decision for a later failure.

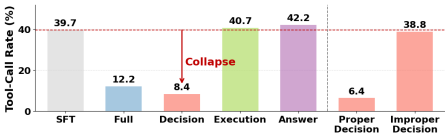


Figure 6: Tool-call rate after controlled training that penalizes the whole trajectory, one stage, or only proper or improper decisions.

Takeaway. *Tool collapse* comes from ***misaligned reward assignment***, which penalizes proper tool-call decisions for later failures, so the learning signal of the decision has to be separated from failures in later stages.

3.2 Does Tool-RLVR Teach When to Call Tools?

We next show that small MLLM agents do not learn *when* to call tools during Tool-RLVR training, and then investigate the learning signal that the tool-call decision receives. We find that calls on questions where the tool is not necessary dominate this signal, so the decision follows the outcome of those calls regardless of whether the tool was necessary. Under this ***necessity-agnostic tool learning***, the tool-call rate moves up or down as a whole rather than following necessity.

Tool-Call Rate Moves Together. We divide the questions into Tool-Necessary (TN) questions from datasets built to train visual tool use and Tool-Unnecessary (TU) questions from datasets built to train reasoning such as logic or math, and track the tool-call rate of each group in Figure 7. GRPO and ARPO reduce tool use on both groups, whereas AXPO increases it on both, so none of the methods teaches when to call tools.

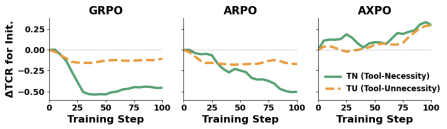


Figure 7: Change in tool-call rate on TN and TU questions during training.

Discussion. Since necessity also depends on the ability of the model, this split by dataset source gives only a coarse proxy for necessity, which we revisit in Section 4.2.

TU Questions Dominate the Decision Signal. Figure 8 compares how much the tool-using trajectories of each group contribute to the learning signal of the tool-call decision. The TU signal is consistently larger than the TN signal. Since the update of the decision grows with the size of its learning signal, the tool-call decision of small MLLM agents is shaped mainly by calls on TU questions, where the tool is not necessary.

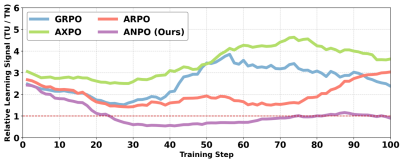


Figure 8: Ratio of the decision learning signal from TU questions to that from TN questions.

The Dominant Signal Pushes Tool Use in One Direction. We next examine the sign of this dominant signal in Figure 9. Under GRPO and ARPO, the TU signal is negative and larger than the TN signal, which pushes the model away from tools. Under AXPO, both signals are positive, since it rewards the tool call whenever a resampled continuation succeeds. These signs match the call rates in Figure 7, where GRPO and ARPO fall into *tool collapse* and AXPO overuses tools. Although the methods differ in direction, all of them train the tool-call decision regardless of whether the tool was necessary, so the dominant TU signal moves tool use in one direction.



Figure 9: Learning signal for the tool decision from TN (green) and TU (orange) questions during training.

Takeaway. The one-directional call rate comes from ***necessity-agnostic tool learning***, which trains the decision by the outcome alone, so the decision signal has to reflect whether the tool is necessary for each question.

4 ANPO: AGENTIC NECESSITY-AWARE POLICY OPTIMIZATION

Section 3 identifies two requirements for the learning signal of the tool decision. It should be separated from failures after the call (Section 3.1), and it should reflect whether the tool helps on the current question rather than merely whether a tool-using trajectory succeeds (Section 3.2). We propose ANPO, which directly satisfies these requirements through two components. **Stage-wise Reward Assignment (SRA)** addresses ***misaligned reward assignment*** by assigning a reward to each stage, so that a later failure in a trajectory does not directly penalize its own decision. **Necessity-Aware Decision Reward (NDR)** addresses ***necessity-agnostic tool learning*** by strengthening the decision advantage on questions where the tool is necessary. The overview of ANPO is shown in Figure 10.

Dual-Arm Rollouts. For each question, we generate two arms of trajectories: N_T *tool-on trajectories* with the tools available and N_R *tool-off trajectories* with the tools removed. We use the two arms to estimate how necessary the tool is for each question. To this end, we compute $\bar{r}_T$, the average correctness, only over the tool-on trajectories that actually call a tool, not over the whole tool-on arm, and $\bar{r}_R$ over all tool-off trajectories. Tool-on trajectories without a tool call are thus used in neither $\bar{r}_T$ nor $\bar{r}_R$.

4.1 STAGE-WISE REWARD ASSIGNMENT (SRA)

Stages of a Trajectory. To separate the tool-call decision from later failures, we divide each tool-using trajectory into the three stages defined in Section 2: decision, execution, and answer. For each tool call, the tokens from the start of the trajectory or the previous tool result up to <*tool_call*> form the decision stage, and the call arguments form the execution stage. The tokens after the final tool result, which produce the final answer, form the answer stage. Thus, a trajectory has as many decision and execution stages as tool calls, followed by one answer stage. Tool-on trajectories without a tool call and tool-off trajectories consist of a single answer stage and are updated only by their answer advantage.

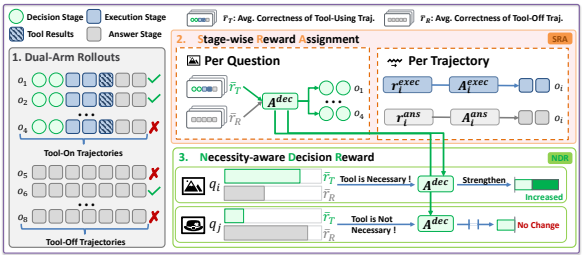


Figure 10: Overview of ANPO. From dual-arm rollouts, SRA assigns each stage its own advantage, and NDR strengthens the decision where the tool is necessary.

Reward and Advantage of Each Stage. SRA gives each stage its own learning signal instead of assigning the trajectory reward to every token. *Decision.* Motivated by Section 3.1, we reward the decision of each question as a whole, not each trajectory separately, giving every tool-using trajectory of the question the same decision reward $r_i^{\text{dec}} = \bar{r}_T$, which we call *broadcast*. As this reward is identical within a question, the average reward equals each individual reward, so we use $\bar{r}_R$ as the baseline instead,

$$A_i^{\text{dec}} = \frac{r_i^{\text{dec}} - \bar{r}_R}{\text{std}(\{r_j^{\text{dec}} - \bar{r}_R\}_{j \in B})}, \tag{1}$$

where B is the set of tool-using trajectories in the batch. *Answer.* The answer stage receives the GRPO advantage A_i^{ans} of r_i within each arm, so the two arms are compared through the decision. *Execution.* The execution stage receives A_i^{exec}, the average of A_i^{ans} and the advantage of an execution reward r_i^{exec}, which is 1 when the call runs without error. Averaging with A_i^{ans} keeps a call that runs but does not help the answer, such as Python code with only comments, from being rewarded.

Why Broadcast the Decision Reward. Section 3.1 shows that penalizing proper decisions contributes to *tool collapse*. One simple alternative is to keep the per-trajectory reward r_i and only change its baseline. However, r_i still reflects failures in later stages, so a proper decision in a failed trajectory can remain penalized. Table 1 reports, among proper decisions in failed trajectories, the fraction that receives a non-negative decision advantage under different reward and baseline choices. Among the tested choices, broadcasting $\bar{r}_T$ leaves the most proper decisions unpenalized. The average accuracy $\bar{r}$ of both arms could also serve as its baseline and gives the same sign, but we use $\bar{r}_R$ because $\bar{r}_T - \bar{r}_R$ directly compares the tool-on arm with the tool-off arm.

Table 1: Proper decisions left unpenalized in failed trajectories.

Reward − baseline	Not penalized (%) ↑
$r_i - \bar{r}$ (GRPO)	0.0
$r_i - \bar{r}_T$	22.4
$r_i - \bar{r}_R$	27.9
$\bar{r}_T - \bar{r}_R$	42.2

4.2 NECESSITY-AWARE DECISION REWARD (NDR)

Section 3.2 shows that, in existing Tool-RLVR, the decision signal is dominated by one sign and moves tool use in one direction rather than teaching when to call. SRA makes this signal question-specific: $\bar{r}_T - \bar{r}_R$ is positive when tool-using trajectories are more accurate than tool-off trajectories and negative otherwise. Following prior work (Yan et al., 2026; Wang et al., 2026a), we call a question *necessary* when $\bar{r}_T > \bar{r}_R$. However, Appendix A.3 shows that the negative signal still outweighs the positive one under SRA, so NDR strengthens the positive decision signal, which requires deciding where and how much to strengthen it.

Where to Strengthen. A naive choice is to use fixed labels such as the TN/TU source labels of Section 3.2. However, these labels give only coarse proxies for necessity, and the necessity of the tool changes as the model improves during training. Table 2 shows that many questions unsolved without the tool at first are solved without it later in training.[2] We therefore estimate tool necessity for the current model from the two arms and strengthen the decision on necessary questions, where $\bar{r}_T > \bar{r}_R$. These are exactly the questions on which SRA turns the decision advantage positive, so NDR only strengthens a signal that SRA makes positive.

Table 2: Questions unsolved without the tool that are solved later.

	Unsolved	Solved	Ratio
GRPO	1,068	349	.33
ARPO	1,028	364	.35
AXPO	639	247	.39

How Much to Strengthen. Necessary questions also differ in how much they need the tool. Here, we measure it with $\bar{r}_R$ of each question, not of each trajectory or of the whole batch. A high $\bar{r}_R$ means that the model already solves the question without the tool, whereas a low $\bar{r}_R$ leaves more room for the tool to help. We therefore set the strength of each question q to $\alpha_q = 1 - \bar{r}_R$. NDR adds this strength to the decision reward $\bar{r}_T$ of SRA only on necessary questions,

$$r_i^{\text{dec}} = \bar{r}_T + \alpha_q \, \mathbf{1}[\bar{r}_T > \bar{r}_R] \,, \tag{2}$$

where the indicator applies the strength only to necessary questions, and the decision advantage is computed as in Eq. 1. Other choices of the strength, such as a constant strength, are compared in Section 5.3.

How SRA and NDR Work Together. SRA separates the decision reward from later failures and decides its sign by comparing the two arms. NDR then strengthens the positive side according to how much the current model needs the tool. Together, they give a decision signal that differs by question, so tool use does not move in one direction.

4.3 TRAINING PROCEDURE

We use a two-phase training procedure, SFT followed by Tool-RLVR, following the baselines (Kang et al., 2026). For SFT, a larger teacher model generates tool-using trajectories with our tool schemas, and we fine-tune the base model on them with next-token prediction. We then apply Tool-RLVR from the SFT checkpoint, using questions that the SFT model still cannot solve together with additional hard questions. We retain the GRPO objective and assign each token the advantage of its corresponding stage,

$$\hat{A}_{i,t} = A_i^{\text{dec}} \, \mathbf{1}[t \in \text{dec}_i] + A_i^{\text{exec}} \, \mathbf{1}[t \in \text{exec}_i] + A_i^{\text{ans}} \, \mathbf{1}[t \in \text{ans}_i], \tag{3}$$

where dec_i, exec_i, and ans_i denote the three stages of each trajectory o_i.

5 EXPERIMENTS

5.1 EXPERIMENTAL SETTINGS

Models, Tools, and Baselines. We use Qwen3.5-2B and 4B (Qwen Team, 2026) with four tools: Python, Zoom-In, object segmentation (Carion et al., 2026), and depth estimation (Lin et al., 2025). We compare ANPO with GRPO (Shao et al., 2024), ARPO (Dong et al., 2025b), AKBE (Chen et al., 2026a), and AXPO (Kang et al., 2026), all starting from the same SFT model with the same data, tools, and prompts. Qwen3.5-9B and 27B are reported for comparison. During Tool-RLVR, every method samples eight rollouts per question and some baselines generate more, whereas ANPO splits the eight into $N_T = N_R = 4$ trajectories. Further details are given in Appendix B.

[2] In Table 2, the no-tool accuracy of a question is the accuracy of its trajectories that do not call the tool, and we keep questions with at least four such trajectories at both appearances. A question is unsolved when this accuracy is at most one quarter at its first appearance, and solved when it is at least one half at its second appearance, about one epoch later.

Table 3: Comparison of ANPO with Tool-RLVR methods on 15 benchmarks (Pass@1, %). The best and second-best results within each size are marked in bold and underlined.

Method	Recognition					Reasoning						Perception							Avg.
	Doc VQA	Chart QA	Text VQA	GQA	**Avg.**	Math Verse	Math Vista	Math Vision	Dyna Math	Logic Vista	**Avg.**	V^*	HR Bench	Tally QA	Visual Probe_H	VLMs AreBlind	Baby Vision	**Avg.**	
Qwen3.5-2B																			
Base	65.7	67.7	75.2	43.3	62.9	41.8	55.7	16.3	52.1	32.5	39.7	68.9	65.4	72.7	15.6	43.8	7.0	45.5	48.2
SFT	83.8	88.2	81.0	53.1	76.5	55.4	72.3	27.1	62.5	46.0	52.7	75.5	71.9	78.8	19.9	61.8	9.2	52.8	59.1
+GRPO	84.9	89.4	81.7	55.0	77.7	65.9	76.4	41.8	67.1	55.5	61.3	77.7	73.6	81.1	20.7	63.1	14.1	55.1	63.2
+ARPO	86.0	89.2	80.3	56.0	77.8	64.7	75.2	42.4	67.5	55.4	61.0	76.8	72.1	82.3	21.4	62.2	14.0	54.8	63.0
+AKBE	86.3	89.2	80.3	55.5	77.8	64.0	75.5	41.0	66.7	54.1	60.2	75.3	71.4	80.9	19.6	59.7	12.8	53.3	62.2
+AXPO	86.4	**89.7**	81.8	52.8	77.7	66.4	76.5	42.6	68.6	55.9	62.0	79.8	74.8	81.8	22.8	64.9	14.8	56.5	64.0
+ANPO	**87.2**	88.5	**82.8**	**56.6**	**78.8**	**67.7**	**78.0**	**44.6**	**69.3**	**58.0**	**63.5**	**82.9**	**75.7**	**84.0**	**23.6**	**66.3**	**15.4**	**58.0**	**65.4**
Δ vs. Best	+0.8	−1.2	+1.0	+0.6	+1.0	+1.3	+1.5	+2.0	+0.7	+2.1	+1.5	+3.1	+0.9	+1.7	+0.8	+1.4	+0.6	+1.5	+1.4
Qwen3.5-4B																			
Base	89.4	86.2	83.2	50.1	77.2	64.3	76.0	44.4	69.1	55.1	61.8	83.8	76.4	80.6	16.5	73.5	13.1	57.3	64.1
SFT	90.8	88.3	83.9	53.5	79.1	67.7	79.1	47.1	70.0	60.9	65.0	85.5	79.0	80.2	28.8	76.5	13.2	60.5	67.0
+GRPO	91.9	89.3	85.8	58.4	81.3	73.8	81.2	59.7	74.4	67.0	71.2	88.9	80.4	84.4	**29.3**	81.5	19.0	63.9	71.0
+ARPO	**92.4**	89.0	86.0	57.6	81.2	74.6	81.6	59.8	73.9	66.9	71.4	88.4	80.6	83.8	24.3	81.9	19.4	63.0	70.7
+AKBE	91.0	80.5	84.0	58.5	78.5	74.5	81.0	60.7	74.2	67.0	71.5	87.6	79.9	84.5	25.2	81.5	19.9	63.1	70.0
+AXPO	92.0	**90.2**	84.5	59.1	81.5	75.2	81.4	59.0	73.9	67.4	71.4	**90.8**	81.2	84.6	26.2	82.1	20.6	64.2	71.2
+ANPO	91.9	89.6	**86.5**	**59.3**	**81.8**	**75.6**	**81.7**	**61.1**	**74.9**	**68.4**	**72.3**	89.0	**82.0**	**85.6**	27.1	**85.6**	**21.0**	**65.1**	**72.0**
Δ vs. Best	−0.5	−0.6	+0.5	+0.2	+0.3	+0.4	+0.1	+0.4	+0.5	+1.0	+0.8	−1.8	+0.8	+1.0	−2.2	+3.5	+0.4	+0.9	+0.8
Qwen3.5-9B																			
Base	87.3	85.9	86.5	51.1	77.7	65.4	82.8	51.6	72.6	58.7	66.2	88.2	78.2	83.7	25.0	78.5	14.4	61.3	67.3
Qwen3.5-27B																			
Base	92.3	85.9	88.4	57.6	81.1	75.0	81.1	65.6	74.7	70.5	73.4	91.5	82.5	84.3	29.4	79.8	19.8	64.6	71.9

Evaluation. To assess both diverse abilities and tool selectivity, we evaluate all methods on fifteen benchmarks in three groups, ***Recognition***, ***Reasoning***, and ***Perception***, which differ in how much a visual tool is expected to help. We report Pass@1, the accuracy averaged over four rollouts, and the tool-call rate (TCR), the fraction of trajectories that call at least one tool. Pass@4, which counts a question as solved if any of the four rollouts is correct, is reported in Appendix C.1.

5.2 Quantitative Results

Overall Performance. In Table 3, we compare ANPO with the Tool-RLVR baselines at two model sizes. We have the following key observations: **1)** ANPO achieves the highest average accuracy on every benchmark group at both sizes. The gains appear not only on perception, where the visual tool is expected to help most, but also on reasoning, which suggests that ANPO improves more than tool use alone. **2)** At 2B size, ANPO outperforms the Qwen3.5-4B base model and falls only slightly below the 9B base model, and at 4B size, it is comparable to the 27B base model. This suggests that ANPO narrows the gap between small MLLM agents and much larger base models. **3)** ANPO and AXPO, the methods that do not fall into *tool collapse*, outperform the other baselines on perception, which is consistent with retaining tool use where the tool helps.

Tool-Call Rate. We report the tool-call rate of the 2B models on each benchmark group in Table 4. **1)** GRPO, ARPO, and AKBE all fall into *tool collapse*, whereas ANPO keeps an average tool-call rate close to that of SFT. **2)** AXPO calls tools more often than SFT on every group, whereas ANPO calls them most on perception and least on reasoning, adjusting its tool use to each group.

Table 4: Tool-call rate (%) on each benchmark group.

	Recog.	Reason.	Percep.	Avg.
SFT	17.4	23.6	67.9	39.7
+GRPO	1.3	1.5	11.1	5.3
+ARPO	1.6	2.5	5.7	3.6
+AKBE	0.0	0.1	0.1	0.1
+AXPO	78.3	36.1	78.8	64.4
+ANPO	49.8	7.7	57.5	38.9

5.3 Ablation Studies

Effect of Core Components. We conduct ablation experiments to understand the effect of each component of ANPO in Table 5. **1)** Dual-arm rollouts alone are not enough, as both the accuracy and the tool-call rate stay close to those of GRPO. **2)** SRA mitigates *tool collapse*, bringing the tool-call rate on perception back to the level of SFT while keeping it low on reasoning, but the gain in accuracy remains limited. **3)** Since NDR strengthens the decision advantage of SRA, we apply it on top of SRA. Applying NDR to every question regardless of tool necessity raises the tool-call rate on both groups, which improves perception but lowers reasoning below SRA. **4)** Applying

Table 5: Ablation of the components of ANPO. Starting from GRPO, each row adds one component, and ANPO applies NDR only to necessary questions.

Method	Reasoning		Perception	
	Pass@1	TCR	Pass@1	TCR
SFT	52.7	23.6	52.8	67.9
GRPO	61.3	1.5	55.1	11.1
+ Dual-Arm	61.4	3.9	55.3	16.3
+ SRA	62.4	3.9	56.4	69.2
+ NDR (all)	61.5	12.9	57.2	78.4
ANPO	63.5	7.7	58.0	57.5

NDR only to necessary questions gives the highest accuracy on both groups with fewer tool calls than applying it to every question.

Table 6: Design choices of ANPO. Each block changes one choice from the full model (shaded).

Variant	Reasoning Pass@1	Reasoning TCR	Perception Pass@1	Perception TCR
Rollouts				
Native	61.5	10.4	58.2	76.8
Dual-Arm	63.5	7.7	58.0	57.5
Normalization				
Pooled	62.1	3.8	56.2	46.0
Within arm	63.5	7.7	58.0	57.5

(a) Rollouts and normalization

Variant	Reasoning Pass@1	Reasoning TCR	Perception Pass@1	Perception TCR
Decision reward				
Per trajectory	62.8	13.6	55.6	51.7
Broadcast	63.5	7.7	58.0	57.5
Necessity signal				
TN/TU label	59.1	11.7	55.2	71.2
$\bar{r}_T > \bar{r}_R$	63.5	7.7	58.0	57.5

(b) Decision and necessity

Variant	Reasoning Pass@1	Reasoning TCR	Perception Pass@1	Perception TCR
Constant strength				
0.3	62.6	11.3	55.2	75.2
0.1	62.5	8.4	55.8	55.0
Adaptive strength				
Batch mean	61.1	5.8	55.2	71.5
Per question	63.5	7.7	58.0	57.5

(c) Strength

Design Choices. In Table 6, we compare each design choice of ANPO with a simpler alternative on the 2B model. *Rollouts and normalization.* When the model decides by itself whether to call the tool (Native vs. Dual-Arm), it calls the tool excessively and its reasoning accuracy drops. Normalizing the answer advantage over both arms together (Pooled vs. Within arm) lowers accuracy on both groups. This shows that both the separate arms and the normalization within each arm contribute to the overall accuracy of ANPO. *Decision and necessity.* Rewarding the decision per trajectory (Per trajectory vs. Broadcast) lowers the perception tool-call rate and accuracy, as the decision is no longer protected from later failures. A fixed label (TN/TU label vs. $\bar{r}_T > \bar{r}_R$) raises the tool-call rate while lowering accuracy, as it does not follow the model during training. *Strength.* A constant strength or a batch-mean $\bar{r}_R$ (Constant and Batch mean vs. Per question) both fall short of ANPO, which supports computing the strength per question.

5.4 Additional Analyses

Tool Use Follows the Difficulty. In Figure 11, we repeat the controlled experiment of Figure 2(bottom). ANPO achieves the best performance at every difficulty level. As for tool use, AXPO calls the tool on almost every question and GRPO and ARPO on almost none, regardless of difficulty. In contrast, ANPO calls the tool less as the target becomes easier, which suggests that it decides whether to call the tool according to the difficulty of the question.

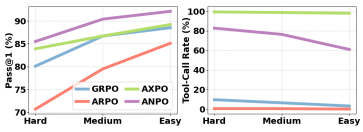


Figure 11: Pass@1 and TCR under controlled difficulty on HR-Bench.

Calls the Tool Where It Is Necessary. We next ask whether ANPO calls the tool on the questions where it is necessary. To this end, we label a question as necessary when the policy fails without the tool but succeeds with it, and as unnecessary when the policy succeeds without the tool, as detailed in Appendix D.1. Figure 12 compares the call rate of each policy (Natural) with the call rate it would have if it called the tool on exactly the necessary questions (Oracle). ANPO stays close to its oracle, whereas AXPO calls the tool far above it. Over all reasoning and perception benchmarks, the per-question correlation between the call rate and the label is also higher for ANPO (0.757) than for AXPO (0.675), which suggests that ANPO calls the tool on the questions where it is necessary.

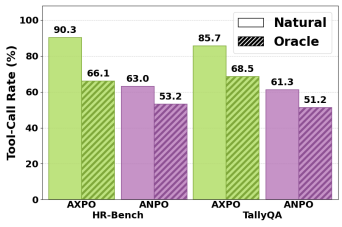


Figure 12: Natural versus oracle TCR on HR-Bench and TallyQA.

ANPO Gains the Most from Tool Access. To check *how* well each method uses the tool, we compare its accuracy with and without tool access on perception tasks in Figure 13. GRPO and ARPO show little difference between the two settings, which indicates that they gain little from the tool. In contrast, ANPO gains more from tool access than AXPO and also achieves the highest accuracy without the tool. This suggests that ANPO both benefits from calling the tool and answers more accurately without it. We also report the tool-call rate and accuracy per tool in Appendix D.2.

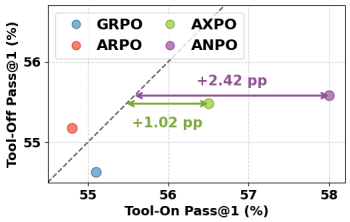


Figure 13: Pass@1 on perception with and without tool access.

Qualitative Examples. Figure 14 shows examples of *how* and *when* ANPO uses tools. On the left, a math question requires precise numerical computation. GRPO and AXPO approximate the computation by hand, and the errors accumulate into a wrong answer. In contrast, ANPO performs the precise computation with Python and reaches the correct answer, which shows that ANPO uses

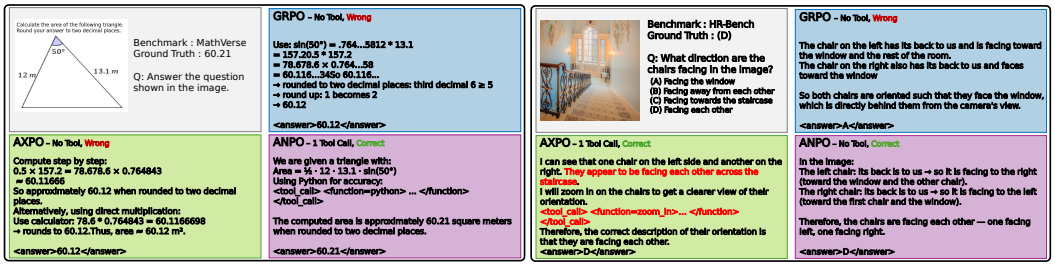


Figure 14: Qualitative examples. (Left) ANPO uses Python for a precise computation that the other methods approximate by hand. (Right) AXPO makes a *redundant tool call* after reaching the answer, whereas ANPO answers directly.

the tool effectively. On the right, for a fine-grained perception question, GRPO answers incorrectly, and AXPO calls the tool even though it has already confirmed the answer (red), a *redundant tool call* (Ma et al., 2026; Feng et al., 2026). In contrast, ANPO answers directly after checking the image, which shows that it skips the tool when it can already answer.

Efficiency Comparison. Table 7 compares the training cost and tool use of the 2B models. We measure the training time per step, the average number of tool calls per question, and the redundant call rate. The redundant call rate is the fraction of tool-using trajectories that already state the correct answer before their first tool call, as in the right example of Figure 14, and is judged by GPT-4o-mini (OpenAI et al., 2024) as detailed in Appendix D.3. We have the following observations: **1)** ANPO trains close to GRPO and faster than ARPO and AXPO, since it adds no rollouts, whereas ARPO and AXPO require additional sampling or branching. **2)** ANPO makes fewer calls than AXPO while avoiding the near-zero calls of GRPO and ARPO. **3)** ANPO has the lowest redundant call rate, which shows that it rarely calls the tool once its own reasoning has already reached the answer.

Table 7: Training time, tool calls, and redundant calls of the 2B models.

	Train. time (s)	Avg. calls	Redundant (%)
GRPO	780	0.08	9.6
ARPO	863	0.06	21.7
AXPO	920	0.96	16.8
ANPO	820	0.54	5.1

6 Related Works

Tool-RLVR has been studied first for search and computation in text (Li et al., 2025b; Feng et al., 2025a; Qian et al., 2025) and then for visual tools (Lai et al., 2025; Zhao et al., 2026). Among them, ARPO (Dong et al., 2025b) branches rollouts after a tool call when the token entropy rises, but still credits each trajectory as a whole, so a failure after a proper decision penalizes the decision. AXPO (Kang et al., 2026) resamples continuations from the tool call in groups where every tool-using trajectory fails, but judges the call by whether a continuation succeeds rather than by whether the tool is needed, which requires additional rollouts and adds training cost. AKBE (Chen et al., 2026a) compares with tool-free rollouts during training, but reduces the comparison to a binary label of whether the tool is needed and uses it only as an imitation target. Another line learns when to call the tool by routing each query with a special token (Ma et al., 2026), scoring how much the tool helps each question before training (Wang et al., 2026a), or labeling necessity with additional forced-tool rollouts (Li et al., 2025a). In contrast, ANPO judges necessity during training without additional rollouts, a special token, or a label fixed in advance, and gives every tool-using trajectory of a question the same decision signal regardless of whether it succeeds. More detailed related works are given in Appendix E.

7 Conclusion

In this work, we study how small MLLM agents learn to use visual tools under Tool-RLVR and identify two root causes, ***misaligned reward assignment*** and ***necessity-agnostic tool learning***. To address them, we propose ANPO, which rewards each stage of tool use separately through SRA and strengthens the decision on questions where the tool is necessary through NDR. These simple components, grounded in our thorough analysis, change only how the advantage is assigned to the tokens of each trajectory, without additional rollouts or models. Extensive experiments show that ANPO achieves the highest accuracy among Tool-RLVR methods while calling the tool mainly where it is necessary. Regarding the limitations and future work, please refer to Appendix F.

AI USE STATEMENT

In this work, we used generative AI tools for writing and debugging code, running and monitoring experiments, and revising the manuscript to improve the tone, grammar, and readability of the text. We have not used generative AI tools for designing the method or determining the claims of this work. We have reviewed all AI-assisted work and verified that the revisions preserved the intended scientific meaning. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

To ensure the reproducibility of the results throughout the paper, we describe the experimental settings and the training details in Section 5.1 and Appendix B, respectively. Appendix B covers the training data (Appendix B.1), the benchmarks (Appendix B.2), the policy gradient objective (Appendix B.3), the implementation of ANPO (Appendix B.4), the hyperparameters (Appendix B.5), and the prompts and tool schemas (Appendix B.6). All models start from public Qwen3.5 checkpoints, and all training and evaluation data come from public datasets. The code is available at `https://anonymous.4open.science/r/ANPO-FF7C/`.

ETHICS STATEMENT

This work is conducted exclusively with publicly available models and datasets, which do not contain any personally identifiable information. The trained agents call only local tools, namely ZoomIn, Python in a sandbox that cannot write files, object segmentation, and depth estimation, and have no access to the internet or external services. To the best of our knowledge, this work does not raise any ethical concerns. We have read and adhere to the ICLR Code of Ethics.

REFERENCES

Manoj Acharya, Kushal Kafle, and Christopher Kanan. Tallyqa: Answering complex counting questions, 2018. URL https://arxiv.org/abs/1810.12440.

Xiang An, Yin Xie, Kaicheng Yang, Wenkang Zhang, Xiuwei Zhao, Zheng Cheng, Yirui Wang, Songcen Xu, Changrui Chen, Didi Zhu, Chunsheng Wu, Huajie Tan, Chunyuan Li, Jing Yang, Jie Yu, Xiyao Wang, Bin Qin, Yumeng Wang, Zizhen Yan, Ziyong Feng, Ziwei Liu, Bo Li, and Jiankang Deng. Llava-onevision-1.5: Fully open framework for democratized multimodal training, 2025. URL https://arxiv.org/abs/2509.23661.

Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. Qwen-vl: A versatile vision-language model for understanding, localization, text reading, and beyond, 2023. URL https://arxiv.org/abs/2308.12966.

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

Hunar Batra, Haoqin Tu, Hardy Chen, Yuanze Lin, Cihang Xie, and Ronald Clark. Spatialthinker: Reinforcing scene graph-grounded spatial reasoning via dense rewards, 2026. URL https://arxiv.org/abs/2511.07403.

Yuxuan Cai, Jiangning Zhang, Haoyang He, Xinwei He, Ao Tong, Zhenye Gan, Chengjie Wang, Zhucun Xue, Yong Liu, and Xiang Bai. Llava-kd: A framework of distilling multimodal large language models, 2025. URL https://arxiv.org/abs/2410.16236.

Nicolas Carion, Laura Gustafson, Yuan-Ting Hu, Shoubhik Debnath, Ronghang Hu, Didac Suris, Chaitanya Ryali, Kalyan Vasudev Alwala, Haitham Khedr, Andrew Huang, Jie Lei, Tengyu Ma, Baishan Guo, Arpit Kalla, Markus Marks, Joseph Greer, Meng Wang, Peize Sun, Roman Rädle, Triantafyllos Afouras, Effrosyni Mavroudi, Katherine Xu, Tsung-Han Wu, Yu Zhou, Liliane Momeni, Rishi Hazra, Shuangrui Ding, Sagar Vaze, Francois Porcher, Feng Li, Siyuan Li, Aishwarya Kamath, Ho Kei Cheng, Piotr Dollár, Nikhila Ravi, Kate Saenko, Pengchuan Zhang, and Christoph Feichtenhofer. Sam 3: Segment anything with concepts, 2026. URL https://arxiv.org/abs/2511.16719.

Dingwei Chen, Zefang Zong, Zhipeng Ma, Leo Luo, Yang Li, Chengming Li, Peng Chen, and Jie Jiang. Efficient agentic reinforcement learning with on-policy intrinsic knowledge boundary enhancement, 2026a. URL https://arxiv.org/abs/2605.26952.

Liang Chen, Haozhe Zhao, Tianyu Liu, Shuai Bai, Junyang Lin, Chang Zhou, and Baobao Chang. An image is worth 1/2 tokens after layer 2: Plug-and-play inference acceleration for large vision-language models, 2024. URL https://arxiv.org/abs/2403.06764.

Liang Chen, Weichu Xie, Yiyan Liang, Hongfeng He, Hans Zhao, Zhibo Yang, Zhiqi Huang, Haoning Wu, Haoyu Lu, Y. charles, Yiping Bao, Yuantao Fan, Guopeng Li, Haiyang Shen, Xuanzhong Chen, Wendong Xu, Shuzheng Si, Zefan Cai, Wenhao Chai, Ziqi Huang, Fangfu Liu, Tianyu Liu, Baobao Chang, Ming Wu, Xiaobo Hu, Kaiyuan Chen, Yixin Ren, Yang Liu, Yuan Gong, and Kuan Li. Babyvision: Visual reasoning beyond language, 2026b. URL https://arxiv.org/abs/2601.06521.

Matt Deitke, Christopher Clark, Sangho Lee, Rohun Tripathi, Yue Yang, Jae Sung Park, Mohammadreza Salehi, Niklas Muennighoff, Kyle Lo, Luca Soldaini, Jiasen Lu, Taira Anderson, Erin Bransom, Kiana Ehsani, Huong Ngo, YenSung Chen, Ajay Patel, Mark Yatskar, Chris Callison-Burch, Andrew Head, Rose Hendrix, Favyen Bastani, Eli VanderBilt, Nathan Lambert, Yvonne Chou, Arnavi Chheda, Jenna Sparks, Sam Skjonsberg, Michael Schmitz, Aaron Sarnat, Byron Bischoff, Pete Walsh, Chris Newell, Piper Wolters, Tanmay Gupta, Kuo-Hao Zeng, Jon Borchardt, Dirk Groeneveld, Crystal Nam, Sophie Lebrecht, Caitlin Wittlif, Carissa Schoenick, Oscar Michel, Ranjay Krishna, Luca Weihs, Noah A. Smith, Hannaneh Hajishirzi, Ross Girshick, Ali Farhadi, and Aniruddha Kembhavi. Molmo and pixmo: Open weights and open data for state-of-the-art vision-language models, 2024. URL https://arxiv.org/abs/2409.17146.

Chengqi Dong, Chuhuai Yue, Hang He, yandong liu, Fenghe Tang, S Kevin Zhou, Xiaohan Wang, Jiajun Chai, and Guojun Yin. Tapo: Tool-aware policy optimization via credit transfer for multimodal search agents, 2026. URL https://arxiv.org/abs/2606.05784.

Guanting Dong, Licheng Bao, Zhongyuan Wang, Kangzhi Zhao, Xiaoxi Li, Jiajie Jin, Jinghan Yang, Hangyu Mao, Fuzheng Zhang, Kun Gai, Guorui Zhou, Yutao Zhu, Ji-Rong Wen, and Zhicheng Dou. Agentic entropy-balanced policy optimization, 2025a. URL https://arxiv.org/abs/2510.14545.

Guanting Dong, Hangyu Mao, Kai Ma, Licheng Bao, Yifei Chen, Zhongyuan Wang, Zhongxia Chen, Jiazhen Du, Huiyang Wang, Fuzheng Zhang, Guorui Zhou, Yutao Zhu, Ji-Rong Wen, and Zhicheng Dou. Agentic reinforced policy optimization, 2025b. URL https://arxiv.org/abs/2507.19849.

Mengfei Du, Binhao Wu, Zejun Li, Xuanjing Huang, and Zhongyu Wei. Embspatial-bench: Benchmarking spatial understanding for embodied tasks with large vision-language models, 2024. URL https://arxiv.org/abs/2406.05756.

Jiazhan Feng, Shijue Huang, Xingwei Qu, Ge Zhang, Yujia Qin, Baoquan Zhong, Chengquan Jiang, Jinxin Chi, and Wanjun Zhong. Retool: Reinforcement learning for strategic tool use in llms, 2025a. URL https://arxiv.org/abs/2504.11536.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for llm agent training, 2025b. URL https://arxiv.org/abs/2505.10978.

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.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers, 2023. URL https://arxiv.org/abs/2210.17323.

GLM-5 Team. Glm-5: from vibe coding to agentic engineering, 2026. URL https://arxiv.org/abs/2602.15763.

Xianda Guo, Ruijun Zhang, Yiqun Duan, Yuhang He, Dujun Nie, Wenke Huang, Chenming Zhang, Shuai Liu, Hao Zhao, and Long Chen. Surds: Benchmarking spatial understanding and reasoning in driving scenarios with vision language models, 2025. URL https://arxiv.org/abs/2411.13112.

Drew A. Hudson and Christopher D. Manning. Gqa: A new dataset for real-world visual reasoning and compositional question answering, 2019. URL https://arxiv.org/abs/1902.09506.

Jaehyun Jang, Eunseop Yoon, Hee Suk Yoon, SooHwan Eom, Mark A. Hasegawa-Johnson, and Chang D. Yoo. Token-level response-visual attention guidance for multimodal llms knowledge distillation, 2026. URL https://arxiv.org/abs/2607.02593.

Boseong Jeon, Yunho Choi, and Taehan Kim. Shallow-π: Knowledge distillation for flow-based vlas, 2026. URL https://arxiv.org/abs/2601.20262.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning, 2025. URL https://arxiv.org/abs/2503.09516.

Minki Kang, Shizhe Diao, Ryo Hachiuma, Sung Ju Hwang, Pavlo Molchanov, Yu-Chiang Frank Wang, and Byung-Kwan Lee. Agent explorative policy optimization for multimodal agentic reasoning, 2026. URL https://arxiv.org/abs/2605.28774.

Dong-Hee Kim, Reuben Tan, and Donghyun Kim. Diversity over frequency: Rethinking tool use in visual chain-of-thought agents, 2026a. URL https://arxiv.org/abs/2606.00096.

Jiwan Kim, Kibum Kim, Sangwoo Seo, and Chanyoung Park. Compodistill: Attention distillation for compositional reasoning in multimodal llms, 2026b. URL https://arxiv.org/abs/2510.12184.

Kibum Kim, Jiwan Kim, Kyle Min, Yueqi Wang, Jinyoung Moon, Julian McAuley, and Chanyoung Park. Sink-token-aware pruning for fine-grained video understanding in efficient video llms, 2026c. URL https://arxiv.org/abs/2604.20937.

Kimi Team. Kimi k3: Open frontier intelligence, 2026. URL https://arxiv.org/abs/2607.24653.

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, 2025. URL https://arxiv.org/abs/2509.07969.

Byung-Kwan Lee, Sangyun Chung, Chae Won Kim, Beomchan Park, and Yong Man Ro. Phantom of latent for large language and vision models, 2024a. URL https://arxiv.org/abs/2409.14713.

Byung-Kwan Lee, Beomchan Park, Chae Won Kim, and Yong Man Ro. Moai: Mixture of all intelligence for large language and vision models, 2024b. URL https://arxiv.org/abs/2403.07508.

Byung-Kwan Lee, Ryo Hachiuma, Yong Man Ro, Yu-Chiang Frank Wang, and Yueh-Hua Wu. Unified reinforcement and imitation learning for vision-language models, 2025a. URL https://arxiv.org/abs/2510.19307.

Byung-Kwan Lee, Ryo Hachiuma, Yu-Chiang Frank Wang, Yong Man Ro, and Yueh-Hua Wu. Vlsi: Verbalized layers-to-interactions from large to small vision language models, 2025b. URL https://arxiv.org/abs/2412.01822.

Byung-Kwan Lee, Yu-Chiang Frank Wang, and Ryo Hachiuma. Masking teacher and reinforcing student for distilling vision-language models, 2025c. URL https://arxiv.org/abs/2512.22238.

Byung-Kwan Lee, Ryo Hachiuma, Yong Man Ro, Yu-Chiang Frank Wang, and Yueh-Hua Wu. Genrecal: Generation after recalibration from large to small vision-language models, 2026. URL https://arxiv.org/abs/2506.15681.

Chunyuan Li, Haotian Liu, Liunian Harold Li, Pengchuan Zhang, Jyoti Aneja, Jianwei Yang, Ping Jin, Houdong Hu, Zicheng Liu, Yong Jae Lee, and Jianfeng Gao. Elevater: A benchmark and toolkit for evaluating language-augmented visual models, 2022. URL https://arxiv.org/abs/2204.08790.

Xuchen Li, Xuzhao Li, Jiahui Gao, Renjie Pi, Shiyu Hu, and Wentao Zhang. Look less, reason more: Rollout-guided adaptive pixel-space reasoning, 2025a. URL https://arxiv.org/abs/2510.01681.

Xuefeng Li, Haoyang Zou, and Pengfei Liu. Torl: Scaling tool-integrated rl, 2025b. URL https://arxiv.org/abs/2503.23383.

Yesheng Liang, Haisheng Chen, Zihan Zhang, Song Han, and Zhijian Liu. Paroquant: Pairwise rotation quantization for efficient reasoning llm inference, 2026. URL https://arxiv.org/abs/2511.10645.

Haotong Lin, Sili Chen, Junhao Liew, Donny Y. Chen, Zhenyu Li, Guang Shi, Jiashi Feng, and Bingyi Kang. Depth anything 3: Recovering the visual space from any views, 2025. URL https://arxiv.org/abs/2511.10647.

Honglin Lin, Zheng Liu, Yun Zhu, Chonghan Qin, Juekai Lin, Xiaoran Shang, Conghui He, Wentao Zhang, and Lijun Wu. Mmfinereason: Closing the multimodal reasoning gap via open data-centric methods, 2026a. URL https://arxiv.org/abs/2601.21821.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. Awq: Activation-aware weight quantization for llm compression and acceleration, 2026b. URL https://arxiv.org/abs/2306.00978.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning, 2023. URL https://arxiv.org/abs/2304.08485.

Jingping Liu, Ziyan Liu, Zhedong Cen, Yan Zhou, Yinan Zou, Weiyan Zhang, Haiyun Jiang, and Tong Ruan. Can multimodal large language models understand spatial relations? In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 620–632. Association for Computational Linguistics, 2025a. doi: 10.18653/v1/2025.acl-long.31. URL http://dx.doi.org/10.18653/v1/2025.acl-long.31.

Shih-Yang Liu, Xin Dong, Ximing Lu, Shizhe Diao, Mingjie Liu, Min-Hung Chen, Hongxu Yin, Yu-Chiang Frank Wang, Kwang-Ting Cheng, Yejin Choi, Jan Kautz, and Pavlo Molchanov. Dler: Doing length penalty right - incentivizing more intelligence per token via reinforcement learning, 2025b. URL https://arxiv.org/abs/2510.15110.

Ziyu Liu, Zeyi Sun, Yuhang Zang, Xiaoyi Dong, Yuhang Cao, Haodong Duan, Dahua Lin, and Jiaqi Wang. Visual-rft: Visual reinforcement fine-tuning, 2025c. URL https://arxiv.org/abs/2503.01785.

Pan Lu, Ran Gong, Shibiao Jiang, Liang Qiu, Siyuan Huang, Xiaodan Liang, and Song-Chun Zhu. Inter-gps: Interpretable geometry problem solving with formal language and symbolic reasoning, 2021. URL https://arxiv.org/abs/2105.04165.

Pan Lu, Liang Qiu, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, Tanmay Rajpurohit, Peter Clark, and Ashwin Kalyan. Dynamic prompt learning via policy gradient for semi-structured mathematical reasoning, 2023. URL https://arxiv.org/abs/2209.14610.

Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. Mathvista: Evaluating mathematical reasoning of foundation models in visual contexts, 2024. URL https://arxiv.org/abs/2310.02255.

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, 2026. URL https://arxiv.org/abs/2605.19852.

Wufei Ma, Haoyu Chen, Guofeng Zhang, Yu-Cheng Chou, Jieneng Chen, Celso M de Melo, and Alan Yuille. 3dsrbench: A comprehensive 3d spatial reasoning benchmark, 2025a. URL https://arxiv.org/abs/2412.07825.

Zixian Ma, Jianguo Zhang, Zhiwei Liu, Jieyu Zhang, Juntao Tan, Manli Shu, Juan Carlos Niebles, Shelby Heinecke, Huan Wang, Caiming Xiong, Ranjay Krishna, and Silvio Savarese. Latte: Learning to think with vision specialists, 2025b. URL https://arxiv.org/abs/2412.05479.

Kenneth Marino, Mohammad Rastegari, Ali Farhadi, and Roozbeh Mottaghi. Ok-vqa: A visual question answering benchmark requiring external knowledge, 2019. URL https://arxiv.org/abs/1906.00067.

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, 2022. URL https://arxiv.org/abs/2203.10244.

Minesh Mathew, Dimosthenis Karatzas, and C. V. Jawahar. Docvqa: A dataset for vqa on document images, 2021. URL https://arxiv.org/abs/2007.00398.

Fanqing Meng, Lingxiao Du, Zongkai Liu, Zhixiang Zhou, Quanfeng Lu, Daocheng Fu, Tiancheng Han, Botian Shi, Wenhai Wang, Junjun He, Kaipeng Zhang, Ping Luo, Yu Qiao, Qiaosheng Zhang, and Wenqi Shao. Mm-eureka: Exploring the frontiers of multimodal reasoning with rule-based reinforcement learning, 2025. URL https://arxiv.org/abs/2503.07365.

OpenAI, :, Aaron Hurst, Adam Lerer, Adam P. Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, Aleksander Madry, Alex Baker-Whitcomb, Alex Beutel, Alex Borzunov, Alex Carney, Alex Chow, Alex Kirillov, Alex Nichol, Alex Paino, Alex Renzin, Alex Tachard Passos, Alexander Kirillov, Alexi Christakis, Alexis Conneau, Ali Kamali, Allan Jabri, Allison Moyer, Allison Tam, Amadou Crookes, Amin Tootoochian, Amin Tootoonchian, Ananya Kumar, Andrea Vallone, Andrej Karpathy, Andrew Braunstein, Andrew Cann, Andrew Codispoti, Andrew Galu, Andrew Kondrich, Andrew Tulloch, Andrey Mishchenko, Angela Baek, Angela Jiang, Antoine Pelisse, Antonia Woodford, Anuj Gosalia, Arka Dhar, Ashley Pantuliano, Avi Nayak, Avital Oliver, Barret Zoph, Behrooz Ghorbani, Ben Leimberger, Ben Rossen, Ben Sokolowsky, Ben Wang, Benjamin Zweig, Beth Hoover, Blake Samic, Bob McGrew, Bobby Spero, Bogo Giertler, Bowen Cheng, Brad Lightcap, Brandon Walkin, Brendan Quinn, Brian Guarraci, Brian Hsu, Bright Kellogg, Brydon Eastman, Camillo Lugaresi, Carroll Wainwright, Cary Bassin, Cary Hudson, Casey Chu, Chad Nelson, Chak Li, Chan Jun Shern, Channing Conger, Charlotte Barette, Chelsea Voss, Chen Ding, Cheng Lu, Chong Zhang, Chris Beaumont, Chris Hallacy, Chris Koch, Christian Gibson, Christina Kim, Christine Choi, Christine McLeavey, Christopher Hesse, Claudia Fischer, Clemens Winter, Coley Czarnecki, Colin Jarvis, Colin Wei, Constantin Koumouzelis, Dane Sherburn, Daniel Kappler, Daniel Levin, Daniel Levy, David Carr, David Farhi, David Mely, David Robinson, David Sasaki, Denny Jin, Dev Valladares, Dimitris Tsipras, Doug Li, Duc Phong Nguyen, Duncan Findlay, Edede Oiwoh, Edmund Wong, Ehsan Asdar, Elizabeth Proehl, Elizabeth Yang, Eric Antonow, Eric Kramer, Eric Peterson, Eric Sigler, Eric Wallace, Eugene Brevdo, Evan Mays, Farzad Khorasani, Felipe Petroski Such, Filippo Raso, Francis Zhang, Fred von Lohmann, Freddie Sulit, Gabriel Goh, Gene Oden, Geoff Salmon, Giulio Starace, Greg Brockman, Hadi Salman, Haiming Bao, Haitang Hu, Hannah Wong, Haoyu Wang, Heather Schmidt, Heather Whitney, Heewoo Jun, Hendrik Kirchner, Henrique Ponde de Oliveira Pinto, Hongyu Ren, Huiwen Chang, Hyung Won Chung, Ian Kivlichan, Ian O'Connell, Ian O'Connell, Ian Osband, Ian Silber, Ian Sohl, Ibrahim Okuyucu, Ikai Lan, Ilya Kostrikov, Ilya Sutskever, Ingmar Kanitscheider, Ishaan Gulrajani, Jacob Coxon, Jacob Menick, Jakub Pachocki, James Aung, James Betker, James Crooks, James Lennon, Jamie Kiros, Jan Leike, Jane Park, Jason Kwon, Jason Phang, Jason Teplitz, Jason Wei, Jason Wolfe, Jay Chen, Jeff Harris, Jenia Varavva, Jessica Gan Lee, Jessica Shieh, Ji Lin, Jiahui Yu, Jiayi Weng, Jie Tang, Jieqi Yu, Joanne Jang, Joaquin Quinonero Candela, Joe Beutler, Joe Landers, Joel Parish, Johannes Heidecke, John Schulman, Jonathan Lachman, Jonathan McKay, Jonathan Uesato, Jonathan Ward, Jong Wook Kim, Joost Huizinga, Jordan Sitkin, Jos Kraaijeveld, Josh Gross, Josh Kaplan, Josh Snyder, Joshua Achiam, Joy Jiao, Joyce Lee, Juntang Zhuang,

Justyn Harriman, Kai Fricke, Kai Hayashi, Karan Singhal, Katy Shi, Kavin Karthik, Kayla Wood, Kendra Rimbach, Kenny Hsu, Kenny Nguyen, Keren Gu-Lemberg, Kevin Button, Kevin Liu, Kiel Howe, Krithika Muthukumar, Kyle Luther, Lama Ahmad, Larry Kai, Lauren Itow, Lauren Workman, Leher Pathak, Leo Chen, Li Jing, Lia Guy, Liam Fedus, Liang Zhou, Lien Mamitsuka, Lilian Weng, Lindsay McCallum, Lindsey Held, Long Ouyang, Louis Feuvrier, Lu Zhang, Lukas Kondraciuk, Lukasz Kaiser, Luke Hewitt, Luke Metz, Lyric Doshi, Mada Aflak, Maddie Simens, Madelaine Boyd, Madeleine Thompson, Marat Dukhan, Mark Chen, Mark Gray, Mark Hudnall, Marvin Zhang, Marwan Aljubeh, Mateusz Litwin, Matthew Zeng, Max Johnson, Maya Shetty, Mayank Gupta, Meghan Shah, Mehmet Yatbaz, Meng Jia Yang, Mengchao Zhong, Mia Glaese, Mianna Chen, Michael Janner, Michael Lampe, Michael Petrov, Michael Wu, Michele Wang, Michelle Fradin, Michelle Pokrass, Miguel Castro, Miguel Oom Temudo de Castro, Mikhail Pavlov, Miles Brundage, Miles Wang, Minal Khan, Mira Murati, Mo Bavarian, Molly Lin, Murat Yesildal, Nacho Soto, Natalia Gimelshein, Natalie Cone, Natalie Staudacher, Natalie Summers, Natan LaFontaine, Neil Chowdhury, Nick Ryder, Nick Stathas, Nick Turley, Nik Tezak, Niko Felix, Nithanth Kudige, Nitish Keskar, Noah Deutsch, Noel Bundick, Nora Puckett, Ofir Nachum, Ola Okelola, Oleg Boiko, Oleg Murk, Oliver Jaffe, Olivia Watkins, Olivier Godement, Owen Campbell-Moore, Patrick Chao, Paul McMillan, Pavel Belov, Peng Su, Peter Bak, Peter Bakkum, Peter Deng, Peter Dolan, Peter Hoeschele, Peter Welinder, Phil Tillet, Philip Pronin, Philippe Tillet, Prafulla Dhariwal, Qiming Yuan, Rachel Dias, Rachel Lim, Rahul Arora, Rajan Troll, Randall Lin, Rapha Gontijo Lopes, Raul Puri, Reah Miyara, Reimar Leike, Renaud Gaubert, Reza Zamani, Ricky Wang, Rob Donnelly, Rob Honsby, Rocky Smith, Rohan Sahai, Rohit Ramchandani, Romain Huet, Rory Carmichael, Rowan Zellers, Roy Chen, Ruby Chen, Ruslan Nigmatullin, Ryan Cheu, Saachi Jain, Sam Altman, Sam Schoenholz, Sam Toizer, Samuel Miserendino, Sandhini Agarwal, Sara Culver, Scott Ethersmith, Scott Gray, Sean Grove, Sean Metzger, Shamez Hermani, Shantanu Jain, Shengjia Zhao, Sherwin Wu, Shino Jomoto, Shirong Wu, Shuaiqi, Xia, Sonia Phene, Spencer Papay, Srinivas Narayanan, Steve Coffey, Steve Lee, Stewart Hall, Suchir Balaji, Tal Broda, Tal Stramer, Tao Xu, Tarun Gogineni, Taya Christianson, Ted Sanders, Tejal Patwardhan, Thomas Cunninghman, Thomas Degry, Thomas Dimson, Thomas Raoux, Thomas Shadwell, Tianhao Zheng, Todd Underwood, Todor Markov, Toki Sherbakov, Tom Rubin, Tom Stasi, Tomer Kaftan, Tristan Heywood, Troy Peterson, Tyce Walters, Tyna Eloundou, Valerie Qi, Veit Moeller, Vinnie Monaco, Vishal Kuo, Vlad Fomenko, Wayne Chang, Weiyi Zheng, Wenda Zhou, Wesam Manassra, Will Sheu, Wojciech Zaremba, Yash Patil, Yilei Qian, Yongjik Kim, Youlong Cheng, Yu Zhang, Yuchen He, Yuchen Zhang, Yujia Jin, Yunxing Dai, and Yury Malkov. Gpt-4o system card, 2024. URL https://arxiv.org/abs/2410.21276.

Cheng Qian, Emre Can Acikgoz, Qi He, Hongru Wang, Xiusi Chen, Dilek Hakkani-Tür, Gokhan Tur, and Heng Ji. Toolrl: Reward is all tool learning needs, 2025. URL https://arxiv.org/abs/2504.13958.

Runqi Qiao, Qiuna Tan, Peiqing Yang, Yanzi Wang, Xiaowan Wang, Enhui Wan, Sitong Zhou, Guanting Dong, Yuchen Zeng, Yida Xu, Jie Wang, Chong Sun, Chen Li, and Honggang Zhang. We-math 2.0: A versatile mathbook system for incentivizing visual mathematical reasoning, 2025. URL https://arxiv.org/abs/2508.10433.

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

Pooyan Rahmanzadehgervi, Logan Bolton, Mohammad Reza Taesiri, and Anh Totti Nguyen. Vision language models are blind: Failing to translate detailed visual features into words, 2025. URL https://arxiv.org/abs/2407.06581.

Viresh Ranjan, Udbhav Sharma, Thu Nguyen, and Minh Hoai. Learning to count everything, 2021. URL https://arxiv.org/abs/2104.08391.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL https://arxiv.org/abs/1707.06347.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL https://arxiv.org/abs/2402.03300.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, EuroSys '25, pp. 1279–1297. ACM, March 2025. doi: 10.1145/3689031.3696075. URL http://dx.doi.org/10.1145/3689031.3696075.

Fangxun Shu, Yue Liao, Le Zhuo, Chenning Xu, Lei Zhang, Guanghao Zhang, Haonan Shi, Long Chen, Tao Zhong, Wanggui He, Siming Fu, Haoyuan Li, Bolin Li, Zhelun Yu, Si Liu, Hongsheng Li, and Hao Jiang. Llava-mod: Making llava tiny via moe knowledge distillation, 2024. URL https://arxiv.org/abs/2408.15881.

Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, Akhila Ananthram, Akshay Nathan, Alan Luo, Alec Helyar, Aleksander Madry, Aleksandr Efremov, Aleksandra Spyra, Alex Baker-Whitcomb, Alex Beutel, Alex Karpenko, Alex Makelov, Alex Neitz, Alex Wei, Alexandra Barr, Alexandre Kirchmeyer, Alexey Ivanov, Alexi Christakis, Alistair Gillespie, Allison Tam, Ally Bennett, Alvin Wan, Alyssa Huang, Amy McDonald Sandjideh, Amy Yang, Ananya Kumar, Andre Saraiva, Andrea Vallone, Andrei Gheorghe, Andres Garcia Garcia, Andrew Braunstein, Andrew Liu, Andrew Schmidt, Andrey Mereskin, Andrey Mishchenko, Andy Applebaum, Andy Rogerson, Ann Rajan, Annie Wei, Anoop Kotha, Anubha Srivastava, Anushree Agrawal, Arun Vijayvergiya, Ashley Tyra, Ashvin Nair, Avi Nayak, Ben Eggers, Bessie Ji, Beth Hoover, Bill Chen, Blair Chen, Boaz Barak, Borys Minaiev, Botao Hao, Bowen Baker, Brad Lightcap, Brandon McKinzie, Brandon Wang, Brendan Quinn, Brian Fioca, Brian Hsu, Brian Yang, Brian Yu, Brian Zhang, Brittany Brenner, Callie Riggins Zetino, Cameron Raymond, Camillo Lugaresi, Carolina Paz, Cary Hudson, Cedric Whitney, Chak Li, Charles Chen, Charlotte Cole, Chelsea Voss, Chen Ding, Chen Shen, Chengdu Huang, Chris Colby, Chris Hallacy, Chris Koch, Chris Lu, Christina Kaplan, Christina Kim, CJ Minott-Henriques, Cliff Frey, Cody Yu, Coley Czarnecki, Colin Reid, Colin Wei, Cory Decareaux, Cristina Scheau, Cyril Zhang, Cyrus Forbes, Da Tang, Dakota Goldberg, Dan Roberts, Dana Palmie, Daniel Kappler, Daniel Levine, Daniel Wright, Dave Leo, David Lin, David Robinson, Declan Grabb, Derek Chen, Derek Lim, Derek Salama, Dibya Bhattacharjee, Dimitris Tsipras, Dinghua Li, Dingli Yu, DJ Strouse, Drew Williams, Dylan Hunn, Ed Bayes, Edwin Arbus, Ekin Akyurek, Elaine Ya Le, Elana Widmann, Eli Yani, Elizabeth Proehl, Enis Sert, Enoch Cheung, Eri Schwartz, Eric Han, Eric Jiang, Eric Mitchell, Eric Sigler, Eric Wallace, Erik Ritter, Erin Kavanaugh, Evan Mays, Evgenii Nikishin, Fangyuan Li, Felipe Petroski Such, Filipe de Avila Belbute Peres, Filippo Raso, Florent Bekerman, Foivos Tsimpourlas, Fotis Chantzis, Francis Song, Francis Zhang, Gaby Raila, Garrett McGrath, Gary Briggs, Gary Yang, Giambattista Parascandolo, Gildas Chabot, Grace Kim, Grace Zhao, Gregory Valiant, Guillaume Leclerc, Hadi Salman, Hanson Wang, Hao Sheng, Haoming Jiang, Haoyu Wang, Haozhun Jin, Harshit Sikchi, Heather Schmidt, Henry Aspegren, Honglin Chen, Huida Qiu, Hunter Lightman, Ian Covert, Ian Kivlichan, Ian Silber, Ian Sohl, Ibrahim Hammoud, Ignasi Clavera, Ikai Lan, Ilge Akkaya, Ilya Kostrikov, Irina Kofman, Isak Etinger, Ishaan Singal, Jackie Hehir, Jacob Huh, Jacqueline Pan, Jake Wilczynski, Jakub Pachocki, James Lee, James Quinn, Jamie Kiros, Janvi Kalra, Jasmyn Samaroo, Jason Wang, Jason Wolfe, Jay Chen, Jay Wang, Jean Harb, Jeffrey Han, Jeffrey Wang, Jennifer Zhao, Jeremy Chen, Jerene Yang, Jerry Tworek, Jesse Chand, Jessica Landon, Jessica Liang, Ji Lin, Jiancheng Liu, Jianfeng Wang, Jie Tang, Jihan Yin, Joanne Jang, Joel Morris, Joey Flynn, Johannes Ferstad, Johannes Heidecke, John Fishbein, John Hallman, Jonah Grant, Jonathan Chien, Jonathan Gordon, Jongsoo Park, Jordan Liss, Jos Kraaijeveld, Joseph Guay, Joseph Mo, Josh Lawson, Josh McGrath, Joshua Vendrow, Joy Jiao, Julian Lee, Julie Steele, Julie Wang, Junhua Mao, Kai Chen, Kai Hayashi, Kai Xiao, Kamyar Salahi, Kan Wu, Karan Sekhri, Karan Sharma, Karan Singhal, Karen Li, Kenny Nguyen, Keren Gu-Lemberg, Kevin King, Kevin Liu, Kevin Stone, Kevin Yu, Kristen Ying, Kristian Georgiev, Kristie Lim, Kushal Tirumala, Kyle Miller, Lama Ahmad, Larry Lv, Laura Clare, Laurance Fauconnet, Lauren Itow, Lauren Yang, Laurentia Romaniuk, Leah Anise, Lee Byron, Leher Pathak, Leon Maksin, Leyan Lo, Leyton Ho, Li Jing, Liang Wu, Liang Xiong, Lien Mamitsuka, Lin Yang, Lindsay McCallum, Lindsey Held, Liz Bourgeois, Logan Engstrom, Lorenz Kuhn, Louis Feuvrier, Lu Zhang, Lucas Switzer, Lukas Kondraciuk, Lukasz Kaiser, Manas Joglekar, Mandeep Singh, Mandip Shah, Manuka Stratta, Marcus Williams, Mark Chen, Mark Sun, Marselus Cayton, Martin Li, Marvin Zhang, Marwan Aljubeh, Matt Nichols, Matthew Haines, Max Schwarzer, Mayank Gupta, Meghan Shah, Melody Y. Guan, Melody Huang, Meng Dong, Mengqing Wang, Mia Glaese, Micah Carroll, Michael Lampe, Michael Malek, Michael Sharman, Michael Zhang, Michele Wang, Michelle Pokrass, Mihai Florian, Mikhail Pavlov, Miles Wang, Ming Chen, Mingxuan Wang, Minnia Feng,

Mo Bavarian, Molly Lin, Moose Abdool, Mostafa Rohaninejad, Nacho Soto, Natalie Staudacher, Natan LaFontaine, Nathan Marwell, Nelson Liu, Nick Preston, Nick Turley, Nicklas Ansman, Nicole Blades, Nikil Pancha, Nikita Mikhaylin, Niko Felix, Nikunj Handa, Nishant Rai, Nitish Keskar, Noam Brown, Ofir Nachum, Oleg Boiko, Oleg Murk, Olivia Watkins, Oona Gleeson, Pamela Mishkin, Patryk Lesiewicz, Paul Baltescu, Pavel Belov, Peter Zhokhov, Philip Pronin, Phillip Guo, Phoebe Thacker, Qi Liu, Qiming Yuan, Qinghua Liu, Rachel Dias, Rachel Puckett, Rahul Arora, Ravi Teja Mullapudi, Raz Gaon, Reah Miyara, Rennie Song, Rishabh Aggarwal, RJ Marsan, Robel Yemiru, Robert Xiong, Rohan Kshirsagar, Rohan Nuttall, Roman Tsiupa, Ronen Eldan, Rose Wang, Roshan James, Roy Ziv, Rui Shu, Ruslan Nigmatullin, Saachi Jain, Saam Talaie, Sam Altman, Sam Arnesen, Sam Toizer, Sam Toyer, Samuel Miserendino, Sandhini Agarwal, Sarah Yoo, Savannah Heon, Scott Ethersmith, Sean Grove, Sean Taylor, Sebastien Bubeck, Sever Banesiu, Shaokyi Amdo, Shengjia Zhao, Sherwin Wu, Shibani Santurkar, Shiyu Zhao, Shraman Ray Chaudhuri, Shreyas Krishnaswamy, Shuaiqi, Xia, Shuyang Cheng, Shyamal Anadkat, Simón Posada Fishman, Simon Tobin, Siyuan Fu, Somay Jain, Song Mei, Sonya Egoian, Spencer Kim, Spug Golden, SQ Mah, Steph Lin, Stephen Imm, Steve Sharpe, Steve Yadlowsky, Sulman Choudhry, Sungwon Eum, Suvansh Sanjeev, Tabarak Khan, Tal Stramer, Tao Wang, Tao Xin, Tarun Gogineni, Taya Christianson, Ted Sanders, Tejal Patwardhan, Thomas Degry, Thomas Shadwell, Tianfu Fu, Tianshi Gao, Timur Garipov, Tina Sriskandarajah, Toki Sherbakov, Tomek Korbak, Tomer Kaftan, Tomo Hiratsuka, Tongzhou Wang, Tony Song, Tony Zhao, Troy Peterson, Val Kharitonov, Victoria Chernova, Vineet Kosaraju, Vishal Kuo, Vitchyr Pong, Vivek Verma, Vlad Petrov, Wanning Jiang, Weixing Zhang, Wenda Zhou, Wenlei Xie, Wenting Zhan, Wes McCabe, Will DePue, Will Ellsworth, Wulfie Bain, Wyatt Thompson, Xiangning Chen, Xiangyu Qi, Xin Xiang, Xinwei Shi, Yann Dubois, Yaodong Yu, Yara Khakbaz, Yifan Wu, Yilei Qian, Yin Tat Lee, Yinbo Chen, Yizhen Zhang, Yizhong Xiong, Yonglong Tian, Young Cha, Yu Bai, Yu Yang, Yuan Yuan, Yuanzhi Li, Yufeng Zhang, Yuguang Yang, Yujia Jin, Yun Jiang, Yunyun Wang, Yushi Wang, Yutian Liu, Zach Stubenvoll, Zehao Dou, Zheng Wu, and Zhigang Wang. Openai gpt-5 system card, 2026. URL https://arxiv.org/abs/2601.03267.

Amanpreet Singh, Vivek Natarajan, Meet Shah, Yu Jiang, Xinlei Chen, Dhruv Batra, Devi Parikh, and Marcus Rohrbach. Towards vqa models that can read, 2019. URL https://arxiv.org/abs/1904.08920.

Haoxiang Sun, Lizhen Xu, Bing Zhao, Wotao Yin, Wei Wang, Boyu Yang, Rui Wang, and Hu Wei. Deepvision-103k: A visually diverse, broad-coverage, and verifiable mathematical dataset for multimodal reasoning, 2026. URL https://arxiv.org/abs/2602.16742.

Yunbo Tang, Chengyi Yang, Shiyu Liu, Zhishang Xiang, Zerui Chen, Qinggang Zhang, and Jinsong Su. Saas: Self-aware reinforcement learning for over-search mitigation in agentic search, 2026. URL https://arxiv.org/abs/2605.29796.

Shengbang Tong, Ellis Brown, Penghao Wu, Sanghyun Woo, Manoj Middepogu, Sai Charitha Akula, Jihan Yang, Shusheng Yang, Adithya Iyer, Xichen Pan, Ziteng Wang, Rob Fergus, Yann LeCun, and Saining Xie. Cambrian-1: A fully open, vision-centric exploration of multimodal llms, 2024. URL https://arxiv.org/abs/2406.16860.

Changpeng Wang, Haozhe Wang, Xi Chen, Junhan Liu, Taofeng Xue, Chong Peng, Donglian Qi, Fangzhen Lin, and Yunfeng Yan. From illusion to intention: Visual rationale learning for vision-language reasoning, 2025a. URL https://arxiv.org/abs/2511.23031.

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.

Haozhe Wang, Chao Qu, Zuming Huang, Wei Chu, Fangzhen Lin, and Wenhu Chen. Vl-rethinker: Incentivizing self-reflection of vision-language models with reinforcement learning, 2025b. URL https://arxiv.org/abs/2504.08837.

Haozhe Wang, Alex Su, Weiming Ren, Fangzhen Lin, and Wenhu Chen. Pixel reasoner: Incentivizing pixel-space reasoning with curiosity-driven reinforcement learning, 2025c. URL https://arxiv.org/abs/2505.15966.

Hongru Wang, Cheng Qian, Wanjun Zhong, Xiusi Chen, Jiahao Qiu, Shijue Huang, Bowen Jin, Mengdi Wang, Kam-Fai Wong, and Heng Ji. Acting less is reasoning more! teaching model to act efficiently, 2025d. URL https://arxiv.org/abs/2504.14870.

Ke Wang, Junting Pan, Weikang Shi, Zimu Lu, Mingjie Zhan, and Hongsheng Li. Measuring multimodal mathematical reasoning with math-vision dataset, 2024a. URL https://arxiv.org/abs/2402.14804.

Weiyun Wang, Zhangwei Gao, Lixin Gu, Hengjun Pu, Long Cui, Xingguang Wei, Zhaoyang Liu, Linglin Jing, Shenglong Ye, Jie Shao, Zhaokai Wang, Zhe Chen, Hongjie Zhang, Ganlin Yang, Haomin Wang, Qi Wei, Jinhui Yin, Wenhao Li, Erfei Cui, Guanzhou Chen, Zichen Ding, Changyao Tian, Zhenyu Wu, Jingjing Xie, Zehao Li, Bowen Yang, Yuchen Duan, Xuehui Wang, Zhi Hou, Haoran Hao, Tianyi Zhang, Songze Li, Xiangyu Zhao, Haodong Duan, Nianchen Deng, Bin Fu, Yinan He, Yi Wang, Conghui He, Botian Shi, Junjun He, Yingtong Xiong, Han Lv, Lijun Wu, Wenqi Shao, Kaipeng Zhang, Huipeng Deng, Biqing Qi, Jiaye Ge, Qipeng Guo, Wenwei Zhang, Songyang Zhang, Maosong Cao, Junyao Lin, Kexian Tang, Jianfei Gao, Haian Huang, Yuzhe Gu, Chengqi Lyu, Huanze Tang, Rui Wang, Haijun Lv, Wanli Ouyang, Limin Wang, Min Dou, Xizhou Zhu, Tong Lu, Dahua Lin, Jifeng Dai, Weijie Su, Bowen Zhou, Kai Chen, Yu Qiao, Wenhai Wang, and Gen Luo. Internvl3.5: Advancing open-source multimodal models in versatility, reasoning, and efficiency, 2025e. 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, 2024b. URL https://arxiv.org/abs/2408.15556.

Xiyao Wang, Zhengyuan Yang, Chao Feng, Hongjin Lu, Linjie Li, Chung-Ching Lin, Kevin Lin, Furong Huang, and Lijuan Wang. Sota with less: Mcts-guided sample selection for data-efficient visual reasoning self-improvement, 2025f. URL https://arxiv.org/abs/2504.07934.

Zhiheng Wang, Bo Peng, Lai Wei, and Chaochao Lu. The illusion of visual tool-use: A causal audit of thinking with images, 2026b. URL https://arxiv.org/abs/2608.06270.

Zihan Wang, Chi Gui, Xing Jin, Qineng Wang, Licheng Liu, Kangrui Wang, Shiqi Chen, Linjie Li, Zhengyuan Yang, Pingyue Zhang, Yiping Lu, Jiajun Wu, Li Fei-Fei, Lijuan Wang, Yejin Choi, and Manling Li. Ragen-2: Reasoning collapse in agentic rl, 2026c. URL https://arxiv.org/abs/2604.06268.

Junjie Wen, Yichen Zhu, Jinming Li, Minjie Zhu, Kun Wu, Zhiyuan Xu, Ning Liu, Ran Cheng, Chaomin Shen, Yaxin Peng, Feifei Feng, and Jian Tang. Tinyvla: Towards fast, data-efficient vision-language-action models for robotic manipulation, 2025. URL https://arxiv.org/abs/2409.12514.

Penghao Wu and Saining Xie. V*: Guided visual search as a core mechanism in multimodal llms, 2023. URL https://arxiv.org/abs/2312.14135.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models, 2024a. URL https://arxiv.org/abs/2211.10438.

Yijia Xiao, Edward Sun, Tianyu Liu, and Wei Wang. Logicvista: Multimodal llm logical reasoning benchmark in visual contexts, 2024b. URL https://arxiv.org/abs/2407.04973.

Shilin Yan, Jintao Tong, Hongwei Xue, Xiaojun Tang, Yangyang Wang, Kunyu Shi, Guannan Zhang, Ruixuan Li, and Yixiong Zou. Act wisely: Cultivating meta-cognitive tool use in agentic multimodal models, 2026. URL https://arxiv.org/abs/2604.08545.

Jingkang Yang, Shuai Liu, Hongming Guo, Yuhao Dong, Xiamengwei Zhang, Sicheng Zhang, Pengyun Wang, Zitang Zhou, Binzhu Xie, Ziyue Wang, Bei Ouyang, Zhengyu Lin, Marco Cominelli, Zhongang Cai, Yuanhan Zhang, Peiyuan Zhang, Fangzhou Hong, Joerg Widmer, Francesco Gringoli, Lei Yang, Bo Li, and Ziwei Liu. Egolife: Towards egocentric life assistant, 2026a. URL https://arxiv.org/abs/2503.03803.

Senqiao Yang, Yukang Chen, Zhuotao Tian, Chengyao Wang, Jingyao Li, Bei Yu, and Jiaya Jia. Visionzip: Longer is better but not necessary in vision language models, 2026b. URL https://arxiv.org/abs/2412.04467.

Yifan Yang, Zhen Zhang, Jiayi Tian, Liyan Tan, and Zheng Zhang. Iapo: Input attribution-aware policy optimization for tool use in small multimodal agents, 2026c. URL https://arxiv.org/abs/2606.11652.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models, 2023. URL https://arxiv.org/abs/2210.03629.

Hee Suk Yoon, Eunseop Yoon, Jaehyun Jang, SooHwan Eom, Ji Woo Hong, Mark Hasegawa-Johnson, Qi Dai, Chong Luo, and Chang D. Yoo. Decomposed on-policy distillation for vision-language reasoning: Steering gradients for visual grounding, 2026. URL https://arxiv.org/abs/2606.00564.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. Dapo: An open-source llm reinforcement learning system at scale, 2025. URL https://arxiv.org/abs/2503.14476.

Jingyi Zhang, Jiaxing Huang, Huanjin Yao, Shunyu Liu, Xikun Zhang, Shijian Lu, and Dacheng Tao. R1-vl: Learning to reason with multimodal large language models via step-wise group relative policy optimization, 2025a. URL https://arxiv.org/abs/2503.12937.

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

Liang Zhang, Anwen Hu, Haiyang Xu, Ming Yan, Yichen Xu, Qin Jin, Ji Zhang, and Fei Huang. Tinychart: Efficient chart understanding with visual token merging and program-of-thoughts learning, 2024a. URL https://arxiv.org/abs/2404.16635.

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?, 2024b. URL https://arxiv.org/abs/2403.14624.

Xintong Zhang, Zhi Gao, Bofei Zhang, Pengxiang Li, Xiaowen Zhang, Yang Liu, Tao Yuan, Yuwei Wu, Yunde Jia, Song-Chun Zhu, and Qing Li. Adaptive chain-of-focus reasoning via dynamic visual search and zooming for efficient vlms, 2025b. URL https://arxiv.org/abs/2505.15436.

Yi-Fan 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, 2025c. URL https://arxiv.org/abs/2508.11630.

Shitian Zhao, Shaoheng Lin, Ming Li, Haoquan Zhang, Wenshuo Peng, Kaipeng Zhang, and Chen Wei. Pyvision-rl: Forging open agentic vision models via rl, 2026. URL https://arxiv.org/abs/2602.20739.

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.

Chengke Zou, Xingang Guo, Rui Yang, Junyu Zhang, Bin Hu, and Huan Zhang. Dynamath: A dynamic visual benchmark for evaluating mathematical reasoning robustness of vision language models, 2025. URL https://arxiv.org/abs/2411.00836.

A Analysis Details

A.1 Setup of the Tool-Use Probe in Figure 2

For the controlled visual difficulty in Figure 2(bottom), we use the single-image questions of HR-Bench (Wang et al., 2024b) in which SAM3 (Carion et al., 2026) finds the target object. For each level, we crop a square window around the target so that the target covers 0.1% (hard), 0.5% (medium), or 3% (easy) of the window, and resize the window to 1024×1024 pixels. Figure 15 shows one question at the three levels. Table 8 shows the no-tool accuracy and the tool-call rate at each level.



Figure 15: An HR-Bench question at the (Left) hard, (Middle) medium, and (Right) easy levels. The question is "What is the speed limit indicated by the sign?" and the answer is 30 km/h at every level.

Table 8: HR-Bench crop ladder of Figure 2 and Figure 11. (Top) No-tool accuracy and tool-call rate (%) of the base models. (Bottom) Pass@1 and tool-call rate (%) of the 2B models trained with Tool-RLVR.

	No-tool accuracy			Tool-call rate		
Model	Hard	Medium	Easy	Hard	Medium	Easy
Qwen3.5-2B	72.7	82.0	87.2	98.3	98.3	97.1
Qwen3.5-4B	76.8	82.6	86.3	95.7	93.9	86.3
Qwen3.5-27B	81.7	86.2	88.6	87.1	77.2	65.7
Qwen3.5-397B	82.9	85.7	89.3	86.7	74.0	56.7

	Pass@1			Tool-call rate		
Method	Hard	Medium	Easy	Hard	Medium	Easy
GRPO	80.1	86.7	88.5	9.6	6.4	3.1
ARPO	70.7	79.5	85.1	0.4	0.4	0.0
AXPO	83.9	86.7	89.2	99.4	98.8	98.0
ANPO	85.5	90.4	92.1	82.8	76.4	61.0

A.2 Experiment Details of Section 3

Tool Collapse under Tool-RLVR. Figure 4 is measured on the training rollouts of the GRPO and ARPO models reported in Table 3. The questions where visual tools are expected to help are the perception questions of the Tool-RLVR data in Appendix B.1 that come from DeepEyes-47k (Zheng et al., 2026), PixelReasoner-SFT (Wang et al., 2025c), and Mini-o3-Coldstart (Lai et al., 2025), which are mainly used as training data for Tool-RLVR.

A Failed Answer Does Not Imply a Failed Decision. Figure 5(left) uses the failed tool-using trajectories from the first 30 training steps of GRPO and ARPO, pooled together. We use the first 30 steps because the tool-call rate is still high in this range. We use GPT-5.4-mini (Singh et al., 2026) as the judge and judge each stage in a separate request, with the question, the image, and only the part of the trajectory that the stage needs, and the reference answer only for the answer stage. The judge returns one of `valid`, `invalid`, and `unclear`, and we count `valid` as proper. When we ran the judge, no stage was labeled `unclear`. The prompts are given below.

System prompt

```
You are a blinded annotator for a small multimodal tool agent.  The
research hypothesis may be false.  Do not favor it.  Judge what THIS
small agent actually got out of each stage, not what you or a stronger
model could have done.  Use unclear when evidence is insufficient.
Return JSON only.  You are judging the {STAGE} segment only.
Information from other segments must not alter this label.
```

Decision

```
Judge only the INFORMATION YIELD of the tool call(s).  You are shown
the question, the original image, each call and exactly what it
returned.
valid - at least one observation surfaced concrete evidence bearing on
the question that was not already readable in the original image (a
magnified detail, a count, a location, a computed value).  It counts
even if the agent later misused it.
invalid - the observations returned nothing the original image did not
already show, or nothing relevant to the question.
Do NOT judge whether the call was well-formed, and do NOT judge the
later reasoning or the final answer.  You are not shown the final
answer.
```

Execution

```
Judge only whether the call(s) were MECHANICALLY SOUND, as a code
reviewer would.  Ignore the image and ignore whether the information
was useful.
valid - every call is well-formed for its tool and does what a call of
that tool should do:  python that actually computes and prints
something, a crop that targets a real sub-region, a concept name a
segmenter can act on; and the returned observation is a real result
rather than an error or an empty payload.
invalid - a call is malformed or degenerate:  python that is only
comments, or that fabricates its input data instead of using the tool
output, or that prints nothing; a crop covering the whole image or an
empty box; a call whose observation is an error or empty.
Do NOT penalize a well-formed call merely because a different tool or
region would have been a better choice.
```

Answer

```
Judge only the post-observation REASONING. Treat the calls and
observations as fixed.  Decide whether the assistant interpreted them
soundly and derived an answer that follows from them together with the
reference answer.
valid - the stated reasoning follows from the observations and
supports the answer.
invalid - the reasoning contradicts the observations, ignores them, or
makes a leap the evidence does not support.
Do not retroactively relabel the tool choice or the call arguments.
```

Figure 5(right) replaces one stage of the small model with Qwen3.5-27B while the SFT 2B model writes the rest, with the same prompt and tools and at most five turns, and the tools run on the new calls. To replace the decision, the 27B model writes the first turn up to the opening <*tool_call*> tag, and the 2B model writes everything after it. To replace the execution, the 2B model stops at each <*tool_call*> tag, the 27B model writes the call up to <*/tool_call*>, and the 2B model continues, for every call in the trajectory. To replace the answer stage, we replay the 2B trajectory up to the first tool result, checking that the re-executed tool returns the same result, and the 27B model writes all later turns.

Penalizing Proper Decisions Suppresses Tool Use. Figure 6 freezes the failed trajectories of GRPO and updates the model with GRPO for 10 steps, changing only which tokens receive the penalty. The tool-call rate is measured on the fifteen evaluation benchmarks of Table 3. Full, Decision,

Execution, and Answer penalize the same failed trajectories and differ only in the penalized stage. Proper decision and Improper decision penalize only the decision stage, and only in the trajectories whose decision is judged proper or improper. They therefore also differ in the number of penalized questions. These differences change the total penalty, which is the sum of $|A_i|$ over the penalized tokens. To remove this effect, we also train each setting with its total penalty matched (Matched), by multiplying the advantage of the penalized tokens by a constant (Scale). For Full, Decision, Execution, and Answer, we match the total penalty of Full, which removes the difference in the number of tokens across stages. We match Proper decision and Improper decision, separately from the other settings, to the total penalty of Proper decision, because matching them to Full would multiply their advantage by a very large constant. This also removes the difference in the number of questions between the two. Table 9 reports the tool-call rate with and without this matching.

Table 9: Tool-call rate (%) after controlled training, without (Original) and with (Matched) the total penalty matched, and the scale applied to the advantage in the matched setting.

Penalized tokens	Original	Matched	Scale
SFT (no training)	39.7	39.7	–
Full	12.2	12.2	1.0
Decision	8.4	6.6	3.4
Execution	40.7	39.0	7.9
Answer	42.2	41.7	1.7
Proper decision	6.4	6.4	1.0
Improper decision	38.8	34.9	3.9

Tool-Call Rate Moves Together. Figure 7 is measured directly on the training rollouts of the models reported in Table 3. We use the perception questions of the Tool-RLVR data in Appendix B.1 as TN questions and the reasoning questions as TU questions.

TU Questions Dominate the Decision Signal. Figure 8 is computed from the training rollouts of the same models.

The Dominant Signal Pushes Tool Use in One Direction. Figure 9 is computed from the training rollouts of the same models.

A.3 Decision Signal under SRA

We compare the learning signal for the tool-call decision from TN and TU questions under SRA only and under ANPO in Figure 16.

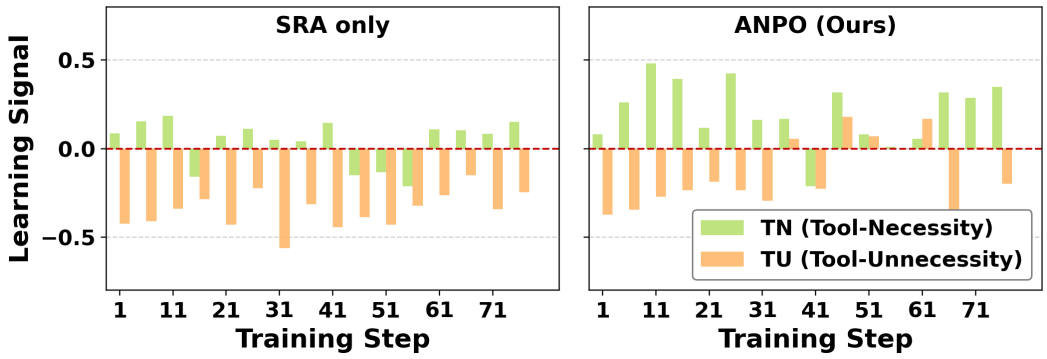


Figure 16: Learning signal for the tool decision from TN (green) and TU (orange) questions during training under (Left) SRA only and (Right) ANPO.

B TRAINING DETAILS

This section describes the data, the benchmarks, the objective, the implementation of ANPO, the hyperparameters, and the prompts and tools. All methods share the same data, prompts, tools, reward, and evaluation protocol, and they differ only in how trajectories are sampled and how the advantage is assigned.

B.1 TRAINING DATA CONSTRUCTION

SFT Data Construction. We build the SFT set in four steps: (1) candidate pool → (2) solvability filter → (3) teacher generation → (4) quality filter and data distribution. The final set contains 19,613 trajectories from 26 source datasets.

(1) Candidate pool. Since the base model already performs well on recognition, we collect diverse data mainly for reasoning and perception, about 142K questions from the following datasets.

- **Recognition**: OK-VQA (Marino et al., 2019), ChartQA (Masry et al., 2022), TinyChart (Zhang et al., 2024a).
- **Reasoning**: ThinkLite-VL-70k (Wang et al., 2025f), MMFineReason-123K (Lin et al., 2026a), MMK12 (Meng et al., 2025), ViRL39K (Wang et al., 2025b), We-Math2.0 (Qiao et al., 2025), Geometry3K (Lu et al., 2021), TabMWP (Lu et al., 2023), PyVision-RL (Zhao et al., 2026).
- **Perception**: DeepEyes-47k (Zheng et al., 2026), PixelReasoner-SFT (Wang et al., 2025c), Mini-o3-Coldstart (Lai et al., 2025), CoF-SFT-5.4k (Zhang et al., 2025b), TallyQA (train) (Acharya et al., 2018), PixMo-Count (Deitke et al., 2024), FSC-147 (Ranjan et al., 2021), ODinW-13 (Li et al., 2022), TVG (Zhang et al., 2026), STVQA-7K (Batra et al., 2026), SURDS (Guo et al., 2025), SpatialMQA (Liu et al., 2025a), EmbSpatial-SFT (Du et al., 2024), 3DSRBench (Ma et al., 2025a), CV-Bench (Tong et al., 2024).

We remove questions that overlap with the evaluation benchmarks. We then sample about 66K candidates, with no single source above 40% of its domain.

(2) Solvability filter. We remove questions that the base model to be trained already solves without tools. A tool demonstration on these questions could teach a call that the question does not need. About 26K candidates remain.

(3) Teacher generation. Qwen3.5-27B generates the trajectories for SFT training. The source datasets differ in question format, answer format, and tool description, so we unify all of them into one format with the same tools, tool schemas, and system prompt that Tool-RLVR uses. The SFT model and the Tool-RLVR policy therefore see identical inputs.

(4) Quality filter and data distribution. We keep only the trajectories that end with a correct answer and remove degenerate and duplicate ones. The final SFT set uses 19,613 trajectories.

Tool-RLVR Data Construction. We build the Tool-RLVR pool in three steps: (1) candidate pool → (2) quality filter → (3) source cap. The final pool contains 17,431 questions from 27 source datasets.

(1) Candidate pool. We collect the questions that the SFT model fails to solve and add harder questions for further exploration. They come from the following datasets.

- **Recognition**: OK-VQA (Marino et al., 2019), ChartQA (Masry et al., 2022), TinyChart (Zhang et al., 2024a).
- **Reasoning**: MMFineReason-123K (Lin et al., 2026a), ThinkLite-VL-70k (Wang et al., 2025f), ViRL39K (Wang et al., 2025b), MMK12 (Meng et al., 2025), We-Math2.0 (Qiao et al., 2025), Geometry3K (Lu et al., 2021), TabMWP (Lu et al., 2023), PyVision-RL (Zhao et al., 2026).
- **Perception**: DeepEyes-47k (Zheng et al., 2026), PixelReasoner-SFT (Wang et al., 2025c), Mini-o3-Coldstart (Lai et al., 2025), CoF-SFT-5.4k (Zhang et al., 2025b), TallyQA (train) (Acharya et al., 2018), PixMo-Count (Deitke et al., 2024), FSC-147 (Ranjan et al., 2021), ODinW-13 (Li et al., 2022), TVG (Zhang et al., 2026), STVQA-7K (Batra et al., 2026), SURDS (Guo et al., 2025), SpatialMQA (Liu et al., 2025a), EmbSpatial-SFT (Du et al., 2024), 3DSRBench (Ma et al., 2025a), CV-Bench (Tong et al., 2024).

- **Added for Tool-RLVR**: DeepVision-103K (math-77k, visual_logic-26k) (Sun et al., 2026).

(2) Quality filter. We remove questions that the rule-based scorer cannot grade reliably, such as questions with open-ended long answers and multiple-choice questions whose gold option is missing, which leaves 21,211 candidates.

(3) Source cap. We cap every source at 20% of the pool so that no single source dominates, sampling within each source by offline difficulty to keep the difficulty spread. The final Tool-RLVR pool uses 17,431 questions.

B.2 Benchmarks

We evaluate on the following fifteen benchmarks.

- **Recognition**: DocVQA (Mathew et al., 2021), ChartQA (Masry et al., 2022), TextVQA (Singh et al., 2019), GQA (Hudson & Manning, 2019).
- **Reasoning**: MathVerse (Zhang et al., 2024b), MathVista (Lu et al., 2024), MathVision (Wang et al., 2024a), DynaMath (Zou et al., 2025), LogicVista (Xiao et al., 2024b).
- **Perception**: $V^\star$ (Wu & Xie, 2023), HR-Bench (Wang et al., 2024b), TallyQA (Acharya et al., 2018), $\text{VisualProbe}_{\text{H}}$ (Lai et al., 2025), VLMsAreBlind (Rahmanzadehgervi et al., 2025), BabyVision (Chen et al., 2026b).

We sample four rollouts per question, score each rollout with the same rule-based scorer as the training reward, and report Pass@1 as the mean accuracy over the four rollouts. Pass@4 counts a question as solved if any of the four rollouts is correct. The scorer reads the last <answer> tag after removing tool calls and tool results from the text, so a tool output cannot supply the answer.

B.3 Policy Gradient Objective

All methods maximize the PPO-clip surrogate (Schulman et al., 2017) without a KL penalty or an entropy bonus. Letting $\rho_{i,t} = \pi_\theta(o_{i,t} \mid q, o_{i,<t}) / \pi_{\theta_{\text{old}}}(o_{i,t} \mid q, o_{i,<t})$ denote the per-token importance ratio against the rollout policy $\pi_{\theta_{\text{old}}}$ and $\hat{A}_{i,t}$ the advantage of token t in trajectory o_i,

$$\mathcal{J}(\theta) = \mathbb{E}\Big[\frac{1}{\sum_{i\in\mathcal{M}} |o_i|} \sum_{i\in\mathcal{M}} \sum_{t=1}^{|o_i|} \min\big(\rho_{i,t}\hat{A}_{i,t},\ \text{clip}(\rho_{i,t}, 1-\epsilon_{\text{low}}, 1+\epsilon_{\text{high}})\,\hat{A}_{i,t}\big)\Big], \tag{4}$$

where $\mathcal{M}$ is the set of trajectories in a mini-batch, $|o_i|$ counts the tokens generated by the model in o_i, and tokens returned by tools are excluded from the sum. GRPO and the baselines assign the trajectory-level advantage to every token, $\hat{A}_{i,t} = A_i$ of Section 2, and AKBE adds its own auxiliary term to this objective. ANPO instead uses the stage advantage of Eq. 3. We use the asymmetric clipping range of DAPO (Yu et al., 2025), $\epsilon_{\text{low}} = 0.2$ and $\epsilon_{\text{high}} = 0.28$. Each training step splits its 128 questions into four mini-batches, so the policy is updated four times per step.

B.4 Implementation Details of ANPO

Two arms. For each question, ANPO and all Tool-RLVR methods use 8 rollouts. ANPO samples $N_T = 4$ tool-on trajectories and $N_R = 4$ tool-off trajectories, while the other methods, which do not split the rollouts, generate 8 rollouts naturally. AKBE (Chen et al., 2026a) generates 4 additional rollouts and AXPO (Kang et al., 2026) also generates additional rollouts, while ARPO (Dong et al., 2025b) splits the same eight rollouts into 4 initial rollouts and 4 branches from tool calls. ANPO generates no additional rollouts either, so its rollout generation costs less than that of AKBE and AXPO. We normalize the decision advantage by the batch-level standard deviation following prior work (Liu et al., 2025b), because the broadcast reward has no within-question variance. The decision advantage is set to zero when the batch has fewer than two tool-using trajectories or when the standard deviation is zero.

Reward and dynamic sampling. The reward is the correctness of the final answer in <answer> tags, and a trajectory that exhausts its generation budget receives zero. No format reward and no length penalty are used. For every method, each step generates 256 questions and keeps the first

128 whose trajectories are neither all correct nor all wrong, following the dynamic sampling of DAPO (Yu et al., 2025), with up to three generation batches per step. For ANPO, this filter looks at the same eight trajectories per question as the other methods, covering both arms. Among the kept questions, ANPO moves to the back those in which one arm is all correct or all wrong, such as a question whose four tool-on trajectories all fail while its tool-off trajectories are mixed. We use Qwen3.5 in its non-thinking mode in all training and evaluation.

B.5 HYPERPARAMETERS

All Tool-RLVR methods start from the same SFT model and share the values below, and the method-specific values follow the official settings of each baseline. We implement all methods on verl (Sheng et al., 2025) with vLLM for generation. All methods are trained on 8 NVIDIA H200 GPUs.

SFT hyperparameters.

- Learning rate, schedule, and warmup: 1×10^{-5}, cosine, 10%
- Epochs and global batch: 3 and 128
- Optimizer, weight decay, and gradient clipping: AdamW (β_1=0.9, β_2=0.95), 0.01, and 1.0

Tool-RLVR hyperparameters. The following values are shared by all methods. We use the AdamW optimizer with a learning rate of 1×10^{-6} and a constant scheduler without warm-up, with a weight decay of 0.01 and gradient clipping at 1.0. Each batch contains 128 questions with 8 rollouts per question. We use dynamic sampling to prevent the sparse reward problem, drawing 256 candidate questions per step with up to three generation batches. Each step is split into mini-batches of 32 questions and runs one epoch. The clipping range is $[0.8,\ 1.28]$, and the KL coefficient is zero. All parameters except the vision encoder are trained.

For rollouts, we sample with temperature 0.7. A trajectory has at most 5 turns and 8,192 generated tokens, excluding tool results. Each trajectory contains at most 6 images, and 10 for ARPO.

Baseline hyperparameters.

- ARPO (Dong et al., 2025b): 4 initial rollouts + 4 branch slots, branch probability 0.5, entropy weight 0.2
- AKBE (Chen et al., 2026a): 8 tool rollouts + 4 tool-free rollouts, imitation weight 0.05
- AXPO (Kang et al., 2026): 8 rollouts + 4 resampled continuations, resampling temperature equal to rollout

B.6 PROMPTS AND TOOL SCHEMAS

SFT, Tool-RLVR, and evaluation use the same two system prompts. The tool-on prompt is used whenever the tools are available, and the tool-off prompt is used for the tool-off trajectories of ANPO and for the tool-free rollouts of AKBE. The schemas of the four tools are passed through the `tools` field of the Qwen3.5 chat template and are identical across SFT, Tool-RLVR, and evaluation. Every question ends with the instruction "Answer the question and put the final answer inside `<answer></answer>`."

Tool-on system prompt

```
You are a careful vision-language problem-solver.  Answer the question
about the image(s).
Before calling any tool or committing to an answer, first reason in
words about what the question needs and whether a tool is required.
Do not act silently.
Use tools deliberately, not by reflex:
- Call a tool ONLY when it provides information that looking and
reasoning alone cannot.
- If you can already answer from what you see, answer directly.  Do
NOT call a tool to confirm something you are already confident about.
```

```
- After each tool result, reflect on whether it gave what you needed.
If it looks wrong, empty, or off-target, do not trust it blindly ---
reconsider or try again.
- Once you have enough information, stop and give your answer.
Put your final answer like <answer>your final answer here</answer>.
```

Tool-off system prompt

```
You are a careful vision-language problem-solver.  Answer the question
about the image(s).
Reason in words step by step about what the image shows and what the
question asks.
Put your final answer like <answer>your final answer here</answer>.
```

Tool schemas. The schemas of Python, Zoom-In, object segmentation, and depth estimation in full are below. All coordinates are normalized to $[0, 1000]$ relative to the original image.

Python (`python`).

```
{
  "type":  "function",
  "function":  {
    "name":  "python",
    "description":  "Run Python code in a sandbox for calculation,
counting, geometry, or pixel-level image work.\n\n1.  Each call runs in a
FRESH process -- variables, imports, and state do NOT persist across
calls.  Re-create what you need in every call.\n2.  The conversation
images are pre-loaded as `_pil_images`, a list of PIL Image objects.
`_pil_images[0]` is the original image; images produced by other tools
(e.g.  an image_zoom_in crop) are appended in the order they were
produced.  `image_1`, `image_2`, ...  and `image` (= _pil_images[0]) are
aliases for the same objects.\n3.  Already imported for you:  numpy as
np, matplotlib.pyplot as plt (Agg backend), PIL.Image, math, json, io,
base64, os, sys -- do not re-import them.\n4.  Available packages:
numpy, scipy, sympy, pandas, matplotlib, PIL/Pillow, cv2, scikit-learn,
networkx, shapely, statsmodels, and the standard library.\n5.  print()
what you want to see; the value of the last expression is also shown
automatically.\n6.  To send an image back to yourself:  call plt.show()
for plots (figures are captured automatically), or show_image(img) for a
PIL image.\n7.  Execution is limited to 15 seconds.  Do not write
infinite loops.  Writing files to disk is not allowed.",
    "parameters":  {
      "type":  "object",
      "properties":  {
        "code":  {
          "type":  "string",
          "description":  "Python code to execute.  Use it for arithmetic,
multi-step calculation, equation solving, geometry from coordinates you
already have (e.g.  boxes returned by segment), or pixel-level work on
_pil_images[0].  When a calculation is error-prone -- trigonometry,
multi-step arithmetic, equation solving, or area/volume formulas --
prefer running it here rather than computing it mentally."
        }
      },
      "required":  [
        "code"
      ]
    }
  }
}
```

Zoom-In (`image_zoom_in`).

```
{
  "type":  "function",
  "function":  {
```

```
    "name": "image_zoom_in",
    "description": "Crop and magnify a rectangular region of the image
for a closer look at small text or fine detail. Returns the cropped
image region.",
    "parameters": {
      "type": "object",
      "properties": {
        "bbox": {
          "type": "array",
          "items": {
            "type": "number"
          },
          "description": "[x1,y1,x2,y2] normalized to 0-1000 (0=left/top
edge, 1000=right/bottom edge)."
        }
      },
      "required": [
        "bbox"
      ]
    }
  }
}
```

Object segmentation (`segment`).

```
{
  "type": "function",
  "function": {
    "name": "segment",
    "description": "Detect, count, and locate every instance of a visual
concept using a concept segmenter (SAM3). Use this to count objects or
to get their locations. Returns the total count and, for each instance,
a bounding box and a confidence score.",
    "parameters": {
      "type": "object",
      "properties": {
        "concept": {
          "type": "string",
          "description": "One or more SHORT noun phrases naming the object
type(s) to find. To count or locate SEVERAL object types at once,
separate them with commas, e.g. 'tree, sign, car' --- every phrase is
processed in this single call (do this instead of calling segment
multiple times, to save turns). Each phrase must be a noun with at most
two modifiers, e.g. 'red car', 'person', 'apple'. Do NOT use clauses,
questions, or spatial relations (not 'the man on the left wearing red')."
        }
      },
      "required": [
        "concept"
      ]
    }
  }
}
```

Depth estimation (`compare_depth`).

```
{
  "type": "function",
  "function": {
    "name": "compare_depth",
    "description": "Compare the relative depth of two points and report
which one is closer to the camera (DepthAnything3).",
    "parameters": {
      "type": "object",
      "properties": {
        "a": {
          "type": "array",
```

```
          "items": {
            "type": "number"
          },
          "description": "[x,y] normalized to 0-1000. First point."
        },
        "b": {
          "type": "array",
          "items": {
            "type": "number"
          },
          "description": "[x,y] normalized to 0-1000. Second point."
        }
      },
      "required": [
        "a",
        "b"
      ]
    }
  }
}
```

B.7 TOOLS

A tool that fails returns an error message instead of a result, which counts as a failed execution for r_i^{exec}. The segmentation and depth models run on a separate GPU server that the rollout workers call over HTTP. Every image returned by a tool is resized to at most $2048 \times 32 \times 32$ pixels, padded to an aspect ratio of at most 180, and enlarged to a shorter side of at least 28 pixels, which keeps it within the input limits of the Qwen3.5 vision encoder.

C ADDITIONAL RESULTS

C.1 PASS@4 RESULTS

We report Pass@4 of ANPO and the baselines in Table 10.

Table 10: Comparison of ANPO with Tool-RLVR methods on 15 benchmarks (Pass@4, %). The best and second-best results within each size are marked in bold and underlined.

	Recognition					Reasoning						Perception							Avg.
Method	Doc VQA	Chart QA	Text VQA	GQA	**Avg.**	Math Verse	Math Vista	Math Vision	Dyna Math	Logic Vista	**Avg.**	V*	HR Bench	Tally QA	Visual Probe$_H$	VLMs AreBlind	Baby Vision	**Avg.**	
Qwen3.5-2B																			
Base	89.2	89.9	88.9	61.5	82.4	64.5	78.1	37.8	72.1	59.2	62.3	88.5	83.8	86.4	36.8	74.3	17.0	64.5	68.5
SFT	93.5	93.8	90.4	65.5	85.8	72.0	83.9	49.1	76.7	70.1	70.3	89.5	87.3	88.0	42.5	79.8	25.5	68.7	73.8
+GRPO	93.0	94.0	90.6	67.7	86.3	80.0	86.8	67.0	78.4	79.9	78.4	90.6	87.6	90.7	40.5	82.5	32.0	70.7	77.4
+ARPO	92.9	93.8	89.2	67.1	85.8	79.2	86.3	**69.4**	79.6	79.7	78.8	89.5	85.1	90.7	37.7	81.3	34.5	69.8	77.1
+AKBE	92.7	93.4	88.0	67.1	85.3	77.7	86.2	65.8	76.7	79.5	77.2	88.4	85.8	90.5	34.9	74.4	32.0	67.7	75.5
+AXPO	93.5	**94.4**	90.3	65.4	85.9	78.9	87.3	66.1	78.8	78.8	78.0	93.2	88.6	91.8	42.5	83.4	32.5	72.0	77.7
+ANPO	**94.3**	94.0	**91.6**	**68.3**	**87.1**	**81.2**	**87.6**	**69.4**	**80.2**	**81.9**	**80.1**	**93.7**	**91.1**	**93.0**	**44.3**	**87.1**	**35.0**	**74.0**	**79.5**
Qwen3.5-4B																			
Base	94.8	92.0	89.6	62.8	84.8	78.1	86.4	65.1	80.8	75.9	77.3	94.8	87.5	88.8	29.3	88.9	33.0	70.4	76.5
SFT	95.0	92.6	90.7	64.1	85.6	80.0	87.1	69.1	79.6	78.8	78.9	95.3	89.1	89.0	37.2	90.2	34.5	72.6	78.2
+GRPO	95.2	93.3	92.2	67.6	87.1	81.5	86.9	76.6	80.6	82.5	81.6	95.3	90.5	90.9	44.7	90.1	38.5	75.0	80.4
+ARPO	95.3	93.5	**92.9**	67.8	**87.4**	82.1	87.1	77.0	81.0	83.7	82.2	95.3	91.4	91.9	42.5	93.2	40.0	75.7	81.0
+AKBE	95.5	93.0	90.5	66.0	86.3	82.2	86.7	**78.0**	81.2	83.9	82.4	95.8	90.6	91.2	41.5	91.4	37.0	74.6	80.3
+AXPO	94.6	92.9	91.0	67.7	86.6	83.5	87.2	**78.0**	81.0	84.5	82.8	96.9	**92.9**	91.3	45.3	94.8	41.5	77.1	81.5
+ANPO	**95.6**	**93.7**	92.1	**68.3**	**87.4**	**83.9**	**88.5**	77.6	**82.6**	**84.9**	**83.5**	**97.9**	92.4	**93.0**	**46.2**	**96.8**	**42.5**	**78.1**	**82.4**

C.2 TOOL-CALL RATE PER BENCHMARK

We report the tool-call rate of ANPO and the baselines on every benchmark in Table 11.

C.3 PER-BENCHMARK ABLATION RESULTS

We report the per-benchmark Pass@1 and tool-call rate of the ablation in Table 12.

Table 11: Tool-call rate (%) of the 2B and 4B models.

	Recognition					Reasoning						Perception						Avg.
Method	Doc VQA	Chart QA	Text VQA	GQA	**Avg.**	Math Verse	Math Vista	Math Vision	Dyna Math	Logic Vista	**Avg.**	V*	HR Bench	Tally QA	Visual Probe$_H$	VLMs AreBlind	Baby Vision	**Avg.**	
Qwen3.5-2B																			
Base	97.4	59.2	89.6	90.6	84.2	41.7	42.4	39.1	48.3	38.9	42.1	98.3	87.6	86.7	99.5	94.9	71.2	89.7	72.4
SFT	31.0	7.2	18.6	13.0	17.4	27.0	17.2	25.6	29.2	19.1	23.6	97.9	78.2	63.1	90.1	24.9	53.4	67.9	39.7
+ GRPO	1.7	1.2	1.4	0.9	1.3	2.4	0.7	1.1	3.0	0.4	1.5	17.9	16.6	1.2	28.5	1.0	1.5	11.1	5.3
+ ARPO	1.9	1.4	1.4	1.9	1.6	5.3	1.4	1.8	2.1	1.9	2.5	3.4	5.7	0.8	19.6	1.2	3.6	5.7	3.6
+ AKBE	0.0	0.0	0.0	0.0	0.0	0.1	0.1	0.0	0.2	0.1	0.1	0.0	0.2	0.0	0.2	0.0	0.0	0.1	0.1
+ AXPO	88.0	63.0	77.5	84.6	78.3	35.5	34.3	43.1	42.8	24.8	36.1	99.7	90.3	85.7	99.5	46.0	51.5	78.8	64.4
+ **ANPO**	57.5	20.7	58.6	62.2	49.8	9.0	9.6	5.0	8.9	6.0	7.7	80.8	63.0	61.3	89.9	25.6	24.8	57.5	38.9
Qwen3.5-4B																			
Base	90.1	13.5	52.4	46.9	50.7	34.0	26.1	20.6	31.8	24.2	27.3	96.6	75.3	74.6	92.9	79.2	61.0	79.9	54.6
SFT	43.0	16.8	19.3	19.2	24.6	31.9	19.3	29.2	30.8	18.0	25.8	97.1	80.2	61.3	94.6	25.9	52.1	68.5	42.6
+ GRPO	10.9	3.6	13.5	4.2	8.0	3.9	6.4	0.5	2.7	0.3	2.8	24.9	27.5	16.7	31.6	16.1	30.0	24.5	12.9
+ ARPO	20.2	8.5	7.3	5.2	10.3	5.8	6.9	1.9	4.8	1.6	4.2	39.2	29.7	16.2	39.9	12.7	21.5	26.5	14.8
+ AKBE	10.8	6.3	4.5	6.4	7.0	2.0	9.3	0.8	4.6	1.1	3.6	34.9	27.5	5.6	18.6	4.3	22.0	18.8	10.6
+ AXPO	96.9	82.5	91.8	87.0	89.5	49.3	42.9	48.0	59.8	36.2	47.2	100.0	90.0	82.5	98.4	79.3	86.0	89.4	75.4
+ **ANPO**	82.6	19.3	37.8	21.0	40.2	7.9	14.6	8.1	11.7	10.2	10.5	94.6	74.7	72.3	91.5	51.9	75.8	76.8	44.9

Table 12: Ablation of the components of ANPO on 2B.

	Recognition					Reasoning						Perception						Avg.
Method	Doc VQA	Chart QA	Text VQA	GQA	**Avg.**	Math Verse	Math Vista	Math Vision	Dyna Math	Logic Vista	**Avg.**	V*	HR Bench	Tally QA	Visual Probe$_H$	VLMs AreBlind	Baby Vision	**Avg.**	
Pass@1																			
SFT	83.8	88.2	81.0	53.1	76.5	55.4	72.3	27.1	62.5	46.0	52.7	75.5	71.9	78.8	19.9	61.8	9.2	52.8	59.1
GRPO	84.9	89.4	81.7	55.0	77.7	65.9	76.4	41.8	67.1	55.5	61.3	77.7	73.6	81.1	20.7	63.1	14.1	55.1	63.2
+ Dual-Arm	85.3	88.4	81.7	54.4	77.5	64.6	76.5	42.0	67.7	56.3	61.4	79.8	74.1	81.7	21.2	61.1	13.6	55.3	63.2
+ SRA	85.7	88.9	82.4	56.1	78.3	66.7	76.7	43.4	68.5	56.9	62.4	80.4	74.2	82.5	22.7	64.4	14.4	56.4	64.3
+ NDR (all)	85.8	89.4	82.3	56.2	78.4	66.1	76.3	41.4	67.4	56.3	61.5	81.2	74.4	82.3	24.2	65.7	14.9	57.2	64.3
ANPO	87.2	88.5	82.8	56.6	78.8	67.7	78.0	44.6	69.3	58.0	63.5	82.9	75.7	84.0	23.6	66.3	15.4	58.0	65.4
Tool-call rate																			
SFT	31.0	7.2	18.6	13.0	17.4	27.0	17.2	25.6	29.2	19.1	23.6	97.9	78.2	63.1	90.1	24.9	53.4	67.9	39.7
GRPO	1.7	1.2	1.4	0.9	1.3	2.4	0.7	1.1	3.0	0.4	1.5	17.9	16.6	1.2	28.5	1.0	1.5	11.1	5.3
+ Dual-Arm	9.6	3.1	4.6	6.9	6.1	4.1	5.7	2.8	4.4	2.6	3.9	22.4	26.5	3.8	37.5	1.8	5.8	16.3	9.4
+ SRA	20.1	8.3	9.5	16.0	13.5	5.0	3.3	3.0	4.7	3.8	3.9	100.0	83.3	78.3	96.7	36.3	20.3	69.2	32.6
+ NDR (all)	87.4	23.8	64.5	87.6	65.8	12.1	15.4	13.4	14.3	9.2	12.9	99.9	86.8	99.5	99.3	40.7	44.3	78.4	53.2
ANPO	57.5	20.7	58.6	62.2	49.8	9.0	9.6	5.0	8.9	6.0	7.7	80.8	63.0	61.3	89.9	25.6	24.8	57.5	38.9

C.4 Per-Benchmark Design Choices

We report the per-benchmark Pass@1 and tool-call rate of the design choices in Table 13.

Table 13: Design choices of ANPO on 2B.

	Recognition					Reasoning						Perception						Avg.
Variant	Doc VQA	Chart QA	Text VQA	GQA	**Avg.**	Math Verse	Math Vista	Math Vision	Dyna Math	Logic Vista	**Avg.**	V*	HR Bench	Tally QA	Visual Probe$_H$	VLMs AreBlind	Baby Vision	**Avg.**	
Pass@1																			
Native	86.7	86.5	81.7	56.1	77.8	64.6	76.7	42.5	68.2	55.5	61.5	84.9	75.6	83.6	23.5	65.6	16.2	58.2	64.5
Pooled	86.9	88.1	82.7	56.1	78.5	66.2	76.8	42.6	69.2	55.6	62.1	81.8	72.7	82.6	21.2	64.6	14.3	56.2	64.1
Per trajectory	86.8	88.0	81.8	55.7	78.1	66.8	77.8	44.2	69.0	56.3	62.8	78.5	73.5	82.5	19.4	65.1	14.2	55.6	64.0
TN/TU label	85.4	85.4	81.3	55.1	76.8	65.8	73.2	39.7	64.6	52.1	59.1	80.8	74.2	77.6	20.6	64.2	13.6	55.2	62.2
Batch mean	85.7	86.5	82.1	55.3	77.4	65.6	76.2	40.9	68.1	54.9	61.1	81.2	71.7	81.4	20.5	63.1	13.5	55.2	63.1
Constant 0.3	83.3	83.8	81.7	54.5	75.8	66.6	77.4	43.6	68.8	56.8	62.6	81.7	72.8	82.5	17.7	62.8	13.9	55.2	63.2
Constant 0.1	87.3	89.3	82.4	56.2	78.8	67.6	77.5	42.6	69.3	55.6	62.5	82.9	73.3	82.4	17.7	64.3	14.1	55.8	64.2
ANPO	87.2	88.5	82.8	56.6	78.8	67.7	78.0	44.6	69.3	58.0	63.5	82.9	75.7	84.0	23.6	66.3	15.4	58.0	65.4
Tool-call rate																			
Native	61.7	13.7	61.6	77.5	53.6	14.4	15.2	8.2	5.5	8.6	10.4	100.0	93.3	99.3	96.7	51.3	20.3	76.8	48.5
Pooled	42.2	4.8	40.1	44.7	33.0	5.8	4.6	2.4	4.9	1.0	3.8	63.6	53.0	60.2	79.9	9.4	9.6	46.0	28.4
Per trajectory	42.8	5.7	33.8	35.1	29.3	15.8	14.6	14.0	12.5	11.1	13.6	94.8	68.1	49.8	82.3	6.6	8.4	51.7	33.0
TN/TU label	95.6	70.6	69.5	84.1	79.9	11.5	12.3	15.0	11.9	7.6	11.7	100.0	84.7	97.4	98.1	37.6	9.2	71.2	53.7
Batch mean	74.9	20.0	41.4	38.7	43.8	4.1	12.4	2.2	4.9	5.4	5.8	99.8	88.7	91.7	96.7	16.6	35.6	71.5	42.2
Constant 0.3	85.3	43.1	76.5	80.7	71.4	14.6	12.6	8.9	11.1	9.4	11.3	100.0	85.1	90.4	99.1	36.1	40.8	75.2	52.9
Constant 0.1	65.9	24.7	40.1	44.7	43.9	11.1	10.1	6.8	9.2	4.9	8.4	93.6	64.0	62.2	89.9	9.4	10.8	55.0	36.5
ANPO	57.5	20.7	58.6	62.2	49.8	9.0	9.6	5.0	8.9	6.0	7.7	80.8	63.0	61.3	89.9	25.6	24.8	57.5	38.9

D Additional Analyses

D.1 Oracle Tool-Call Rate per Benchmark

The natural TCR is the fraction of the rollouts of each policy that call the tool at least once. For the oracle label, we let the model answer each question with greedy decoding, which gives its most likely answer, once with the tool-off prompt and once with the tool-on prompt. A question does not need the tool if the answer with the tool-off prompt is correct. Otherwise, we let the policy answer with the tool, and the question needs the tool if this answer is correct. If this answer is also incorrect, the question fits neither label. We report the natural and oracle TCR of AXPO and ANPO on the reasoning and perception benchmarks under two treatments of the questions that fit neither label. Table 14 counts them as not needing the tool, as in Figure 12, and Table 15 excludes them from both rates, since they fit neither label. Over all questions of these benchmarks, the Pearson correlation between the fraction of the four rollouts that call the tool and the oracle label of each question is 0.675 for AXPO and 0.757 for ANPO under the treatment of Table 14, and 0.765 and 0.812 under that of Table 15.

Table 14: Natural versus oracle TCR on the reasoning and perception benchmarks. Questions that fit neither label are counted as unnecessary.

	AXPO		ANPO	
Benchmark	Natural	Oracle	Natural	Oracle
Reasoning				
MathVerse	35.5	13.3	9.0	4.5
MathVista	34.3	12.1	9.6	5.6
MathVision	43.1	10.9	5.0	3.6
DynaMath	42.8	9.1	8.9	6.2
LogicVista	24.8	9.7	6.0	4.0
Perception				
V*	99.7	77.6	80.8	65.0
HR-Bench	90.3	66.1	63.0	53.2
TallyQA	85.7	68.5	61.3	51.2
$\text{VisualProbe}_{\text{H}}$	99.5	21.7	89.9	27.4
VLMsAreBlind	46.0	27.7	25.6	15.2

Table 15: Natural versus oracle TCR on the reasoning and perception benchmarks. Questions that fit neither label are excluded.

	AXPO		ANPO	
Benchmark	Natural	Oracle	Natural	Oracle
Reasoning				
MathVerse	37.5	16.6	9.4	5.8
MathVista	32.0	14.2	9.8	6.4
MathVision	42.2	16.4	6.4	6.1
DynaMath	37.8	11.6	10.7	7.0
LogicVista	26.0	12.3	9.4	6.1
Perception				
V*	99.3	79.2	84.1	68.2
HR-Bench	86.7	69.9	64.7	55.5
TallyQA	88.5	71.3	66.6	56.7
$\text{VisualProbe}_{\text{H}}$	94.1	40.3	89.3	51.3
VLMsAreBlind	51.1	35.8	27.6	19.4

D.2 Tool-Call Rate and Accuracy per Tool

Table 16 reports, for each tool, the tool-call rate of the 2B models and their accuracy on the questions where they call that tool, averaged over the 15 benchmarks. The accuracy of each model is computed only on the questions where that model calls the tool, so its denominator differs across models and tools. GRPO rarely calls any tool, so its accuracy is based on a small number of questions.

Table 16: Tool-call rate (%) / accuracy (%) on the questions where the tool is called, for each tool of the 2B models.

	Zoom-In	Python	Segmentation	Depth
GRPO	3.5 / 41.6	1.4 / 58.7	2.7 / 54.8	0.0 / –
AXPO	36.1 / 56.5	20.8 / 55.1	26.4 / 58.9	4.3 / 59.5
ANPO	21.4 / 61.4	12.0 / 58.1	10.9 / 59.5	3.6 / 63.7

D.3 Redundant Tool Calls

We judge every tool-using trajectory of the 2B models in Table 7 with GPT-4o-mini (OpenAI et al., 2024) at temperature 0 in two steps. First, the judge sees only the question, its answer choices, and the reasoning before the first tool call, without the reference answer, and decides whether the reasoning has already committed to one final answer. Second, the judge decides whether the committed answer is equivalent to the reference answer, and an answer that matches it exactly is accepted without this step. A trajectory counts as a redundant tool call when its committed answer is equivalent to the reference answer. The prompts are given below.

Step 1: Commitment

```
You audit the reasoning of a vision-language model that can call
external tools (image zoom, python).
You are given a QUESTION (with answer choices if any) and the model's
REASONING written BEFORE its first tool call.
Decide whether the model had ALREADY committed to a specific final
answer before calling the tool.

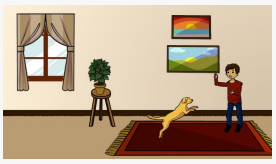
Rules:
- committed = true ONLY if the reasoning states or clearly concludes
ONE specific final answer to the question (e.g.  "the answer is B",
"so there are 4 birds", "= 12", "the sign says STOP").
- committed = false if the reasoning only lists or discusses the
options, restates the question, names several candidate values without
choosing, plans what to check, or says it cannot tell yet.
- If it commits to an answer and then calls the tool "to verify /
double-check", that still counts as committed.
- Copy the committed answer exactly as the reasoning states it
(letter, number, or short phrase).  Do not guess.
Respond with JSON only:  {"committed":  true|false, "answer_stated":
"<string or null>", "confidence":  "high"|"medium"|"low"}
```

Step 2: Equivalence

```
You compare a model's stated answer with the reference answer of a
question.
Treat as equivalent:  same option letter or the option's text, same
number after normalization (units, commas, trailing zeros, percent
sign), same entity with trivial spelling or article differences.
Otherwise not equivalent.
Respond with JSON only:  {"equivalent":  true|false}
```

D.4 Qualitative Examples

We show qualitative examples of GRPO, AXPO, and ANPO on the reasoning and perception benchmarks.



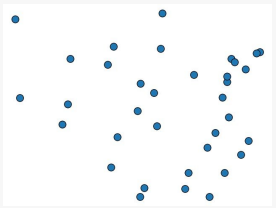
Benchmark MathVista. **Ground truth** `No`.
Prompt "Is the smaller picture below the larger picture? (A) Yes (B) No"

GRPO ✓
Think. *"... So the smaller picture is NOT below the larger picture. Therefore, the correct answer is No."*
No tool call.
Answer. `B` ✓

AXPO ×
Think. *"... I need to determine if the smaller picture is located below the larger picture. I will zoom in on the wall area to clearly see the two pictures and their relative positions."*
Tool call.
`image_zoom_in(bbox=[469, 85,816,513])`
Observation. Zoomed region [330,32,574,196] of the 704x384 image -> (488, 328) (attached). (If this is ...
Answer. `A` ×

ANPO (Ours) ✓
Think. *"... So the smaller picture is positioned above the larger picture, not below it. Therefore, the statement "the smaller picture is below the larger picture" is false."*
No tool call.
Answer. `B` ✓

Figure 17: Qualitative example of the 2B models on MathVista.



Benchmark DynaMath. **Ground truth** `35`.
Prompt "How many nodes are in this empty graph?"

GRPO ×
Think. *"... I think 29 or 30. I'll choose 30."*
No tool call.
Answer. `30` ×

AXPO ×
Think. *"... ensure I can clearly distinguish each node and avoid missing any or merging nearby ones."*
Tool call.
`image_zoom_in(bbox=[0, 0,1000,1000])`
Observation. Zoomed region [0,0,640,480] of the 640x480 image -> (1280, 9...
+2 more calls.
Answer. `32` ×

ANPO (Ours) ✓
Think. *"... Let's call the segmenter with the concept "blue circle" to get a count of each node."*
Tool call.
`segment(concept=blue circle)`
Observation. Found 35 'blue circle': #1 box[206, 579, 236, 617] score 0.93, #2 box[388, 790, 417, 829] ...
Answer. `35` ✓

Figure 18: Qualitative example of the 2B models on DynaMath.



Benchmark V*. **Ground truth** `A`.
Prompt "What is the color of the man's cap? (A) white (B) blue (C) red (D) black"

GRPO ×
Think. *"I need to find a man in the image and determine the color of his cap. I will zoom in on the right side of the image where a man appears to be standing near the blue building."*
Tool call.
`image_zoom_in(bbox=[925, 546, 956, 617])`
Observation. Zoomed region [1628,628,1682,710] of the 1760x1152 image -> (108, 164) (attached). (If thi...
Answer. `D` ×

AXPO ×
Think. *"...I will zoom in on the area where people are visible to get a clearer view."*
Tool call.
`image_zoom_in(bbox=[929, 550, 956, 618])`
Observation. Zoomed region [1635,633,1682,711] of the 1760x1152 image -> ...
+4 more calls.
Answer. `(no answer)` ×

ANPO (Ours) ✓
Think. *"...in the scene, as they are located in the background on the left side of the image."*
Tool call.
`image_zoom_in(bbox=[0, 550, 150, 660])`
Observation. Zoomed region [0,633,264,760] of the 1760x1152 image -> (528...
+1 more call.
Answer. `A` ✓

Figure 19: Qualitative example of the 2B models on V*.



Benchmark VisualProbe$_H$. **Ground truth** `KEBA`.
Prompt "What is the word above "TURECH""

GRPO ×
No tool call.
Answer. `THAI` ×

AXPO ×
Think. *"...I can see some signs on the buildings, but I need to zoom in to find the specific text."*
Tool call.
`image_zoom_in(bbox=[880, 400, 1000, 480])`
Observation. Zoomed region [1464,499,1664,599] of the 1664x1248 image -> ...
+1 more call.
Answer. `SPECIALITY` ×

ANPO (Ours) ✓
Think. *"...Let me zoom in on that area to get a clearer view."*
Tool call.
`image_zoom_in(bbox=[880, 400, 990, 470])`
Observation. Zoomed region [1464,499,1647,586] of the 1664x1248 image -> ...
+1 more call.
Answer. `KEBA` ✓

Figure 20: Qualitative example of the 2B models on VisualProbe$_H$.



Benchmark TallyQA. **Ground truth** `4`.
Prompt "How many giraffes are in the photo?"

GRPO ×
Think. *"... The reflection in the water shows the same three shapes – which confirms our count. Therefore, the answer is:"*
No tool call.
Answer. `3` ×

AXPO ×
Think. *"... I will zoom in on the group to get a clearer view and count each one distinctly."*
Tool call. `segment(concept=giraffes`
Observation. Found 4 'giraffes': #1 box[508, 49, 754, 422] score 0.92, #2...
Think. *"... I will proceed with the count of three clearly visible giraffes."*
Tool call. `python(code=# Count the number of giraffes detected giraffes_count = 4 p...)`
Observation. Number of giraffes detected: 4
Answer. `3` ×

ANPO (Ours) ✓
Think. *"... I can see what appears to be a group of giraffes standing near a tree and rocks. Let me use the segmenter to get an accurate count of the giraffes."*
Tool call. `segment(concept=giraffe)`
Observation. Found 4 'giraffe': #1 box[508, 47, 754, 422] score 0.96, #2 box[196, 76, 492, 451] score 0...
Answer. `4` ✓

Figure 21: Qualitative example of the 2B models on TallyQA.



Benchmark VLMsAreBlind. **Ground truth** `Yes`.
Prompt "Are the two circles overlapping?"

GRPO ×
Think. *"... Therefore, based on visual inspection, the two circles are not overlapping."*
No tool call.
Answer. `No` ×

AXPO ×
Think. *"... I need to determine if they are overlapping. Let me zoom in to get a clearer view of their relationship."*
Tool call. `image_zoom_in(bbox=[350, 350,750,750])`
Observation. Zoomed region [403,403,864,864] of the 1152x1152 image -> (922, 922) (attached). (If this ...
Answer. `No` ×

ANPO (Ours) ✓
Think. *"... - This means the two circles are intersecting; that is, they are overlapping. Therefore, the answer to whether the two circles are overlapping is yes."*
No tool call.
Answer. `Yes` ✓

Figure 22: Qualitative example of the 2B models on VLMsAreBlind.

E Detailed Related Works

E.1 Multimodal Large Language Models

Multimodal large language models connect a vision encoder to a language model through visual instruction tuning (Liu et al., 2023; Bai et al., 2023; Wang et al., 2025e; Bai et al., 2025), and recent models such as Qwen3.5 (Qwen Team, 2026), Kimi (Kimi Team, 2026) and GLM-5 (GLM-5 Team, 2026) are trained to act as native multimodal agents. In agentic systems built on these models, the model emits a call, receives an external output, and conditions the remaining reasoning on it (Yao et al., 2023). For a visual agent, the call acts on the image itself, by cropping a region, running a program over it, or invoking a perception model such as SAM3 (Carion et al., 2026) or Depth Anything 3 (Lin et al., 2025), and the returned view enters the context as new evidence (Ma et al., 2025b).

E.2 Efficient Multimodal Large Language Models

Large MLLMs impose substantial computational and memory demands, which makes them hard to deploy on resource-constrained devices such as robots and personal assistants. This has driven active research on small yet capable MLLMs, which can be broadly divided into two categories. **1) Changing MLLM Internals:** These methods modify the architecture or the parameters of the model. **Quantization** (Frantar et al., 2023; Xiao et al., 2024a; Lin et al., 2026b; Liang et al., 2026) reduces the numerical precision of the weights, for example from 16-bit to 8-bit or 4-bit, while keeping the accuracy close to that of the full-precision model. **Distillation** transfers the abilities of a large teacher to a small student. Early work distills the final outputs of the teacher (Cai et al., 2025; Shu et al., 2024), and later work transfers intermediate signals such as layer-wise responses (Lee et al., 2025b), attention over visual tokens (Jang et al., 2026; Kim et al., 2026b), and features across heterogeneous teachers and students (Lee et al., 2026). Recent work further combines distillation with reinforcement learning (Lee et al., 2025c;a; Yoon et al., 2026), and the same idea has been extended to vision-language-action models (Jeon et al., 2026). **Small MLLM Architecture** designs small models that keep the abilities of larger ones, by integrating the outputs of external computer vision models (Lee et al., 2024b) or enlarging the latent dimension without increasing the model size (Lee et al., 2024a). **2) Optimizing the Inference Stage:** These methods reduce the cost of using a model without changing its weights, such as **Token Pruning** (Chen et al., 2024; Yang et al., 2026b; Kim et al., 2026c), which removes visual tokens that contribute little to the answer. For MLLM agents, the cost also depends on how often the agent calls tools, and methods that reduce the number of tool calls are discussed in Section E.3.

E.3 Tool-RLVR for MLLMs

Tool-RLVR Methods. Tool-RLVR was first developed for search and computation in text (Jin et al., 2025; Li et al., 2025b; Feng et al., 2025a; Qian et al., 2025) and then extended to visual tools (Zheng et al., 2026; Wang et al., 2025c; Lai et al., 2025; Zhang et al., 2025c; Zhao et al., 2026). Several methods improve how rollouts are sampled around tool calls. ARPO (Dong et al., 2025b) branches additional rollouts right after a tool result when the token entropy rises, and AEPO (Dong et al., 2025a) balances the sampling budget between global and branch rollouts and rescales the gradients of high-entropy tokens. AXPO (Kang et al., 2026) targets tool-using groups in which every rollout fails, resamples continuations from the tool call, and credits the prefix by whether any continuation succeeds. AKBE (Chen et al., 2026a) runs tool-free rollouts alongside the tool rollouts and imitates a selected correct trajectory with a cross-entropy term, a tool-free one when any tool-free rollout succeeds and the tool-using one with the fewest calls otherwise. Concurrent with our work, TAPO (Dong et al., 2026) shows that assigning the trajectory-level advantage to every token penalizes useful tool calls in failed trajectories of search agents, and compensates this negative credit by comparing calls with similar parameters within the batch. IAPO (Yang et al., 2026c) improves tool use in small multimodal agents by aligning their attribution over the input with that of a stronger teacher. Unlike these methods, which adjust the credit of each trajectory or call with a criterion fixed in advance, such as parameter similarity or a teacher, ANPO corrects the decision per question by sharing one decision reward among its tool-using trajectories, and judges necessity from the rollouts of the model being trained, a design grounded in the analysis of Section 3. These methods credit the decision to call with the outcome of each trajectory or reduce the tool-free comparison to a binary choice, and AKBE and AXPO require additional rollouts.

Tool-Efficiency Methods. Another line aims to reduce tool calls regardless of the question. OTC-PO (Wang et al., 2025d) multiplies a tool reward that favors fewer calls by the correctness reward, and Metis (Yan et al., 2026), trained with HDPO, adds an efficiency channel computed only among correct trajectories. Both lower the call rate on average but give no signal to call more on a question where calling is right. A method that lowers the call rate everywhere and one that lowers it only where the tool is unnecessary are indistinguishable in an average over benchmarks, which is why we report the call rate by benchmark group.

Tool-Necessity Methods. A third line aims to call the tool only where it is necessary. AutoTool (Ma et al., 2026) routes each query with a mode token, `<tool_on>` or `<tool_off>`, which requires the model to learn new tokens. AdaTooler-V (Wang et al., 2026a) scales the tool reward by a benefit score computed once with a larger model before training, which does not follow the model as it changes during training. Concurrent with our work, TACO (Feng et al., 2026) probes the answer before and after a tool call within a single trajectory and uses the difference as the value of the call, so the value still rests on that one trajectory. SAAS (Tang et al., 2026) contrasts search-enabled and search-disabled rollouts to find the knowledge boundary of a text search agent under the current policy, and turns it into a trajectory-level penalty on unnecessary searches, which suppresses calls but does not strengthen them where the tool is needed. Look-Less (Li et al., 2025a) compares forced-tool and tool-free rollouts of each question and thresholds their accuracies into a binary necessity label, which is then used as an alignment reward on the remaining rollouts. In contrast, ANPO shares one decision reward among the tool-using trajectories of a question, separating the decision from the correctness of each trajectory, and uses the difference between the two arms directly as the decision advantage, without additional rollouts, a mode token, or a fixed label, following the analysis of Section 3.

F Limitations and Future Work

The design of our work rests on the controlled analyses of Section 3, which isolate how each stage and the necessity of the tool shape the learning signal of the tool-call decision, but it is not supported by a theoretical analysis. Nonetheless, the ablations in Section 5.3 suggest that each component changes the tool-call rate and accuracy in the direction these analyses predict. We expect that a theoretical account of how the stage-wise signal shapes the tool-call decision, and in particular the probability of the special tool-call token $<tool_call>$, would further clarify when our work helps, and leave it to future work.

Our work also divides a trajectory into stages at each $<tool_call>$ tag. Between two tool calls, the tokens that read the earlier tool result are counted as the decision of the next call, although they may also do what the answer stage does. We expect that a finer division of the tokens between tool calls would further separate the two signals, and leave it to future work.

We also use four visual tools of different kinds, namely Python, Zoom-In, object segmentation, and depth estimation, but a much larger and more diverse set of tools is not examined. In addition, the segmentation and depth tools run on locally served SAM3 and DA3 models, and the cost of serving them is not included in our efficiency analysis. Consequently, the efficiency comparison reflects the cost of training the policy rather than the full cost of running the tools.

Regarding the scope of this paper, we train models of up to 4B parameters from one model family, and whether our work applies at larger scales is left open. Extending our work to other model families (Wang et al., 2025e; An et al., 2025) and to richer tool environments, where agents call external tools such as search engines (Jin et al., 2025) and act over longer horizons with many turns (Feng et al., 2025b; Wang et al., 2026c), is a natural next step. Such environments also require the agent to learn which tool to call and to judge the necessity of each call.