Agent Explorative Policy Optimization for Multimodal Agentic Reasoning

Minki Kang[*] KAIST
Shizhe Diao NVIDIA
Ryo Hachiuma NVIDIA
Sung Ju Hwang KAIST
Pavlo Molchanov NVIDIA
Yu-Chiang Frank Wang NVIDIA
Byung-Kwan Lee[†] NVIDIA

Abstract

Vision-language models with extended reasoning succeed on complex problems, but many real-world problems require external tools that internal reasoning alone often cannot resolve. Agentic reasoning therefore interleaves two behaviors with a structural asymmetry: thinking (the self-contained default) and tool use (a high-variance auxiliary acting). We refer to this asymmetry as the *Thinking-Acting Gap*. Under standard RL recipes like GRPO, the gap manifests as two diagnostic symptoms during training: tool use is attempted on only $\sim$30% of rollouts, and when attempted, the tool-using rollouts within a group are all-wrong on $\sim$40% of questions, suppressing the learning signal at the tool calls that needed it. We propose **AXPO** (**A**gent e**X**plorative **P**olicy **O**ptimization): for each all-wrong tool-using subgroup, AXPO fixes the thinking prefix and resamples the tool call and its continuation, paired with uncertainty-based prefix selection. Across nine multimodal benchmarks and three scales of Qwen3-VL-Thinking, SFT + AXPO outperforms SFT + GRPO at average (+1.8 pp Pass@1 and +1.8 pp Pass@4 at 8B on average) and 8B with SFT + AXPO surpasses the 32B Base on Pass@4 with 4× fewer parameters.

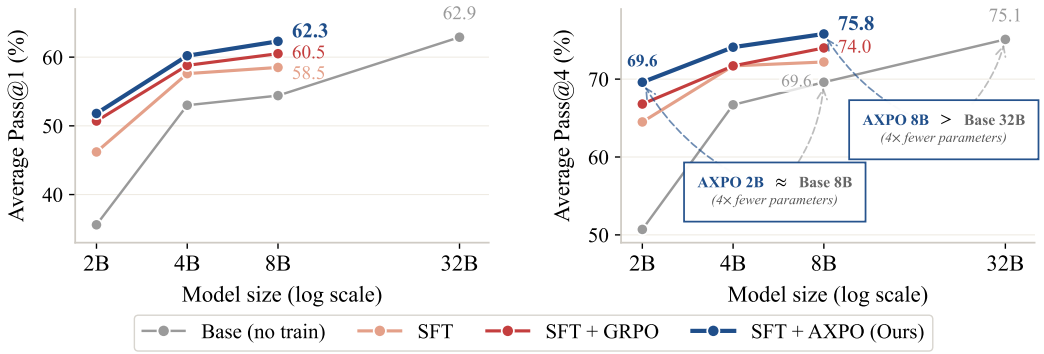


Figure 1: **Main result: AXPO narrows the agentic gap to a 4× larger baseline.** Average Pass@1 (left) and Pass@4 (right) over nine multimodal benchmarks across Qwen3-VL-Thinking sizes. *At 8B, AXPO surpasses the 32B Base baseline on Pass@4* while outperforming GRPO at every scale.

[*]Work done during internship at NVIDIA. [†]Project Lead. Correspondence to Minki Kang (minkikang@kaist.ac.kr) and Byung-Kwan Lee (byungkwanl@nvidia.com).

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

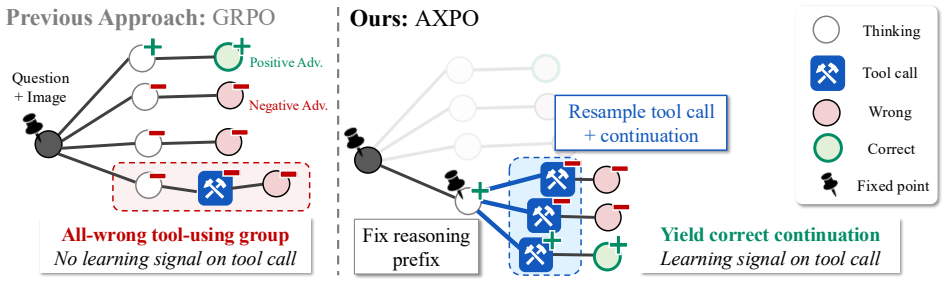


Figure 2: **Concept.** AXPO concentrates exploration on the tool call for agentic reasoning. **Left:** under GRPO, tool-using rollouts are scarce and often fail, leaving no positive learning signal at the tool call where it is most needed. **Right:** AXPO fixes the thinking prefix from the failed rollout and resamples the tool call and its continuation, recovering a correct rollout and producing a positive learning signal on the tool call.

1 Introduction

Large language models (LLMs) and vision-language models (VLMs) with extended reasoning capabilities have made remarkable progress, producing chains of thought that rival much larger models on mathematical and scientific reasoning benchmarks [1–3]. These reasoning models invest test-time compute into internal search and reflection to arrive at accurate answers. However, many real-world problems lie beyond what internal reasoning alone can resolve: retrieving up-to-date knowledge, performing complex numerical computation, and understanding fine-grained visual details all require access to external resources [4–8]. Tools such as web search, code interpreters, and image manipulation utilities can fill this gap, enabling what we call ***Multimodal Agentic Reasoning***: the model autonomously decides when to think further and when to act through tools, much as it learns to allocate thinking budget, but now extended to interaction with environments [9–14].

Agentic reasoning carries a deeper asymmetry between its two constituent behaviors: *Thinking* and *Tool use (Acting)*. *Thinking* is the default mode, self-contained within the model's token space: every token is under its control and errors can be corrected by exploring a different chain of thought [15–17]. *Tool use*, by contrast, is an auxiliary behavior that is simultaneously powerful but fragile: a short token sequence triggers an external response the model cannot predict or control, and small changes in the call can produce drastically different outcomes (e.g., invalid python code with wrong indent) [18]. Reasoning models are heavily post-trained for thinking, but far less for acting. We refer to this asymmetry between a safe default and a high-variance auxiliary as the ***Thinking-Acting Gap***.

Due to this gap, eliciting agentic reasoning remains a challenge despite the capabilities of recent reasoning VLMs [19]. Relying solely on prompting is often insufficient to ensure an appropriate tool call when needed. This necessitates post-training pipelines, such as Supervised Fine-Tuning (SFT) and Reinforcement Learning (RL), as evidenced in Figure 1. SFT reliably teaches the *mechanics* of tool use but struggles to teach the model to utilize tools better for situations the teacher's demonstrations did not cover. The resulting agent inherits the teacher's failure modes and often plateaus near its performance [20, 21]. Therefore, the model requires a training signal that goes beyond imitation, which motivates utilizing RL with verifiable outcome rewards after SFT [18, 22, 23].

However, RL methods such as GRPO [24–26] are suboptimal to make models proficient at acting, because two patterns suppress the tool-call learning signal. First, tool use remains a minority of rollouts compared with thinking-only (Figure 3a). Second, when tool use is attempted, the entire group of rollouts is far more likely to fail together (Figure 3b). Under GRPO, this asymmetry weakens the tool-call learning signal in both regimes: tool-using rollouts trail thinking-only successes in mixed groups (negative advantage) and zero out in all-wrong groups [27, 28]. These observations motivate targeted exploration for tool use during RL. This raises a critical question: *how can we effectively induce such exploration?* The Thinking-Acting Gap suggests the answer. Thinking is the native mode of the model and the tool call is the separate mode that turns an intent into a concrete action where the same intent admits diverse implementations (Figure 3c).

Building on this observation, our key insight is that resampling at the tool call lets the model explore diverse agentic reasoning: we anchor the thinking prefix and resample the tool call and its continuation (Figure 2). We call this *tool-call resampling*. The operation has no analog in pure reasoning, where there is no comparable boundary between a committed intent and a high-variance action. When a

tool-using trajectory fails, the thinking prefix may be entirely sound. Re-deriving the full trajectory from scratch wastes budget on prefixes that were not the bottleneck, whereas tool-call resampling concentrates exploration precisely where the Thinking-Acting Gap manifests. We propose **AXPO** (**A**gent e**X**plorative **P**olicy **O**ptimization) based on this principle to narrow the gap during RL.

Concretely, AXPO instantiates three design choices on top of a small extra rollout budget: (1) *where to resample*, only on all-wrong tool-using subgroups where a single recovery flips the GRPO advantage; (2) *which prefix*, the highest-uncertainty ones; and (3) *how resamples enter the gradient*, a per-prefix GRPO advantage on continuations and a binary recovery indicator on the source prefix. Unlike prior reward-shaping methods [18, 23, 29] or rollout scaling methods [30, 31], AXPO restructures the rollout distribution itself at the locus of the Thinking-Acting Gap.

Across nine multimodal benchmarks at Qwen3-VL-Thinking 2B/4B/8B [19], SFT + AXPO outperforms SFT + GRPO at average with Pass@4 gains of $+2.8/+2.3/+1.8$ pp. **At 8B, SFT + AXPO surpasses the 32B Base** on Pass@4 and reaches 99% of its Pass@1, despite using $4\times$ fewer parameters (Figure 1). The improvement concentrates on tool-intensive benchmarks, while preserving the performance improvement in reasoning benchmarks compared to SFT + GRPO baseline. Notably, $+25\%$ extra resampling budget with AXPO delivers $+1.1$ pp over $+100\%$ extra rollout budget with GRPO, confirming that the gain comes from where compute is spent.

Our contributions are as follows:

- We identify the **Thinking-Acting Gap**, an asymmetry between thinking and tool use that manifests in group-based agentic RL as two symptoms: tool use is attempted on 30% of rollouts and tool-using subgroups (the tool-using rollouts within a group) are all-wrong on 40% of questions, yielding non-positive advantage on tool-call tokens.
- We propose **AXPO**, an RL algorithm that closes the gap via *tool-call resampling*: for all-wrong tool-using subgroups, AXPO fixes the thinking prefix at uncertain tool calls and resamples from them, providing at least as much tool-use recovery coverage as from-scratch sampling at fixed budget when the stated per-prefix condition holds (Proposition 1).
- We validate AXPO on nine multimodal benchmarks across three scales of Qwen3-VL-Thinking, with gains for all sizes compared to GRPO baseline, and the **resulting 2B model matches the 8B Base and the 8B model surpasses the 32B Base on Pass@4**.

2 Analysis of RL in Agentic Reasoning

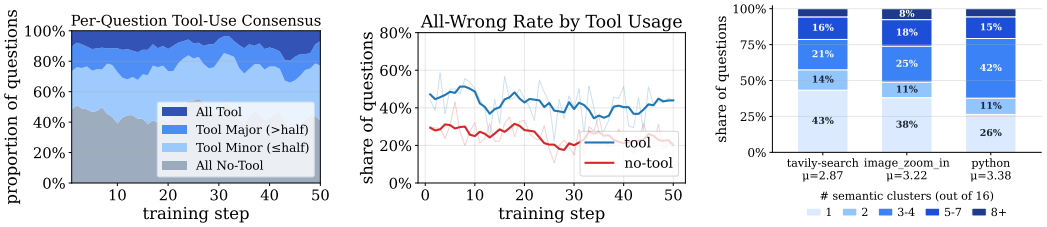
2.1 Preliminaries

Agentic Reasoning. We study the agentic reasoning of Vision-Language Models (VLMs) π_θ given an input $\boldsymbol{x}$ (text question with visual image) and instruction prompt (A.2), interleaving thinking and tool use. At turn t, the model emits a thinking segment $\boldsymbol{t}_t$ (in `<think> ... </think>`) and an action $\boldsymbol{a}_t$ (a tool call formatted as JSON with tool type and arguments, in `<tool_call> ... </tool_call>`), then receives an observation $\boldsymbol{o}_t = \texttt{exec}(\boldsymbol{a}_t)$ (in `<tool> ... </tool>`). We write $\boldsymbol{y}_t = \boldsymbol{t}_t\boldsymbol{a}_t\boldsymbol{o}_t$ with $\boldsymbol{t}_t\boldsymbol{a}_t \sim \pi_\theta(\cdot \mid \boldsymbol{x}, \boldsymbol{y}_{<t})$. This continues until the final answer (in `<answer> ... </answer>`) is emitted or the turn limit is reached, yielding $\tau = (\boldsymbol{x}, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_T, \boldsymbol{y}_{\text{ans}})$. Our tool set covers three standard [7, 18, 23]: a Python interpreter, a web search, and an image zoom-in tool.

RL training with GRPO. Our starting point is the standard agentic post-training pipeline [18, 23]: supervised fine-tuning (SFT) on expert trajectories followed by reinforcement learning (RL) with verifiable rewards. Given an input $\boldsymbol{x}$, the policy π_θ generates a group of N rollouts $\{\tau_i\}_{i=1}^N$, each receiving a binary outcome reward $r_i \in \{0, 1\}$ based on answer correctness. Group Relative Policy Optimization (GRPO) [24] computes a group-normalized advantage $A_i = (r_i - \text{mean}(\{r_j\}_{j=1}^N))/\text{std}(\{r_j\}_{j=1}^N)$, which is assigned uniformly to every token in τ_i and used to update the policy via the PPO-clip surrogate $\mathcal{L}_{\text{clip}}$ [32]. We defer the explicit form of $\mathcal{L}_{\text{clip}}$ in Appendix A.1.

2.2 Diagnosing the Thinking-Acting Gap

We now ground the Thinking-Acting Gap in three diagnostic measurements collected during RL training (Figure 3). Our starting point is the SFT checkpoint, trained with GRPO on outcome rewards.



(a) Proportion of questions by tool-using rollout count (8 rollouts/question) across RL steps under GRPO.

(b) All-wrong rate by subgroup type under GRPO: the tool-using fails together far more than the no-tool.

(c) Semantic cluster count of 16 resampled tool calls from a fixed failed-rollout prefix, per tool type.

Figure 3: **Diagnosing the Thinking-Acting Gap under GRPO.** Tool use is infrequent and tool-using subgroups fail together more often, while resampling at a fixed prefix yields diverse tool calls.

Tool use is under-attempted. A defining property of agentic reasoning, unlike conventional agentic RL [31, 33], is that tool use is *optional*: the model can either produce a `<tool_call>` tag or skip straight to `<answer>`. This optionality makes action fragile under RL. Figure 3a bins questions by their tool-using rollout count among $N = 8$. The fraction of rollouts containing at least one tool call is roughly 27% at step 1 and stays within 20-35% during RL training, well below the thinking-only majority. GRPO exerts no pressure to grow the tool-using minority, and the behavior that distinguishes agentic from non-agentic reasoning remains under-represented in each group.

When attempted, the tool-using subgroup is prone to being all-wrong. Within a GRPO group of N rollouts for one question, we call the rollouts that emit a tool call the *tool-using subgroup* and the rest the *no-tool subgroup*, and the tool-using subgroup is *all-wrong* when none of its rollouts is correct, even if some no-tool rollouts in the same group succeed. Conditional on a question where tool use is attempted, how often does the tool-using subgroup fail together? Figure 3b shows this all-wrong rate is $\sim$40% for tool-using subgroups compared to $\sim$25% for no-tool subgroups. Under group-normalized advantage, this asymmetry yields non-positive advantage on tool-call tokens in both regimes: in mixed groups they trail the no-tool success signal, and in fully all-wrong groups the numerator $r_i - \text{mean}(\{r_j\})$ collapses to zero. Either way, effective learning signal on tool calls is suppressed on the questions where tool use would matter most.

Tool calls admit substantive resampling at a fixed thinking. Resampling at the tool call is only worthwhile if it is a real divergence point under the policy. If the thinking prefix already commits to the concrete action, resampling at the tool-call boundary changes nothing. To validate, we fix the thinking prefix $\boldsymbol{t}_1$ at the first tool-call boundary, draw 16 continuations from $\pi_\theta(\cdot \mid \boldsymbol{x}, \boldsymbol{t}_1)$, and use an LLM judge to cluster the resulting tool calls by semantic intent. Figure 3c shows that the resampled tool calls do not collapse to a single class. Across tools, the 16 samples occupy 2.9-3.4 distinct semantic clusters on average. The thinking prefix therefore does not determine the exact tool call and the tool call is an under-explored divergence point whose exploration must be supplied externally.

Takeaways: GRPO under-trains the tool call

- Tool use remains a minority behavior under GRPO: $\sim$20–35% of rollouts across RL steps, well below the thinking-only majority (Figure 3a).
- When attempted, the tool-using subgroup is all-wrong on $\sim$40% of questions (vs. 25% for the no-tool subgroup), so tool-call tokens get negative advantage in mixed groups and zero in fully all-wrong groups (Figure 3b).
- Resampling from a fixed prefix preserves substantive diversity in the tool call (2.9-3.4 semantic clusters per 16 samples in Figure 3c): the tool call is a real divergence point.

3 Agent eXplorative Policy Optimization (AXPO)

3.1 Tool-call resampling

In agentic reasoning, the policy must learn both thinking and tool use where the latter is the under-trained behavior under GRPO (Section 2.2). Rather than scaling rollouts N to generate tool-using

successes uniformly, we commit to a thinking prefix that promises a tool call and concentrate stochasticity on the continuation.

Concretely, let τ^{src} denote a **tool-using** rollout from the standard GRPO group of $\boldsymbol{x}$, which we call the *source rollout*, and let $\boldsymbol{t}_1^{\text{src}}$ be its prefix that includes the opening `<tool_call>` tag, so that decoding from $\boldsymbol{t}_1^{\text{src}}$ is constrained to produce a valid tool call. *Tool-call resampling* draws K continuations $\{\boldsymbol{y}_k^{\text{res}}\}_{k=1}^{K} \sim \pi_\theta(\cdot \mid \boldsymbol{x}, \boldsymbol{t}_1^{\text{src}})$, executes the tool calls inside each, and rolls every resampled trajectory forward until an `<answer>` tag or the turn limit. Each resampled trajectory $\tau_k^{\text{res}} = (\boldsymbol{t}_1^{\text{src}}, \boldsymbol{y}_k^{\text{res}})$ therefore shares the fixed thinking prefix with τ^{src} and diverges at the tool call, concentrating stochasticity on the action that the thinking was preparing for. Each resampled trajectory receives the same binary outcome reward $r_k^{\text{res}} \in \{0, 1\}$ as a standard rollout.

Theoretical motivation. *Why grow tool-using rollout coverage?* Recent analysis [30] shows that scaling per-prompt rollouts N yields a policy-improvement guarantee that grows with the correct-token mass the group covers. However, our learning target is correct *tool-using* rollouts, which can provide positive learning signals for tool-call tokens. A from-scratch rollout invokes a tool with probability $q = \Pr_{\pi_\theta}[\texttt{tool use} \mid \boldsymbol{x}]$, so a fraction $1 - q$ of the raw sampling budget goes to non-tool rollouts that cannot directly reinforce tool-call tokens. Tool-call resampling sidesteps this waste by committing to a prefix $\boldsymbol{t}_1^{\text{src}}$ that has already crossed the `<tool_call>` boundary: every resampled continuation from $\boldsymbol{t}_1^{\text{src}}$ is tool-using by construction, so the full resampling budget lands on the signal we actually want.

Let $p(\boldsymbol{t}_1^{\text{src}}) = \Pr_{\pi_\theta}[R = 1 \mid \boldsymbol{x}, \boldsymbol{t}_1^{\text{src}}]$ be the success probability of a continuation from prefix $\boldsymbol{t}_1^{\text{src}}$, and $p^{\text{tool}} = \Pr_{\pi_\theta}[R = 1 \mid \boldsymbol{x}, \texttt{tool use}]$ the per-tool-using-rollout success rate of standard rollout. The following proposition makes the gain precise.

Proposition 1 (Conditional coverage advantage of tool-call resampling)**.** *Let $\boldsymbol{t}_1^{src}$ be a prefix that has crossed the tool call boundary with $p(\boldsymbol{t}_1^{src}) \geq q\, p^{tool}$. Then for any $N \geq 1$,*

$$\underbrace{1 - \big(1 - p(\boldsymbol{t}_1^{src})\big)^N}_{\textit{resampling: } N \textit{ tool-using continuations}} \quad \geq \quad \underbrace{1 - (1 - q\, p^{tool})^N}_{\textit{raw: } N \textit{ rollouts, } q\textit{-fraction tool-using}} , \tag{1}$$

with strict inequality whenever $p(\boldsymbol{t}_1^{src}) > q\, p^{tool}$ and $q\, p^{tool} \in (0, 1)$. Both sides measure the probability that the group contains at least one correct tool-using rollout.

Conditioning on a tool-committed prefix directs every additional sample toward a tool-using continuation. Under the stated per-prefix condition, Proposition 1 guarantees at least as much coverage of correct tool-using rollouts as from-scratch sampling at any fixed N, with a strict gain when the inequality is strict. The gain therefore depends on the recoverability of the selected thinking prefix. The proof and additional coverage analysis are in Appendix E.

Selected prefixes have higher mean continuation success. To examine the condition on the failed-rollout prefixes actually selected by the condition used in AXPO, we draw four continuations per selected prefix and four from-scratch rollouts for the same question. A from-scratch success must both invoke a tool and answer correctly, matching $q\, p^{\text{tool}}$ in Proposition 1. Averaged over samples during training, selected-prefix continuations succeed at 5.60%, compared with 3.53% for from-scratch tool-and-correct rollouts, a $1.59\times$ ratio. This supports a higher mean success probability over the selected-prefix distribution, without establishing that every selected prefix satisfies the condition. The additional per-prefix estimates and an audit of recovery credit are in Appendix C.6.

3.2 Resample candidate selection

Trigger and budget allocation. AXPO resamples only on groups in which the tool-using subgroup is non-empty and entirely wrong (an *all-wrong tool-using subgroup*; no-tool rollouts in the same group may succeed). These are exactly the groups where tool-call tokens receive non-positive advantage under GRPO (Section 2.2): zero when the entire group is all-wrong, negative when only no-tool rollouts succeed. A recovered continuation therefore delivers the largest gradient lift per resampled continuation. Proposition 1 would in principle admit groups with at least one correct tool-using rollout as well, but those can already produce positive advantage on tool-call tokens and gain comparatively little from a marginal recovery. Resampling every triggered prefix would add up to BNK continuations on top of BN standard rollouts (batch size B), so AXPO caps the extra cost

up to $r \cdot BN$ per step with $r < 1$ and allocates breadth-first: every triggered question receives its top-ranked prefix before any receives a second. Additional trigger-allocation measurements are in Appendix C.4.

Uncertainty-based prefix ranking. Not every admitted prefix is worth resampling. Figure 3c shows ∼30% of prefixes collapse to a single cluster across 16 resamples. AXPO therefore ranks candidates by the mean policy probability over the tool-call tokens in τ^{src} (a tractable uncertainty proxy [34]) and resamples the lowest-confidence prefixes first (see Appendix C.7 for the choice of confidence over predictive entropy).

3.3 Advantage calculation

Given the resampled trajectories from Section 3.2, we now specify how their advantages enter the gradient. Because each τ_k^{res} shares its prefix with the source rollout τ^{src}, training both as independent rollouts would assign contradictory advantages to the prefix tokens: the failed source penalizes the prefix while a successful resample rewards it. AXPO decomposes the advantage streams so that every token is updated from exactly one source.

For each selected prefix $\boldsymbol{t}_1^{\text{src}}$, the K resampled continuations form an independent advantage group. We compute a per-prefix GRPO advantage

$$\hat{A}_k^{\text{res}}(\boldsymbol{t}_1^{\text{src}}) \;=\; \frac{r_k^{\text{res}} - \text{mean}\big(\{r_j^{\text{res}}\}_{j=1}^{K}\big)}{\text{std}\big(\{r_j^{\text{res}}\}_{j=1}^{K}\big)}, \qquad k = 1, \ldots, K, \tag{2}$$

which is applied only to the continuation tokens $\boldsymbol{y}_k^{\text{res}}$ where the prefix thinking tokens are masked. The source trajectory's prefix tokens are updated through τ^{src} with a binary recovery reward that fires whenever resampling recovers at least one correct continuation: $r^{\text{prefix}}(\boldsymbol{t}_1^{\text{src}}) = \mathbb{1}[\exists\, k \in \{1, \ldots, K\} : r_k^{\text{res}} = 1]$. This recovery reward replaces the original (zero) source-rollout reward in the source group's GRPO normalization, yielding a per-prefix advantage that we apply to the source's prefix tokens as follows:

$$\hat{A}^{\text{prefix}}(\boldsymbol{t}_1^{\text{src}}) \;=\; \frac{r^{\text{prefix}}(\boldsymbol{t}_1^{\text{src}}) - \text{mean}\big(\{\tilde{r}_j\}_{j=1}^{N}\big)}{\text{std}\big(\{\tilde{r}_j\}_{j=1}^{N}\big)}, \qquad \tilde{r}_j \;=\; \begin{cases} r^{\text{prefix}}(\boldsymbol{t}_1^{\text{src}}), & j = \text{source index}, \\ r_j, & \text{otherwise}, \end{cases} \tag{3}$$

and the source continuation is masked. Combining both streams with the standard clipped surrogate from PPO [32] yields the resampling contribution for a selected prefix $\boldsymbol{t}_1^{\text{src}}$ to maximize:

$$\mathcal{L}_{\text{AXPO}}(\boldsymbol{t}_1^{\text{src}}) \;=\; \underbrace{\mathcal{L}_{\text{clip}}\big(\boldsymbol{t}_1^{\text{src}};\, \hat{A}^{\text{prefix}}(\boldsymbol{t}_1^{\text{src}})\big)}_{\text{prefix, source only}} \;+\; \sum_{k=1}^{K} \underbrace{\mathcal{L}_{\text{clip}}\big(\boldsymbol{y}_k^{\text{res}};\, \hat{A}_k^{\text{res}}(\boldsymbol{t}_1^{\text{src}})\big)}_{\text{resampled continuation}}. \tag{4}$$

Non-triggered groups and rollouts not selected for resampling retain the standard GRPO loss from Section 2.1. Equation 4 is added once per prefix selected in Section 3.2. The recovery-indicator r^{prefix} avoids the gradient conflict of propagating mixed-sign continuation advantages back to the shared prefix: $\hat{A}^{\text{prefix}}$ is monotone in r^{prefix}, so a successful continuation can yield positive credit for the prefix, depending on the source group's reward contrast.

4 Experiments

4.1 Experimental setup

Base models and tools. We use Qwen3-VL-Thinking [19] at three sizes (2B, 4B, 8B) for the primary experiments and additionally evaluate the SFT-then-RL pipeline on InternVL3.5-8B [35]. The tool set is the three families introduced in Section 2.1: a Python interpreter, a web search engine (Tavily search API[2]), and an image zoom-in tool. All tools share a unified `<tool_call>` interface. Full prompt templates and tool schemas are in Appendix A.2.

[2] https://www.tavily.com/

Table 1: Main results — Pass@1 (%, ↑) on **nine multimodal benchmarks**, grouped into three categories: Reasoning, Perception, and Search. All methods use **Qwen3-VL-Thinking** [19] at four sizes (2B, 4B, 8B, 32B). Best result per column within each size block is **bolded**; second-best is underlined. Δ rows show AXPO's gain over SFT + GRPO, the strongest comparable training recipe where AXPO outperforms it in most of evaluations. Pass@4 results are reported in Table 6.

	Reasoning			Perception				Search		Avg.
Method	Math Vision	Dyna Math	Math -VR	V*	Visual Probe	HRBen. -4K	HRBen. -8K	HR-MM Search	MM Search	
Qwen3-VL-2B-Thinking										
Base	18.6	60.5	16.9	63.0	5.0	64.6	55.9	8.2	27.7	35.6
GRPO	33.7	64.9	34.6	67.1	23.8	65.0	59.4	**18.4**	**41.4**	45.4
SFT	28.9	64.1	30.2	75.9	24.5	73.1	68.8	14.5	35.9	46.2
+ GRPO	34.8	66.4	35.8	**81.7**	**36.1**	74.3	70.6	15.7	40.6	50.7
+ **AXPO** *(Ours)*	**35.0**	**66.9**	**38.9**	81.3	35.1	**77.9**	**72.4**	18.1	40.3	**51.8**
Δ *vs. SFT + GRPO*	+0.2	+0.5	+3.1	−0.4	−1.0	+3.6	+1.8	+2.4	−0.3	+1.1
Qwen3-VL-4B-Thinking										
Base	48.0	76.6	50.8	80.6	24.8	73.6	67.5	16.9	37.8	53.0
GRPO	52.5	76.2	**57.9**	82.7	29.7	75.5	71.1	12.7	23.8	53.6
SFT	49.1	76.3	55.3	87.0	34.7	79.5	74.1	**20.6**	41.5	57.6
+ GRPO	51.6	**77.2**	57.4	**90.2**	38.0	78.8	73.8	20.0	42.3	58.8
+ **AXPO** *(Ours)*	**52.8**	76.4	57.2	88.9	**43.6**	**80.9**	**78.3**	20.1	**43.3**	**60.2**
Δ *vs. SFT + GRPO*	+1.2	−0.8	−0.2	−1.3	+5.6	+2.1	+4.5	+0.1	+1.0	+1.4
Qwen3-VL-8B-Thinking										
Base	47.1	75.9	54.9	77.7	31.8	72.8	66.1	21.0	42.7	54.4
GRPO	54.4	77.2	58.9	85.7	40.1	77.1	73.9	20.6	**45.1**	59.2
SFT	50.9	76.8	58.4	84.8	38.4	79.5	74.4	23.0	40.6	58.5
+ GRPO	55.3	78.2	60.4	87.7	40.1	79.5	74.9	24.4	44.0	60.5
+ **AXPO** *(Ours)*	**56.1**	**79.0**	**60.6**	**87.8**	**45.8**	**83.3**	**77.0**	**25.9**	45.0	**62.3**
Δ *vs. SFT + GRPO*	+0.8	+0.8	+0.2	+0.1	+5.7	+3.8	+2.1	+1.5	+1.0	+1.8
Qwen3-VL-32B-Thinking (inference-only baseline)										
Base	56.5	83.3	64.1	89.1	40.3	85.3	78.9	22.8	46.1	62.9

Training data & pipeline. SFT uses 64K correct trajectories drawn from ViRL [36], fvqa [37], and PyVision-RL [18], generated by Qwen3-VL-32B-Thinking [19] where ∼25% contain a tool call. RL uses a 16K filtered subset plus 22K questions from MMFineReason-hard [38]. We follow the SFT then RL pipeline of Section 2.1. AXPO augments the rollout loop with tool-call resampling, holding all other hyperparameters constant against SFT + GRPO. Full details are in Appendix A.1.

Benchmarks and metrics. We evaluate on nine multimodal benchmarks grouped into three task categories: **Reasoning** (MathVision [39], DynaMath [40], Math-VR [41]), **Perception** (V* [8], VisualProbe [7], HR-Bench-4K and HR-Bench-8K [42]), and **Search** (HR-MMSearch [43], MM-Search [44]); per-benchmark descriptions in Appendix A.3. All benchmarks are evaluated in the agentic setting with the same tool interface used in training. We report Pass@1 and Pass@4 (%), averaged over four rollouts per question, with answer matching logic tailored for each benchmark.

Baselines. We compare AXPO against four baselines: (1) **Base**: the base model only with prompting without additional training (2) **SFT**: the SFT checkpoint without RL (3) **GRPO**: RL from the base model without an SFT stage and (4) **SFT + GRPO**: the standard SFT-then-RL recipe. Our method, **SFT + AXPO**: applies tool-call resampling and advantage calculation from Section 3 on top of the same SFT-then-RL pipeline.

4.2 Main Results: AXPO narrows the agentic gap to a 4× larger baseline.

Table 1 reports Pass@1 across nine benchmarks at four model sizes (per-benchmark Pass@4 results in Table 6, Appendix B.2). SFT + AXPO outperforms SFT + GRPO at average performance on Pass@1 (+1.1 / +1.4 / +1.8 pp at 2B/4B/8B). On Pass@4, SFT + AXPO also beats SFT + GRPO at every scale (+2.8 / +2.3 / +1.8 pp). **At 8B, SFT + AXPO reaches** 99% **of the 32B Base's Pass@1 and surpasses it on Pass@4** (75.8 **vs.** 75.1)**, despite having** 4× **fewer parameters**. AXPO is the only recipe whose 8B point reaches the 32B line on Pass@1, and the only one whose Pass@4 crosses it (Figure 1).

Table 2: **Generalization to InternVL3.5-8B: Pass@1** (%, ↑) on the same nine benchmarks as Table 1. Best result per column is **bolded**, and second-best is underlined. Δ rows report AXPO minus SFT + GRPO (pp). Pass@4 results are in Table 7.

Method	Reasoning			Perception				Search		Avg.
	Math Vision	Dyna Math	Math -VR	V*	Visual Probe	HRBen. -4K	HRBen. -8K	HR-MM Search	MM Search	
InternVL3.5-8B										
Base	37.0	62.5	30.2	36.4	13.4	40.6	32.6	11.3	18.8	31.4
SFT	47.7	74.7	49.7	67.2	19.6	63.0	56.4	**18.8**	36.3	48.2
+ GRPO	47.4	74.7	50.3	70.4	**28.1**	67.6	58.6	15.3	32.9	49.5
+ **AXPO** *(Ours)*	**50.3**	**75.3**	**52.2**	**72.4**	26.9	**70.1**	**60.6**	17.9	**40.5**	**51.8**
Δ *vs. SFT + GRPO*	+2.9	+0.6	+1.9	+2.0	−1.2	+2.5	+2.0	+2.6	+7.6	+2.3

Table 3: **AXPO ablations** on **Qwen3-VL-8B-Thinking**. Each row removes (or alters) one AXPO design choice from the full method. All rows share the SFT initialization. Best per column is **bolded**; second-best underlined.

Method	MathVision		V*		VisualProbe		HR-MMSearch		**Average**	
SFT +	P@1	P@4	P@1	P@4	P@1	P@4	P@1	P@4	P@1	P@4
AXPO w/o prefix fix (rollout from scratch)	56.0	75.0	86.9	**96.9**	43.6	**68.9**	21.9	35.7	52.1	69.1
w/o all-wrong trigger	53.6	73.7	87.2	95.3	42.9	62.3	22.8	38.7	51.6	67.5
w/ no-tool rollout resampling	54.9	71.4	**89.1**	96.3	41.5	62.3	21.1	33.4	51.7	65.9
w/o uncertainty rank	53.0	72.4	85.7	93.2	43.9	63.2	21.4	34.4	51.0	65.8
w/o prefix credit	57.2	**76.6**	86.3	95.3	40.6	62.3	21.6	36.7	51.4	67.7
w/o separated advantage group	**57.7**	**76.6**	87.2	95.8	40.1	61.3	19.8	34.4	51.2	67.0
AXPO *(Ours)*	56.1	75.7	87.8	95.3	**45.8**	67.9	**25.9**	**42.0**	**53.9**	**70.2**

Decomposing the 8B Pass@1 column of Table 1, SFT + AXPO gains +0.6 pp on Reasoning, +2.9 pp on Perception, and +1.3 pp on Search over the SFT + GRPO baseline. In relative terms, the gain concentrates in Perception (+4%) and is smallest on Reasoning (+1%). Perception is where image-tool use most directly determines correctness, while on Reasoning the bottleneck is often the thinking rather than the tool call. A detailed discussion of the results is provided in Appendix C.3.

The aggregate gains are supported by three training seeds at 8B and paired per-question bootstrap comparisons at all three scales (Appendix B.4).

AXPO's gains transfer to InternVL3.5-8B. To test whether the improvement extends beyond Qwen3-VL-Thinking, we complete the SFT-then-RL pipeline on InternVL3.5-8B. Table 2 shows that SFT + AXPO improves Pass@1 over SFT + GRPO on eight of nine benchmarks, with a macro-average gain of +2.3 pp. The largest gain is on MMSearch (+7.6 pp), while VisualProbe decreases by 1.2 pp, indicating broad but heterogeneous transfer across benchmarks. On Pass@4, AXPO improves seven of nine benchmarks and the macro-average by +1.8 pp (Table 7). These results extend the aggregate benefit to a second model family. A replication of the diagnostic imbalance and paired significance tests are in Appendix C.1 and Appendix B.4, respectively.

4.3 Ablations: All components of AXPO matter

Table 3 ablates each AXPO design choice on 8B. Removing the all-wrong trigger reduces average Pass@1/Pass@4 from 53.9/70.2 to 51.6/67.5. Removing prefix fixing, the resampling target, uncertainty-based prefix selection, prefix credit, or the per-prefix advantage group also degrades both averages. See Appendix C.4 for ablation details.

4.4 AXPO outperforms alternative RL recipes

Table 4 compares AXPO against alternative training recipes: reward shaping (tool penalty / bonus), an extra rollout budget, and other RL algorithms (RLTF [47], CISPO [48], ARPO [31]). AXPO exceeds every alternative on both Pass@1 and Pass@4. The most direct more compute control, rollout 2×, doubles rollout budget yet underperforms than AXPO. The gain therefore comes from *where* compute is spent, not from *how much*. Reward-shaping methods both fall below the GRPO baseline of 51.9, indicating the gap is not closed by reward shaping either in agentic reasoning. Detailed analysis is in Appendix C.5. Per-step wall-time including tool execution and estimated model FLOPs are reported in Appendix B.8.

Table 4: **Comparison to alternative RL recipes** on **Qwen3-VL-8B-Thinking**. All rows share the SFT initialization. *Reward shaping & sampling* adds a per-tool-call reward or doubles the GRPO rollout budget. *RL algorithms* replaces GRPO with another alternative. Best per column is **bolded**; second-best underlined.

Method	MathVision		V*		VisualProbe		HR-MMSearch		**Average**	
SFT +	P@1	P@4	P@1	P@4	P@1	P@4	P@1	P@4	P@1	P@4
GRPO (baseline)	55.3	75.0	87.7	**96.9**	40.1	63.2	24.4	38.0	51.9	68.3
Reward shaping & sampling										
GRPO + Tool penalty [29]	53.0	69.7	83.1	90.1	30.4	46.2	17.4	30.5	46.0	59.1
+ Tool bonus [18, 45, 46]	52.9	72.7	86.0	93.2	42.7	59.4	21.6	37.0	50.8	65.6
+ rollout 2× ($N = 16$)	55.6	73.4	87.0	95.8	43.6	67.0	23.2	36.7	52.4	68.2
RL algorithms										
RLTF [47]	42.4	65.5	79.8	84.3	30.2	42.5	20.6	31.8	43.3	56.0
CISPO [48]	54.3	71.1	86.0	93.7	29.2	48.1	21.5	35.7	47.8	62.2
ARPO [31]	53.2	71.7	**89.3**	95.8	41.0	60.4	21.6	34.4	51.3	65.6
AXPO *(Ours)*	**56.1**	**75.7**	87.8	95.3	**45.8**	**67.9**	**25.9**	**42.0**	**53.9**	**70.2**

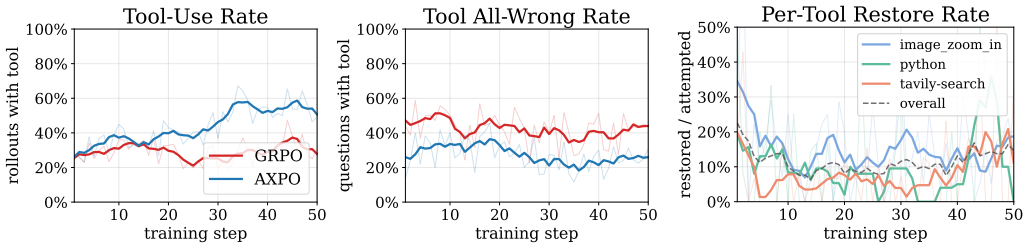


Figure 4: Both Thinking-Acting Gap symptoms reverse during AXPO training but stay flat under GRPO. **Left:** tool-use rate on original rollouts. **Middle:** all-wrong rate on tool-using subgroups, measured after resampling for AXPO. **Right:** per-tool failed-subgroup recovery rate, the share of all-wrong tool-using subgroups that resampling converts into subgroups with a correct tool-using trajectory.

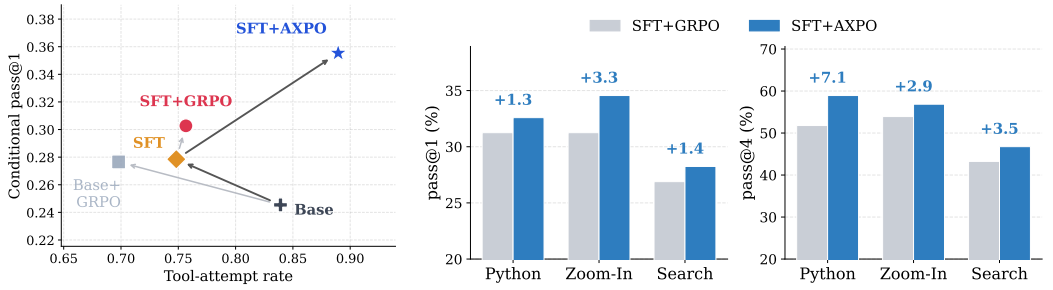
4.5 AXPO narrows the Thinking-Acting Gap during RL

Training dynamics. Figure 4 tracks the two diagnostic symptoms of Section 2.2 over RL steps. Under SFT + GRPO, tool use stays near 30% and the all-wrong rate on tool-using subgroups remains near 40%. Under SFT + AXPO, tool use increases by 28 pp over training. The tool-use rate is measured on the original on-policy rollouts before any resampling, so this increase reflects the learned policy's own tool-use decisions. The right panel separately shows that tool-call resampling recovers approximately 12% of all-wrong tool-using subgroups per step, supplying correct continuations on questions where every original tool-using rollout failed.

Policy improvement and immediate recovery both reduce failures. The AXPO all-wrong rate plotted in Figure 4 is measured after resampling and therefore includes same-step recovery. To separate this effect from changes in the policy, we also measure failures on the original rollouts. Averaged over training, the all-wrong rate is 42.3% under GRPO and 34.7% under AXPO before resampling, with resampling further reducing the AXPO rate to 27.4%. Thus, roughly half of the total reduction is already present before resampling, while the remainder comes from immediate recovery. The two measurements show that AXPO both learns a policy with fewer all-wrong tool-using subgroups and recovers additional successes during each training step. Trigger diagnostics are in Appendix C.4.

The GRPO gap persists on the same questions. Tools may be used more often on harder questions, potentially explaining the imbalance in Section 2.2 and Figure 3b. We therefore restrict the GRPO comparison to questions with both tool-using and thinking-only rollouts. Tool-using rollouts still succeed 7.5 pp less often (Appendix C.2), suggesting a reliability gap beyond question selection. This motivates measuring both tool-use frequency and reliability at evaluation.

Test-time dynamics. More globally, Figure 5a positions each training stage on the (tool-attempt rate, conditional Pass@1) plane measured on three benchmarks: every other stage either trades tool frequency for quality (Base → GRPO or SFT) or holds tool frequency fixed while pushing only



(a) Training stages on the tool-attempt rate vs. conditional pass@1 plane. Only AXPO expands both axes simultaneously.

(b) Per-tool quality on the matched-tool-use subset (questions where both SFT+GRPO and SFT+AXPO invoke the tool). AXPO improves pass@1 and pass@4 across all three tool families.

Figure 5: **Post-training analysis** pooled over MathVision, VisualProbe-Hard, and HR-MMSearch on 8B model. Conditional Pass@1 is measured on 715 tool-attempted questions from each benchmark.

quality (SFT $\rightarrow$ SFT + GRPO), while **only SFT + AXPO advances along both axes simultaneously**. The post-training matched-tool analysis (Figure 5b) is the per-question analysis of this joint advance: on questions where SFT + GRPO and SFT + AXPO both invoke a tool, AXPO improves Pass@1 across all three tool families, indicating the gain is in tool interaction quality.

5 Related Works

Agentic reasoning for VLMs. Agentic reasoning extends the reasoning loop beyond the model's token space, allowing it to emit a tool call, receive an external output, and condition subsequent reasoning on it [49, 9]. Tools span knowledge access [5, 4], computation [10, 11, 50–52], and visual perception [45, 7, 12, 46, 53, 54]; recent VLM systems combine image manipulation with code and search [44, 23, 18, 14, 55], and follow a two-stage SFT-then-RL recipe.

RL for agentic reasoning. On thinking-only trajectories, group-relative methods like GRPO [24–26] normalize rewards within N rollouts and uniformly assign the resulting advantage; BroRL [30] analyzes how scaling N closes coverage gaps. Three lines extend this family to tool-using rollouts: *direct ports* that reuse outcome-reward RL without changing the sampling structure [46, 23, 56–59, 22, 43], *mid-trajectory branching* that resamples the continuation after a tool call to absorb tool-output uncertainty [31, 60], and *reward shaping* that adds bonus or penalty terms on tool use [18, 29]. We include more detailed comparison with related works in Appendix D.

Our contribution. AXPO contrasts with each line. Against direct use of GRPO, it restores learning signal on the all-wrong tool-using subgroups identified in Section 2.2, which uniform sampling cannot reach. Against mid-trajectory branching [31, 60], it resamples *at* the tool-call boundary rather than after the tool observation, exploring the tool call itself rather than the continuation given a tool output. Against reward shaping, it acts on the rollout distribution while leaving the reward intact.

6 Conclusion

We identified a *Thinking-Acting Gap* in group-based agentic RL, an asymmetry between thinking (the self-contained default) and tool use (a high-variance auxiliary) that manifests during training as two symptoms: tool use is attempted on a minority of rollouts and tool-using subgroups (the tool-using rollouts within a group) are disproportionately all-wrong, so the advantage at tool-call tokens is non-positive on exactly the questions where tool use would help. AXPO narrows this gap by augmenting the rollout loop with *tool-call resampling*, which fixes the thinking prefix and concentrates exploration on the tool call itself, paired with uncertainty-based prefix selection. AXPO outperforms SFT + GRPO on average across nine benchmarks at all three model scales, with gains varying by benchmark. At 8B, it surpasses the 32B Base on Pass@4 with $4\times$ fewer parameters.

Limitations. Our study assumes verifiable outcome rewards for RL and trains models up to 8B parameters. We discuss limitations and the future directions in details in Appendix F.

Acknowledgments

This work was supported by NVIDIA Research, the Institute for Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (RS-2019-II190075, Artificial Intelligence Graduate School Program (KAIST)), and the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. RS-2023-00256259).

References

[1] DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

[2] Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, Alex Iftimie, Alex Karpenko, Alex Tachard Passos, Alexander Neitz, Alexander Prokofiev, Alexander Wei, Allison Tam, Ally Bennett, Ananya Kumar, Andre Saraiva, Andrea Vallone, Andrew Duberstein, Andrew Kondrich, Andrey Mishchenko, Andy Applebaum, Angela Jiang, Ashvin Nair, Barret Zoph, Behrooz Ghorbani, Ben Rossen, Benjamin Sokolowsky, Boaz Barak, Bob McGrew, Borys Minaiev, Botao Hao, Bowen Baker, Brandon Houghton, Brandon McKinzie, Brydon Eastman, Camillo Lugaresi, Cary Bassin, Cary Hudson, Chak Ming Li, Charles de Bourcy, Chelsea Voss, Chen Shen, Chong Zhang, Chris Koch, Chris Orsinger, Christopher Hesse, Claudia Fischer, Clive Chan, Dan Roberts, Daniel Kappler, Daniel Levy, Daniel Selsam, David Dohan, David Farhi, David Mely, David Robinson, Dimitris Tsipras, Doug Li, Dragos Oprica, Eben Freeman, Eddie Zhang, Edmund Wong, Elizabeth Proehl, Enoch Cheung, Eric Mitchell, Eric Wallace, Erik Ritter, Evan Mays, Fan Wang, Felipe Petroski Such, Filippo Raso, Florencia Leoni, Foivos Tsimpourlas, Francis Song, Fred von Lohmann, Freddie Sulit, Geoff Salmon, Giambattista Parascandolo, Gildas Chabot, Grace Zhao, Greg Brockman, Guillaume Leclerc, Hadi Salman, Haiming Bao, Hao Sheng, Hart Andrin, Hessam Bagherinezhad, Hongyu Ren, Hunter Lightman, Hyung Won Chung, Ian Kivlichan, Ian O'Connell, Ian Osband, Ignasi Clavera Gilaberte, and Ilge Akkaya. Openai o1 system card. *arXiv*, 2412.16720, 2024. URL https://doi.org/10.48550/arXiv.2412.16720.

[3] Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling, 2025. URL https://arxiv.org/abs/2501.19393.

[4] Bowen Jin, Hansi Zeng, Zhenrui Yue, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv*, 2503.09516, 2025. URL https://doi.org/10.48550/arXiv.2503.09516.

[5] Xiaoxi Li, Guanting Dong, Jiajie Jin, Yuyao Zhang, Yujia Zhou, Yutao Zhu, Peitian Zhang, and Zhicheng Dou. Search-o1: Agentic search-enhanced large reasoning models. *arXiv*, 2501.05366, 2025. URL https://doi.org/10.48550/arXiv.2501.05366.

[6] Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *Trans. Mach. Learn. Res.*, 2023, 2023. URL https://openreview.net/forum?id=YfZ4ZPt8zd.

[7] 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. *arXiv*, 2509.07969, 2025. URL https://doi.org/10.48550/arXiv.2509.07969.

[8] Penghao Wu and Saining Xie. V*: Guided visual search as a core mechanism in multimodal llms. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024*, pages 13084–13094. IEEE, 2024. URL https://doi.org/10.1109/CVPR52733.2024.01243.

[9] Junde Wu, Jiayuan Zhu, and Yuyuan Liu. Agentic reasoning: Reasoning llms with tools for the deep research. *arXiv*, 2502.04644, 2025. URL https://doi.org/10.48550/arXiv.2502.04644.

[10] Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Minlie Huang, Nan Duan, and Weizhu Chen. Tora: A tool-integrated reasoning agent for mathematical problem solving. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=Ep0TtjVoap.

[11] Heng Lin and Zhongwen Xu. Understanding tool-integrated reasoning. *arXiv*, 2508.19201, 2025. URL https://doi.org/10.48550/arXiv.2508.19201.

[12] Shitian Zhao, Haoquan Zhang, Shaoheng Lin, Ming Li, Qilong Wu, Kaipeng Zhang, and Chen Wei. Pyvision: Agentic vision with dynamic tooling. *arXiv*, 2507.07998, 2025. URL https://doi.org/10.48550/arXiv.2507.07998.

[13] Hongjin Su, Shizhe Diao, Ximing Lu, Mingjie Liu, Jiacheng Xu, Xin Dong, Yonggan Fu, Peter Belcak, Hanrong Ye, Hongxu Yin, Yi Dong, Evelina Bakhturina, Tao Yu, Yejin Choi, Jan Kautz, and Pavlo Molchanov. Toolorchestra: Elevating intelligence via efficient model and tool orchestration. *arXiv*, 2511.21689, 2025. URL https://doi.org/10.48550/arXiv.2511.21689.

[14] Yushi Hu, Weijia Shi, Xingyu Fu, Dan Roth, Mari Ostendorf, Luke Zettlemoyer, Noah A. Smith, and Ranjay Krishna. Visual sketchpad: Sketching as a visual chain of thought for multimodal language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=GNSMl1P5VR.

[15] Sara Vera Marjanovic, Arkil Patel, Vaibhav Adlakha, Milad Aghajohari, Parishad BehnamGhader, Mehar Bhatia, Aditi Khandelwal, Austin Kraft, Benno Krojer, Xing Han Lù, Nicholas Meade, Dongchan Shin, Amirhossein Kazemnejad, Gaurav Kamath, Marius Mosbach, Karolina Stanczak, and Siva Reddy. Deepseek-r1 thoughtology: Let's think about LLM reasoning. *Trans. Mach. Learn. Res.*, 2026, 2026. URL https://openreview.net/forum?id=BZwKsiRnJI.

[16] Shu Yang, Junchao Wu, Xin Chen, Yunze Xiao, Xinyi Yang, Derek F. Wong, and Di Wang. Understanding aha moments: from external observations to internal mechanisms, 2025. URL https://arxiv.org/abs/2504.02956.

[17] Jeonghye Kim, Xufang Luo, Minbeom Kim, Sangmook Lee, Dongsheng Li, and Yuqing Yang. Understanding reasoning in llms through strategic information allocation under uncertainty, 2026. URL https://arxiv.org/abs/2603.15500.

[18] Shitian Zhao, Shaoheng Lin, Ming Li, Haoquan Zhang, Wenshuo Peng, Kaipeng Zhang, and Chen Wei. Pyvision-rl: Forging open agentic vision models via RL. *arxiv*, 2602.20739, 2026. URL https://doi.org/10.48550/arXiv.2602.20739.

[19] Qwen Team. Qwen3-vl technical report. *arXiv*, 2511.21631, 2025. URL https://doi.org/10.48550/arXiv.2511.21631.

[20] Minki Kang, Jongwon Jeong, Seanie Lee, Jaewoong Cho, and Sung Ju Hwang. Distilling LLM agent into small models with retrieval and code tools. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=VkicTqszOn.

[21] Stéphane Ross, Geoffrey J. Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In Geoffrey J. Gordon, David B. Dunson, and Miroslav Dudík, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2011, Fort Lauderdale, USA, April 11-13, 2011*, JMLR Proceedings, pages 627–635. JMLR.org, 2011. URL http://proceedings.mlr.press/v15/ross11a/ross11a.pdf.

[22] Joykirat Singh, Raghav Magazine, Yash Pandya, and Akshay Nambi. Agentic reasoning and tool integration for llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2505.01441.

[23] Jack Hong, Chenxiao Zhao, ChengLin Zhu, Weiheng Lu, Guohai Xu, and Xing Yu. Deepeyesv2: Toward agentic multimodal model, 2025. URL https://arxiv.org/abs/2511.05271.

[24] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv*, 2402.03300, 2024. URL https://doi.org/10.48550/arXiv.2402.03300.

[25] Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, 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, Honglin Yu, Weinan Dai, Yuxuan Song, Xiang Wei, Haodong Zhou, Jingjing Liu, Wei Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yong-Xu Wu, and Mingxuan Wang. Dapo: An open-source llm reinforcement learning system at scale. *arXiv*, 2503.14476, 2025. URL https://api.semanticscholar.org/CorpusID:277104124.

[26] Mingjie Liu, Shizhe Diao, Ximing Lu, Jian Hu, Xin Dong, Yejin Choi, Jan Kautz, and Yi Dong. ProRL: Prolonged reinforcement learning expands reasoning boundaries in large language models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=YPsJha5HXQ.

[27] Yuxiao Qu, Amrith Setlur, Virginia Smith, Ruslan Salakhutdinov, and Aviral Kumar. POPE: learning to reason on hard problems via privileged on-policy exploration. *arXiv*, 2601.18779, 2026. URL https://doi.org/10.48550/arXiv.2601.18779.

[28] Baohao Liao, Hanze Dong, Xinxing Xu, Christof Monz, and Jiang Bian. Self-hinting language models enhance reinforcement learning. *arXiv*, 2602.03143, 2026. URL https://doi.org/10.48550/arXiv.2602.03143.

[29] 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, 2025. URL https://arxiv.org/abs/2504.14870.

[30] Jian Hu, Mingjie Liu, Ximing Lu, Fang Wu, Zaïd Harchaoui, Shizhe Diao, Yejin Choi, Pavlo Molchanov, June Yang, Jan Kautz, and Yi Dong. Brorl: Scaling reinforcement learning via broadened exploration. *arXiv*, 2510.01180, 2025. URL https://doi.org/10.48550/arXiv.2510.01180.

[31] 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. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=TX4k7BF6a0.

[32] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv*, 1707.06347, 2017. URL http://arxiv.org/abs/1707.06347.

[33] 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, 2026. URL https://arxiv.org/abs/2604.06268.

[34] Yichao Fu, Xuewei Wang, Yuandong Tian, and Jiawei Zhao. Deep think with confidence. *arXiv*, 2025. URL https://arxiv.org/abs/2508.15260.

[35] 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. *arXiv*, 2508.18265, 2025. URL https://doi.org/10.48550/arXiv.2508.18265.

[36] 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. *arXiv*, 2504.08837, 2025. URL https://doi.org/10.48550/arXiv.2504.08837.

[37] Jinming Wu, Zihao Deng, Wei Li, Yiding Liu, Bo You, Bo Li, Zejun Ma, and Ziwei Liu. Mmsearch-r1: Incentivizing lmms to search. 2025. URL https://arxiv.org/abs/2506.20670.

[38] 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. *arXiv*, 2601.21821, 2026. URL https://doi.org/10.48550/arXiv.2601.21821.

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

[40] 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. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=VOAMTA8jKu.

[41] Chengqi Duan, Kaiyue Sun, Rongyao Fang, Manyuan Zhang, Yan Feng, Ying Luo, Yufang Liu, Ke Wang, Peng Pei, Xunliang Cai, Hongsheng Li, Yi Ma, and Xihui Liu. Codeplot-cot: Mathematical visual reasoning by thinking with code-driven images, 2025. URL https://arxiv.org/abs/2510.11718.

[42] Wenbin Wang, Liang Ding, Minyan Zeng, Xiabin Zhou, Li Shen, Yong Luo, Wei Yu, and Dacheng Tao. Divide, conquer and combine: A training-free framework for high-resolution image perception in multimodal large language models. In Toby Walsh, Julie Shah, and Zico Kolter,

editors, *Thirty-Ninth AAAI Conference on Artificial Intelligence, Thirty-Seventh Conference on Innovative Applications of Artificial Intelligence, Fifteenth Symposium on Educational Advances in Artificial Intelligence, AAAI 2025, Philadelphia, PA, USA, February 25 - March 4, 2025*, pages 7907–7915. AAAI Press, 2025. URL https://doi.org/10.1609/aaai.v39i8.32852.

[43] Yong Xien Chng, Tao Hu, Wenwen Tong, Xueheng Li, Jiandong Chen, Haojia Yu, Jiefan Lu, Hewei Guo, Hanming Deng, Xie Chen, Gao Huang, Dahua Lin, and Lewei Lu. Sensenova-mars: Empowering multimodal agentic reasoning and search via reinforcement learning. *arXiv*, 2512.24330, 2025. URL https://doi.org/10.48550/arXiv.2512.24330.

[44] Dongzhi Jiang, Renrui Zhang, Ziyu Guo, Yanmin Wu, Jiayi Lei, Pengshuo Qiu, Pan Lu, Zehui Chen, Guanglu Song, Peng Gao, Yu Liu, Chunyuan Li, and Hongsheng Li. Mmsearch: Benchmarking the potential of large models as multi-modal search engines. *arXiv*, 2409.12959, 2024. URL https://doi.org/10.48550/arXiv.2409.12959.

[45] Alex Su, Haozhe Wang, Weiming Ren, Fangzhen Lin, and Wenhu Chen. Pixel reasoner: Incentivizing pixel space reasoning via curiosity-driven reinforcement learning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=VeZkY3JjWV.

[46] Ziwei Zheng, Michael Yang, Jack Hong, Chenxiao Zhao, Guohai Xu, Le Yang, Chao Shen, and XingYu. Deepeyes: Incentivizing "thinking with images" via reinforcement learning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=xUyMXkI958.

[47] Yuda Song, Lili Chen, Fahim Tajwar, Remi Munos, Deepak Pathak, J. Andrew Bagnell, Aarti Singh, and Andrea Zanette. Expanding the capabilities of reinforcement learning via text feedback, 2026. URL https://arxiv.org/abs/2602.02482.

[48] MiniMax, :, Aili Chen, Aonian Li, Bangwei Gong, Binyang Jiang, Bo Fei, Bo Yang, Boji Shan, Changqing Yu, Chao Wang, Cheng Zhu, Chengjun Xiao, Chengyu Du, Chi Zhang, Chu Qiao, Chunhao Zhang, Chunhui Du, Congchao Guo, Da Chen, Deming Ding, Dianjun Sun, Dong Li, Enwei Jiao, Haigang Zhou, Haimo Zhang, Han Ding, Haohai Sun, Haoyu Feng, Huaiguang Cai, Haichao Zhu, Jian Sun, Jiaqi Zhuang, Jiaren Cai, Jiayuan Song, Jin Zhu, Jingyang Li, Jinhao Tian, Jinli Liu, Junhao Xu, Junjie Yan, Junteng Liu, Junxian He, Kaiyi Feng, Ke Yang, Kecheng Xiao, Le Han, Leyang Wang, Lianfei Yu, Liheng Feng, Lin Li, Lin Zheng, Linge Du, Lingyu Yang, Lunbin Zeng, Minghui Yu, Mingliang Tao, Mingyuan Chi, Mozhi Zhang, Mujie Lin, Nan Hu, Nongyu Di, Peng Gao, Pengfei Li, Pengyu Zhao, Qibing Ren, Qidi Xu, Qile Li, Qin Wang, Rong Tian, Ruitao Leng, Shaoxiang Chen, Shaoyu Chen, Shengmin Shi, Shitong Weng, Shuchang Guan, Shuqi Yu, Sichen Li, Songquan Zhu, Tengfei Li, Tianchi Cai, Tianrun Liang, Weiyu Cheng, Weize Kong, Wenkai Li, Xiancai Chen, Xiangjun Song, Xiao Luo, Xiao Su, Xiaobo Li, Xiaodong Han, Xinzhu Hou, Xuan Lu, Xun Zou, Xuyang Shen, Yan Gong, Yan Ma, Yang Wang, Yiqi Shi, Yiran Zhong, Yonghong Duan, Yongxiang Fu, Yongyi Hu, Yu Gao, Yuanxiang Fan, Yufeng Yang, Yuhao Li, Yulin Hu, Yunan Huang, Yunji Li, Yunzhi Xu, Yuxin Mao, Yuxuan Shi, Yuze Wenren, Zehan Li, Zelin Li, Zhanxu Tian, Zhengmao Zhu, Zhenhua Fan, Zhenzhen Wu, Zhichao Xu, Zhihang Yu, Zhiheng Lyu, Zhuo Jiang, Zibo Gao, Zijia Wu, Zijian Song, and Zijun Sun. Minimax-m1: Scaling test-time compute efficiently with lightning attention, 2025. URL https://arxiv.org/abs/2506.13585.

[49] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

[50] 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. *arXiv*, 2504.11536, 2025. URL https://arxiv.org/abs/2504.11536.

[51] Ke Wang, Houxing Ren, Aojun Zhou, Zimu Lu, Sichun Luo, Weikang Shi, Renrui Zhang, Linqi Song, Mingjie Zhan, and Hongsheng Li. Mathcoder: Seamless code integration in llms for enhanced mathematical reasoning. *arXiv*, 2310.03731, 2023. URL https://arxiv.org/abs/2310.03731.

[52] Minki Kang, Jongwon Jeong, and Jaewoong Cho. T1: Tool-integrated self-verification for test-time compute scaling in small language models, 2025. URL https://arxiv.org/abs/2504.04718.

[53] Google Deepmind. Introducing agentic vision in gemini 3 flash. https://blog.google/innovation-and-ai/technology/developers-tools/agentic-vision-gemini-3-flash/, 2025.

[54] OpenAI. Thinking with images. https://openai.com/index/thinking-with-images/, 2025.

[55] Runqi Qiao, Qiuna Tan, Minghan Yang, Guanting Dong, Peiqing Yang, Shiqiang Lang, Enhui Wan, Xiaowan Wang, Yida Xu, Lan Yang, Chong Sun, Chen Li, Jing Lyu, and Honggang Zhang. V-thinker: Interactive thinking with images, 2025. URL https://arxiv.org/abs/2511.04460.

[56] Ning Shang, Yifei Liu, Yi Zhu, Li Lyna Zhang, Weijiang Xu, Xinyu Guan, Buze Zhang, Bingcheng Dong, Xudong Zhou, Bowen Zhang, Ying Xin, Ziming Miao, Scarlett Li, Fan Yang, and Mao Yang. rstar2-agent: Agentic reasoning technical report. *arXiv*, 2508.20722, 2025. URL https://doi.org/10.48550/arXiv.2508.20722.

[57] 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.

[58] Xuefeng Li, Haoyang Zou, and Pengfei Liu. Torl: Scaling tool-integrated RL. *arXiv*, 2503.23383, 2025. URL https://doi.org/10.48550/arXiv.2503.23383.

[59] Yifan Zhang, Xingyu Lu, Shukang Yin, Chaoyou Fu, Wei Chen, Xiao Hu, Bin Wen, Kaiyu Jiang, Changyi Liu, Tianke Zhang, Haonan Fan, Kaibing Chen, Jiankang Chen, Haojie Ding, Kaiyu Tang, Zhang Zhang, Liang Wang, Fan Yang, Tingting Gao, and Guorui Zhou. Thyme: Think beyond images. *arXiv*, 2508.11630, 2025. URL https://doi.org/10.48550/arXiv.2508.11630.

[60] 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. *arXiv*, 2025. URL https://arxiv.org/abs/2510.14545.

[61] 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. *arXiv preprint arXiv: 2409.19256*, 2024.

[62] Sijun Tan, Michael Luo, Colin Cai, Tarun Venkat, Kyle Montgomery, Aaron Hao, Tianhao Wu, Arnav Balyan, Manan Roongta, Chenguang Wang, Li Erran Li, Raluca Ada Popa, and Ion Stoica. rllm: A framework for post-training language agents. https://pretty-radio-b75.notion.site/rLLM-A-Framework-for-Post-Training-Language-Agents\-21b81902c146819db63cd98a54ba5f31, 2025. Notion Blog.

[63] Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Ming-Hsuan Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-vl technical report. *arXiv*, 2502.13923, 2025. URL https://doi.org/10.48550/arXiv.2502.13923.

[64] Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Yang Yue, Shiji Song, and Gao Huang. Does reinforcement learning really incentivize reasoning capacity in LLMs beyond the base model? In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=4OsgYD7em5.

[65] Tianzhe Chu, Yuexiang Zhai, Jihan Yang, Shengbang Tong, Saining Xie, Dale Schuurmans, Quoc V. Le, Sergey Levine, and Yi Ma. Sft memorizes, rl generalizes: A comparative study of foundation model post-training, 2025. URL https://arxiv.org/abs/2501.17161.

[66] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

[67] Baian Chen, Chang Shu, Ehsan Shareghi, Nigel Collier, Karthik Narasimhan, and Shunyu Yao. Fireact: Toward language agent fine-tuning. *arXiv*, 2310.05915, 2023. URL https://doi.org/10.48550/arXiv.2310.05915.

[68] Feijie Wu, Weiwu Zhu, Yuxiang Zhang, Soumya Chatterjee, Jiarong Zhu, Fan Mo, Rong Luo, and Jing Gao. PORTool: Importance-aware policy optimization with rewarded tree for multi-tool-integrated reasoning, 2026. URL https://arxiv.org/abs/2510.26020v2.

[69] Alexandra Carpentier and Rémi Munos. Adaptive stratified sampling for Monte-Carlo integration of differentiable functions. In *Advances in Neural Information Processing Systems*, volume 25, 2012. URL https://papers.neurips.cc/paper_files/paper/2012/hash/42a0e188f5033bc65bf8d78622277c4e-Abstract.html.

[70] Andrew W. Moore and Christopher G. Atkeson. Prioritized sweeping: Reinforcement learning with less data and less time. *Machine Learning*, 13:103–130, 1993.

[71] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. In *International Conference on Learning Representations*, 2016. URL https://arxiv.org/abs/1511.05952.

[72] Junho Kim, Byung-Kwan Lee, and Yong Man Ro. Distilling robust and non-robust features in adversarial examples by information bottleneck. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 17148–17159. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/8e5e15c4e6d09c8333a17843461041a9-Paper.pdf.

[73] Byung-Kwan Lee, Junho Kim, and Yong Man Ro. Masking adversarial damage: Finding adversarial saliency for robust and sparse network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15126–15136, June 2022.

[74] Junho Kim, Byung-Kwan Lee, and Yong Man Ro. Demystifying causal features on adversarial examples and causal inoculation for robust network by adversarial instrumental variable regression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12302–12312, June 2023.

[75] Byung-Kwan Lee, Junho Kim, and Yong Man Ro. Mitigating adversarial vulnerability through causal parameter estimation by adversarial double machine learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4499–4509, October 2023.

[76] Byung-Kwan Lee, Youngjoon Yu, and Yong Man Ro. Towards adversarial robustness of bayesian neural network through hierarchical variational inference, 2021. URL https://openreview.net/forum?id=Cue2ZEBf12.

[77] Yeonju Kim, Junho Kim, Byung-Kwan Lee, Sebin Shin, and Yong Man Ro. Mitigating dataset bias in image captioning through clip confounder-free captioning network. In *2023 IEEE International Conference on Image Processing (ICIP)*, pages 1720–1724, 2023. doi: 10.1109/ICIP49359.2023.10222502.

[78] Junho Kim, Byung-Kwan Lee, and Yong Man Ro. Causal unsupervised semantic segmentation. *Pattern Recognition*, 171:112173, 2026. ISSN 0031-3203. doi: https://doi.org/10.1016/j.patcog.2025.112173. URL https://www.sciencedirect.com/science/article/pii/S0031320325008349.

A Experimental Setup

This section expands on Section 4.1: training data and hyperparameters (Appendix A.1), the agentic system prompt and tool schemas (Appendix A.2), and per-benchmark descriptions (Appendix A.3).

A.1 Training

We expand on Section 4.1 with the full training data composition and hyperparameter list.

SFT data composition. SFT uses 64,274 trajectories drawn from three sources: ViRL [36], fvqa [37], and PyVision-RL [18]. All trajectories are generated by Qwen3-VL-32B-Thinking [19] acting as the teacher model. $\sim$25% of the trajectories contain at least one tool call; the remainder solve the problem with internal reasoning alone. We retain only trajectories whose final answer is correct against the ground-truth label.

RL data composition. RL uses 15,591 problems obtained by filtering the SFT problem set: we remove problems that the SFT checkpoint solves on all 4 rollouts, and problems that the 32B teacher fails on all 4 rollouts. This filtering convention follows prior agentic RL work [23, 26] and removes both trivial problems (zero learning signal) and unreachable problems (no positive signal achievable at this scale). We additionally include 22,000 hard questions from MMFineReason-hard[3] to expose the policy to problems unseen during SFT, bringing the RL training set to $\sim$37K problems.

Policy gradient objective. Both SFT + GRPO and SFT + AXPO *maximize* the standard PPO-clip surrogate [32] regularized by a KL penalty toward a fixed reference policy π_{ref} (the SFT-initialized checkpoint). Letting $\rho_t(\theta) = \pi_\theta(y_t \mid \boldsymbol{x}, \boldsymbol{y}_{<t}) / \pi_{\theta_{\text{old}}}(y_t \mid \boldsymbol{x}, \boldsymbol{y}_{<t})$ denote the per-token importance ratio against the rollout policy $\pi_{\theta_{\text{old}}}$ and $\hat{A}$ the trajectory-level advantage broadcast uniformly to every token of $\boldsymbol{y}$,

$$\begin{aligned} \mathcal{L}_{\text{clip}}(\boldsymbol{y};\, \hat{A}) \;=\; \mathbb{E}_t\Big[\; & \min\big(\rho_t(\theta)\,\hat{A},\; \text{clip}(\rho_t(\theta),\, 1-\epsilon_{\text{low}},\, 1+\epsilon_{\text{high}})\,\hat{A}\big) \\ & - \beta\, D_{\text{KL}}[\pi_\theta(\cdot \mid \boldsymbol{x}, \boldsymbol{y}_{<t}) \,\|\, \pi_{\text{ref}}(\cdot \mid \boldsymbol{x}, \boldsymbol{y}_{<t})]\;\Big]. \end{aligned} \tag{5}$$

The training objective is $\max_\theta\; \mathbb{E}_{\boldsymbol{y}\sim\pi_{\theta_{\text{old}}}}\Big[\mathcal{L}_{\text{clip}}(\boldsymbol{y};\, \hat{A})\Big]$. GRPO instantiates $\hat{A} = A_i$ for trajectory τ_i from Section 2.1; AXPO applies the same surrogate per stream in Equation 4 (per-prefix advantage on continuations, recovery-indicator advantage on the source prefix). We use the asymmetric DAPO clipping range $\epsilon_{\text{low}} = 0.2$, $\epsilon_{\text{high}} = 0.4$ and KL coefficient $\beta = 1 \times 10^{-3}$.

Hyperparameters. We adopt the ProRL [26] recipe for the RL stage. The key hyperparameters are:

- Sampling temperature: 1.0
- KL coefficient: 1×10^{-3}
- Asymmetric clipping: $\epsilon_{\text{low}} = 0.2$, $\epsilon_{\text{high}} = 0.4$ (DAPO-style [25])
- Group size: $N = 8$ rollouts per question
- Maximum number of turns per trajectory: 3
- Hardware: 8$\times$ A100 80GB GPUs

All hyperparameters are held constant between SFT + GRPO and SFT + AXPO; only the rollout loop differs (see below).

For implementation of the training pipeline, we used verl [61] and rllm [62] libraries.

AXPO-specific hyperparameters. AXPO adds two knobs on top of the GRPO recipe:

- Branching budget: $r = 0.25$ of total rollout (i.e., extra continuations per training step are capped at $0.25 \cdot BN$).
- Per-prefix resample count: $K = 4$ continuations per selected prefix.

[3] A question subset of the MMFineReason corpus that Qwen3-VL-4B entirely fails on 4 rollouts.

A.2 System Prompt and Tool Interface

We use the same agentic system prompt for SFT, RL, and evaluation across all model sizes where the prompt is modified version of the system prompt of deepeyes v2 [23]. Tools are exposed via a unified `<tool_call>` interface used in qwen-agent [19] and the model is free to invoke any subset of the three tools (`python`, `tavily-search`, `image_zoom_in`) on every question. The full text of the prompt is reproduced below.

System Prompt

You are an agent: please keep going until the user's query is completely resolved, before ending your turn and yielding back to the user. Only terminate your turn when you are sure that the problem is solved.
Solve the following problem step by step. In your reasoning process, if the answer cannot be determined, you can write Python code in a Jupyter Notebook to process the image and extract more information from it. The stdout and stderr content, along with the images generated by "`plt.show()`" will be returned to better assist with the user query.
You MUST use the python tool to analyze or transform images whenever it could improve your understanding. This includes but is not limited to zooming in, rotating, adjusting contrast, computing statistics, or isolating features. If you find you lack sufficient knowledge to confidently answer the question, you MUST conduct search to thoroughly seek the internet for information. No matter how complex the query, you will not give up until you find the corresponding information. You can use the tavily-search tool, which will search the web and return relevant information based on your query.
Notes for tools:
image_zoom_in tool

1. **image_zoom_in** crops and magnifies a specific region of the image so you can inspect fine details.
2. When you need to look at a specific region more closely, prefer **image_zoom_in** over writing python code to crop — it is simpler, faster, and returns a high-resolution view directly.
3. After a zoom-in, the cropped image is appended to `_pil_images` in the python tool (the observation will tell you the exact index), so you can further process it with code if needed.

python tool

1. **python** runs in a stateful Jupyter notebook kernel — variables, imports, and state persist across calls.
2. The conversation images are pre-loaded as `_pil_images` (list of PIL Image objects) and `_images` (list of file paths). Images produced by other tools (e.g. `image_zoom_in`) are also appended to these lists.
3. Use `plt.show()` to visualize processed images; generated plots are automatically captured and returned.
4. Available packages include numpy, scipy, sympy, matplotlib, PIL/Pillow, and the standard library.
5. **Always use Python for non-trivial computation** — trigonometric calculations, multi-step arithmetic, equation solving, and area/volume formulas. Do NOT compute these mentally; write code to ensure numerical accuracy.
6. Your program should always return in finite time. Do not write infinite loops.
7. Writing file to disk is not allowed.

tavily-search tool

1. You MUST engage in many interactions, delving deeply into the topic to explore all possible aspects until a satisfactory answer is found.
2. Before presenting a Final Answer, you will **cross-check** and **validate the information** you've gathered to confirm its accuracy and reliability.
3. You will carefully analyze each information source to ensure that all data is current, relevant, and from credible origins.

general

> You MUST think and plan before each function call, and reflect on the outcomes of the previous function calls. DO NOT do this entire process by making function calls only, as this can impair your ability to solve the problem and think insightfully.
> Remember to place the final answer in the last part using the format:
>
> <answer>
> \boxed{"The final answer goes here."}
> </answer>

Tool schemas. Each tool is registered with the model using the OpenAI function-calling schema (`name`, `description`, JSON `parameters`) and is concatenated with the system prompt above at every step. The three schemas in full are below.

`python`.

```
{
  "type": "function",
  "function": {
    "name": "python",
    "description": "Execute Python code in a persistent Jupyter kernel. State
        ↪  (variables, imports) is preserved across calls. Returns stdout/stderr
        ↪  and captured images. Conversation images are pre-loaded as
        ↪  `_pil_images` (list of PIL.Image) and `_images` (list of file paths).
        ↪  Use `_pil_images[0]` to access the first image directly - do NOT use
        ↪  plt.imread() or open files manually.",
    "parameters": {
      "type": "object",
      "properties": {
        "code": {
          "type": "string",
          "description": "Python code to execute in the sandbox environment."
        },
        "timeout": {
          "type": "integer",
          "description": "Maximum execution time in seconds before timing out",
          "default": 12
        }
      },
      "required": ["code"]
    }
  }
}
```

`image_zoom_in`.

```
{
  "type": "function",
  "function": {
    "name": "image_zoom_in",
    "description": "Zoom in on a specific region of an image by cropping it based
        ↪  on a bounding box (bbox) and an optional object label",
    "parameters": {
      "type": "object",
      "properties": {
        "bbox_2d": {
          "type": "array",
          "items": {"type": "number"},
          "minItems": 4,
          "maxItems": 4,
          "description": "The bounding box of the region to zoom in, as [x1, y1,
              ↪  x2, y2] in 0-1000 relative coordinates, where (x1, y1) is the
              ↪  top-left corner and (x2, y2) is the bottom-right corner"
        },
        "label": {
```

```
          "type": "string",
          "description": "The name or label of the object in the specified bounding
              ↪ box"
        },
        "img_idx": {
          "type": "number",
          "description": "The index of the image to zoom in on (starting from 0)",
          "default": 0
        }
      },
      "required": ["bbox_2d", "label"]
    }
  }
}
```

`tavily-search`.

```
{
  "type": "function",
  "function": {
    "name": "tavily-search",
    "description": "Search the web for information on a specific query",
    "parameters": {
      "type": "object",
      "properties": {
        "query": {
          "type": "string",
          "description": "The search query"
        },
        "max_results": {
          "type": "integer",
          "description": "Maximum number of search results to return. Defaults to
              ↪ 5."
        }
      },
      "required": ["query"]
    }
  }
}
```

Search domain blacklist. To prevent answer leakage during training and evaluation, `tavily-search` is configured with the API-level parameter `exclude_domains = ["huggingface.co"]`, which filters out the site most likely to index benchmark questions or solutions. The exclusion is enforced uniformly across training and evaluation rollouts.

A.3 Benchmark Details

We evaluate on nine multimodal benchmarks grouped by which tool family is most expected to help: **Reasoning** (Python interpreter), **Perception** (image zoom-in), and **Search** (web search). All benchmarks are evaluated in the agentic setting with the same tool interface used in training; the model is free to use any subset of the three tools or none at all on every question.

Reasoning. These benchmarks evaluate multi-step reasoning over text, math, and figures, where a Python interpreter is the natural tool for symbolic and numerical computation.

- **MathVision** [39]: competition-style math problems paired with diagrams (geometry, algebra, combinatorics). Answers are short numerical or symbolic expressions; we use the public mini test set containing 304 questions.
- **DynaMath** [40]: a robustness benchmark in which each seed problem is rendered with multiple parametric variants (e.g., differing numerical values, axis ranges). The benchmark stress-tests whether a model has learned the underlying procedure rather than memorizing one instance; we evaluate on all variants.

- **Math-VR** [41]: a mathematical visual reasoning benchmark introduced with CodePlot-CoT, in which problems are paired with diagrams (figures, plots, geometric constructions) that the model must inspect to compute the answer. The benchmark targets the regime where the visual content is load-bearing for the math derivation rather than decorative.

Perception. These benchmarks contain high-resolution images with information that is hard to read at the default vision encoder resolution. The image zoom-in tool is the natural recovery mechanism.

- **V*** [8]: a fine-grained visual-search benchmark in which the answer depends on a small object embedded in a complex scene. Designed to expose the failure of monolithic image encoders on high-resolution detail.
- **VisualProbe** [7]: a multimodal probing benchmark introduced with mini-o3, where each question targets a specific small region (text, sign, label) within a high-resolution scene. We use the *hard* subset, which selects questions Qwen3-VL-Thinking-8B fails without tools.
- **HR-Bench-4K and HR-Bench-8K** [42]: high-resolution single-image perception at 4096^2 and 8192^2 resolutions, respectively. Questions probe small text, distant objects, and fine textures; the two splits share a common question pool with images rescaled to the target resolution.

Search. These benchmarks require external knowledge that is unlikely to be in the model's parametric memory or that requires multi-hop lookup. The Tavily web search API is the natural tool.

- **HR-MMSearch** [43]: high-resolution multimodal search questions whose answers are composed by chaining two or more web lookups (e.g., "in which year was the mayor of the city where this fashion designer was born"). The benchmark stresses both perception (identifying entities from the image) and multi-hop search composition.
- **MMSearch** [44]: an earlier multimodal search benchmark covering current-events questions whose answers depend on web content released after the model's training cutoff. Predominantly single-hop, but answers are often time-sensitive.

Evaluation protocol. For all benchmarks, we draw four rollouts per question and report Pass@1 (per-question accuracy averaged over four rollouts) and Pass@4 (a question is correct if any of the four rollouts is correct). Final-answer matching against the gold label is performed by an exact match for math and multiple-choice question and F1 score for search questions. For benchmarks that are hard to evaluate with exact match (e.g., MMSearch), we use a llm-as-a-judge using gpt-5-mini model.

Evaluation hyperparameters.

- Sampling: temperature 0.7, top-p 0.8, top-k 20, presence penalty 1.5
- Rollouts per question: $N = 4$ (used for Pass@1 / Pass@4)
- Maximum agentic turns per trajectory: 5
- Maximum response length: $32{,}768$ tokens

B Additional Experimental Results

We report tool utilization at evaluation (Appendix B.1) and per-benchmark Pass@4 for Qwen3-VL-Thinking and InternVL3.5-8B (Appendix B.2). We then quantify Pass@1 variation across evaluation rollouts (Appendix B.3) and report results across training seeds and paired significance tests (Appendix B.4). The remaining comparisons cover inference-time tool access without post-training (Appendix B.5), transfer to an unseen image-search tool (Appendix B.6), prior agentic VLMs (Appendix B.7), and end-to-end training cost (Appendix B.8).

B.1 Tool Utilization Rates at Evaluation

Table 5 reports tool utilization (the fraction of evaluation rollouts that emit at least one `<tool_call>`) on each of the nine benchmarks, mirroring the structure of Table 1. Across all three trainable scales (2B/4B/8B), SFT + AXPO produces the highest average tool utilization among trained methods

Table 5: **Tool utilization rate (%, ↑)** per benchmark, mirroring Table 1. Tool utilization is the fraction of evaluation rollouts that emit at least one `<tool_call>` on each benchmark. Non-agent rollouts (no tools) have tool utilization 0 by construction and are omitted; "Base" is the inference-only baseline with tools enabled. Best per column within each size block in **bold**.

	Reasoning			Perception				Search		Avg.
Method	Math Vision	Dyna Math	Math -VR	V*	Visual Probe	HRBen. -4K	HRBen. -8K	HR-MM Search	MM Search	
Qwen3-VL-2B-Thinking										
Base	78.3	86.4	76.7	94.8	99.1	87.5	93.0	92.8	97.3	89.5
GRPO	**96.1**	**99.0**	**93.2**	96.9	100.0	90.0	**94.0**	**100.0**	**100.0**	**96.6**
SFT	38.8	34.3	8.0	95.8	100.0	79.5	80.5	91.5	95.0	69.3
+ GRPO	54.6	30.7	17.1	90.1	100.0	72.0	80.5	94.1	97.0	70.7
+ **AXPO** *(Ours)*	63.8	63.9	27.1	**99.0**	100.0	**93.0**	93.0	98.7	99.0	81.9
Qwen3-VL-4B-Thinking										
Base	25.7	17.0	1.6	94.8	99.1	79.5	85.5	85.2	95.7	64.9
GRPO	2.3	2.4	0.0	65.4	90.6	36.5	42.5	35.1	63.0	37.5
SFT	53.9	38.9	10.6	94.8	100.0	75.5	83.5	93.1	97.7	72.0
+ GRPO	38.8	27.7	8.6	80.6	100.0	64.0	74.5	91.5	95.7	64.6
+ **AXPO** *(Ours)*	**73.0**	**96.2**	**13.3**	**100.0**	100.0	**100.0**	**100.0**	**100.0**	**100.0**	**86.9**
Qwen3-VL-8B-Thinking										
Base	62.2	50.7	22.5	94.8	100.0	84.5	85.0	94.4	**98.7**	77.0
GRPO	38.2	22.0	7.8	87.4	99.1	72.0	75.5	91.1	97.3	65.6
SFT	47.7	34.1	9.8	90.6	99.1	74.0	76.0	93.4	96.7	69.0
+ GRPO	46.1	30.7	7.6	92.1	99.1	73.0	78.0	97.0	97.7	69.0
+ **AXPO** *(Ours)*	**75.7**	**63.9**	**29.1**	**99.0**	100.0	**90.0**	**92.0**	**98.4**	98.3	**82.9**
Qwen3-VL-32B-Thinking (inference-only baseline)										
Base	51.6	36.5	15.1	94.8	100.0	75.0	77.5	94.8	97.3	71.4

(81.9 / 86.9 / 82.9), and the gap to SFT + GRPO grows with size only on benchmarks where tool use is load-bearing (e.g., MathVision, DynaMath, Math-VR at 8B: $+30/+33/+22$ pp), while on tool-saturated benchmarks (V*, VisualProbe, MMSearch) both methods sit near 100%. Thus AXPO improves both tool-use frequency *and*, as Figure 5b suggests, conditional tool-use quality. The matched-tool analysis does not rule out frequency effects but rather shows that gains remain even on the subset of questions where both SFT + GRPO and SFT + AXPO invoke the same tool family.

Higher tool use persists at evaluation. At evaluation, where resampling is disabled, the 8B tool-use rate across the nine benchmarks is 69.0% for both SFT and SFT + GRPO, and 82.9% for SFT + AXPO (Table 5).

B.2 Per-Benchmark Pass@4 Results

The main manuscript reports Pass@1 in Table 1; Table 6 reports the corresponding Pass@4 numbers across the same nine benchmarks and four model sizes, with identical method ordering and bold/underline conventions. The two tables share an Average column, which is the basis for the headline numbers in Section 4.2 and the per-stage decomposition in Appendix C.3.

Table 7 breaks down the InternVL Pass@4 results by benchmark. DynaMath and HRBench-8K are the two benchmarks where AXPO trails GRPO on Pass@4. The paired-bootstrap intervals in Table 10 exclude zero for both aggregate metrics.

B.3 Pass@1 Variance Across Independent Rollouts

Table 1 reports Pass@1 averaged over four rollouts per question for compactness. To quantify rollout-induced variance, we re-evaluate every (model, benchmark) cell with **four independent evaluation rollouts** (each itself an average over four rollouts per question, matching the main-table protocol) and report the resulting mean ± std in Table 8. Per-cell std values fall in the 0.4–3.6 pp range for nearly all entries, with isolated outliers on the smallest (2B) and most variance-prone (HRBench-8K) settings. These standard deviations describe evaluation-rollout variability. Training-seed results and paired per-question significance tests are reported separately in Appendix B.4.

Table 6: **Main results: Pass@4** (%, ↑) on **nine multimodal benchmarks**, grouped into three categories: Reasoning, Perception, and Search. All methods use **Qwen3-VL-Thinking** [19] at four sizes (2B, 4B, 8B, 32B). Base (tools, no training) is the inference-only baseline; the Non-agent (no tools) baseline is reported separately in Table 11. Best result per column within each size block is **bolded**; second-best is underlined. Δ rows show AXPO's gain over SFT + GRPO, the strongest comparable training recipe where AXPO outperforms it in most of evaluations. Pass@1 results are reported in Table 1.

	Reasoning			Perception				Search		Avg.
Method	Math Vision	Dyna Math	Math -VR	V*	Visual Probe	HRBen. -4K	HRBen. -8K	HR-MM Search	MM Search	
Qwen3-VL-2B-Thinking										
Base	32.6	76.6	28.9	81.7	15.1	79.0	77.0	18.7	46.3	50.7
GRPO	53.3	78.2	52.0	80.6	43.4	75.5	72.5	29.5	54.7	60.0
SFT	53.0	80.2	49.2	91.1	50.0	86.5	**88.0**	28.5	53.7	64.5
+ GRPO	55.9	80.2	58.0	**94.2**	**60.4**	83.5	85.5	28.2	55.7	66.8
+ **AXPO** *(Ours)*	**61.2**	**83.6**	**60.0**	**94.2**	**60.4**	**88.5**	**88.0**	**33.4**	**57.3**	**69.6**
Δ *vs. SFT + GRPO*	+5.3	+3.4	+2.0	+0.0	+0.0	+5.0	+2.5	+5.2	+1.6	+2.8
Qwen3-VL-4B-Thinking										
Base	64.8	82.6	65.1	95.8	44.3	83.5	81.0	27.5	55.7	66.7
GRPO	69.1	84.2	70.5	92.7	48.1	87.0	85.5	22.0	42.0	66.8
SFT	69.4	86.4	69.5	95.3	51.9	89.5	88.0	**35.7**	**59.7**	71.7
+ GRPO	69.7	85.8	71.9	**97.9**	57.5	88.0	84.5	33.1	57.0	71.7
+ **AXPO** *(Ours)*	**74.3**	**87.0**	**72.1**	**97.9**	**65.1**	**91.0**	**90.0**	32.1	57.0	**74.1**
Δ *vs. SFT + GRPO*	+4.6	+1.2	+0.2	+0.0	+7.6	+3.0	+5.5	−1.0	+0.0	+2.3
Qwen3-VL-8B-Thinking										
Base	63.5	85.2	68.3	92.7	55.7	85.5	82.5	33.8	59.0	69.6
GRPO	69.7	84.6	70.3	94.2	62.3	87.0	82.0	34.4	60.0	71.6
SFT	71.4	86.0	72.1	93.7	57.5	**90.5**	87.0	36.4	55.3	72.2
+ GRPO	75.0	**87.0**	**74.3**	**96.9**	63.2	87.5	84.5	38.0	59.3	74.0
+ **AXPO** *(Ours)*	**75.7**	**87.0**	74.1	95.3	**67.9**	90.0	**89.0**	**42.0**	**61.0**	**75.8**
Δ *vs. SFT + GRPO*	+0.7	+0.0	−0.2	−1.6	+4.7	+2.5	+4.5	+4.0	+1.7	+1.8
Qwen3-VL-32B-Thinking (inference-only baseline)										
Base	73.0	89.4	73.9	97.4	64.2	90.5	90.0	38.0	59.3	75.1

Table 7: **Generalization to InternVL3.5-8B: Pass@4** (%, ↑) on the same nine benchmarks as Table 1. Best result per column is **bolded**, and second-best is underlined. Δ rows report AXPO minus SFT + GRPO (pp). Pass@1 results are in Table 2.

	Reasoning			Perception				Search		Avg.
Method	Math Vision	Dyna Math	Math -VR	V*	Visual Probe	HRBen. -4K	HRBen. -8K	HR-MM Search	MM Search	
InternVL3.5-8B										
Base	57.6	79.0	49.2	63.9	28.3	60.0	56.5	22.6	39.7	50.8
SFT	67.1	**88.2**	67.3	87.4	40.6	84.0	79.0	**34.4**	54.3	66.9
+ GRPO	68.8	86.6	66.7	88.5	47.2	84.5	**83.5**	28.5	53.3	67.5
+ **AXPO** *(Ours)*	**71.1**	86.2	**70.1**	**92.7**	**50.0**	**85.0**	81.0	29.2	**58.7**	**69.3**
Δ *vs. SFT + GRPO*	+2.3	−0.4	+3.4	+4.2	+2.8	+0.5	−2.5	+0.7	+5.4	+1.8

B.4 Training Seeds and Paired Significance Tests

The aggregate gain persists across training seeds. To assess variation across training runs, we add two training seeds for SFT + GRPO and SFT + AXPO on Qwen3-VL-8B-Thinking. Table 9 reports macro-averages over the nine benchmarks. AXPO improves both metrics in every seed. The three-seed means are 60.7% versus 59.4% on Pass@1 and 74.2% versus 73.0% on Pass@4.

Paired evaluation supports the aggregate gains. We pair each question's SFT + AXPO and SFT + GRPO outcomes and bootstrap over evaluation questions, reporting the difference macro-averaged over the nine benchmarks. We use 10,000 bootstrap resamples for the three Qwen3-VL-Thinking scales. Table 10 gives the gains and their 95% confidence intervals. All intervals exclude zero across the three Qwen3-VL-Thinking scales and InternVL3.5-8B. These question-level comparisons complement the training-seed results, which measure variability across separately trained policies, and the rollout-variance results in Appendix B.3.

Table 8: **Pass@1 mean ± std** (%, ↑) across **four independent rollouts** on nine multimodal benchmarks. Subscripts denote standard deviation across the four rollouts. Variance is comparable across methods, with most std values in the 0.4–3.6 pp range; the column-averaged std for SFT + AXPO is 1.4 pp at 4B and 1.2 pp at 8B.

	Reasoning			Perception				Search		Avg.
Method	Math Vision	Dyna Math	Math -VR	$V^\star$	Visual Probe	HRBen. -4K	HRBen. -8K	HR-MM Search	MM Search	
Qwen3-VL-2B-Thinking										
Base	18.6 ± 1.8	60.5 ± 1.2	16.9 ± 1.5	63.0 ± 2.9	5.0 ± 1.4	64.6 ± 0.9	55.9 ± 0.5	8.2 ± 1.5	27.7 ± 1.4	35.6 ± 1.5
GRPO	33.7 ± 1.0	64.9 ± 0.8	34.6 ± 1.0	67.1 ± 1.8	23.8 ± 3.4	65.0 ± 1.3	59.4 ± 1.8	18.4 ± 1.3	41.4 ± 0.5	45.4 ± 1.4
SFT	28.9 ± 1.3	64.1 ± 1.6	30.2 ± 1.0	75.9 ± 2.4	24.5 ± 2.8	73.1 ± 1.1	68.8 ± 1.6	14.5 ± 1.7	35.9 ± 1.4	46.2 ± 1.7
+ GRPO	34.8 ± 1.3	66.4 ± 1.0	35.8 ± 2.2	81.7 ± 3.2	36.1 ± 1.5	74.2 ± 0.8	70.6 ± 2.1	15.7 ± 1.0	40.6 ± 1.2	50.7 ± 1.6
+ **AXPO** *(Ours)*	35.0 ± 1.1	66.9 ± 2.2	38.9 ± 1.0	81.3 ± 3.1	35.1 ± 5.0	77.9 ± 1.9	72.4 ± 1.6	18.1 ± 0.4	40.2 ± 1.7	51.8 ± 2.0
Qwen3-VL-4B-Thinking										
Base	48.0 ± 1.2	76.6 ± 0.7	50.8 ± 0.5	80.6 ± 1.2	24.8 ± 2.3	73.6 ± 1.2	67.5 ± 2.0	16.9 ± 0.4	37.8 ± 1.5	53.0 ± 1.2
GRPO	52.5 ± 2.3	76.2 ± 1.0	57.9 ± 1.1	82.7 ± 2.8	29.7 ± 1.9	75.5 ± 1.3	71.1 ± 1.5	12.7 ± 1.2	23.8 ± 1.7	53.6 ± 1.6
SFT	49.1 ± 1.3	76.3 ± 1.0	55.3 ± 1.2	87.0 ± 1.9	34.7 ± 1.7	79.5 ± 1.3	74.1 ± 1.0	20.6 ± 1.7	41.5 ± 1.5	57.6 ± 1.4
+ GRPO	51.6 ± 0.6	77.2 ± 0.4	57.4 ± 1.2	90.2 ± 1.1	38.0 ± 2.1	78.8 ± 0.8	73.8 ± 1.5	20.0 ± 1.8	42.3 ± 0.8	58.8 ± 1.2
+ **AXPO** *(Ours)*	52.8 ± 1.4	76.4 ± 0.5	57.2 ± 0.9	88.9 ± 1.1	43.6 ± 2.6	80.9 ± 1.7	78.2 ± 1.4	20.1 ± 1.3	43.3 ± 1.4	60.2 ± 1.4
Qwen3-VL-8B-Thinking										
Base	47.1 ± 2.0	75.9 ± 0.9	54.9 ± 1.1	77.7 ± 2.2	31.8 ± 1.8	72.8 ± 1.7	66.1 ± 2.2	21.0 ± 1.0	42.7 ± 1.7	54.4 ± 0.6
GRPO	54.4 ± 0.7	77.2 ± 0.4	58.9 ± 1.4	85.7 ± 1.3	40.1 ± 3.2	77.1 ± 2.2	73.9 ± 0.5	20.6 ± 1.7	45.1 ± 0.6	59.2 ± 1.3
SFT	50.9 ± 0.7	76.8 ± 0.6	58.4 ± 1.9	84.8 ± 2.0	38.4 ± 3.1	79.5 ± 0.6	74.4 ± 2.3	23.0 ± 1.6	40.6 ± 2.2	58.5 ± 1.7
+ GRPO	55.3 ± 1.8	78.2 ± 1.0	60.4 ± 1.5	87.7 ± 2.4	40.1 ± 3.6	79.5 ± 1.1	74.9 ± 1.1	24.4 ± 1.2	44.0 ± 0.4	60.5 ± 0.5
+ **AXPO** *(Ours)*	56.1 ± 1.4	79.0 ± 0.9	60.6 ± 0.8	87.8 ± 0.4	45.8 ± 1.1	83.2 ± 0.6	77.0 ± 4.3	25.9 ± 0.7	45.0 ± 0.4	62.3 ± 1.2
Qwen3-VL-32B-Thinking (inference-only baseline)										
Base	56.5 ± 1.4	83.3 ± 0.6	64.1 ± 1.1	89.1 ± 1.1	40.3 ± 2.4	85.2 ± 1.6	78.9 ± 3.7	22.8 ± 1.2	46.1 ± 1.6	62.9 ± 1.6

Table 9: **Three training seeds on Qwen3-VL-8B-Thinking.** Scores are macro-averages over nine benchmarks. GRPO and AXPO share the SFT-then-RL pipeline. Δ is AXPO minus GRPO (pp).

	Pass@1 (%)			Pass@4 (%)		
Seed	GRPO	AXPO	Δ	GRPO	AXPO	Δ
Original	60.5	62.3	+1.8	74.0	75.8	+1.8
Seed 2	59.0	59.7	+0.7	72.7	73.2	+0.5
Seed 3	58.6	60.0	+1.4	72.4	73.5	+1.1
Mean	59.4	60.7	+1.3	73.0	74.2	+1.2

Table 10: **Paired per-question bootstrap results.** Gains and 95% confidence intervals are in percentage points for SFT + AXPO minus SFT + GRPO, macro-averaged over nine benchmarks. These intervals quantify evaluation-question uncertainty for the evaluated checkpoints.

Model	Δ Pass@1	95% CI	Δ Pass@4	95% CI
Qwen3-VL-2B-Thinking	+1.09	[0.04, 2.15]	+2.79	[1.10, 4.53]
Qwen3-VL-4B-Thinking	+1.37	[0.27, 2.48]	+2.34	[0.86, 3.83]
Qwen3-VL-8B-Thinking	+1.76	[0.77, 2.79]	+1.80	[0.28, 3.31]
InternVL3.5-8B	+2.3	[1.21, 3.43]	+1.8	[0.09, 3.52]

B.5 Inference-Time Tool Access without Post-Training

How much does tool access help before post-training? For each Qwen3-VL-Thinking size, we evaluate the same checkpoint in two settings. Non-agent answers using internal reasoning alone, and Base has access to the same tools as the trained models. Table 11 reports Pass@1 and Pass@4 on the nine benchmarks.

Tool access improves average Pass@4 at all four sizes. Average Pass@1 improves at 4B, 8B, and 32B and falls by 1.3 pp at 2B. Both search benchmarks improve on both metrics at every size. For example, at 8B, Pass@1 rises from 15.3% to 42.7% on MMSearch and falls from 52.5% to 47.1% on MathVision. These results establish what the base models achieve with tools before the post-training stages analyzed in Appendix C.3.

Table 11: **Non-agent vs Base**: effect of turning tools on without any training, across nine multimodal benchmarks and four model sizes. *Non-agent* disables tool calls (the model answers from internal reasoning alone). *Base* enables the same tool interface as the trained methods but receives no post-training. Both Pass@1 and Pass@4 (%, ↑) are reported. The Pass@1 / Pass@4 deltas isolate the marginal value of inference-time tool access at each scale, motivating the post-training pipeline analyzed in Appendix C.3.

	Reasoning			Perception				Search		Avg.
Method	Math Vision	Dyna Math	Math -VR	V*	Visual Probe	HRBen. -4K	HRBen. -8K	HR-MM Search	MM Search	
Pass@1										
Qwen3-VL-2B-Thinking										
Non-agent	27.9	66.5	24.8	64.8	18.2	57.8	57.3	5.1	9.5	36.9
Base	18.6	60.5	16.9	63.0	5.0	64.6	55.9	8.2	27.7	35.6
Qwen3-VL-4B-Thinking										
Non-agent	44.8	76.1	45.9	79.6	21.9	74.4	68.1	6.1	13.3	47.8
Base	48.0	76.6	50.8	80.6	24.8	73.6	67.5	16.9	37.8	53.0
Qwen3-VL-8B-Thinking										
Non-agent	52.5	75.5	50.6	78.0	27.6	70.8	68.1	8.5	15.3	49.7
Base	47.1	75.9	54.9	77.7	31.8	72.8	66.1	21.0	42.7	54.4
Qwen3-VL-32B-Thinking										
Non-agent	61.2	84.2	64.2	84.4	24.8	81.4	74.4	11.2	19.8	56.2
Base	56.5	83.3	64.1	89.1	40.3	85.2	78.9	22.8	46.1	62.9
Pass@4										
Qwen3-VL-2B-Thinking										
Non-agent	43.8	79.4	39.4	77.5	31.1	73.0	75.5	10.2	16.3	49.6
Base	32.6	76.6	28.9	81.7	15.1	79.0	77.0	18.7	46.3	50.7
Qwen3-VL-4B-Thinking										
Non-agent	62.0	82.6	60.4	89.0	31.1	83.5	78.5	9.8	20.3	57.5
Base	64.8	82.6	65.1	95.8	44.3	83.5	81.0	27.5	55.7	66.7
Qwen3-VL-8B-Thinking										
Non-agent	70.7	85.0	63.7	85.3	38.7	79.0	77.0	16.4	22.7	59.8
Base	63.5	85.2	68.3	92.7	55.7	85.5	82.5	33.8	59.0	69.6
Qwen3-VL-32B-Thinking										
Non-agent	78.9	91.2	75.3	91.6	33.0	88.5	80.5	16.4	30.0	65.0
Base	73.0	89.4	73.9	97.4	64.2	90.5	90.0	38.0	59.3	75.1

Table 12: **Generalization to an unseen tool.** MMSearch Pass@1 / Pass@4 (%, ↑), with and without an image-search tool added at inference time. Image search is not in the training tool set (Python, Tavily web search, image zoom-in). AXPO converts the new tool into Pass@1 gains more effectively than GRPO (+4.6 pp vs. +2.8 pp), while Pass@4 gains are similar across the two methods.

		SFT + GRPO	SFT + AXPO
w/o image search	Pass@1	44.0	**45.0**
	Pass@4	59.3	**61.0**
w/ image search	Pass@1	46.8	**49.6**
	Pass@4	65.3	**66.7**

B.6 Generalization to an Unseen Tool: Image Search

Training in Section 4.1 uses three tool families: a Python interpreter, a Tavily web search engine, and an image zoom-in tool. None of these is an *image-based* retrieval tool that takes an image as the query and returns visually similar pages, which we cannot directly utilize due to excessive API costs. To probe whether AXPO's tool-using policy transfers beyond the training tool inventory, we extend the inference-time tool set with an image-search tool and re-evaluate on **MMSearch** [44], the search benchmark with the largest fraction of image-grounded queries. Since we did not have access to a production image-retrieval endpoint at the budget required for full evaluation, we approximate the tool by prompting GPT-5.4 with the query image and instructing it to return the kind of structured response a visual-search API would emit (caption, candidate entity names, related URLs). This proxy

Table 13: Comparison with prior agentic VLM systems on five benchmarks where numbers are publicly available. All numbers are Pass@1 (%, ↑). "–" indicates the method does not report on that benchmark. Average is computed only over methods with all five entries. Best per column is **bolded**; second-best is underlined. Prior agentic systems use different base models (mostly Qwen2.5-VL-7B), so absolute values are not strictly comparable; the Δ *vs. base* rows report each group's best improvement over its own base, isolating the post-training gain from the base-model gap. AXPO delivers a larger average lift over its base (+8.7 pp) than the strongest prior agentic recipe over Qwen2.5-VL-7B (+6.9 pp).

Method	MathVision	DynaMath	V*	HR-Bench-4K	HR-Bench-8K	**Avg.**
Qwen2.5-VL-7B base + prior agentic methods						
Qwen2.5-VL-7B-Instruct [63]	25.6	53.3	78.5	71.6	67.9	59.4
PixelReasoner [45]	–	–	84.3	74.0	66.9	–
Mini-o3 [7]	–	–	88.2	77.5	73.3	–
DeepEyes [46]	26.6	55.0	85.6	75.1	72.6	63.0
Thyme [59]	27.6	–	82.2	77.0	72.0	–
DeepEyes-v2 [23]	28.9	57.2	81.8	77.9	73.8	–
PyVision-RL [18]	28.7	61.6	**88.7**	78.1	74.3	66.3
Δ *vs. base (best prior)*	*+3.3*	***+8.3***	***+10.2***	*+6.5*	*+6.4*	*+6.9*
Qwen3-VL-8B-Thinking base + ours						
Qwen3-VL-8B-Thinking (Agent) [19]	47.1	75.9	77.7	72.8	66.1	67.9
SFT + AXPO *(Ours)*	**56.1**	**79.0**	87.8	**83.3**	**77.0**	**76.6**
Δ *vs. base (ours)*	***+9.0***	*+3.1*	*+10.1*	***+10.5***	***+10.9***	***+8.7***

is imperfect, but it preserves the key property the experiment probes: an entirely new tool interface, unseen at training time, whose output the model must integrate into its reasoning.

Table 12 shows the result. *Without* image search at inference, SFT + GRPO and SFT + AXPO are essentially tied (Pass@1: 44.0 vs. 45.0; Pass@4: 59.3 vs. 61.0). *With* image search at inference, the gap opens on Pass@1: AXPO gains +4.6 pp (45.0 → 49.6) while GRPO gains only +2.8 pp (44.0 → 46.8); on Pass@4 both methods gain similarly (+6.0 and +5.7). The Pass@1 asymmetry suggests AXPO not only *can* use the unseen tool when given enough samples (Pass@4 captures this) but converts that capability into reliable first-shot answers, whereas GRPO's tool-using policy is more narrowly fitted to the training tool inventory. We read this as evidence that resampling-induced exposure to diverse tool-call resolutions during training transfers to tool families the model has never been trained on, although the claim is bounded to one benchmark and one new tool type.

B.7 Comparison with Prior Agentic VLMs

Table 13 situates SFT + AXPO against prior agentic VLM systems on the five benchmarks for which their published numbers are available (MathVision, DynaMath, V*, HR-Bench-4K, HR-Bench-8K) from previous work [18, 23]. This comparison is not strictly controlled: prior systems use different base models (mostly Qwen2.5-VL-7B), different training dataset scale, and report on benchmark subsets, so part of any gap reflects base-model improvements rather than method choice. With that caveat, SFT + AXPO at 8B leads on four of the five benchmarks and on the cross-benchmark average (76.6 vs. 66.3 for the strongest prior system reporting on all five). The exception is V*, where pixel-level reasoning systems (Mini-o3, PyVision-RL) sit within ∼1 pp; this is a benchmark where image zoom-in is the load-bearing tool and tool-using rollouts are already near saturation across systems, so the headroom for gradient-rescue is small. The largest gaps appear on math-over-image benchmarks (MathVision, DynaMath), where prior agentic VLM training has invested less and where AXPO's tool-call gradient survival is most directly load-bearing.

B.8 End-to-End Training Cost

Targeted resampling costs less than doubling the rollout count. We measure end-to-end active wall-time on identical 8 A100 GPUs over matched training steps, including external tool execution. We separately estimate model FLOPs from all generated trajectories, including resampled trajectories. Table 14 reports both measurements. Relative to GRPO with $N = 8$, AXPO adds 22.7% wall-time and approximately 9.8% model FLOPs. The rollout-2× control uses 67.8% more model FLOPs than AXPO and takes 26.1 rather than 18.9 minutes per step, while obtaining lower average Pass@1 and Pass@4 in Table 4. The rollout-count budget and measured systems cost are distinct quantities.

Table 14: **Training cost per step on eight A100 GPUs.** Active wall-time includes external tool execution. Estimated model FLOPs include all generated trajectories, including resampled continuations.

Per training step	GRPO $N = 8$	AXPO	GRPO $N = 16$
Active wall-time	15.4 min	18.9 min	26.1 min
Est. model FLOPs	397.6 PFLOPs	436.6 PFLOPs	732.4 PFLOPs

C Additional Analyses

We examine whether the Thinking-Acting Gap also appears on InternVL3.5-8B (Appendix C.1) and how it may depend on the SFT policy (Appendix C.2). We then analyze the stages of the SFT-then-RL pipeline (Appendix C.3), the component ablations including the all-wrong trigger's effect on budget and recovery (Appendix C.4), and alternative RL recipes (Appendix C.5). Finally, we examine recovery and credit assignment on selected prefixes (Appendix C.6) and compare tool-call confidence with exact policy entropy (Appendix C.7).

C.1 Thinking-Acting Gap on InternVL3.5-8B

We measure the quantities from Section 2.2 during GRPO training on InternVL3.5-8B. Tool-using subgroups are all-wrong more often than no-tool subgroups at both the start (49.7% vs. 38.9%) and end (37.1% vs. 29.0%) of training. The rollout-level tool-use rate falls from 25.2% to 8.8%, while the fraction of groups containing no tool-using rollout rises from 51.9% to 77.7%. Thus, both the reliability imbalance and a shortage of tool-using trajectories occur beyond the Qwen3-VL-Thinking family in our evaluated setting.

C.2 Dependence of the Thinking-Acting Gap on the SFT Policy

Across GRPO training rollouts, tool-using and thinking-only rollouts succeed at 39.6% and 51.7%, respectively. The within-prompt comparison in Section 4.5 does not isolate the causal effect of invoking a tool.

The gap depends on the starting policy. We view the Thinking-Acting Gap as a failure mode that can persist during group-based RL after SFT, rather than an immutable property of tool use. A stronger or more tool-rich SFT distribution could increase both tool-use frequency and reliability and reduce the gap. GRPO does not explicitly penalize tool use, but less successful tool-using rollouts receive lower group-normalized advantages, allowing an initial imbalance to persist or worsen. AXPO targets this remaining imbalance during on-policy training. The same mechanism can arise in text-only agents with optional tool use, so we do not claim that it is unique to multimodal reasoning.

C.3 Decomposing the SFT-then-RL Recipe

Table 6 contains four pre-AXPO baselines per model size: Base (tools, no training), GRPO (RL from base), SFT (SFT only), and SFT + GRPO. The Non-agent (no tools, no training) configuration in Table 11 provides a control for the marginal value of inference-time tool access, which is orthogonal to the agentic-RL story. Reading the average Pass@1 / Pass@4 columns across these baselines isolates the contribution of each pipeline stage and motivates the SFT-then-RL recipe that AXPO builds on. All numbers in this section come from the **Average** columns of Table 1 (Pass@1) and Table 6 (Pass@4).

Why is post-training needed at all? Comparing Non-agent and Base at the same model size measures the effect of *turning tools on without any training*. The gain is strongly scale-dependent: Pass@1 / Pass@4 deltas are -1.3 / $+1.1$at 2B, $+5.2$ / $+9.2$at 4B, $+4.7$ / $+9.8$at 8B, and $+6.7$ / $+10.1$at 32B. At 2B the Base (agentic) baseline is actually *worse* than the Non-agent baseline on Pass@1, showing that small models pay an agent-prompt overhead that outweighs the gain from correct tool use. Larger models recover Pass@1 and accumulate a Pass@4 ceiling gain, but even at 32B the gap to SFT + AXPO at 8B on Pass@4 (75.8 vs. 75.1) shows that inference-time tool access alone does not close the agentic gap. Post-training is therefore necessary at every trainable scale.

Why SFT before RL? Comparing GRPO (RL from base) against SFT + GRPO measures the value of an SFT initialization before RL. Pass@1 gaps shrink with size ($+5.3$ at 2B, $+5.2$ at 4B, $+1.3$ at 8B), but Pass@4 gaps remain positive at every scale ($+6.8$, $+4.9$, $+2.4$). RL from the base policy can find a single high-reward mode (Pass@1 partially recovered at 8B), but it consistently sacrifices diversity, capping the Pass@4 ceiling [64]. SFT broadens the cold-start distribution over tool-using trajectories before group-based exploration begins; without it, all-wrong tool-using subgroups dominate from step 1 and the policy collapses onto a narrow single-mode solution [23].

Why RL after SFT? SFT minimizes log-likelihood on a fixed set of teacher trajectories: it can make the student *imitate* the teacher's tool-use patterns, but the optimization signal lives entirely on the teacher's distribution. At test time the student samples from its own distribution and small token-level deviations from the teacher accumulate into prefixes the SFT objective never scored, so imitation alone does not guarantee that on-policy rollouts inherit the teacher's correctness (*exposure bias*). RL after SFT narrows this gap by training on the policy's own rollouts under an outcome reward, adapting the policy to its own induced trajectory distribution rather than to the teacher's [65]. Empirically, comparing SFT against SFT + GRPO isolates this on-policy adaptation effect given the SFT initialization: Pass@1 improves at every scale ($+4.5$ at 2B, $+1.2$ at 4B, $+2.0$ at 8B), the regime where on-policy reliability matters most, while Pass@4 gains are smaller and uneven ($+2.3$, $+0.0$, $+1.8$), consistent with GRPO consolidating mass on trajectories the SFT policy can already reach rather than discovering new ones. RL is therefore necessary on top of SFT, but standard GRPO's adaptation is biased toward the SFT manifold, motivating AXPO's tool-call resampling.

Why AXPO on top of SFT, instead of GRPO? The previous two paragraphs leave a structural gap: SFT sets the Pass@4 ceiling and RL sharpens within it, so further gains require a mechanism that simultaneously extends the ceiling *and* sharpens. AXPO does both. Pass@1 gains over SFT + GRPO are $+1.1$/ $+1.4$/ $+1.8$at 2B/4B/8B, and Pass@4 gains are $+2.8$/ $+2.3$/ $+1.8$. The Pass@4 lift is the diagnostic signature that AXPO operates on a different axis from SFT-then-GRPO sharpening: by recovering all-wrong tool-using subgroups whose gradient signal would otherwise vanish (Section 2.2), AXPO turns previously dead continuations into trainable ones, expanding the policy's reachable correct-trajectory set rather than only redistributing mass within it.

C.4 Detailed Ablation Analysis

Table 3 ablates each AXPO design choice on 8B at the same rollout budget. Pass@1 scores below are averages over MathVision, V*, VisualProbe, and HR-MMSearch.

Prefix fixing. Removing prefix fixing reduces AXPO to 52.1, near the SFT + GRPO baseline of 51.9: each resample becomes a fresh rollout from $\pi_\theta(\cdot \mid \boldsymbol{x})$, so only a q-fraction of resamples are tool-using and the $(1-q)$ waste factor that Proposition 1 removes returns in full.

Tool-using resampling target. Targeting questions where all no-tool rollouts fail for resampling instead reaches 51.7, since the policy already covers the no-tool subgroup under GRPO and additional samples there cannot recover the missing tool-using signal. Resampling pays off only when the resampling target is the under-covered tool-using subgroup.

Uncertainty-based prefix selection. Replacing uncertainty rank with random ranking drops to 51.0 (-2.9 pp): the prefix is still fixed so the coverage gain is preserved, but random ranking routes the budget to prefixes where the policy has already converged toward a wrong continuation, wasting resamples on determined commitments.

Prefix credit. Dropping the recovery-indicator credit on the source prefix yields 51.4 (-2.5 pp): prefix tokens receive no positive signal even when resampling recovers a continuation, so the policy is not steered toward producing high-yield prefixes in the first place.

Per-prefix advantage group. Merging resampled continuations into the source group instead of forming a per-prefix advantage yields 51.2 (-2.7 pp): the shared prefix then collects contradictory advantages from the failed source and successful resamples while no-tool successes in the source group dilute the recovered signal.

The all-wrong trigger directs budget toward missing successes. In the trigger ablation in Table 3, removing the all-wrong trigger sends roughly 50% of the resampling budget to subgroups that already contain a correct tool-using trajectory. Attempts per all-wrong subgroup fall from approximately 8.4 to 5.0, and the number of all-wrong subgroups recovered per step falls from 3.4 to 2.4.

C.5 Detailed Comparison Analysis

Table 4 compares AXPO to alternative training recipes. All numbers are 8B Pass@1 averages.

Reward shaping does not recover the gain of AXPO. A tool penalty [29] drops to 46.0 and a tool bonus [18, 45, 46] reaches only 50.8, both *below* the GRPO baseline of 51.9. Reward-channel interventions either suppress tool use globally (penalty) or reward tool calls regardless of whether they help (bonus); neither addresses the coverage problem that AXPO targets, which is conditional on the prefix being tool-using and on resampling restoring at least one correct continuation.

Extra rollout budget does not close the gap. Doubling the GRPO rollout count reaches 52.4, still 1.5 pp below SFT + AXPO. Its measured wall-time and estimated model FLOPs both exceed AXPO's (Appendix B.8). The extra budget re-derives non-tool-using prefixes and cannot break the all-wrong tool-using subgroups that resampling is designed to recover; throwing more samples at the same $(1-q)$ waste factor does not change the structural problem.

Alternative RL algorithms. We replace GRPO with three recently proposed alternatives. RLTF [47], an algorithm that utilizes the text feedback from the external teacher, reaches only 43.3 Avg with 40% tool utilization, suffering tool-call collapse over training (consistent with reports of hint leakage degrading text-feedback RL on agentic tasks). CISPO [48], an importance-sampling variant, reaches 47.8 Avg: its clip rule reduces gradient bias but does not address the all-wrong subgroup whose advantage is zero by construction. ARPO [31], an agent-specific branching method that draws additional continuations after a tool observation when post-observation entropy is high, is the closest comparison and the strongest competitor at 51.3 Avg, yet still trails AXPO by 2.6 pp: post-observation branching cannot recover when the tool call itself is the failure mode, which is the regime Section 2.2 identifies as binding. The RL-algorithm sweep confirms AXPO's gain is not a generic effect of any agent-aware RL trick but specifically reflects tool-call resampling's coverage restoration on tool-using thinking prefixes.

C.6 Recovery on AXPO-Selected Prefixes

Per-prefix estimates are coarse at four samples. Let $\widehat{\Delta}_i$ be the difference between the estimated continuation success rate for selected prefix i and the estimated from-scratch tool-and-correct rate for its question. The empirical margin is positive for 10.6% of prefixes, negative for 8.3%, and tied for 81.1%. Among the 18.9% non-tied cases, 26.7% have $\widehat{\Delta}_i \geq 0.50$ and 4.0% have $\widehat{\Delta}_i \leq -0.50$. A margin of magnitude 0.50 corresponds to a difference of two successful trajectories out of four. The mean margin is $+2.07$ pp. Because each rate is estimated from only four samples, we use these margins as a coarse description and emphasize the aggregate success-rate comparison.

Recovery credit can reinforce an imperfect prefix. Coverage of a correct final answer does not establish the semantic correctness of the source thinking prefix. We audit all 183 binary-credited recoveries observed in the 8B AXPO training run using GPT-5.6-Sol as an external judge. The judge identifies an apparent error in the source prefix in 19 cases (10.4%). Thus, an imperfect prefix can receive recovery credit, although it is a minority pattern in this audit. Filtering with prefix quality or credit assignment remains a future direction for improving outcome-based recovery.

C.7 Confidence as a Tractable Proxy for Tool-Call Entropy

Section 3 ranks candidate prefixes by the mean probability the source policy assigned to the tool-call tokens of τ^{src} (henceforth *confidence*). The natural alternative is the predictive entropy $H(\pi_\theta(\cdot \mid \boldsymbol{x}, \boldsymbol{y}_{<t}))$ of the next-token distribution at each tool-call position, but this choice is precluded by training-loop structure. Per-token sampling probabilities are emitted essentially for free by the rollout engine (vLLM [66]), whereas predictive entropy requires either an extra forward pass over the full

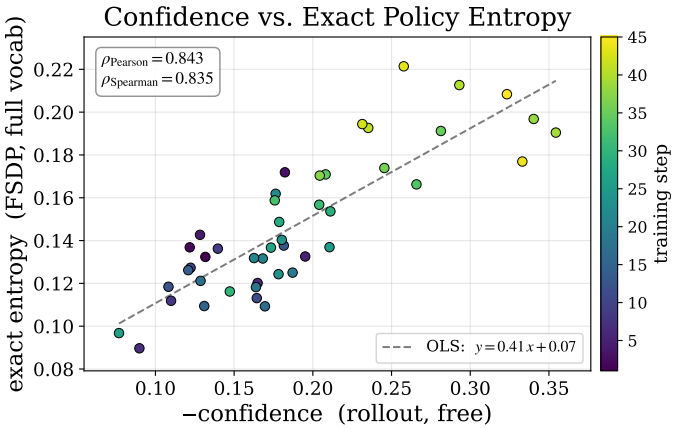


Figure 6: **Confidence vs. exact policy entropy at tool-call positions.** For each training step, we plot the mean confidence (vLLM rollout, free) against the exact policy entropy (FSDP forward, full vocabulary) on tool-call tokens of failed tool-using rollouts. The two signals are strongly correlated (Pearson $\rho = 0.843$, Spearman $\rho = 0.835$), justifying confidence as a tractable uncertainty proxy.

vocabulary under the trained policy (FSDP) or vLLM-side full-vocabulary logprob expansion at decode time. Our RL pipeline under verl [61] alternates rollout (vLLM) and gradient updates (FSDP), so inserting a per-prefix entropy computation would force an additional vLLM $\rightarrow$ FSDP $\rightarrow$ vLLM round-trip per training step (or equivalently expensive vLLM-side full-vocabulary logprob expansion), a cost we wished to avoid.

To verify that confidence preserves the ranking an entropy-based scheme would induce, we measured the per-step correlation between mean-token-confidence and exact policy entropy on tool-call positions of failed tool-using rollouts collected during training (Figure 6). Across training steps, the two signals are strongly correlated (Pearson $\rho = 0.843$, Spearman $\rho = 0.835$, OLS fit $y = 0.41x + 0.07$ on entropy vs negated confidence). Confidence therefore preserves the relative ordering an entropy-based scheme would produce, and we adopt it as the ranking signal in AXPO.

D Extended Related Works

This appendix expands on Section 5 with the longer descriptions that the main text abridges.

Two-stage SFT-then-RL recipe for agentic VLMs. Recent agentic VLM systems share a two-stage recipe: supervised fine-tuning on tool-using trajectories distilled from larger teachers [67, 20], followed by outcome-reward RL [18, 23, 46]. They differ mainly in the distillation corpus, the tool interface, and the choice of base model. Most prior agentic-RL VLM studies focus on a single tool family (typically image manipulation), whereas our setup combines code, search, and image zoom-in on a thinking-capable base (Qwen3-VL-Thinking) [19], a configuration under which the RL stage becomes the primary bottleneck.

Group-relative RL and BroRL. RL with verifiable outcome rewards is the standard post-SFT stage for reasoning models [1, 2]. GRPO [24–26] samples N rollouts per question, normalizes rewards within the group, and assigns the resulting advantage uniformly to every token. BroRL [30] argues that increasing N reduces unsampled blind spots at a rate determined by per-token baseline probability. Both analyses were developed for thinking-only trajectories, where the rollout structure is uniform across tokens and the per-token baseline probability decays slowly. Proposition 1 instead compares correct tool-using rollout coverage under prefix-fixed and from-scratch sampling, conditional on the selected prefix satisfying the stated recoverability threshold.

Mid-trajectory branching for tool-using RL. ARPO [31] and AEPO [60] adapt group-relative RL to tool-using rollouts by branching *after* a tool observation, drawing additional continuations from $\pi_\theta(\cdot \mid \boldsymbol{x}, \boldsymbol{t}_1, \boldsymbol{a}_1, \boldsymbol{o}_1)$ when post-observation entropy is high. The exploration target is the continuation given a fixed tool call, which absorbs variance introduced by the tool output but cannot recover when

the tool call itself is wrong (wrong sub-image, wrong python code, wrong search query). AXPO instead branches *at* the tool-call boundary and triggers only on all-wrong tool-using subgroups, with a per-prefix advantage that credits the source prefix exactly when its branched continuation succeeds, making it a targeted intervention on the regime Section 2.2 identifies as binding rather than a general rollout-budget reshaper. Empirically, AXPO outperforms ARPO on the 8B comparison (53.9 vs. 51.3 Avg, Table 4), consistent with the diagnosis that the tool call itself, not the post-observation continuation, is the dominant failure locus. The two strategies are compatible, and combining tool-call and post-observation branching is a natural extension we leave to future work.

Tree-based credit assignment. PORTool [68] organizes tool-use trajectories into a prefix-sharing tree and combines trajectory-level and fork-relative advantages to assign credit to intermediate tool decisions. AXPO instead targets the absence of successful tool-using trajectories when tool use is optional, with an all-wrong trigger, resampling before tool execution, and separate recovery credit for the prefix and continuation. Its contribution lies in this combination, rather than branching alone.

Connection to non-uniform sampling. Adaptive stratified sampling [69], prioritized sweeping [70], and prioritized experience replay [71] already allocate sampling or updates non-uniformly. We do not claim this general principle as new. AXPO identifies the all-wrong tool-using subgroup as a specific recovery target in group-based RL. This failure mode can also arise in text-only agents with optional tool use (Appendix C.2).

E Coverage Argument for Tool-call resampling

This appendix formalizes the conditional coverage guarantee in Proposition 1. At fixed budget N, resampling from a tool-committed prefix has at least as much probability of recovering a correct tool-using rollout as from-scratch sampling when $p(\boldsymbol{t}_1^{\text{src}}) \geq q\, p^{\text{tool}}$, with strict improvement under the proposition's strictness conditions. This result concerns recovery coverage, not the semantic correctness of the prefix or an unconditional guarantee of policy improvement.

Setup. Let π_θ be the policy, $\boldsymbol{x}$ a question, and $R(\tau) \in \{0, 1\}$ the outcome reward. Decompose $\tau = (\boldsymbol{t}_1, \boldsymbol{y}_{\geq 1})$ into a thinking prefix and its continuation (tool call, tool output, and the remainder of the trajectory). Let $\mathcal{T} = \{\boldsymbol{t}_1 : \boldsymbol{t}_1$ includes the opening `<tool_call>` tag$\}$ be the set of tool-committed prefixes, and define

$$q := \Pr_{\pi_\theta}[\boldsymbol{t}_1 \in \mathcal{T} \mid \boldsymbol{x}], \qquad p(\boldsymbol{t}_1) := \Pr_{\pi_\theta}[R = 1 \mid \boldsymbol{x}, \boldsymbol{t}_1], \qquad p^{\text{tool}} := \mathbb{E}_{\boldsymbol{t}_1 \sim \pi_\theta(\cdot \mid \boldsymbol{x},\, \boldsymbol{t}_1 \in \mathcal{T})}[p(\boldsymbol{t}_1)].$$

So q is the rate at which the policy commits to tool use from $\boldsymbol{x}$, and p^{tool} is the per-tool-using-rollout success rate. Both are directly measurable (Figure 3a, Figure 3b). We compare two samplers, each drawing N trajectories, on the event *"at least one sample is a correct tool-using rollout"*:

- **Raw rollout**: $\tau \sim \pi_\theta(\cdot \mid \boldsymbol{x})$. Per-sample probability of a correct tool-using rollout is $\Pr[\boldsymbol{t}_1 \in \mathcal{T}] \cdot \Pr[R = 1 \mid \boldsymbol{t}_1 \in \mathcal{T}] = q\, p^{\text{tool}}$.
- **Prefix-fixed resampling**: fix $\boldsymbol{t}_1^{\text{src}} \in \mathcal{T}$ and draw continuations from $\pi_\theta(\cdot \mid \boldsymbol{x}, \boldsymbol{t}_1^{\text{src}})$. Every resampled continuation is tool-using by construction ($q \to 1$), and per-resample success is $p(\boldsymbol{t}_1^{\text{src}})$.

Coverage, i.e., the probability that at least one of the N samples is a correct tool-using rollout, equals $1 - (1 - p_{\text{eff}})^N$ with $p_{\text{eff}} = q\, p^{\text{tool}}$ under raw rollout and $p_{\text{eff}} = p(\boldsymbol{t}_1^{\text{src}})$ under resampling.

Proof of Proposition 1. The function $f(p) = 1 - (1 - p)^N$ has derivative $f'(p) = N(1 - p)^{N-1}$, non-negative on $[0, 1]$ and strictly positive on $[0, 1)$, so f is non-decreasing on $[0, 1]$ and strictly increasing on $[0, 1)$. Under the assumption $p(\boldsymbol{t}_1^{\text{src}}) \geq q\, p^{\text{tool}}$, applying f to both sides gives $1 - (1 - p(\boldsymbol{t}_1^{\text{src}}))^N \geq 1 - (1 - q\, p^{\text{tool}})^N$. Strictness follows whenever $p(\boldsymbol{t}_1^{\text{src}}) > q\, p^{\text{tool}}$ and $q\, p^{\text{tool}} \in (0, 1)$. The mechanism is $q \to 1$ by construction: resampling eliminates raw's wasted $(1 - q)$ fraction of non-tool rollouts, so the baseline threshold is $q\, p^{\text{tool}}$ rather than p^{tool}.

Relation between coverage and the learning signal. With binary rewards, an all-wrong tool-using subgroup receives negative advantage if some no-tool rollouts succeed, and zero advantage if the entire group fails. Recovering a correct tool-using rollout can therefore provide a missing learning

signal. However, coverage is not identical to the probability of positive GRPO advantage: a group in which every rollout succeeds also has zero reward contrast. Proposition 1 establishes the probability of obtaining at least one correct tool-using rollout under its condition. The subsequent gradient depends on the reward composition and credit assignment in Section 3.3, and a correct final answer does not verify the source prefix's reasoning.

Checking the condition on selected prefixes. For prefixes drawn from the policy conditional on tool use, $\mathbb{E}_{\boldsymbol{t}_1}[p(\boldsymbol{t}_1)] = p^{\text{tool}} \geq q\, p^{\text{tool}}$, strictly so when $q < 1$ and $p^{\text{tool}} > 0$. AXPO instead selects prefixes from failed rollouts using uncertainty, so this identity does not establish its condition on the selected distribution. The comparison in Section 3.1 estimates mean per-sample success. Average coverage also depends on how success probabilities vary across prefixes. Appendix C.6 reports the coarse per-prefix margins and the prefix-quality audit.

Relation to BroRL. Hu et al. [30] motivate larger rollout groups through coverage of successful trajectories. AXPO changes where additional samples are drawn: conditioning on a tool-committed prefix makes every continuation tool-using, whereas a from-scratch rollout invokes a tool with probability q. This removes sampling effort spent on no-tool trajectories, but is beneficial for correct tool-use coverage only when the retained prefix is sufficiently recoverable. Proposition 1 states that condition explicitly, and the selected-prefix measurements assess it empirically.

F Limitations and Future Work

We expand on the limitations sketched in the main-text conclusion and discuss future directions.

Verifiable outcome rewards. AXPO's subgroup-level trigger ("all-wrong tool-using subgroup") and per-prefix advantage in Eq. 2 both rely on a binary, automatically verifiable outcome signal $r \in \{0, 1\}$. Tasks where verifiability is partial (open-ended generation, multi-criterion subjective evaluation) require a different definition of "failed subgroup" before tool-call resampling applies. The resampling mechanism itself is reward-agnostic, but the trigger and the all-wrong learning signal argument in Section 2.2 are not.

Compute scope of training. Trained models are at most 8B parameters; 32B is reported inference-only as the SFT teacher. This boundary reflects available compute rather than a methodological choice. Whether the Thinking-Acting Gap and AXPO's recovery generalize to 32B+ trainable scale, where the base policy is closer to the tool-use ceiling and the all-wrong-subgroup fraction may be smaller, is open. Furthermore, although the teacher model can theoretically scale to 100B+ parameters, analyzing the effects of teacher scaling on downstream reinforcement learning (RL) remains an area for future work.

First-boundary scope. Our implementation resamples at the first tool-call boundary, where the policy first chooses tools over direct answering. In our training runs, 79.6% of tool-using rollouts make one call and 20.3% make two. Continuations run to the final answer and can include later calls, but the method does not identify which later call caused a failure. The first boundary is therefore a practical choice for this setting, not a uniquely optimal one. Adaptive multi-boundary resampling would need to select boundaries, allocate budget, and assign credit without dense process supervision or combinatorial rollout growth.

Larger tool inventories. On 1,500 tool-calling turns from the 8B AXPO run, GPT-5.6-Sol judges the thinking text to commit to a next-tool identity in 73.9% of cases. Conditional on such commitment, it matches the eventual call 96.5% of the time. These measurements characterize our three-tool setting, not scaling to larger inventories. With more tools, exploration may need to distinguish tool selection from argument refinement, potentially fixing a reliable tool name and resampling only its arguments and continuation.

Future work: richer tool environments. Our tool inventory covers three families that span numerical, retrieval, and visual modalities (Python interpreter, web search, image zoom-in), each shown to be load-bearing in prior agentic VLM work [23, 18, 7]; this multi-tool setup is already broader than most prior agentic-VLM RL studies, which typically focus on a single tool family. One

notable absence is an image-search tool, which prior works [23, 43] report results with but which we could not include during training and full evaluation due to per-call API costs at our scale (we instead approximate it via a GPT-5.4 prompt for the unseen-tool study in Appendix B.6). Tools with longer execution latency (browser agents, GUI control, terminal sessions) and tools with high per-call cost (LLM-as-tool) [13] introduce additional considerations that we do not address. Extending AXPO to such richer, longer-horizon environments is a natural next step.

G Broader Impacts

AXPO is a methodological improvement to the RL training of agentic VLMs and is additive on top of a publicly released base model (Qwen3-VL-Thinking [19]), so it does not introduce a new capability class. Among the three tool families used in this work, the Python interpreter raises standard security concerns for code-executing agents. We can run all tool calls in a sandboxed execution environment during both training and evaluation, consistent with prior agentic-VLM RL work [12]. Beyond this sandboxing, broader safety considerations [72–78] (e.g., misuse of tool-grounded outputs, causality, robustness, deployment-time guardrails, downstream effects of cheaper agentic deployments) are out of scope for this work and were not addressed.

H Qualitative Examples: GRPO vs. AXPO

We show three side-by-side trajectories where SFT + AXPO succeeds (4/4) and SFT + GRPO fails (0/4) at the 8B scale, one per modality (perception, reasoning, search). For each example we display the question, the correct answer, and one representative trajectory per method. Thoughts are extensively abbreviated with ellipses for brevity; tool calls and outputs are reproduced verbatim.

H.1 Perception: tiny-text reading on a Hong Kong street scene

Question

Benchmark: VisualProbe-hard [7]. **Ground truth**: `2563`.
Prompt: "What is the number to the left of 1000?"



AXPO ✓

Step 0, `<think>`**.** *"...checking phone numbers and license plates...looking at the yellow sign on the right side under SUNBEAM THEATRE, there's a section with 'Tel: 2563 1000'...Let me confirm with image_zoom_in."*
Step 0, `<tool_call>`**.**

```
image_zoom_in({"bbox_2d": [870, 450, 945, 517], "label": "Telephone number
sign", "img_idx": 0})
```



Step 1, `<think>`**.** *"The cropped image clearly shows 'Tel: 2563 1000'. The number to the left of 1000 is 2563."*
`<answer> 2563` ✓

GRPO ×

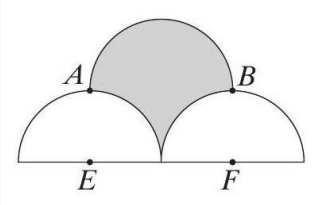
Step 0, `<think>`**.** *"...checking traffic lights...license plates...bus route numbers...none match 1000...maybe the question is a trick. The number to the left of 1000 in numerical order is 999...Let me try image_zoom_in...[reasoning continues without ever emitting a tool call]...Given that this is a common math question, the answer is likely 999."*
No tool calls were emitted.
`<answer> 999` ×

The thinking prefix in both rollouts identifies the right candidate region, but only AXPO commits to a tool call; GRPO talks itself into a numerical-order guess (`999`) and never invokes `image_zoom_in`, despite repeatedly mentioning it as a candidate action. All four GRPO rollouts on this question follow the same pattern.

H.2 Reasoning: geometry disambiguation via python visualization

Question

Benchmark: MathVision [39]. **Ground truth**: 8.
Prompt: "Three semi-circles as shown. $ABEF$ is a rectangle and the radius of each of the semi-circles is 2 cm. E and F are the centers of the bottom semi-circles. The area of the shaded region (in cm^2) is:"

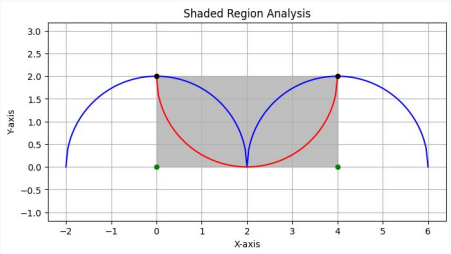


AXPO ✓

Step 0, `<think>`. *"... Setting $E = (0, 0)$, $F = (4, 0)$, the top semicircle has center $(2, 2)$ and radius 2. Computing sector minus triangle for each side gives an overlap of $\pi - 2$... Shaded $= 2\pi - 2(\pi - 2) = 4$. The result matches the area of a 2×2 square, suggesting correctness. Let me also visualise the configuration to confirm."*
Step 0, `<tool_call>`.

```
python({"code": "import matplotlib.pyplot as plt; import
numpy as np;...ax.plot(x1, np.sqrt(4-x1**2), 'b'); ax.plot(x2,
np.sqrt(4-(x2-4)**2), 'b'); ax.plot(x3, 2-np.sqrt(4-(x3-2)**2), 'r');
ax.add_patch(plt.Polygon([(0,2),(0,0),(4,0),(4,2)], color='gray',
alpha=0.5));..."})
```

Tool output (matplotlib figure).



Step 1, `<think>`. *"... The plot shows the bottom semicircles curving **upward** from EF to A and B (forming quarter-disks at corners E, F, total area 2π) and the top semicircle curving **downward** from AB to $(2, 0)$ (a half-disk of area 2π). Since the two areas match, by area-swap the shaded region rearranges to the full rectangle $ABEF$: $4 \times 2 = 8$."*
`<answer> 8` ✓

GRPO ×

Step 0, `<think>`. *"... Top semicircle area $= 2\pi$. Each side gives a sector of $\pi/2$ minus a right triangle of area 2, so each segment is $\pi - 2$. Subtracting both: $2\pi - 2(\pi - 2) = 4$... so the shaded area is 4."*
No tool calls were emitted.
`<answer> 4` ×

Both rollouts begin with the same algebraic derivation that subtracts the two overlapping segments from the top semicircle and arrives at 4, the area of the wrong sub-region. The error is geometric, not arithmetic. AXPO emits a python call to render the configuration; the matplotlib output makes the area-swap structure visible – the bottom quarter-disks at corners E, F inside the rectangle (total area 2π) match the top half-disk hanging from AB (also 2π), so the shaded region rearranges to the rectangle area $8 = 4 \times 2$, and the model revises accordingly. All four GRPO rollouts emit zero tool calls and commit to one of the wrong values (4, 4, 4, 2π). The example illustrates a clean instance of the Thinking-Acting Gap: the algebraic prefix is sound but the area-swap intuition only becomes accessible after a tool call, and GRPO never learns to invoke it on questions of this type.

H.3 Search: multi-hop decomposition into two sequential queries

Question

Benchmark: HR-MMSearch [43]. **Ground truth**: `1984`.
Prompt: "In which year was the mayor, who will be serving in August 2025, born in the city where the fashion designer of this fashion show was born?"



AXPO ✓

Step 0, `<think>`**.** *"The backdrop reads PAUL COSTELLOE; designer is Paul Costelloe. I need (i) his birthplace, (ii) the mayor of that city in August 2025, (iii) the mayor's birth year. Start with the birthplace."*
Step 0, `<tool_call>`**.**
`tavily-search({"query": "Paul Costelloe birthplace", "max_results": 5})`
Tool output (truncated). "*. . . Born in Dublin, Ireland . . . Paul Costelloe (23 June 1945 – 21 November 2025). . .* "
Step 1, `<think>`**.** *"Birthplace = Dublin. Now the Lord Mayor of Dublin in August 2025."*
Step 1, `<tool_call>`**.**
`tavily-search({"query": "current mayor of Dublin Ireland 2024", "max_results": 5})`
Tool output (truncated). "*Ray McAdam (born 1984) is an Irish Fine Gael politician who has served as Lord Mayor of Dublin since June 2025.*"
Step 2, `<think>`**.** *"Designer was born in Dublin; Lord Mayor of Dublin since June 2025 is Ray McAdam, born 1984."*
`<answer> 1984` ✓

GRPO ×

Step 0, `<think>`**.** *"Designer is Paul Costelloe; born in Dublin in 1945. Mayor of Dublin in 2025 – the search results don't directly show this, but if the riddle is that the mayor is the same age as the designer, the answer would be 1945."*
Step 0, `<tool_call>`**.**
`tavily-search({"query": "Paul Costelloe birthplace", ...})`
Tool output (truncated). "*. . . Born in Dublin, Ireland . . . Paul Costelloe (23 June 1945 – 21 November 2025). . .* "
Step 1, `<think>`**.** *". . . search returned the designer's info but not the mayor's. I'll commit to 1945 since the designer was 80 in 2025 and the question may be a riddle."*
No second-hop search.
`<answer> 1945` ×

The two-hop structure (designer → city; city + date → mayor; mayor → birth year) admits a single-query collapse only by accident; the load-bearing decision is whether to issue a second `tavily-search` after the first hop returns. AXPO consistently issues the follow-up; all four GRPO rollouts on this question stop after the first hop and fall back to a riddle-style guess (1945, 1963, 1968, etc.).

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Both abstract and Section 1 accurately reflect the paper's contributions and scope.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We include limitations in Section 6 and fully disclose them in Appendix F due to the page limitation.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: We include theoretical motivation in Section 3.1, and we include related assumptions and proofs in Appendix E.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We describe the AXPO algorithm in Section 3. We report the training and evaluation settings in Appendix A.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We include the details on data used in Appendix A, where all datasets we used are already open-sourced. The code for the proposed algorithm is open-sourced as the supplementary file.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: We include all the details on data, prompt, and hyperparameters used during experiments in Appendix A.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: We report Pass@1 variability across evaluation rollouts in Table 8. We report results across training seeds and paired per-question bootstrap confidence intervals in Appendix B.4.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We indicate the type of compute workers in Appendix A.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: The research conducted in the paper conform with the NeurIPS Code of Ethics in every respect.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: AXPO trains agentic VLMs to invoke a Python interpreter as one of its tools, which raises standard security concerns for code-executing agents. We can run all tool calls in a sandboxed execution environment during both training and evaluation. Beyond this, broader safety considerations (e.g., misuse of tool-grounded outputs, deployment-time guardrails) are out of scope for this work and were not addressed. See Appendix G.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Our work is built on open-source model (Qwen3-VL), dataset, benchmarks, and training libraries and we properly cite the original works of them.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: Qwen3-VL-32B-Thinking generates SFT trajectories. An LLM judge computes correctness rewards for RL examples that are not reliably covered by exact matching or rule-based verification. All RL methods share this reward protocol. AXPO adds tool-call resampling and modifies advantage assignment. We also use LLM judges to cluster tool calls by semantic intent (Section 2.2) and to match answers during evaluation (Appendix A.3).

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.