Think Twice Before Calling Tools: How Pre-Call Reasoning in SFT Shapes Tool-Use Retention under RL

Kaiwen Hu[1*] Hyunin Lee[1] Joseph Suh[1] Serina Chang[1] Shuai Shao[2]
Yong Zhang[2] Namyong Park[2] Xiaoqiang Luo[2] Luke Simon[2] Somayeh Sojoudi[1]

[1]University of California, Berkeley [2]Meta

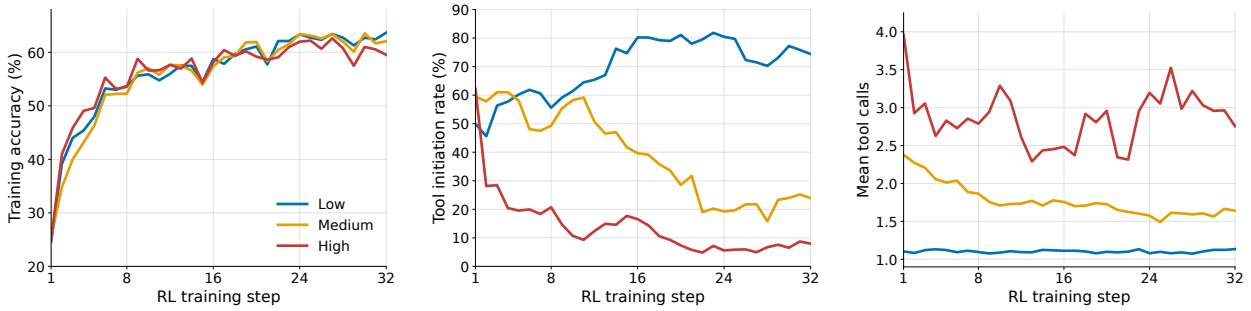
Abstract

Tool use learned during supervised fine-tuning (SFT) is not always retained during reinforcement learning (RL). We study how properties of SFT data may affect this retention by constructing Low, Medium, and High SFT datasets with approximately matched token budgets but different tool-call depths. Under the same RL recipe, the three initializations reach similar RL training accuracy yet exhibit markedly different tool-use dynamics: tool initiation increases in Low, declines in Medium, and collapses in High. We observe that these datasets also differ substantially in the amount of reasoning preceding each tool call. In particular, High trajectories contain considerably less pre-call reasoning, motivating the hypothesis that pre-call reasoning may affect how well tool use is calibrated and, consequently, whether the corresponding behaviors are retained during RL. An idealized theoretical analysis provides a selective-retention view of this process: advantages in overall and tool-conditioned accuracy can be preserved under outcome-only optimization, while tool initiation increases only when tool-using trajectories are more successful than the tool-free alternative. Guided by this analysis, we rank the original High trajectories by average pre-call reasoning length and retain the highest-ranked examples while approximately matching cumulative SFT token exposure. This simple filtering strategy improves post-RL accuracy and tool-conditioned final-answer success without consistently increasing tool initiation across multiple benchmarks. These results suggest that pre-call reasoning is a useful dimension of SFT supervision for understanding and improving tool-use retention under downstream RL.

1 Introduction

Recent advances in Multimodal Large Language Models (MLLMs) have demonstrated outstanding capabilities in areas such as math and coding (Li et al., 2024; Deitke et al., 2025; Bai et al., 2025; Chen et al., 2024). A growing line of work further augments text-based reasoning with external tools, allowing models to zoom into image regions, execute code, retrieve information, and incorporate the resulting observations into subsequent reasoning (Zheng et al., 2026; Jin et al., 2025). Such Tool-Integrated Reasoning (TIR) typically requires a model to decide not only *how* to use a tool, but also *when* it is useful and how its output should affect the remaining solution process (Yao et al., 2022; Schick et al., 2023). Recent systems commonly learn these behaviors through a two-stage pipeline:

*Corresponding author: kaiwen_hu@berkeley.edu.



(a) RL training accuracy (b) Tool initiation rate (c) Conditional tool-call depth

Figure 1: RL training dynamics of the `Low`, `Medium`, and `High` SFT initializations. (a) RL training accuracy. (b) Fraction of rollouts that use at least one tool call within each batch. (c) Mean number of tool calls among tool-using rollouts. Labels indicate the tool-call depth of the SFT data. Despite similar training accuracy, the three initializations exhibit markedly different tool-use behavior. Under the same RL recipe, `Low` maintains frequent tool use, whereas `High` sharply reduces tool initiation while retaining deeper interactions among the remaining tool-use trajectories.

supervised fine-tuning (SFT) establishes tool-use capabilities and interaction formats, followed by reinforcement learning (RL) (Hong et al., 2026; Song et al., 2026). While substantial effort has focused on RL algorithms (Qian et al., 2025; Hou et al., 2026; Zheng et al., 2026), considerably less is understood about how SFT data shapes the RL training dynamics of tool-use behavior.

We investigate this question by varying the *tool-call depth* of SFT trajectories, defined as the number of calls in a tool-using trajectory. Starting from the same base model with Zoom and Code tools, we construct `Low`, `Medium`, and `High` SFT datasets with a matched number of trajectories and total token budgets, and then apply an identical RL training recipe. Our early experiments reveal this is a non-trivial question (Figure 1): despite similar RL training accuracy, their tool-use behaviors diverge sharply. We find that the tool initiation rate gradually increases in `Low`, moderately decreases in `Medium`, and experiences a severe collapse in `High`. Among trajectories that still use tools, the depth ordering remains `High` > `Medium` > `Low`. Thus, demonstrating more tool calls during SFT does not imply that those behaviors will be retained more frequently under outcome-only RL.

To understand this phenomenon, we examine the textual reasoning that precedes each tool call, which we term *pre-call reasoning*. `High` trajectories contain substantially fewer average pre-call reasoning tokens than `Low`. During RL, we further observe that the tool-using trajectories that remain in `Medium` and `High` tend to contain more pre-call reasoning after RL training than at initialization. These observations motivate the hypothesis that *pre-call reasoning is a key signal for tool-use calibration*. Here, tool-use calibration refers to whether the model invokes tools when they are actually useful, rather than simply learning to call them frequently.

Our theoretical analysis formalizes this intuition through a selective-retention view of RL. In an idealized probability-space analysis, advantages in overall accuracy and tool-conditioned accuracy present in the SFT initialization are preserved under the same outcome-only optimization. Tool initiation follows different dynamics: it increases only when tool-using trajectories are more successful than the tool-free alternative, and decreases otherwise. This provides a theoretical view of selective retention: RL need not preserve frequent tool use, but instead favors tool behaviors that are better aligned with task success. Thus, tool-call frequency or depth alone is insufficient to characterize useful tool use.

Guided by this analysis, we rank the original `High` trajectories by their average pre-call reasoning length and retain the highest-ranked ones. We construct two subsets containing approximately 50%

and 25% of the original assistant-supervised token mass per epoch. Under a controlled number of cumulative supervision tokens and the same downstream RL training recipe, the 25% variant outperforms the original `High` on all five benchmarks, while the 50% variant improves on four out of five. Moreover, the filtered initializations achieve substantially higher final-answer success among their self-selected tool-using trajectories on several benchmarks, without a consistent increase in tool initiation, suggesting that effective SFT should prioritize supervision supporting useful tool decisions rather than tool-call frequency.

Our contributions are threefold:

- We show that the tool-call depth of SFT data leads to sharply different tool-use dynamics under the same outcome-only RL recipe: more frequent tool use during SFT does not necessarily translate into more persistent or better-calibrated tool use after RL.
- We identify pre-call reasoning as a key factor shaping *tool-use calibration* and develop a selective-retention analysis showing how differences in overall and tool-conditioned success are preserved under outcome-only optimization, while tool initiation depends on whether tool use is more successful than the tool-free alternative.
- We show that filtering the original `High` SFT data by average pre-call reasoning length, under approximately matched cumulative supervision budgets, improves downstream accuracy and tool-enabled performance without consistently increasing tool initiation, outperforming a matched random-selection control.

2 Related Work

Tool-Integrated Multimodal Reasoning. TIR couples language-based deliberation with external actions, allowing models to gather evidence, perform computation, and update their plans during inference. ReAct establishes the general interleaving of reasoning and acting, while Toolformer studies self-supervised learning for API selection, invocation, and result use (Yao et al., 2022; Schick et al., 2023). Recent multimodal systems extend this paradigm to visual search and executable computation: DeepEyes and VTool-R1 use RL to improve visual tool use (Zheng et al., 2026; Wu et al., 2026); DeepEyesV2 and Thyme combine visual operations with code-based computation in multimodal post-training pipelines (Hong et al., 2026; Zhang et al., 2025b); and CodeDance combines task-aware cold-start SFT with difficulty-adaptive RL for executable visual reasoning (Song et al., 2026). ICRL instead places demonstrations in RL prompts and gradually removes them during training (Ye et al., 2026). These studies primarily focus on enabling multimodal tool use or improving its final performance. We study a complementary question: how the structure of SFT tool-use trajectories determines which tool behaviors remain viable under outcome-only RL training.

SFT Initialization for Reinforcement Learning. Prior work shows that SFT initialization can strongly affect subsequent RL. DeepSeek-R1 and Chu et al. (Guo et al., 2025; Chu et al., 2025) compare the roles of SFT and RL, while PEAR shows that a stronger SFT checkpoint can perform worse after identical RL (Zhang et al., 2026a). This observation closely motivates our study: SFT quality should be evaluated by the behaviors it prepares for RL, rather than by SFT performance alone. Tool-integrated studies explore cold-start data, diverse reasoning primitives, and process supervision (Yao et al., 2026; Feng et al., 2026a; Mao et al., 2026; He et al., 2026). ASTER reports that deep tool trajectories can provide beneficial initializations (Zhang et al., 2026b), which appears different from our observation of tool-initiation collapse for high-depth SFT. However, ASTER

emphasizes interaction density and tool-use budgets, whereas we approximately match the total SFT token budget and study how it is allocated before individual calls. Our results therefore complement this line of work by showing that the value of deep trajectories depends not only on interaction depth, but also on per-call reasoning supervision.

Tool Utility and Reinforcement Learning. Tool frequency alone does not determine tool utility. MED separates intrinsic improvement, tool-induced correction, and tool-induced harm (Ma et al., 2026), while other work studies capability expansion, evidence dependence, and the relationship between invocation frequency and reasoning quality (Guo et al., 2026; Wang et al., 2026; Kim et al., 2026). In particular, the finding that frequent tool invocation need not improve reasoning is closely aligned with our selective-retention view (Kim et al., 2026). Other approaches improve tool learning by modifying rewards, credit assignment, or rollout allocation (Qian et al., 2025; Hou et al., 2026; Li et al., 2026; Xie et al., 2026; Feng et al., 2026b; Dong et al., 2026). We instead keep outcome-only RL fixed and study how SFT data determines the reliability of the tool decisions retained by RL, separating tool frequency from the success of self-selected tool-using trajectories.

3 How SFT Shapes Tool Use

Notations. Let $\mathcal{M}$ denote an MLLM and x an input prompt. Given x, the model interacts with the available tools and produces a trajectory $\tau \sim \mathcal{M}(\cdot \mid x)$. If τ contains T tool calls, we write $\tau = (s_1, \ldots, s_T, r)$, where each *tool-interaction step* s_t consists of the reasoning preceding a tool call, the tool call itself, and the corresponding tool output: $s_t = (\text{reasoning}_t, \text{tool-call}_t, \text{tool-output}_t)$; r is the reasoning after the last tool call that reaches the final answer. We define the *tool-call depth* of τ as $d(\tau) = T$. We define reasoning_t as *pre-call reasoning* of s_t. A tool-free trajectory therefore has $d(\tau) = 0$. An SFT dataset is denoted by $\mathcal{D} = (x_i, \tau_i)_{i=1}^{N}$, where each prompt (or *question*) x_i is paired with its corresponding trajectory τ_i. Throughout the paper, we use the terms *question* and *prompt* interchangeably, as well as *trajectory* and *rollout*, depending on the context.

We study how the tool-use patterns demonstrated during SFT affect subsequent behavior under a fixed outcome-only RL training. We vary the tool-call depth of SFT while holding the downstream training recipe fixed, and analyze both the resulting tool-use dynamics and the reasoning preceding individual tool decisions.

3.1 Experimental Setup

Model and Tools. We use Qwen2.5-VL-7B-Instruct (Bai et al., 2025) as the base MLLM with two tools: **Zoom**, which crops and enlarges an image region, and **Code**, which executes model-generated Python. Models may invoke either tool multiple times.

SFT Data Construction. We construct a shared source pool of tool-using Zoom and Code trajectories by standardizing prompts and tool formats and removing malformed or duplicate trajectories. Zoom trajectories are drawn from CoF (Zhang et al., 2025a), Mini-o3 (Lai et al., 2026), and PixelReasoner (Su et al., 2025), while Code trajectories are drawn from DeepEyesV2 (Hong et al., 2026) and Thyme (Zhang et al., 2025b). From this pool, we construct three SFT datasets of 2,000 trajectories each, grouped by tool-call depth: `Low` ($d(\tau) = 1$), `Medium` ($2 \le d(\tau) \le 3$), and `High` ($d(\tau) \ge 4$). Each dataset contains 750 Zoom trajectories, 750 Code trajectories, and the same 500 tool-free trajectories, with approximately matched trajectory lengths.

RL Data Construction. All SFT initializations are trained with the same fixed set of 16,384 questions with RL drawn from DeepEyes-47K (Zheng et al., 2026), MMK12 (Meng et al., 2025), ChartQA (Masry et al., 2022), and Geometry3K (Lu et al., 2021). We remove explicit tool-use hints and evaluation overlap, and use the same questions and ordering across all RL runs.

Evaluation. We evaluate all checkpoints on a fixed 6,431-question suite consisting of V*Bench (Wu and Xie, 2024) (191), ChartQA (Masry et al., 2022) (2,500), MathVista testmini (Lu et al., 2024) (1,000), WeMath (Qiao et al., 2025) (1,740), and CharXiv (Wang et al., 2024) (1,000). We consider two evaluation settings: `tool-on` evaluation provides both Zoom and Code with up to eight calls and an 8,192-token response budget, while `tool-off` evaluation removes access to the tools while retaining the same evaluation protocol. The prompts are specified in Appendix B.

Training Configuration. Each SFT model is trained for one epoch (50 optimizer updates) with assistant-only supervision. Starting from each SFT checkpoint, we run 32 updates of outcome-only GRPO (Shao et al., 2024), sampling five rollouts for each of 512 questions per update. RL rewards only final-answer correctness and uses the same eight-call and 8,192-token interaction budget as evaluation. All optimization settings, RL data, sampling, and decoding configurations are shared across `Low`, `Medium`, and `High`. More details are provided in Appendix C

3.2 Divergent Tool-Use Dynamics under identical RL training

Figure 1 compares the RL training dynamics of policy initialized from SFT on the `Low`, `Medium`, and `High` dataset under the same RL recipe. Despite different initializations regarding the tool-call depth, all three runs finish at approximately the same training accuracy. Their tool calling behaviors, however, diverge substantially. We have three specific observations.

1. Tool initiation diverges across SFT initializations. In Figure 1b, `Low` increases its tool initiation rate to approximately 75% by the end of training. In contrast, `Medium` exhibits an overall decline to approximately 20%, while `High` drops rapidly in the early steps and remains below 10% near the end. Thus, greater tool-call depth in the SFT trajectories does not necessarily translate into more persistent tool initiation under the same RL recipe.

2. The ordering of conditional tool-call depth persists. Figure 1c describes the mean number of tool-calls among tool-using rollouts, excluding tool-free rollouts. `Low` consistently remains close to 1.1 tool calls, while `Medium` gradually decreases its tool calls from approximately 2.4 to 1.6. After a slight decline from roughly four calls, `High` continues to fluctuate in this panel between 2.3 and 3.5. Thus, in contrast to the tool initiation rate, the results show that the ordering of conditional tool-call depth persists during the RL training.

3. Similar accuracy can mask differences in tool-use calibration. Table 1 shows that RL brings the three initializations to similar final `tool-on` accuracy, despite their sharply different tool-use behaviors. `Low` continues to invoke tools frequently, whereas `Medium` and especially `High` increasingly avoid them. These results show that final task accuracy alone does not reveal how tool use is being retained. Comparing `tool-on` and `tool-off` performance shows that after RL, `tool-on` is approximately on par with `tool-off` for `Low`, but remains worse for `Medium` and `High`, which indicates that the tool use in `Medium` and `High` is not well-calibrated. Taken together, the results suggest that outcome-only RL does not simply preserve the tool-use patterns learned during SFT: tool behaviors that are poorly calibrated to task success are less likely to be retained. This

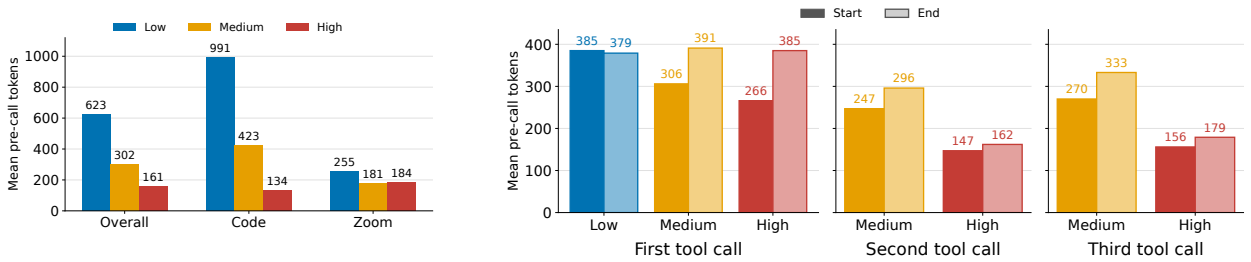
Table 1: Evaluation results for the SFT and RL checkpoints of `Low`, `Medium`, and `High`. Accuracy is pooled over five evaluation benchmarks. Gain denotes the difference between `tool-on` and `tool-off` accuracy. The three initializations exhibit similar final `tool-on` accuracy but critically different tool-use behavior (Figure 1b).

Checkpoint	`Tool-on` Acc	`Tool-off` Acc	Gain	Tool initiation rate	Conditional tool-call depth
`Low` SFT	59.34	59.65	-0.31	67.72	1.21
`Low` RL	**67.67**	67.66	+0.02	75.04	1.32
`Medium` SFT	47.43	62.14	-14.71	58.50	3.98
`Medium` RL	**67.66**	70.56	-2.90	20.82	3.03
`High` SFT	43.45	63.52	-20.07	50.60	5.59
`High` RL	**66.24**	68.90	-2.66	5.38	6.32

motivates our central question: what property of the SFT trajectories determines the calibration of tool use, and consequently whether that behavior survives downstream RL?

3.3 A Hypothesis: Pre-call Reasoning Matters

To investigate the divergent tool-use dynamics in Section 3.2, we examine how much reasoning the model performs *before* each tool call, which we refer to as *pre-call reasoning* (see its definition in Notations from Section 3).



(a) Mean pre-call tokens in SFT

(b) Mean pre-call tokens at RL start and end

Figure 2: Pre-call reasoning in SFT and during RL. (a) Mean number of reasoning tokens preceding each tool call in the `Low`, `Medium`, and `High` SFT datasets. (b) Mean pre-call reasoning tokens for the first three tool calls at the beginning and end of RL training. `High` receives substantially less pre-call supervision per call in SFT, while the tool-using trajectories retained by Medium and High exhibit longer pre-call reasoning after RL.

Figure 2a shows the average number of pre-call tokens, which decreases from 623 tokens in Low to 302 in Medium and 161 in High. The contrast is particularly pronounced for Code, with means of 991, 423, and 134 tokens, respectively. Thus, `Low`, `Medium`, and `High` differ not only in tool-call depth but also in the amount of pre-call reasoning. Intuitively, this may cause the calls in High to be insufficiently planned and the information from each tool call to be improperly integrated.

To test whether this hypothesis is consistent with the RL dynamics, we examine whether the tool-using rollouts retained after RL exhibit longer pre-call reasoning. Figure 2b compares the start (SFT-trained policy or step=0 of RL training policy) and end (step=32 of RL training policy) of the RL training for all three runs. For both `Medium` and `High`, we observe a consistent increase in the mean pre-call reasoning tokens in the first three calls, which implies that such behaviors tend to produce correct final answers and thus be promoted during the RL training. Specifically, we

highlight that this phenomenon is more obvious in the first call, where Medium increases from 306 to 391 and High increases from 266 to 385. This is natural because the model needs to understand the task, perform basic reasoning and plan the tool calling strategies that lay the foundation for the complete solution. We also note that the difference in Low is very little, mainly because the model already allocates sufficient pre-call tokens that produce robust reasoning ability. Taken together, these observations motivate the following hypothesis:

Hypothesis

Pre-call reasoning is a key factor shaping the **tool-use calibration**, which in turn affects whether the corresponding tool-use behaviors are retained under RL training.

Here, **tool-use calibration** refers to whether the model invokes tools when they are actually useful, rather than simply learning to call them frequently. Even if a model learns the mechanics of multi-step tool use and sustains high tool-use rates during inference, insufficient pre-call reasoning may lead to poorly calibrated decisions, causing the model to invoke tools unnecessarily.

3.4 A Theoretical Understanding of Selective Tool Retention

Section 3.3 motivates the hypothesis that pre-call reasoning affects the calibration of tool-using trajectories. We examine how different SFT data (Low, Medium, High) evolve under outcome-only optimization, distinguishing successful tool use from frequent tool invocation.

Modeling Tool Use from SFT to RL. Fix a question x, and let τ denote a complete trajectory containing the model's reasoning, tool calls, tool observations, and final answer. Let $p_i(\tau, t)$ denote the trajectory distribution at RL training time t induced by SFT initialization i, where different values of i correspond to policies trained on different SFT datasets. For the theoretical analysis, we treat t as continuous optimization time; in the discrete setting, t indexes RL optimization steps. Let $R(\tau) \in \{0, 1\}$ denote the outcome reward associated with trajectory τ. Assume positive initial probability on a common finite trajectory set Ω. To isolate probability reallocation under outcome-only optimization, we consider the idealized dynamics

$$\frac{\partial p_i(\tau, t)}{\partial t} = (R(\tau) - A_i(t))p_i(\tau, t), \qquad A_i(t) = \mathbb{E}_{p_i(t)}[R(\tau)]. \tag{1}$$

$A_i(t)$ denotes the probability that a rollout ends with a correct answer at time t. We partition Ω into the nonempty sets $\mathcal{T}$ and $\mathcal{D}$, containing trajectories with and without at least one supported tool execution attempt, respectively. Let $I_i(t)$ denote the probability of calling at least one tool in a rollout at time t, and let $C_i(t)$ and $D_i(t)$ denote conditional accuracy within $\mathcal{T}$ and $\mathcal{D}$. Then

$$A_i(t) = I_i(t)C_i(t) + (1 - I_i(t))D_i(t). \tag{2}$$

Theorem 1 (SFT initialization and selective tool retention)**.** *Under Eq. 1, the overall and tool-conditioned accuracy satisfy*

$$\dot{A}_i(t) = A_i(t)(1 - A_i(t)), \qquad \dot{C}_i(t) = C_i(t)(1 - C_i(t)), \tag{3}$$

whereas tool initiation satisfies

$$\dot{I}_i(t) = I_i(t)(1 - I_i(t))(C_i(t) - D_i(t)). \tag{4}$$

Moreover, if two SFT initializations satisfy $A_1(0) \geq A_2(0)$ *and* $C_1(0) \geq C_2(0)$*, then*

$$A_1(t) \geq A_2(t), \qquad C_1(t) \geq C_2(t) \qquad \text{for all } t \geq 0. \tag{5}$$

These accuracy orderings do not imply $I_1(t) \geq I_2(t)$*.*

Theorem 1 distinguishes two effects of SFT initialization: advantages in overall and tool-conditioned accuracy can persist under the same idealized optimization, whereas tool initiation depends on the relative accuracy of the tool-using and tool-free branches. Consequently, tool-conditioned accuracy can improve while initiation decreases whenever $C_i(t) < D_i(t)$. Demonstrating frequent or deep tool interactions alone, therefore, does not ensure their retention. This motivates selecting SFT trajectories that support reliable pre-call reasoning, rather than preserving tool calls for their own sake. The derivation and proof are provided in Appendix A.

4 From Diagnosis to SFT Data Design

In Section 3, we identified pre-call reasoning as a potential factor shaping tool-use calibration. We now ask whether this insight can guide SFT data selection. Specifically, we use average pre-call reasoning length as a proxy and filter the original `High` trajectories to test whether this simple intervention improves downstream RL performance.

4.1 Experimental Setup

Data Construction with Pre-call-length-guided Filtering. We investigate whether filtering trajectories by pre-call reasoning length improves downstream RL performance. Starting from the `High` SFT dataset $\{\tau_i\}_{i=1}^{N}$, we score each trajectory by its average number of pre-call reasoning tokens per tool call, $s_i = \frac{1}{k_i}\sum_{j=1}^{k_i} l_{i,j}$, where k_i is the number of tool calls in τ_i and $l_{i,j}$ is the number of reasoning tokens preceding the j-th call. We rank trajectories by s_i and select the top K_{25} and K_{50} trajectories such that their total token counts are 25% and 50% of the original dataset, respectively; for example, the total number of tokens in $\{\tau_i\}_{i=1}^{K_{25}}$ is approximately 25% of that in $\{\tau_i\}_{i=1}^{N}$. We denote these subsets as Length-Filtered 25% (`LF25`) and Length-Filtered 50% (`LF50`).

Training setup. We train `LF50` for two epochs and `LF25` for four epochs, so that their cumulative assistant-supervised token exposure approximately matches Original High. Both models start from the same base checkpoint and retain the original SFT objective and token-masking scheme. The subsequent RL training and evaluations are also identical to those in Section 3.

4.2 Experimental Results

Length filtering improves final task performance. Table 2 compares the final RL performance of the `High` and length-filtered SFT initializations. For completeness, we also include a matched Random (25%) control, which we analyze separately in Section 4.3. The `LF25` achieves the best performance on all five benchmarks, while the `LF50` outperforms `High` on four out of five benchmarks. Thus, concentrating the training budget in SFT on a small subset of the original data pool with longer pre-call reasoning yields better results. This result also resonates with the theoretical findings in Section 3.4, where we indicate that a stronger initialization can preserve this advantage in task-level success under the same outcome-only optimization.

Table 2: Final `tool-on` accuracy (%) after RL. Length-Filtered 25% (`LF25`) and 50% (`LF50`) are subsets of `High` containing the highest-ranked trajectories by pre-call reasoning length, with total token budgets of approximately 25% and 50% of the original `High` dataset, respectively. Random (25%) is a matched random-selection control analyzed in Section 4.3. Overall denotes the pooled accuracy across the five benchmarks. Shading indicates the best results.

Method	V*Bench	ChartQA	MathVista	WeMath	CharXiv	Overall
`High`	84.29	85.48	65.10	57.30	31.40	66.24
Length-Filtered (50%)	84.82	85.36	67.70	58.79	37.20	67.92
Length-Filtered (25%)	**86.91**	**87.72**	**68.90**	**60.57**	**38.40**	**69.75**
Random (25%)	81.68	82.20	66.90	60.34	35.80	66.67

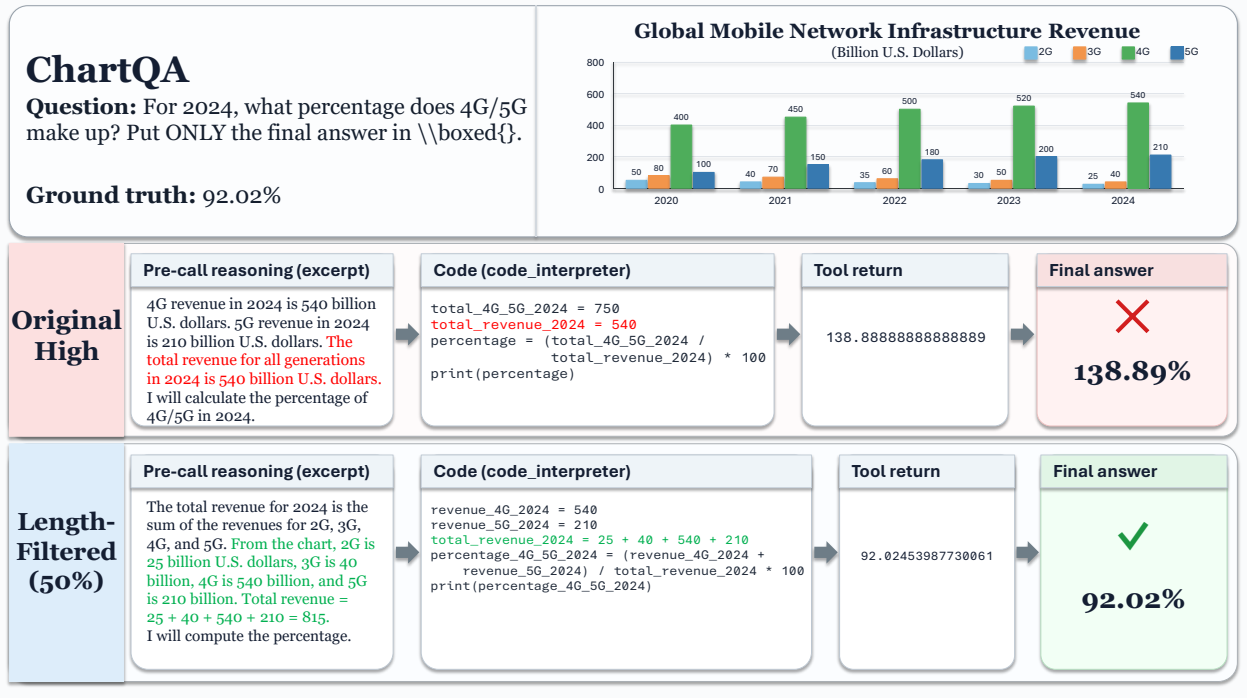
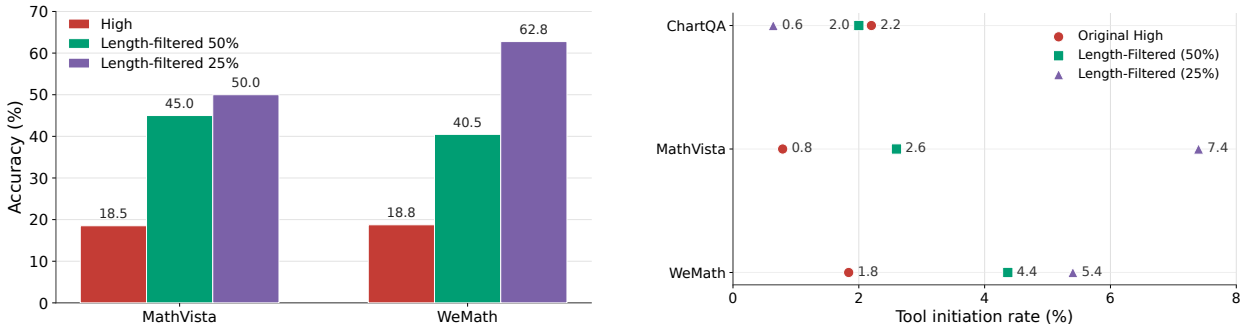


Figure 3: A qualitative ChartQA example illustrating how reasoning before a Code call can determine the computational target. In the original `High` trace, the model incorrectly treats the 4G revenue (540) as the total 2024 revenue, leading the subsequent code to use the wrong denominator. In contrast, the Length-Filtered model correctly sums the revenues of all four generations to obtain 815, yielding the correct percentage of 92.02%.

Higher conditional success among tool-using trajectories. Figure 4a reports the final-answer accuracy conditional on trajectories that used tools, indicating that the trajectories on which the filtered policies choose to invoke tools are more likely to end in a correct answer. This improvement is distinct from simply increasing tool usage: it reflects higher success among the model's self-selected tool-involved decisions. Figure 3 provides a qualitative illustration of this distinction: the two traces invoke the same Code tool, but differ in whether the pre-call reasoning identifies the computation that is actually needed to answer the question. This pattern mirrors the ordering of tool-conditioned accuracy in our analysis: a stronger initialization can maintain a higher ratio of successful tool-use trajectories under the same RL training.



(a) The final accuracy of the tool-using trajectories

(b) Tool initiation rates after RL

Figure 4: Tool-use behavior after RL. Length-filtered SFT yields higher final-answer success among self-selected tool-using trajectories, while tool initiation does not increase uniformly across benchmarks. Together, the two views show that the downstream benefit is better reflected by the effectiveness of retained tool-mediated decisions than by how frequently tools are invoked.

Successful completion and tool frequency are distinct. Figure 4b compares tool initiation rates across the three settings. Length filtering does not induce a uniform change in how often tools are invoked. On ChartQA, initiation decreases from 2.2% for Original High to 2.0% and 0.6% for Length-Filtered (50%) and Length-Filtered (25%), respectively. In contrast, initiation increases on MathVista and WeMath. Thus, the downstream gains from length filtering cannot be attributed simply to encouraging more tool calls. This behavior follows the tool-initiation dynamics in Eq. 4: tool use is reinforced only when tool-using trajectories outperform the tool-free alternative.

Taken together, these results show that the benefit of length-filtered SFT is not a uniform increase in tool use. Instead, it improves overall task accuracy and the success rate of self-selected tool-using trajectories, while tool initiation varies across benchmarks. This distinction is central to our findings: what matters after RL is not how many tool calls are retained, but whether the retained tool-use decisions are effective for reaching the correct answer.

4.3 Ablations and Analysis

Q1: Would random selection yield the same gains? To separate length-guided selection from simply training on a smaller, repeated subset, we compare Length-Filtered (25%) with Random (25%). The two variants use the same retained token budget, number of epochs, batch size, optimizer updates, and downstream RL procedure, differing only in how the SFT trajectories are selected.

Table 2 shows that Length-Filtered achieves higher accuracy on all five benchmarks. Random selection improves over Original `High` on MathVista, WeMath, and CharXiv, but reduces accuracy on V*Bench and ChartQA. Thus, random sampling a subset does not reproduce the results of the selection guided by length filtering. This suggests that the improvement cannot be explained by reduced data diversity or repeated exposure alone, and supports average pre-call length as an informative selection signal.

Q2: Does tool availability improve performance after length filtering? In Table 1, we observe that the **tool-availability gain**, which is measured by `tool-on` accuracy minus `tool-off` accuracy, is negative on almost all checkpoints. We therefore ask whether applying the pre-call-length guided filtering can make the remaining tool use beneficial.

Table 3: Tool-availability gain, measured as `tool-on` minus `tool-off` accuracy in percentage points. Positive values indicate higher accuracy with tools available.

Initialization	V*Bench	ChartQA	MathVista	WeMath	CharXiv	Overall
Original High	-1.57	-2.08	-1.80	-3.68	-3.40	-2.66
Length-Filtered (25%)	**+5.23**	**+1.00**	**+1.30**	**+1.37**	**+0.40**	**+1.18**

As shown in Table 3, Length-Filtered (25%) achieves positive tool-availability gains on all five benchmarks. The pooled gain changes from approximately -2.66 points for the original `High` to $+1.18$ points for Length-Filtered (25%). Notably, the filtered model's pooled `tool-off` accuracy remains close to that of the original `High`. Thus, the benefit of the available tools emerges when the pre-call-length-guided filtering is applied to the SFT data.

Together, these analyses strengthen the evidence that pre-call-length-guided SFT filtering benefits tool-enabled performance. The gains are not reproduced by a matched random subset, nor are they accompanied by a comparable improvement in `tool-off` performance. Instead, the selected SFT data appears to produce better calibrated tool use, making useful tool behaviors more likely to be retained under outcome-only RL.

5 Conclusion

We study how SFT shapes the tool-use behaviors retained under outcome-only RL. With approximately matched SFT budgets but different tool-call depths, we observe similar RL training accuracy yet sharply different tool-use dynamics, motivating pre-call reasoning as a key factor in tool-use calibration. Our selective-retention analysis shows that advantages in overall and tool-conditioned accuracy can persist under the same optimization, while tool initiation depends on whether tool use is more successful than the tool-free alternative. Guided by this view, pre-call-reasoning-length-guided filtering improves post-RL accuracy and tool-enabled performance across five benchmarks, outperforming a matched random-selection control without consistently increasing tool initiation. Overall, effective TIR training should prioritize supervision that supports well-calibrated tool use rather than tool-call frequency alone.

A natural future direction is to move beyond fixed proxies such as pre-call reasoning length and develop more adaptive criteria for selecting or generating SFT data that better prepares tool use for downstream RL. More broadly, it would be interesting to automate this process by jointly selecting, rewriting, or generating SFT trajectories based on downstream RL behavior, rather than relying on a manually chosen filtering rule. Finally, evaluating these principles across different tool types, model scales, and RL objectives would help determine how broadly the observed relationship between SFT data, tool-use calibration, and retention generalizes.

References

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

Zhe Chen, Weiyun Wang, Yue Cao, Yangzhou Liu, Zhangwei Gao, Erfei Cui, Jinguo Zhu, Shenglong Ye, Hao Tian, Zhaoyang Liu, et al. Expanding performance boundaries of open-source multimodal models with model, data, and test-time scaling. *arXiv preprint arXiv:2412.05271*, 2024.

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. *arXiv preprint arXiv:2501.17161*, 2025.

Matt Deitke, Christopher Clark, Sangho Lee, Rohun Tripathi, Yue Yang, Jae Sung Park, Mohammadreza Salehi, Niklas Muennighoff, Kyle Lo, Luca Soldaini, et al. Molmo and pixmo: Open weights and open data for state-of-the-art vision-language models. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 91–104. IEEE, 2025.

Guanting Dong, Hangyu Mao, Kai Ma, Licheng Bao, Yifei Chen, Zhongyuan Wang, Zhongxia Chen, Jiazhen Du, Huiyang Wang, Fuzheng Zhang, et al. Agentic reinforced policy optimization. In *International Conference on Learning Representations*, volume 2026, pages 16981–17017, 2026.

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. In *International Conference on Learning Representations*, volume 2026, pages 37909–37926, 2026a.

Mingkuan Feng, Jinyang Wu, Hao Gu, Fangrui Lv, Ruihan Jin, Chuyuan Zhang, Zhengqi Wen, and Jianhua Tao. Taco: Tool-augmented credit optimization for agentic tool use. *arXiv preprint arXiv:2606.30251*, 2026b.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Jiawei Guo, Donglei Yu, Yu Chen, Xiang Wang, Shuai Li, Xinpei Zhao, Huaxing Liu, Qinghao Wang, and Minpeng Liao. Do multimodal agents really benefit from tool use? a systematic study of capability gains. *arXiv preprint arXiv:2606.02357*, 2026.

Yang He, Xiao Ding, Bibo Cai, Yufei Zhang, Kai Xiong, Zhouhao Sun, Bing Qin, and Ting Liu. Deeptool: Scaling interleaved deliberation in tool-integrated reasoning via process-supervised reinforcement learning. *arXiv preprint arXiv:2605.29568*, 2026.

Jack Hong, Chenxiao Zhao, ChengLIn Zhu, Weiheng Lu, Guohai Xu, and XingYu. Deepeyesv2: Toward agentic multimodal model. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust, editors, *International Conference on Learning Representations*, volume 2026, pages 114851–114872, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/badbef7d421640fb038399e9d7271b1c-Paper-Conference.pdf.

Xinhai Hou, Shaoyuan Xu, Manan Biyani, Moyan Li, Jia Liu, Todd C Hollon, and Bryan Wang. Codev: Code with images for faithful visual reasoning via tool-aware policy optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21500–21510, 2026.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*, 2025.

Dong-Hee Kim, Reuben Tan, and Donghyun Kim. Diversity over frequency: Rethinking tool use in visual chain-of-thought agents. *arXiv preprint arXiv:2606.00096*, 2026.

Xin Lai, Junyi Li, Wei Li, Tao Liu, Tianjian Li, and Hengshuang Zhao. Mini-o3: Scaling up reasoning patterns and interaction turns for visual search. In *International Conference on Learning Representations*, volume 2026, pages 76722–76746, 2026.

Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, Feng Li, Hao Zhang, Kaichen Zhang, Peiyuan Zhang, Yanwei Li, Ziwei Liu, et al. Llava-onevision: Easy visual task transfer. *arXiv preprint arXiv:2408.03326*, 2024.

Xuchen Li, Xuzhao Li, Jiahui Gao, Renjie Pi, Shiyu Hu, and Wentao Zhang. Look less, reason more: Rollout-guided adaptive pixel-space reasoning. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4948–4967, 2026.

Pan Lu, Ran Gong, Shibiao Jiang, Liang Qiu, Siyuan Huang, Xiaodan Liang, and Song-Chun Zhu. Inter-gps: Interpretable geometry problem solving with formal language and symbolic reasoning. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6774–6786, 2021.

Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. Mathvista: Evaluating mathematical reasoning of foundation models in visual contexts. In *International Conference on Learning Representations*, volume 2024, pages 23439–23554, 2024.

Yan Ma, Weiyu Zhang, Tianle Li, Linge Du, Xuyang Shen, and Pengfei Liu. What does vision tool-use reinforcement learning really learn? disentangling tool-induced and intrinsic effects for crop-and-zoom. *arXiv preprint arXiv:2602.01334*, 2026.

Delin Mao, Chenghao Sun, Jingwei Song, Chishui Chen, and Linfeng Zhang. Toolvision: Learning when and how to use visual tools with capability-aligned supervision. *arXiv preprint arXiv:2608.08907*, 2026.

Ahmed Masry, Jia Qing Tan, Shafiq Joty, Enamul Hoque, et al. Chartqa: A benchmark for question answering about charts with visual and logical reasoning. In *Findings of the association for computational linguistics: ACL 2022*, pages 2263–2279, 2022.

Fanqing Meng, Lingxiao Du, Zongkai Liu, Zhixiang Zhou, Quanfeng Lu, Daocheng Fu, Tiancheng Han, Botian Shi, Wenhai Wang, Junjun He, et al. Mm-eureka: Exploring the frontiers of multimodal reasoning with rule-based reinforcement learning. *arXiv preprint arXiv:2503.07365*, 2025.

Cheng Qian, Emre Can Acikgoz, Qi He, Hongru WANG, Xiusi Chen, Dilek Hakkani-Tur, Gokhan Tur, and Heng Ji. Toolrl: Reward is all tool learning needs. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen, editors, *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pages 105523–105553. Curran Associates, Inc., 2025. doi: 10.52202/085713-3524. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/97c5b2707228e7e3fb67e4ecc2e0e607-Paper-Conference.pdf.

Runqi Qiao, Qiuna Tan, Guanting Dong, Minhui Wu, Chong Sun, Xiaoshuai Song, Jiapeng Wang, Zhuoma GongQue, Shanglin Lei, YiFan Zhang, Zhe Wei, Miaoxuan Zhang, Runfeng Qiao, Xiao Zong, Yida Xu, Peiqing Yang, Zhimin Bao, Muxi Diao, Chen Li, and Honggang Zhang. We-math: Does your large multimodal model achieve human-like mathematical reasoning? In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 20023–20070, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.983. URL https://aclanthology.org/2025.acl-long.983/.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in neural information processing systems*, 36:68539–68551, 2023.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Qi Song, Honglin Li, Yingchen Yu, Haoyi Zhou, Lin Yang, Song Bai, Qi She, Zilong Huang, and Yunqing Zhao. Codedance: A dynamic tool-integrated mllm for executable visual reasoning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19186–19195, 2026.

Alex Su, Haozhe Wang, Weiming Ren, Fangzhen Lin, and Wenhu Chen. Pixel reasoner: Incentivizing pixel space reasoning via curiosity-driven reinforcement learning. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen, editors, *Advances in Neural Information Processing Systems*,

volume 38, Main Conference, pages 8222–8251. Curran Associates, Inc., 2025. doi: 10.52202/085713-0280. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/0c38f54740062529aa4117a04b583f3c-Paper-Conference.pdf.

Zhiheng Wang, Bo Peng, Lai Wei, and Chaochao Lu. The illusion of visual tool-use: A causal audit of thinking with images. *arXiv preprint arXiv:2608.06270*, 2026.

Zirui Wang, Mengzhou Xia, Luxi He, Howard Chen, Yitao Liu, Richard Zhu, Kaiqu Liang, Xindi Wu, Haotian Liu, Sadhika Malladi, et al. Charxiv: Charting gaps in realistic chart understanding in multimodal llms. *Advances in Neural Information Processing Systems*, 37:113569–113697, 2024.

Mingyuan Wu, Jingcheng Yang, Jize Jiang, Meitang Li, Kaizhuo Yan, Hanchao Yu, Minjia Zhang, Chengxiang Zhai, and Klara Nahrstedt. Vtool-r1: Vlms learn to think with images via reinforcement learning on multimodal tool use. In *International Conference on Learning Representations*, volume 2026, pages 78298–78319, 2026.

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. doi: 10.1109/CVPR52733.2024.01243. URL https://doi.org/10.1109/CVPR52733.2024.01243.

Yutao Xie, Nathaniel Thomas, Nick Hansen, Yang Fu, Li Li, and Xiaolong Wang. Tips: Turn-level information-potential reward shaping for search-augmented llms. In *International Conference on Learning Representations*, volume 2026, pages 156549–156584, 2026.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

Yihang Yao, Guangtao Zeng, Raina Wu, Yang Zhang, Ding Zhao, Zhang-Wei Hong, and Chuang Gan. Tailored primitive initialization is the secret key to reinforcement learning. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 33300–33318, 2026.

Yaoqi Ye, Yiran Zhao, Keyu Duan, Zeyu Zheng, Kenji Kawaguchi, Cihang Xie, and Michael Qizhe Shieh. In-context reinforcement learning for tool use in large language models. *arXiv preprint arXiv:2603.08068*, 2026.

Dylan Zhang, Yufeng Xu, Haojin Wang, Qingzhi Chen, and Hao Peng. Good sft optimizes for sft, better sft prepares for reinforcement learning. *arXiv preprint arXiv:2602.01058*, 2026a.

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

Xuqin Zhang, Quan He, Zhenrui Zheng, Zongzhang Zhang, Xu He, and Dong Li. Aster: Agentic scaling with tool-integrated extended reasoning. *arXiv preprint arXiv:2602.01204*, 2026b.

Yi-Fan Zhang, Xingyu Lu, Shukang Yin, Chaoyou Fu, Wei Chen, Xiao Hu, Bin Wen, Kaiyu Jiang, Changyi Liu, Tianke Zhang, Haonan Fan, Kaibing Chen, Jiankang Chen, Haojie Ding, Kaiyu Tang, Zhang Zhang, Liang Wang, Fan Yang, Tingting Gao, and Guorui Zhou. Thyme: Think beyond images. *arXiv preprint arXiv:2508.11630*, 2025b. cs.CV.

Ziwei Zheng, Minghao Yang, Jack Hong, Chenxiao Zhao, Guohai Xu, Le Yang, Chao Shen, and XingYu. Deepeyes: Incentivizing "thinking with images" via reinforcement learning. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust, editors, *International Conference on Learning Representations*, volume 2026, pages 126775–126798, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/cdb347d7516d52a9280ea9d6708911f6-Paper-Conference.pdf.

A Omitted Proof

We condition throughout on a fixed question x and use the notation of Section 3.4. The finite trajectory set Ω and terminal reward $R : \Omega \to \{0, 1\}$ are fixed. Both initializations assign positive probability to every trajectory in Ω, and both $\mathcal{T}$ and $\mathcal{D}$ are nonempty.

A.1 Derivation of Eq. 1

We derive Eq. 1 from an idealized KL-proximal update over trajectory distributions. This update is used only to isolate reward-driven probability reallocation; it is not an exact reformulation of the GRPO objective or its reference-policy KL regularization.

For clarity, we omit the initialization index i. Let $p(\tau, t)$ denote the probability assigned to trajectory τ at optimization time t, and let $R(\tau) \in \{0, 1\}$ denote its fixed terminal correctness reward.

For an infinitesimal update interval $h > 0$, let $q(\tau)$ denote the candidate trajectory distribution for the next update and consider

$$q^\star = \arg\max_{q \in \Delta(\Omega)} \left\{ \sum_{\tau \in \Omega} q(\tau) R(\tau) - \frac{1}{h} D_{\mathrm{KL}}(q \| p(\cdot, t)) \right\}, \tag{6}$$

where $\Delta(\Omega)$ is the probability simplex. We then set

$$p(\tau, t + h) = q^\star(\tau).$$

Using

$$D_{\mathrm{KL}}(q \| p(\cdot, t)) = \sum_{\tau \in \Omega} q(\tau) \log \frac{q(\tau)}{p(\tau, t)},$$

and introducing a Lagrange multiplier α for the constraint $\sum_\tau q(\tau) = 1$, we obtain

$$\mathcal{L}(q, \alpha) = \sum_\tau q(\tau) R(\tau) - \frac{1}{h} \sum_\tau q(\tau) \log \frac{q(\tau)}{p(\tau, t)} + \alpha \left(\sum_\tau q(\tau) - 1 \right).$$

Differentiating with respect to $q(\tau)$ gives

$$R(\tau) - \frac{1}{h} \left(\log \frac{q(\tau)}{p(\tau, t)} + 1 \right) + \alpha = 0.$$

Rearranging,

$$\log \frac{q(\tau)}{p(\tau, t)} = hR(\tau) + h\alpha - 1.$$

The last two terms are independent of τ and can therefore be absorbed into the normalization constant. Hence,

$$p(\tau, t + h) = \frac{p(\tau, t) e^{hR(\tau)}}{Z(t, h)}, \tag{7}$$

where

$$Z(t, h) = \sum_{\tau'} p(\tau', t) e^{hR(\tau')}.$$

For $h \to 0$,

$$e^{hR(\tau)} = 1 + hR(\tau) + o(h).$$

Therefore,

$$\begin{aligned} Z(t,h) &= \sum_{\tau'} p(\tau', t) \left(1 + hR(\tau') + o(h)\right) \\ &= 1 + h \sum_{\tau'} p(\tau', t) R(\tau') + o(h) \\ &= 1 + hA(t) + o(h), \end{aligned}$$

where

$$A(t) = \sum_{\tau'} p(\tau', t) R(\tau')$$

is the expected final-answer accuracy.

Substituting this expansion into Eq. 7,

$$p(\tau, t+h) = p(\tau, t) \frac{1 + hR(\tau) + o(h)}{1 + hA(t) + o(h)}.$$

Thus,

$$\frac{p(\tau, t+h) - p(\tau, t)}{h} = p(\tau, t) \frac{R(\tau) - A(t) + o(1)}{1 + hA(t) + o(h)}.$$

Taking $h \to 0$ gives

$$\frac{\partial p(\tau, t)}{\partial t} = \big(R(\tau) - A(t)\big) p(\tau, t).$$

A.2 Proof of Theorem 1

Proof. We first derive the dynamics of overall accuracy, tool-conditioned accuracy, and tool initiation, and then show that the accuracy ordering induced by the SFT initialization is preserved.

Overall accuracy. By definition,

$$A_i(t) = \sum_{\tau \in \Omega} p_i(\tau, t) R(\tau).$$

Since $R(\tau)$ is fixed,

$$\begin{aligned} \dot{A}_i(t) &= \sum_{\tau} R(\tau) \frac{\partial p_i(\tau, t)}{\partial t} \\ &= \sum_{\tau} p_i(\tau, t) R(\tau) \big(R(\tau) - A_i(t)\big) \\ &= \sum_{\tau} p_i(\tau, t) R(\tau)^2 - A_i(t) \sum_{\tau} p_i(\tau, t) R(\tau). \end{aligned}$$

Because $R(\tau) \in \{0, 1\}$, we have $R(\tau)^2 = R(\tau)$. Both sums therefore equal $A_i(t)$, giving

$$\dot{A}_i(t) = A_i(t)\big(1 - A_i(t)\big).$$

Tool-initiation dynamics. Define the probability mass of correct tool-using trajectories as

$$N_i(t) := \sum_{\tau \in \mathcal{T}} p_i(\tau, t) R(\tau).$$

By the definition of conditional accuracy,

$$N_i(t) = I_i(t) C_i(t).$$

Since

$$I_i(t) = \sum_{\tau \in \mathcal{T}} p_i(\tau, t),$$

Eq. 1 gives

$$\begin{aligned} \dot{I}_i(t) &= \sum_{\tau \in \mathcal{T}} \frac{\partial p_i(\tau, t)}{\partial t} \\ &= \sum_{\tau \in \mathcal{T}} p_i(\tau, t) \big(R(\tau) - A_i(t)\big) \\ &= N_i(t) - A_i(t) I_i(t) \\ &= I_i(t) \big(C_i(t) - A_i(t)\big). \end{aligned}$$

Using

$$A_i(t) = I_i(t) C_i(t) + \big(1 - I_i(t)\big) D_i(t),$$

we obtain

$$\begin{aligned} C_i(t) - A_i(t) &= C_i(t) - I_i(t) C_i(t) - \big(1 - I_i(t)\big) D_i(t) \\ &= \big(1 - I_i(t)\big) \big(C_i(t) - D_i(t)\big). \end{aligned}$$

Therefore,

$$\dot{I}_i(t) = I_i(t) \big(1 - I_i(t)\big) \big(C_i(t) - D_i(t)\big).$$

Tool-conditioned accuracy. Differentiating $N_i(t)$ gives

$$\begin{aligned} \dot{N}_i(t) &= \sum_{\tau \in \mathcal{T}} R(\tau) \frac{\partial p_i(\tau, t)}{\partial t} \\ &= \sum_{\tau \in \mathcal{T}} p_i(\tau, t) R(\tau) \big(R(\tau) - A_i(t)\big). \end{aligned}$$

Again using $R(\tau)^2 = R(\tau)$,

$$\begin{aligned} \dot{N}_i(t) &= \sum_{\tau \in \mathcal{T}} p_i(\tau, t) R(\tau) - A_i(t) \sum_{\tau \in \mathcal{T}} p_i(\tau, t) R(\tau) \\ &= N_i(t) \big(1 - A_i(t)\big). \end{aligned}$$

Since

$$C_i(t) = \frac{N_i(t)}{I_i(t)},$$

we have

$$\dot{C}_i(t) = \frac{\dot{N}_i(t) I_i(t) - N_i(t) \dot{I}_i(t)}{I_i(t)^2}.$$

Substituting the expressions above,

$$\begin{aligned}\dot{C}_i(t) &= \frac{N_i(t)(1-A_i(t))I_i(t) - N_i(t)I_i(t)(C_i(t)-A_i(t))}{I_i(t)^2} \\ &= \frac{N_i(t)}{I_i(t)}\left[(1-A_i(t)) - (C_i(t)-A_i(t))\right] \\ &= C_i(t)(1-C_i(t)).\end{aligned}$$

Preservation of initialization ordering. From the previous derivation, we know that both $A_i(t)$ and $C_i(t)$ satisfy the scalar differential equation

$$\dot{y}(t) = y(t)(1-y(t)).$$

Consider first an initial value $y(0) = y_0 \in (0,1)$. Separating the variables gives

$$\frac{dy}{y(1-y)} = dt.$$

Integrating the left-hand side with respect to y from y_0 to $y(t)$, and the right-hand side with respect to time from 0 to t, we obtain

$$\int_{y_0}^{y(t)} \frac{1}{y(1-y)}\,dy = \int_0^t ds.$$

The left-hand side is

$$\begin{aligned}\int_{y_0}^{y(t)} \frac{1}{y(1-y)}\,dy &= \int_{y_0}^{y(t)} \left(\frac{1}{y} + \frac{1}{1-y}\right) dy \\ &= \left[\log y - \log(1-y)\right]\Big|_{y_0}^{y(t)} \\ &= \log\frac{y(t)}{1-y(t)} - \log\frac{y_0}{1-y_0}.\end{aligned}$$

Therefore,

$$\log\frac{y(t)}{1-y(t)} - \log\frac{y_0}{1-y_0} = t.$$

Exponentiating both sides gives

$$\frac{y(t)}{1-y(t)} = \frac{y_0}{1-y_0}e^t.$$

Solving for $y(t)$ yields

$$y(t) = \frac{y_0 e^t}{1-y_0+y_0e^t}.$$

Define

$$F_t(u) = \frac{ue^t}{1-u+ue^t}.$$

For every $t \geq 0$, we have

$$\frac{\partial F_t(u)}{\partial u} = \frac{e^t}{(1-u+ue^t)^2} > 0.$$

Therefore, F_t is increasing in its initial value. It follows that

$$A_1(0) \geq A_2(0) \quad \Longrightarrow \quad A_1(t) \geq A_2(t),$$

and similarly,

$$C_1(0) \geq C_2(0) \quad \Longrightarrow \quad C_1(t) \geq C_2(t)$$

for all $t \geq 0$. □

B Prompts

TOOL-OFF PROMPT

```
<|im_start|>system
You are a careful multimodal reasoner. Solve the problem from the image and
  question.
<|im_end|>
<|im_start|>user
<image>
{QUESTION}
{BENCHMARK-SPECIFIC ANSWER-FORMAT INSTRUCTION}
<|im_end|>
<|im_start|>assistant
```

TOOL-ON PROMPT

```
<|im_start|>system
You are a careful multimodal reasoner. Solve the problem from the image and
  question. If tools are available, use them only
when they are genuinely helpful. Use image_zoom_in_tool only for closer visual
  inspection. Use code_interpreter only for
genuine numerical or symbolic computation over quantities and relations you have
  already extracted; it cannot inspect the
original image.

# Tools

You may call one or more functions to assist with the user query.
Function signatures are provided within <tools></tools> XML tags:

<tools>
{
  "function": {
    "description": "Crop and magnify a rectangular region of the input image for
  closer visual inspection.",
    "name": "image_zoom_in_tool",
    "parameters": {
      "properties": {
        "bbox_2d": {
          "type": "array",
          "description": "Bounding box [x1, y1, x2, y2] in original-image pixel
  coordinates."
        },
        "code": null,
        "label": {
```

```
          "type": "string",
          "description": "Optional label for the region."
        }
      },
      "required": ["bbox_2d"],
      "type": "object"
    }
  },
  "type": "function"
}
{
  "function": {
    "description": "Execute Python for genuine computation over quantities and
  relations already extracted from the
    problem; the original image is not an input to this tool.",
    "name": "code_interpreter",
    "parameters": {
      "properties": {
        "bbox_2d": null,
        "code": {
          "type": "string",
          "description": "Python source for numerical or symbolic computation."
        },
        "label": null
      },
      "required": ["code"],
      "type": "object"
    }
  },
  "type": "function"
}
</tools>

Return each function call as JSON within <tool_call></tool_call> XML tags:

<tool_call>
{"name": <function-name>, "arguments": <args-json-object>}
</tool_call>
<|im_end|>
<|im_start|>user
<image>
{QUESTION}
{BENCHMARK-SPECIFIC ANSWER-FORMAT INSTRUCTION}
<|im_end|>
<|im_start|>assistant
```

C Experimental Configuration

C.1 SFT Configuration

All SFT experiments initialize from Qwen2.5-VL-7B-Instruct. For the original `Low`, `Medium`, and `High` settings, each training set contains 2,000 question–trajectory pairs containing 750 Zoom trajectories, 750 Code trajectories, and 500 shared tool-free ones. `Low` contains one-call tool trajectories, `Medium` contains two-to-three-call trajectories, and `High` contains trajectories with at least four calls. The three datasets have approximately matched total token budgets to ensure a fair comparison on the SFT training stage.

We train each model for one epoch, corresponding to 50 optimizer updates, using AdamW with a global batch size of 40. The learning rate peaks at 1×10^{-5} and is cosine-decayed to 1×10^{-6}. The SFT objective is computed only over assistant-generated tokens, including natural-language reasoning, structured tool calls, and final answers. System messages, user messages, and tool observations are masked from the loss. All three settings use the same prompt templates, tool schemas, optimizer configuration, and training schedule.

For the length-filtering experiments in Section 4.1, we retain complete trajectories from the original `High` dataset without modifying their contents. Length-Filtered (50%) and Length-Filtered (25%) retain approximately 50% and 25% of the original assistant-supervised token mass per epoch, corresponding to approximately 36% and 16% of the trajectories, respectively. To approximately match the cumulative assistant-supervised token exposure of the original `High` while keeping the optimization budget fixed, all variants use exactly 50 optimizer updates. The original `High` is trained for one epoch with a global batch size of 40. Length-Filtered (50%) is trained for two epochs with a global batch size of 29, and Length-Filtered (25%) is trained for four epochs with a global batch size of 26. The optimizer, learning-rate schedule, warmup, weight decay, loss masking, and all other SFT settings are kept unchanged.

C.2 RL Configuration

Starting from each SFT checkpoint, we apply the same outcome-only GRPO (Shao et al., 2024) procedure. Training runs for 32 RL updates with a learning rate of 1×10^{-6}. At each update, we sample 512 training questions and generate five rollouts per question. All model variants use the same RL questions, question ordering, sampling configuration, and decoding settings.

Each rollout may invoke Zoom and Code for at most eight tool calls and has an 8,192-token response budget. The reward is determined solely by final-answer correctness. We do not provide auxiliary rewards for tool use, tool-call depth, pre-call reasoning length, or intermediate reasoning quality. Consequently, any change in tool-use behavior during RL arises from optimizing final task success rather than from explicit supervision of tool frequency.

C.3 Evaluation Configuration

We report two inference settings. In `tool-on`, both Zoom and Code are available, with the same maximum of eight tool calls and 8,192-token response budget used during RL. The model decides itself whether to call tools and when to call them. It may invoke multiple tools or simply perform a tool-free rollout based on its own decision. In `tool-off`, the tool schemas are removed and the model must answer directly. Final accuracy is computed from the model's final answer under the benchmark-specific evaluation protocol.

For tool-behavior analysis, we define a *tool-using trajectory* as one containing at least one actual execution attempt of a supported Zoom or Code call. The *tool initiation rate* is the fraction of evaluation examples satisfying this criterion. We further report *tool-conditioned accuracy*, defined as final-answer accuracy restricted to these tool-using trajectories. Because different policies may choose to invoke tools on different questions, this conditional metric is descriptive and should not be interpreted as a same-question causal estimate of the benefit of tool use. Malformed calls or unknown tool names that do not result in an execution are not counted as successful tool initiations.

D Additional Results

Tables 4 and 5 provide the benchmark-level breakdown of the aggregate results reported in Table 1. RL improves final accuracy broadly across benchmarks for all three SFT initializations, despite their substantially different tool-use dynamics discussed in Section 3. The detailed results also show that the aggregate trends are not driven by a single benchmark: after RL, the three initializations achieve broadly comparable `tool-on` performance on several benchmarks, while `tool-off` performance remains competitive.

Table 4: Tool-on accuracy (%) of the `Low`, `Medium`, and `High` SFT and RL checkpoints. Overall reports pooled accuracy across all 6,431 evaluation examples.

Checkpoint	V*	ChartQA	MathVista	WeMath	CharXiv	Overall
Low SFT	79.06	71.80	57.90	54.89	33.60	59.34
Low RL	83.25	84.72	65.60	58.10	40.80	67.67
Medium SFT	54.97	53.96	48.80	53.05	18.50	47.43
Medium RL	85.34	83.40	66.90	62.24	35.10	67.66
High SFT	26.70	48.00	47.30	55.57	10.30	43.45
High RL	84.29	85.48	65.10	57.30	31.40	66.24

Table 5: Tool-off accuracy (%) of the `Low`, `Medium`, and `High` SFT and RL checkpoints. Overall reports pooled accuracy across all 6,431 evaluation examples.

Checkpoint	V*	ChartQA	MathVista	WeMath	CharXiv	Overall
Low SFT	75.39	74.04	61.20	55.17	26.90	59.65
Low RL	83.77	86.16	69.00	56.72	36.00	67.66
Medium SFT	80.63	76.44	62.60	55.80	33.40	62.14
Medium RL	85.34	87.88	68.50	62.93	39.80	70.56
High SFT	79.58	78.20	65.10	55.46	36.20	63.52
High RL	85.86	87.56	66.90	60.98	34.80	68.90