Do Multimodal Agents Really Benefit from Tool Use? A Systematic Study of Capability Gains

Anonymous ACL submission

Abstract

Tool-augmented multimodal agents show strong benchmark gains, often taken as evidence that agents have learned to use tools. We argue that this interpretation can be premature: a tool-call trace alone does not show whether the tool supplied answer-critical information. We study two representative "thinking with images" agents, Thyme and DeepEyesV2, across real-world understanding, OCR, chart understanding, and mathematical reasoning. Each agent is compared with its Tool-Free counterpart and with a Pure-Text Reasoner trained from the same source pool without tool-calling trajectories. Tool access yields little consistent aggregate improvement, does not reliably reduce generated-token cost, and leaves only a small tool-only solved set: 93% of DeepEyesV2's tool-solved problems and 96% of Thyme's are also solved by at least one non-tool setting. Mechanism ablations further show that the full tool-use loop does not consistently outperform either the tool-call format or the returned execution result alone. In the settings we study, the analyzed agents appear to learn tool-calling patterns more reliably than tool-contributed capabilities, suggesting that evaluation should distinguish tool availability from whether tools actually expand what agents can solve.

1 Introduction

A multimodal agent is asked what material a glove in a street-food image is made of. It writes a Python snippet, calls the code interpreter, and answers. Yet the snippet does little: it simply reloads and displays the original image, with no cropping, zooming, or visual processing. A pure-text model, given no tools, answers correctly on the first try. The trajectory contains a tool call, but the call supplies no new visual evidence.

This is not an isolated curiosity. Recent multimodal systems increasingly use intermediate visual or computational operations—including OCR, code execution, region inspection, visual manipulation, or code-mediated reasoning—and their benchmark gains are often linked to these operations (Wu et al., 2025; Liu et al., 2025; Wang et al., 2025b; Wei et al., 2025; Li et al., 2025b). Yet observing such an operation only shows that the model performed one; whether it supplied missing information, or whether the model would have failed without it, is a separate question. We therefore ask: across instances, does tool access actually expand the set of problems an agent can solve?

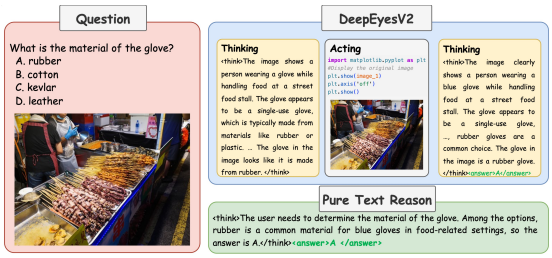


Figure 1: A motivating example illustrating that tool interaction can become procedural rather than genuinely helpful: the agent performs extra tool-mediated steps, yet this added interaction does not necessarily improve the final reasoning outcome.

To examine this question, we first place tool-augmented thinking-with-images agents in a broader benchmark comparison. Across real-world understanding, OCR, chart understanding, and mathematical reasoning benchmarks, we compare several recent tool-augmented multimodal systems with non-tool reasoning baselines (Bai et al., 2025b; Wang et al., 2025a; Meng et al., 2025) and a tool-free Pure-Text Reasoner trained on filtered data from the DeepEyesV2 (Hong et al., 2026) source pool. The comparison shows that tool-augmented agents often achieve strong results, but similar performance can also be obtained without tool-enabled inference.

We then take a closer look at two representative thinking-with-images agents, Thyme (Zhang

et al., 2026b) and DeepEyesV2, through solved-set, process-level, and trajectory-ablation analyses. We choose these two systems because they are among the strongest recent open-source agents in our benchmark suite, expose executable tool-use interfaces, and provide sufficiently complete model and inference resources to support fine-grained diagnostic analysis. This choice lets us move beyond aggregate benchmark scores and ask whether tool access changes which examples the agents can solve. Importantly, our main attribution results do not rely solely on the Pure-Text Reasoner: they are also supported by the Tool-Free Agent, which uses the released agent itself with tool calls suppressed only at inference, and by format-only/result-only ablations that require no retraining.

Three findings emerge. First, tools do not consistently improve accuracy or reduce generated-token cost: removing tool access leaves performance largely unchanged, while the Pure-Text Reasoner remains competitive and sometimes stronger. Second, the tool-only solved set is small: 93.2% of DeepEyesV2's tool-solved problems and 96.0% of Thyme's are also solved by at least one non-tool setting. Third, the agents show no stable capability-expanding tool use: DeepEyesV2 is better preserved by the tool-call format, whereas Thyme is better preserved by the returned result, yet neither pattern yields stable solved-set expansion.

These findings do not imply that tools are useless. Rather, they suggest that the agents studied here learn the protocol of tool calling more reliably than the conditions under which tool outputs change the answer. This matters for evaluation: a correct answer with a tool-call trace is not, by itself, evidence that the tool mattered. Distinguishing tool availability from tool-contributed capability requires asking whether tool use changes what a model can solve.

We make three contributions. First, we reframe tool-use evaluation around whether tool access expands an agent's solvable region, comparing each Tool-Enabled Agent with its inference-time Tool-Free counterpart and a separately trained Pure-Text Reasoner constructed without tool-calling trajectories. Second, we pair sample-level decomposition of tool-call outcomes—preserved successes, unrepaired failures, effective corrections, and induced errors—with process-level attribution of what tool outputs actually contribute. Third, we introduce format-only and result-only trajectory ablations to separate dependence on the tool-calling protocol from dependence on the returned execution result.

2 Related Works

2.1 Tool-Augmented Multimodal Reasoning

A growing line of work moves multimodal models from passive image understanding toward active visual reasoning, often described as "thinking with images" (OpenAI, 2025). Training-free systems orchestrate vision modules through executable programs (Gupta and Kembhavi, 2023; Surís et al., 2023) or produce visual artifacts as intermediate reasoning steps (Hu et al., 2024). Recent trained systems teach multimodal models to invoke tools directly, covering end-to-end RL elicitation (Zheng et al., 2026), pixel-space and zoom-and-focus operations (Su et al., 2026a; Zhang et al., 2025), query-conditional tool selection (Huang et al., 2025), interleaved visual editing (Wu et al., 2026), deep multi-turn trajectories (Lai et al., 2026), and refinements in trajectory synthesis, tool-grounded reward modeling, and experience reuse (Ashraf et al., 2025; Ding et al., 2026; Wang et al., 2026b). The two systems we study, Thyme (Zhang et al., 2026b) and DeepEyesV2 (Hong et al., 2026), follow this paradigm by using code execution for image manipulation and numerical computation during reasoning. Prior work asks how to improve tool use; we ask whether tool access expands what agents can solve.

2.2 Faithfulness and Necessity of Intermediate Reasoning

A parallel literature questions whether tool-mediated reasoning steps are faithful to, or necessary for, the final answer (Turpin et al., 2023). MME-CoT (Jiang et al., 2025) shows that high final-answer accuracy can coexist with logically broken intermediate steps, and Faithful-First RPA (Li et al., 2025a) uses faithfulness signals to plan tool actions during inference. Closer to our setting, MAPO (Yang et al., 2026) identifies a reasoning–action discrepancy in tool-using multimodal agents under outcome-based rewards, and Tool-Use Tax (Zhang et al., 2026a) decomposes prompt formatting, protocol overhead, and execution gain in LLM agents, finding that protocol cost can offset tool gain under semantic noise. These works propose corrective training or inference-time mechanisms; we instead provide a sample-level empirical characterization of when tool calls fail to expand the set of problems already solvable without tools.

Table 1: Main perception benchmark results across real-world understanding, OCR, and chart-related tasks. The bold numbers indicate the best performance on each benchmark, and the underlined numbers indicate the second best.

Model	Tool	Real-World Understanding				OCR		Chart		
		V* Bench	HRBench 4K	HRBench 8K	Tree Bench	OCR Bench	SEED 2 PLUS	CharXiv descriptive	CharXiv reasoning	Chart QA
Qwen2.5-VL	✗	78.5	71.6	67.9	37.0	864	70.4	72.7	40.2	86.2
Pixel-Reasoner	Crop	84.3	74.0	66.9	39.0	-	-	-	-	-
DeepEyes	Crop	**85.6**	75.1	**72.6**	37.5	-	-	-	-	-
Thyme	Code	82.7	77.0	72.5	38.8	865	70.2	73.3	43.8	87.7
Thyme	Tool-Free	83.8	78.5	72.3	37.5	875	69.7	73.3	43.0	87.2
DeepEyesV2	General	82.2	76.1	72.3	41.5	814	69.8	75.7	45.9	**88.1**
DeepEyesV2	Tool-Free	80.6	74.8	71.1	33.1	866	**71.2**	77.0	**46.2**	83.3
Pure-Text Reasoner	✗	83.2	**77.3**	72.5	**42.2**	**888**	70.7	**77.5**	**46.2**	83.4

2.3 Benchmarks for Agentic Multimodal Reasoning

Recent benchmarks broaden multimodal evaluation toward agentic settings that explicitly stress tool use, including agentic thinking-with-images across diverse tasks (Li et al., 2025b), long-horizon hybrid tool use across many sub-domains (Su et al., 2026b), and knowledge-intensive geolocation reasoning with expert-annotated stepwise traces (Geng et al., 2026). These benchmarks make tool use more central to evaluation, but their primary goal is to measure performance under harder or more process-aware tasks. We ask a complementary question: is the observed tool-enabled performance actually attributable to tool access?

3 Study Design

3.1 Study Settings

This section describes the five settings used throughout the paper, the two multimodal agents they are applied to, the benchmarks covered, and how the Pure-Text Reasoner is trained. Three settings test whether tool access expands the set of problems an agent can solve; the remaining two isolate whether behavior depends more on the model-generated tool-call message or on the returned execution result.

3.2 Comparison Settings

Tool-Enabled Agent. The trained multimodal agent is run with its default tool configuration, exactly as released. This is the reference point against which all other settings are compared.

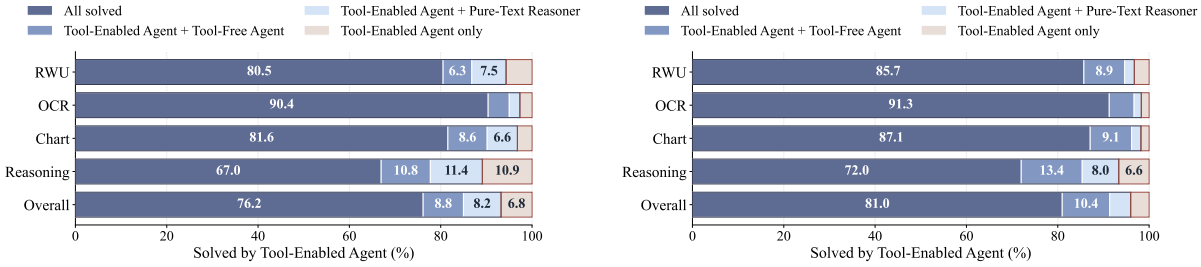
Tool-Free Agent. The same agent model is run with a prompt that forbids code generation and instructs text-only reasoning. This is an out-of-distribution diagnostic setting rather than a matched training control: if the released agent's successes depend strongly on executed tools, suppressing tool calls should produce a visible drop.

Pure-Text Reasoner. A separate baseline trained under a tool-free regime, initialized from the same backbone family and using data drawn from the same source pool as the agents under study. Unlike the Tool-Free Agent, which was trained with tools and has tool access suppressed only at inference time, this model has never seen a tool-calling trajectory. It therefore serves as a tool-free training reference: it estimates what can be achieved from processed data drawn from the same source pool without learning a tool-calling interface. It is not a strict training-procedure control or an upper-bound baseline, but an auxiliary reference for solved-set attribution.

The remaining two settings isolate which part of the tool-use loop a behavior depends on by intervening on already-generated trajectories rather than retraining the model. Both Thyme and DeepEyesV2 invoke tools by generating executable code and folding the returned execution result — an edited image, a numerical value, or parsed text — back into the reasoning chain. In *Tool Format Only*, we post-hoc edit each tool-enabled trajectory by retaining the model-generated tool-call message, including any executable code, and replacing the returned execution result with an empty placeholder before the trajectory re-enters the model for the final answer. In *Tool Result Only*, we do the reverse: the tool-call message is removed from the trajectory, while the corresponding execution result is retained at the position where it would have appeared. Both interventions operate on already-generated trajectories; no retraining is involved. Their contrast reveals whether behavior tracks the

Table 2: Main math reasoning benchmark results across six representative benchmarks. The bold numbers indicate the best performance on each benchmark, and the underlined numbers indicate the second best.

Model	Tool	Math Reasoning					
		MathVista	MathVerse	MathVision	WeMath	DynaMath	LogicVista
Qwen2.5-VL	✗	68.3	45.6	25.6	34.6	53.3	45.9
MM-Eureka	✗	72.6	-	28.1	21.8	-	46.3
VL-Rethinker	✗	73.7	-	28.4	36.3	-	42.7
DeepEyes	Crop	70.1	47.3	26.6	38.9	55.0	47.7
Thyme	Code	71.9	47.3	25.5	37.5	54.1	48.3
Thyme	Tool-Free	71.7	41.6	24.9	37.5	54.3	47.4
DeepEyesV2	General	72.5	**52.2**	28.4	37.8	**57.6**	48.1
DeepEyesV2	Tool-Free	73.8	48.6	28.5	43.1	57.9	49.2
Pure-Text Reasoner	✗	**74.6**	51.6	**29.3**	**46.3**	56.8	**49.2**



(a) Solved-set coverage for DeepEyesV2.

(b) Solved-set coverage for Thyme.

Figure 2: Solved-set coverage of Tool-Enabled Agents relative to non-tool settings across task families. Most problems solved by the Tool-Enabled Agent are also solved by at least one non-tool setting, while the tool-only solved region remains comparatively small across all task families. Each bar is normalized over the samples correctly solved by the corresponding Tool-Enabled Agent. "Tool-Enabled only" denotes samples missed by both the Tool-Free Agent and the Pure-Text Reasoner; "All solved" denotes samples solved by all three settings.

call it produces, the result it receives, or both.

We study Thyme and DeepEyesV2. Both generate executable code during reasoning, run it with an external interpreter, and return the result to the trajectory, but differ in their tool operations and training pipelines. Both provide open-source training and inference code, enabling controlled comparisons, and both are competitive on the benchmarks considered. They are not exhaustive, but cover representative realizations of the current "thinking with images" paradigm.

3.3 Benchmarks

Evaluation spans four families of multimodal tasks, chosen so that conclusions are not pinned to any single capability profile: *real-world understanding* (Wu and Xie, 2024; Wang et al., 2025c, 2026a), *OCR* (Liu et al., 2024; Li et al., 2024), *chart understanding* (Wang et al., 2024b; Masry et al., 2022), and *mathematical reasoning* (Lu et al., 2024; Zhang et al., 2024; Wang et al., 2024a; Qiao et al., 2025; Zou et al., 2025; Xiao et al., 2024).

3.4 Training the Pure-Text Reasoner

The Pure-Text Reasoner is a diagnostic tool-free reference rather than an upper-bound model. It estimates which successes are attainable from the same source pool without learning a tool-use protocol. To align it with DeepEyesV2, we initialize from Qwen2.5-VL-7B and train on processed data derived from the DeepEyesV2 source pool.

Training has two stages. In SFT, we extract 53K examples from the DeepEyesV2 SFT data. Because the original rationales include tool-call steps, which are incompatible with a pure-text setting, we use Qwen2.5-VL-72B to convert them into text-based reasoning traces without tool use. In RL, we apply a pass@k filter to the DeepEyesV2 RL data for computational feasibility, retaining 30K examples with $1 \leq n \leq k-1$ correct sampled responses, and train with GRPO (Shao et al., 2024) using final-answer correctness as the reward. We use Qwen3-235B-A22B-Instruct-2507 (Yang et al., 2025) to judge final-answer correctness for the reward. Additional details are provided in Appendix A.

Table 3: Task-family average of tool-call outcome decomposition on tool-called samples. Across both agents and both non-tool references, TRR is the dominant outcome, indicating that many tool-called samples are already solved correctly without tools, while TER remains comparatively small. This suggests that tool calls more often preserve existing correctness than convert previously incorrect predictions into correct ones. Rates are computed against both the Tool-Free Agent and the Pure-Text Reasoner.

Model	Task	vs Tool-Free Agent				vs Pure-Text Reasoner			
		TRR	TFR	TER	TIR	TRR	TFR	TER	TIR
DeepEyesV2	Real-World Understanding	59.1	24.8	9.0	7.2	59.9	22.8	8.2	9.2
	OCR	69.7	20.4	3.7	6.2	68.0	19.1	5.3	7.5
	Chart	68.1	17.6	7.4	6.8	66.6	16.3	8.9	8.1
	Reasoning	39.8	37.1	11.4	11.6	40.1	36.5	11.1	12.3
	Overall	52.3	29.0	9.3	9.4	52.0	28.0	9.6	10.4
Thyme	Real-World Understanding	71.0	20.0	2.7	6.3	66.7	18.8	7.1	7.5
	OCR	83.7	13.2	0.5	2.7	81.0	10.0	3.2	5.9
	Chart	62.5	31.2	0.0	6.2	37.5	12.5	25.0	25.0
	Reasoning	31.7	43.1	22.0	3.3	30.9	37.4	22.8	8.9
	Overall	71.3	20.3	4.5	4.0	67.9	17.0	7.8	7.2

4 Experiment Results

4.1 Tool Access Does Not Consistently Expand Solved Sets

If tool access provides a robust capability gain, we would expect it to improve accuracy on tool-relevant benchmarks or reduce the generated-token cost needed to reach comparable answers. In our experiments, neither pattern appears consistently.

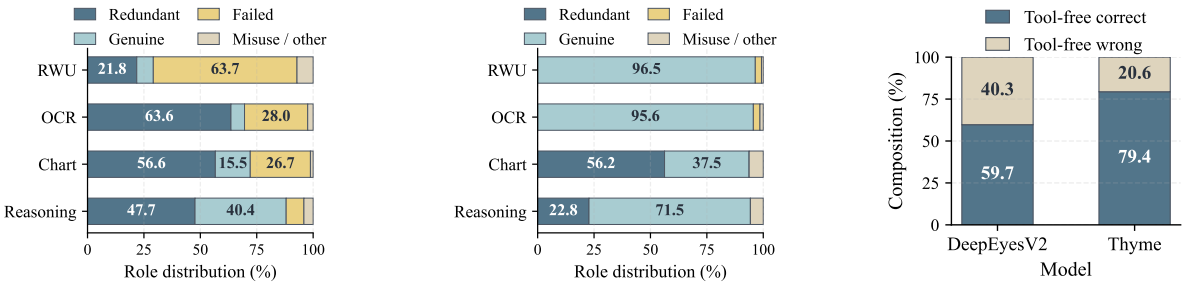
Aggregate accuracy. Across perception and reasoning benchmarks (Tables 1 and 2), the Tool-Enabled Agents are competitive and occasionally top an individual benchmark, but no consistent advantage emerges across task families. Moreover, disabling tools does not lead to a systematic collapse in performance: the Tool-Free variants often remain close to, and occasionally surpass, their tool-enabled counterparts, suggesting that many successes do not depend on executed tool use. On several benchmarks — spanning OCR, charts, and math reasoning — the Pure-Text Reasoner matches or surpasses them, despite being trained on processed data derived from the same source pool without any tool-calling mechanism. Thus, aggregate accuracy does not show a consistent advantage for tool-enabled inference.

Solved-set composition. Aggregate scores can mask which problems an agent actually solves. Two settings can land at the same accuracy by solving largely the same subset, or by solving disjoint ones — and the difference matters: only the latter would suggest tools open up problems otherwise out of reach. This analysis is intended for attribution rather than causal estimation or model ranking: it asks whether tool-enabled successes are also attainable without tool execution under comparable non-tool references. To tell these apart, we condition on the samples each Tool-Enabled Agent solves, and check how many of them are also solved by the Tool-Free Agent or the Pure-Text Reasoner.

Figure 2 shows the breakdown by task family. Across all benchmarks, 93.2% of the problems DeepEyesV2 solves with tools are also solved by at least one non-tool setting; for Thyme the figure is 96.0%. This overlap is not driven solely by the separately trained Pure-Text Reasoner: Appendix Table 7 reports the benchmark-level overlap with the released Tool-Free Agent separately and shows the same qualitative pattern. The pattern holds in every task family. The tool-only solved set — samples correctly answered by the Tool-Enabled Agent but missed by both non-tool settings — remains small overall, with most task-family values in the single digits or low double digits.

No compensating efficiency. Tool-enabled inference also does not consistently reduce generated-token cost. Across task families, Tool-Enabled DeepEyesV2 generates 445 tokens per sample on average, compared with 413 for the Tool-Free Agent and 275 for the Pure-Text Reasoner; for Thyme, the corresponding numbers are 272, 259, and 275. Detailed token counts are reported in Appendix Table 11.

Implications. These results show why aggregate benchmark scores are insufficient for evaluating tool use. Small score differences need not imply



(a) DeepEyesV2.

(b) Thyme.

(c) *Genuine contribution* subset.

Figure 3: Process-level analysis of tool use. Tool calls often reflect redundant confirmation, failed contribution, or contributions to samples that are already solvable without tools, rather than clear capability expansion. The first two subfigures show tool-use role distributions across task families. The third subfigure further shows that even among trajectories labeled as *Genuine contribution*, most samples are already solved by the Tool-Free Agent. Role-distribution sample sizes are: DeepEyesV2 — 2195/3275/7499/15175 and Thyme — 255/410/16/123 for RWU/OCR/Chart/Reasoning, respectively. Small Thyme subsets, especially Chart and Reasoning, should be interpreted descriptively.

Table 4: Per-sample correctness agreement (%) with the Tool-Enabled Agent, computed as the proportion of samples on which the two settings yield the same correctness outcome (both correct or both incorrect). Higher values indicate that the comparison setting more closely matches the correctness pattern of the Tool-Enabled Agent. Parenthesized values report the difference from the Tool-Free Agent.

Task	DeepEyesV2			Thyme		
	Tool-Free	Result Only	Format Only	Tool-Free	Result Only	Format Only
Real-World Understanding	82.9	82.8 (-0.1)	**90.0** (+7.1)	92.4	**99.4** (+7.0)	99.2 (+6.8)
OCR	90.7	92.8 (+2.1)	**93.9** (+3.2)	95.9	**99.0** (+3.2)	96.5 (+0.6)
Chart	83.3	85.8 (+2.5)	**90.8** (+7.5)	93.0	**96.3** (+3.3)	95.9 (+2.9)
Reasoning	76.8	77.7 (+0.8)	**82.9** (+6.0)	84.8	**89.6** (+4.9)	88.8 (+4.0)

that tools open new solvable regions; they may instead reflect changes near an existing decision boundary. This motivates a more local question: when an agent invokes a tool, what does the call actually contribute?

4.2 Tool Calls Confirm More Than They Repair

Section 4.1 showed that tool access does not consistently expand the solved set. We next ask what role tool calls play when they occur, using two complementary views: outcome counts, which show whether tool calling preserves, repairs, or harms correctness, and trajectory attribution, which asks whether the tool output supplies missing information or merely confirms the model's prior inference. Across both views, tool calls are dominated by redundant confirmation and failed repair rather than genuine error correction.

Tool-call outcome decomposition. On tool-called samples, we compare the Tool-Enabled Agent with a non-tool reference and partition outcomes into four mutually exclusive cases: *TRR*, where both settings are correct; *TFR*, where both are wrong; *TER*, where the non-tool reference is wrong but tool calling makes the answer correct; and *TIR*, where the non-tool reference is correct but tool calling makes the answer wrong. Rates are normalized over tool-called samples and sum to 100%. To avoid pinning conclusions to a single reference, we compute them against both the Tool-Free Agent and the Pure-Text Reasoner.

The dominant case, by a wide margin, is *TRR* (Table 3). Against the Tool-Free Agent, overall *TRR* reaches 52.3% for DeepEyesV2 and 71.3% for Thyme, while *TER* stays below 10% for both. Using the Pure-Text Reasoner as the baseline gives the same picture. Most tool calls, in other words, occur on samples that a non-tool reference already answers correctly.

Table 5: Performance comparison across tool-usage modes on Real-World Understanding, OCR, and Chart benchmarks.

Model	Mode	Real-World Understanding				OCR		Chart		
		V* Bench	HRBench 4K	HRBench 8K	Tree Bench	OCR Bench	SEED 2 PLUS	CharXiv descriptive	CharXiv reasoning	Chart QA
DeepEyesV2	Tool-Enabled Agent	82.20	76.13	72.25	41.48	814	69.78	75.68	45.90	88.08
	Tool Format Only	83.77	76.00	70.88	39.01	805	69.78	76.05	44.70	87.08
	Tool Result Only	81.68	73.50	68.75	34.07	828	69.87	73.78	45.90	80.24
Thyme	Tool-Enabled Agent	82.72	77.00	72.50	38.77	865	70.22	73.30	43.80	87.68
	Tool Format Only	83.25	77.25	72.13	39.26	865	70.05	73.15	42.70	87.08
	Tool Result Only	83.25	77.50	72.38	39.26	879	70.36	73.00	42.20	87.56
Pure-Text Reasoner	–	83.25	77.25	72.50	42.22	888	70.66	77.45	46.20	83.44

The four rates are not evenly distributed across task families. On reasoning, the picture flips: *TRR* drops sharply, and *TFR* climbs to 37.1% for DeepEyesV2 and 43.1% for Thyme. Tools are still being called on these samples, but many calls fail to convert incorrect predictions into correct ones. This is precisely where the outcome decomposition is most diagnostic: it shows that tool calls often do not repair the errors they are expected to address. The full benchmark-level decomposition is reported in Appendix Table 8.

Process attribution. Outcome counts cannot separate "tool helped" from "tool happened to coincide with a correct answer". For that, we need trajectory-level attribution. We use Qwen3-VL-30B (Bai et al., 2025a) as the judge model, with a manual audit on a random 40% subset, to label each tool-using trajectory along three axes: whether the tool output was novel relative to the model's pre-tool reasoning, whether it was reliable, and whether the model integrated it into the final answer. The label combinations group naturally into four roles: *Redundant confirmation*, where the output mainly reaffirms what the model already believed; *Genuine contribution*, where the tool supplies new and correct information that the model uses; *Failed / non-contributory*, where the output is unusable; and *Misuse / other*, for incorrect integration and rare combinations. The full prompt and benchmark-level distributions are in Appendix Tables 9 and 10.

What the judge sees (Figure 3) lines up with what the outcome decomposition implies. For DeepEyesV2, *Redundant confirmation* is the largest role on OCR (63.6%) and chart (56.6%) tasks; on real-world understanding, *Failed / non-contributory* takes over (63.7%); even on reasoning, *Redundant confirmation* still leads at 47.7%, with *Genuine contribution* rising to 40.4%. Thyme appears to assign more mass to *Genuine contribution*, although this pattern should be interpreted with care because several Thyme task-family estimates are based on relatively few tool-called samples. As we show next, even this label often corresponds to samples that are already solved without tools.

Many "genuine contribution" cases are already solvable without tools. This leaves one place where tool calls might still be doing real work: the trajectories the judge labels as *Genuine contribution*. Figure 3c examines them. Of those samples, 59.7% (DeepEyesV2) and 79.4% (Thyme) were already solved by the Tool-Free Agent. The trajectories that look most substantive — novel information, correctly integrated — often occur on problems the model could already solve, especially for Thyme. The tool calls that the judge identifies as informative are therefore frequently not necessary for final correctness.

Implications. The outcome counts and trajectory annotations converge on the same pattern: tool calls frequently occur on samples that are already solvable, while samples that require correction often remain incorrect. This does not match the behavior of a model that has learned to reach for a tool when its own knowledge runs out. In this sense, much of the observed tool use looks like a learned reflex — a pattern of calling that is weakly connected to whether a tool is actually needed.

4.3 Call Format or Returned Result: What Drives Tool-Conditioned Behavior?

We next ask which part of the tool-use loop drives tool-conditioned behavior: the model-generated tool-call message, the returned execution result, or both. As defined in Section 3.2, we intervene on already-generated trajectories: *Tool Format Only* keeps the call message but suppresses the result, while *Tool Result Only* keeps the result but removes the call message. We then test which intervention better preserves each agent's correctness pattern.

Behavioral agreement. For two settings A and B, we define their per-sample correctness agree-

Table 6: Performance comparison across tool-usage modes on reasoning benchmarks.

Model	Mode	MathVista	MathVerse	MathVision	WeMath	DynaMath	LogicVista
DeepEyesV2	Tool-Enabled Agent	72.50	52.21	28.39	37.81	57.60	48.10
	Tool Format Only	71.20	47.99	27.27	36.19	57.53	47.18
	Tool Result Only	69.70	40.20	28.06	29.90	48.42	43.37
Thyme	Tool-Enabled Agent	71.90	47.26	25.49	37.52	54.11	48.32
	Tool Format Only	71.30	46.70	26.18	38.67	53.45	50.56
	Tool Result Only	70.90	47.39	25.26	37.43	54.21	47.20
Pure-Text Reasoner	–	74.60	51.62	29.31	46.29	56.81	49.22

ment as $\mathrm{Agree}(A, B) = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}[c_i^A = c_i^B]$, where $c_i^A, c_i^B \in \{0, 1\}$ are correctness indicators on sample i. Higher values indicate that the two settings make the same correctness judgments on more samples. Table 4 reports per-sample correctness agreement (%) between each intervention and the Tool-Enabled Agent — that is, the proportion of samples on which the two settings yield the same correctness outcome — alongside the Tool-Free Agent as a reference floor. The two agents land on opposite sides.

For DeepEyesV2, *Tool Format Only* is closer to the Tool-Enabled Agent than the Tool-Free Agent on every task family, and is the closest of the three comparison settings overall, with agreement gains of $+7.1$, $+7.5$, and $+6.0$ points on real-world understanding, chart, and reasoning. In other words, keeping the tool-call format while removing the returned result disturbs DeepEyesV2 less than the reverse intervention. The generated call format accounts for much of the behavioral pattern attributed to tool use.

Thyme shows the opposite pattern. *Tool Result Only* achieves the highest agreement on all four task families, with gains of $+3.2$ to $+7.0$ points over the Tool-Free Agent. *Tool Format Only* also tracks the Tool-Enabled Agent closely, but remains slightly below *Tool Result Only*. Thus, Thyme's behavior is better preserved by the returned execution result than by the calling pattern alone.

Final-task performance. The behavioral picture is consistent with how each intervention affects task scores (Tables 5 and 6). For DeepEyesV2, performance under *Tool Format Only* remains close to the Tool-Enabled Agent and even slightly improves on V* Bench and CharXiv descriptive; *Tool Result Only*, by contrast, produces visible drops on HRBench, TreeBench, and ChartQA. For Thyme, both interventions stay close to the Tool-Enabled Agent, and *Tool Result Only* matches or surpasses it on OCRBench and HRBench 4K/8K. In both agents, however, the full tool-use pipeline does not deliver a consistent benchmark advantage over the corresponding ablations.

Implications. The two agents preserve tool-conditioned behavior through different parts of the tool loop: DeepEyesV2 through call format, and Thyme through returned results. This divergence is informative, but the overall picture remains similar: the full pipeline shows no consistent advantage over either ablation and does not resolve the limited solved-set expansion in Section 4.1. Tool training therefore appears to shape agent conditioning without reliably expanding what they can solve.

5 Conclusion

It has become natural to assume that adding tools to a multimodal agent, and training it to call them, makes the agent more capable. Our results suggest a more careful distinction. Across two trained agents, four task families, and analyses of correctness, process, generated-token cost, and mechanism, we find little evidence that tool access consistently expands the solved set relative to comparable non-tool settings. Tool calls often occur on samples already solved without tools, fail to repair incorrect predictions, and are preserved by different parts of the tool-use loop in different agents. These patterns point to a gap between learning to call tools and learning when tool outputs change what can be solved: final-answer correctness alone does not distinguish tools that supply missing information from those that merely confirm an available answer. Evaluation and training signals should therefore credit tools for the information they add, not merely for their presence in successful trajectories; otherwise, scaling tool access may reinforce tool-calling frequency rather than meaningful tool use.

Limitations

A few boundaries on this study are worth flagging. First, our analysis centers on two representative "thinking with images" agents and four task families. The agent ecosystem continues to grow, so the patterns we report should be read as characterizing this slice of the field rather than the field as a whole. The Pure-Text Reasoner also has a limited role: although it uses the same backbone scale and source data pool and contains no tool-calling trajectories, it is separately trained with re-annotation and filtering. We therefore use it as a diagnostic reference for solved-set attribution, not as evidence that tool-free training is generally superior.

Second, our framework characterizes tool use at three levels: aggregate performance and solved-set composition, per-sample outcomes with process attribution, and mechanism-level ablations. Complementary perspectives, such as user-perceived helpfulness, multi-turn interaction, or domain-specific deployment, could further enrich this view.

Third, the process-level attribution relies on a strong language model as a judge, although a 40% manual audit supports the major role distinctions. Alternative attribution paradigms, such as multi-judge ensembles or fine-grained human studies, could provide additional triangulation. Nevertheless, the qualitative patterns are consistent across all four analytical dimensions, suggesting that they are not artifacts of any single judge.

References

Tajamul Ashraf, Umair Nawaz, Abdelrahman M Shaker, Rao Anwer, Philip Torr, Fahad Shahbaz Khan, and Salman Khan. 2025. Matrix: Multimodal agent tuning for robust tool-use reasoning. *arXiv preprint arXiv:2510.08567*.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, and 45 others. 2025a. Qwen3-VL technical report. *Preprint*, arXiv:2511.21631.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, and 8 others. 2025b. Qwen2.5-VL technical report. *Preprint*, arXiv:2502.13923.

Shengyuan Ding, Xinyu Fang, Ziyu Liu, Yuhang Zang, Yuhang Cao, Xiangyu Zhao, Haodong Duan, Xiaoyi Dong, Jianze Liang, Bin Wang, Conghui He, Dahua Lin, and Jiaqi Wang. 2026. Arm-thinker: Reinforcing multimodal generative reward models with agentic tool use and visual reasoning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Haodong Duan, Xinyu Fang, Junming Yang, Xiangyu Zhao, Yuxuan Qiao, Mo Li, Amit Agarwal, Zhe Chen, Lin Chen, Yuan Liu, Yubo Ma, Hailong Sun, Yifan Zhang, Shiyin Lu, Tack Hwa Wong, Weiyun Wang, Peiheng Zhou, Xiaozhe Li, Chaoyou Fu, and 13 others. 2025. VLMEvalKit: An open-source toolkit for evaluating large multi-modality models. *Preprint*, arXiv:2407.11691.

Xinyu Geng, Yanjing Xiao, Yuyang Zhang, Hanwen Wang, Xinyan Liu, Rui Min, Tianqing Fang, and Yi R. Fung. 2026. Geobrowse: A geolocation benchmark for agentic tool use with expert-annotated reasoning traces. *Preprint*, arXiv:2604.04017.

Tanmay Gupta and Aniruddha Kembhavi. 2023. Visual programming: Compositional visual reasoning without training. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 14953–14962.

Jack Hong, Chenxiao Zhao, ChengLin Zhu, Weiheng Lu, Guohai Xu, and Xing Yu. 2026. DeepEyesV2: Toward agentic multimodal model. In *The Fourteenth International Conference on Learning Representations*.

Yushi Hu, Weijia Shi, Xingyu Fu, Dan Roth, Mari Ostendorf, Luke Zettlemoyer, Noah A Smith, and Ranjay Krishna. 2024. Visual sketchpad: Sketching as a visual chain of thought for multimodal language models. *Advances in Neural Information Processing Systems*, 37:139348–139379.

Zeyi Huang, Yuyang Ji, Anirudh Sundara Rajan, Zefan Cai, Wen Xiao, Haohan Wang, Junjie Hu, and Yong Jae Lee. 2025. Visualtoolagent (vista): A reinforcement learning framework for visual tool selection. *arXiv preprint arXiv:2505.20289*.

Dongzhi Jiang, Renrui Zhang, Ziyu Guo, Yanwei Li, Yu Qi, Xinyan Chen, Liuhui Wang, Jianhan Jin, Claire Guo, Shen Yan, Bo Zhang, Chaoyou Fu, Peng Gao, and Hongsheng Li. 2025. MME-cot: Benchmarking chain-of-thought in large multimodal models for reasoning quality, robustness, and efficiency. In *Forty-second International Conference on Machine Learning*.

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

Bohao Li, Yuying Ge, Yixiao Ge, Guangzhi Wang, Rui Wang, Ruimao Zhang, and Ying Shan. 2024. Seed-bench: Benchmarking multimodal large language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13299–13308.

Junxian Li, Xinyue Xu, Sai Ma, and Sichao Li. 2025a. Faithact: Faithfulness planning and acting in mllms. *arXiv preprint arXiv:2511.08409*.

Ming Li, Jike Zhong, Shitian Zhao, Haoquan Zhang, Shaoheng Lin, Yuxiang Lai, Wei Chen, Konstantinos Psounis, and Kaipeng Zhang. 2025b. Tir-bench: A comprehensive benchmark for agentic thinking-with-images reasoning. *arXiv preprint arXiv:2511.01833*.

Yuliang Liu, Zhang Li, Mingxin Huang, Biao Yang, Wenwen Yu, Chunyuan Li, Xu-Cheng Yin, Cheng-Lin Liu, Lianwen Jin, and Xiang Bai. 2024. OCR-Bench: on the hidden mystery of ocr in large multi-modal models. *Science China Information Sciences*, 67(12):220102.

Ziyu Liu, Zeyi Sun, Yuhang Zang, Xiaoyi Dong, Yuhang Cao, Haodong Duan, Dahua Lin, and Jiaqi Wang. 2025. Visual-rft: Visual reinforcement fine-tuning. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2034–2044.

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

Ahmed Masry, Xuan Long Do, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. 2022. 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.

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

OpenAI, :, Aaron Hurst, Adam Lerer, Adam P. Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, Aleksander Mądry, Alex Baker-Whitcomb, Alex Beutel, Alex Borzunov, Alex Carney, Alex Chow, Alex Kirillov, and 401 others. 2024. Gpt-4o system card. *Preprint*, arXiv:2410.21276.

OpenAI. 2025. Thinking with images. https://openai.com/index/thinking-with-images/. Accessed: 2026-05-05.

Runqi Qiao, Qiuna Tan, Guanting Dong, MinhuiWu MinhuiWu, 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. 2025. We-math: Does your large multimodal model achieve human-like mathematical reasoning? In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 20023–20070, Vienna, Austria. Association for Computational Linguistics.

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

Alex Su, Haozhe Wang, Weiming Ren, Fangzhen Lin, and Wenhu Chen. 2026a. Pixel reasoner: Incentivizing pixel space reasoning via curiosity-driven reinforcement learning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Zhaochen Su, Jincheng Gao, Hangyu Guo, Zhenhua Liu, Lueyang Zhang, Xinyu Geng, Shijue Huang, Peng Xia, Guanyu Jiang, Cheng Wang, Yue Zhang, Yi R. Fung, and Junxian He. 2026b. Agentvista: Evaluating multimodal agents in ultra-challenging realistic visual scenarios. *Preprint*, arXiv:2602.23166.

Dídac Surís, Sachit Menon, and Carl Vondrick. 2023. Vipergpt: Visual inference via python execution for reasoning. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 11888–11898.

Miles Turpin, Julian Michael, Ethan Perez, and Samuel Bowman. 2023. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting. *Advances in Neural Information Processing Systems*, 36:74952–74965.

Haochen Wang, Xiangtai Li, Zilong Huang, Anran Wang, Jiacong Wang, Tao Zhang, Jiani zheng, Sule Bai, Zijian Kang, Jiashi Feng, Wang Zhuochen, and Zhaoxiang Zhang. 2026a. Traceable evidence enhanced visual grounded reasoning: Evaluation and method. In *The Fourteenth International Conference on Learning Representations*.

Haozhe Wang, Chao Qu, Zuming Huang, Wei Chu, Fangzhen Lin, and Wenhu Chen. 2025a. Vl-rethinker: Incentivizing self-reflection of vision-language models with reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 38, pages 30865–30891. Curran Associates, Inc.

Ke Wang, Junting Pan, Weikang Shi, Zimu Lu, Houxing Ren, Aojun Zhou, Mingjie Zhan, and Hongsheng Li. 2024a. Measuring multimodal mathematical reasoning with math-vision dataset. *Advances in Neural Information Processing Systems*, 37:95095–95169.

Ke Wang, Junting Pan, Linda Wei, Aojun Zhou, Weikang Shi, Zimu Lu, Han Xiao, Yunqiao Yang, Houxing Ren, Mingjie Zhan, and Hongsheng Li. 2025b. MathCoder-VL: Bridging vision and code for enhanced multimodal mathematical reasoning. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 2505–2534, Vienna, Austria. Association for Computational Linguistics.

Shijian Wang, Jiarui Jin, Runhao Fu, Zexuan Yan, Xingjian Wang, Mengkang Hu, Eric Wang, Xiaoxi Li, Kangning Zhang, Li Yao, Wenxiang Jiao, Xuelian Cheng, Yuan Lu, and Zongyuan Ge. 2026b. Muse-agent: A multimodal reasoning agent with stateful experiences. *Preprint*, arXiv:2603.27813.

Wenbin Wang, Liang Ding, Minyan Zeng, Xiabin Zhou, Li Shen, Yong Luo, Wei Yu, and Dacheng Tao. 2025c. Divide, conquer and combine: A training-free framework for high-resolution image perception in multimodal large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 7907–7915.

Zirui Wang, Mengzhou Xia, Luxi He, Howard Chen, Yitao Liu, Richard Zhu, Kaiqu Liang, Xindi Wu, Haotian Liu, Sadhika Malladi, Alexis Chevalier, Sanjeev Arora, and Danqi Chen. 2024b. Charxiv: Charting gaps in realistic chart understanding in multimodal LLMs. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Haoran Wei, Yaofeng Sun, and Yukun Li. 2025. DeepSeek-OCR: Contexts optical compression. *arXiv preprint arXiv:2510.18234*.

Chenfei Wu, Jiahao Li, Jingren Zhou, Junyang Lin, Kaiyuan Gao, Kun Yan, Sheng ming Yin, Shuai Bai, Xiao Xu, Yilei Chen, Yuxiang Chen, Zecheng Tang, Zekai Zhang, Zhengyi Wang, An Yang, Bowen Yu, Chen Cheng, Dayiheng Liu, Deqing Li, and 20 others. 2025. Qwen-Image technical report. *Preprint*, arXiv:2508.02324.

Mingyuan Wu, Jingcheng Yang, Jize Jiang, Meitang Li, Kaizhuo Yan, Hanchao Yu, Minjia Zhang, ChengXiang Zhai, and Klara Nahrstedt. 2026. VTool-R1: VLMs learn to think with images via reinforcement learning on multimodal tool use. In *The Fourteenth International Conference on Learning Representations*.

Penghao Wu and Saining Xie. 2024. V*: Guided visual search as a core mechanism in multimodal llms. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13084–13094.

Yijia Xiao, Edward Sun, Tianyu Liu, and Wei Wang. 2024. LogicVista: Multimodal llm logical reasoning benchmark in visual contexts. *arXiv preprint arXiv:2407.04973*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Wenhao Yang, Yu Xia, Jinlong Huang, Shiyin Lu, Qing-Guo Chen, Zhao Xu, Weihua Luo, Kaifu Zhang, Yuchen Zhou, Xiaobo Xia, Yuanyu Wan, Lijun Zhang, and Tat-Seng Chua. 2026. Walk the talk: Bridging the reasoning-action gap for thinking with images via multimodal agentic policy optimization. *Preprint*, arXiv:2604.06777.

Kaituo Zhang, Zhen Xiong, Mingyu Zhong, Zhimeng Jiang, Zhouyuan Yuan, Zhecheng Li, and Ying Lin. 2026a. Are tools all we need? unveiling the tool-use tax in llm agents. *Preprint*, arXiv:2605.00136.

Renrui Zhang, Dongzhi Jiang, Yichi Zhang, Haokun Lin, Ziyu Guo, Pengshuo Qiu, Aojun Zhou, Pan Lu, Kai-Wei Chang, Yu Qiao, Peng Gao, and Hongsheng Li. 2024. MathVerse: Does your multi-modal llm truly see the diagrams in visual math problems? In *European Conference on Computer Vision*, pages 169–186. Springer.

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

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. 2026b. Thyme: Think beyond images. In *The Fourteenth International Conference on Learning Representations*.

Ziwei Zheng, Michael Yang, Jack Hong, Chenxiao Zhao, Guohai Xu, Le Yang, Chao Shen, and XingYu. 2026. Deepeyes: Incentivizing "thinking with images" via reinforcement learning. In *The Fourteenth International Conference on Learning Representations*.

Chengke Zou, Xingang Guo, Rui Yang, Junyu Zhang, Bin Hu, and Huan Zhang. 2025. DynaMath: A dynamic visual benchmark for evaluating mathematical reasoning robustness of vision language models. In *The Thirteenth International Conference on Learning Representations*.

A Training Data for the Pure-Text Reasoner

To keep the Pure-Text Reasoner aligned with the multimodal agents under comparison, we construct its training data from the DeepEyesV2 training source pool. The resulting model is intended as a diagnostic tool-free reference rather than a stronger competitor or an upper-bound model.

SFT data. The original DeepEyesV2 SFT set contains 65K examples, including both text-only reasoning data and visual reasoning data. Since our study focuses on multimodal tool use, we remove the text-only examples and retain only the visual reasoning data. The original visual rationales contain many tool-calling trajectories, which are incompatible with the training objective of the Pure-Text Reasoner. We therefore re-annotate these examples to remove tool-call segments and retain only text-based reasoning traces.

DeepEyesV2 uses strong closed-source models such as Gemini 2.5 Pro, GPT-4o, and Claude Sonnet 4 for SFT data annotation. Due to resource constraints, we perform the re-annotation with our locally deployed Qwen2.5-VL-72B model. This choice may introduce differences in annotation style and quality, but it does not introduce tool-calling trajectories or execution results into the Pure-Text Reasoner.

RL data. The original DeepEyesV2 RL set contains 82K examples. We use the same RL training configuration as DeepEyesV2 where possible, but full-scale training on the complete RL set is computationally expensive in our environment: with the full 82K examples, one training step takes about 16 minutes, and one epoch requires approximately 634 steps, or about seven days. We therefore apply a pass@k filter and retain 30K examples for RL training. This reduces one epoch to approximately 235 steps, or about 2.6 days.

We acknowledge that this setup does not exactly reproduce the original DeepEyesV2 training conditions. However, our goal is not to construct a stronger training set or to use the Pure-Text Reasoner for model ranking. Instead, it serves as a diagnostic reference for estimating which tool-enabled successes are also attainable under a tool-free reasoning path using data from the same source pool. The main conclusion is also supported by analyses that do not rely on the Pure-Text Reasoner training pipeline, including the Tool-Free counterpart and the Tool Format Only and Tool Result Only ablations. These results similarly show that the full tool-use loop does not consistently yield additional capability beyond its component signals.

B Benchmarks and Evaluation Metrics

Following the evaluation settings of DeepEyesV2 and Thyme, we evaluate the compared models on a broad set of multimodal benchmarks covering real-world understanding, OCR, chart understanding, and mathematical reasoning. This benchmark selection is intended to avoid drawing conclusions from a single task type or capability profile.

Real-world understanding. For real-world understanding, we evaluate on V*Bench, HRBench, and TreeBench, which test fine-grained visual perception, high-resolution image understanding, and structured visual reasoning in natural scenes.

OCR. For OCR-related tasks, we evaluate on OCRBench and SEED-Bench-2-Plus. These benchmarks measure the ability of multimodal models to recognize and reason over text under diverse visual conditions.

Chart understanding. For chart-related reasoning, we evaluate on CharXiv and ChartQA. CharXiv includes both descriptive and reasoning-oriented chart understanding tasks, while ChartQA requires answering questions over visualized data with both perceptual and logical reasoning.

Mathematical reasoning. For mathematical reasoning, we evaluate on MathVista, MathVerse, MathVision, WeMath, DynaMath, and LogicVista. These benchmarks cover visual mathematical reasoning, diagram understanding, dynamic or multi-step mathematical inference, and logical reasoning in visual contexts.

All evaluations are conducted using VLMEvalKit (Duan et al., 2025), following the evaluation framework used by DeepEyesV2 and Thyme. For benchmarks that require model-based answer judgment, we use GPT-4o (OpenAI et al., 2024) as the judge model across all settings to ensure a consistent evaluation protocol.

Repeated evaluation stability. We repeated the benchmark evaluations under the same evaluation protocol. Across repeated runs, the reported accuracies or normalized scores varied by at most 1.3 points, and the qualitative patterns used in our

Table 7: Benchmark-level solved-set decomposition conditioned on samples correctly answered by the Tool-Enabled Agent. $N_{\text{tool-solved}}$ denotes the number of such samples and serves as the denominator for all percentages in each row. *Tool-Enabled only* denotes samples missed by both non-tool references, while *Also solved without tools* denotes samples also solved by at least one non-tool setting. The last three columns decompose this overlap. Summing 'Tool-Enabled only' and 'Also Pure-Text only' gives the share of Tool-Enabled successes missed by the Tool-Free Agent alone; these values help verify that the Section 4.1 pattern is not solely driven by the Pure-Text Reasoner.

Model	Benchmark	$N_{\text{tool-solved}}$	Tool-Enabled only	Also solved without tools	Also Tool-Free only	Also Pure-Text only	All three
DeepEyesV2	VStarBench	156	3.8	96.2	1.9	4.5	89.7
	HRBench4K	606	4.3	95.7	5.1	5.9	84.7
	HRBench8K	564	3.9	96.1	6.9	3.9	85.3
	TreeBench	168	18.5	81.5	12.5	28.0	41.1
	OCRBench	814	0.7	99.3	2.9	0.9	95.5
	SEEDBench2_Plus	1589	3.7	96.3	5.5	3.1	87.8
	CharXiv_descriptive_val	3027	2.3	97.7	6.1	5.1	86.5
	CharXiv_reasoning_val	437	13.3	86.7	16.5	12.6	57.7
	ChartQA_TEST	2202	2.4	97.6	10.4	7.5	79.6
	MathVista_MINI	725	2.6	97.4	8.8	5.8	82.8
	MathVerse_MINI	2039	13.0	87.0	10.1	13.8	63.1
	MathVision	804	25.7	74.3	17.0	17.0	40.2
	WeMath	1103	4.7	95.3	10.1	10.1	75.2
	DynaMath	2886	9.1	90.9	9.7	9.8	71.3
	LogicVista	215	17.7	82.3	18.1	14.0	50.2
Thyme	VStarBench	158	1.3	98.7	5.1	1.3	92.4
	HRBench4K	618	1.8	98.2	7.8	1.5	89.0
	HRBench8K	580	3.3	96.7	8.3	2.1	86.4
	TreeBench	157	10.8	89.2	19.7	5.7	63.7
	OCRBench	865	0.2	99.8	3.8	0.5	95.5
	SEEDBench2_Plus	1599	2.5	97.5	6.3	2.2	89.0
	CharXiv_descriptive_val	2930	1.7	98.3	6.7	2.0	89.6
	CharXiv_reasoning_val	438	6.4	93.6	21.2	9.1	63.2
	ChartQA_TEST	2183	1.0	99.0	9.9	0.6	88.5
	MathVista_MINI	725	2.2	97.8	9.0	4.3	84.6
	MathVerse_MINI	1862	5.8	94.2	13.2	8.3	72.8
	MathVision	756	19.8	80.2	19.8	13.8	46.6
	WeMath	1136	5.0	95.0	10.9	8.6	75.4
	DynaMath	2711	4.8	95.2	13.4	6.8	75.0
	LogicVista	216	14.4	85.6	19.0	9.7	56.9

analysis were stable. Specifically, no consistent aggregate advantage, small tool-only solved regions, and close agreement between full tool use and trajectory ablations hold across runs.

C Solved-Set Analysis Details

Figure 2 in the main text provides a task-family view of whether tool-enabled inference expands the set of problems an agent can solve. Here we provide the corresponding benchmark-level decomposition. Instead of only asking whether a Tool-Enabled Agent achieves a high aggregate score, we ask a more concrete question for each benchmark: among the examples it answers correctly, how many are actually new successes that neither non-tool reference can solve?

For each benchmark, we take the samples correctly answered by the Tool-Enabled Agent as the conditioned set. We denote the size of this set as $N_{\text{tool-solved}}$, which serves as the denominator for all percentages in Table 7. The column *Tool-Enabled only* reports the proportion of these correctly answered samples that are missed by both non-tool references, i.e., the Tool-Free Agent and the Pure-Text Reasoner. This is the strictest measure of tool-only solved cases. By contrast, *Also solved without tools* reports the proportion of tool-enabled successes that are also solved by at least one non-tool setting, and is therefore the complement of *Tool-Enabled only*.

The last three columns further explain where this overlap comes from. *Also Tool-Free only* denotes samples solved by the Tool-Enabled Agent and the Tool-Free Agent, but not by the Pure-Text Reasoner. *Also Pure-Text only* denotes samples solved by the Tool-Enabled Agent and the Pure-Text Reasoner, but not by the Tool-Free Agent. *All three* denotes samples solved by all three settings. These columns together decompose the non-tool overlap and show whether the shared solved set comes from the inference-only tool-free variant, the separately trained pure-text reference, or both.

The benchmark-level results make the main pattern in Figure 2 more explicit. Across most benchmarks, the Tool-Enabled Agent does not solve a largely separate set of examples. Instead, most of its correct answers overlap with at least one

non-tool reference. This overlap is especially high on OCR and chart benchmarks, where many tool-enabled successes are also reachable without executing tools. Some benchmarks, such as TreeBench, MathVision, and LogicVista, show larger tool-only portions, suggesting that tool access can still help on localized subsets. However, these cases do not dominate the overall picture. The table therefore supports the conclusion in Section 4.1: in the settings we study, tool access does not consistently expand the solved set, and many correct tool-enabled trajectories correspond to examples that are already solvable through non-tool reasoning paths.

D Tool-Call Outcome Decomposition Details

Table 8 provides the benchmark-level decomposition behind the task-family averages reported in Table 3. The table keeps the same four-way outcome taxonomy as the main text, but reports it separately for each benchmark and for both non-tool references. All rates are computed on the subset of samples where the Tool-Enabled Agent actually invokes a tool, so the table focuses specifically on trajectories with observed tool-use behavior rather than on all evaluated samples.

This finer-grained view shows that the task-family trends in the main text are not driven by a single benchmark. On OCR and chart-related benchmarks, tool-called samples frequently fall into shared-correctness regions with the non-tool references, indicating that many successful tool-use trajectories are not unique to tool-enabled inference. On reasoning benchmarks, the distribution shifts more strongly toward shared failures, especially on harder datasets such as MathVision, MathVerse, and LogicVista. These cases show that tool invocation often continues to occur on difficult samples, but does not necessarily convert them into correct answers.

The benchmark-level results also reveal where tool use is more locally beneficial. Some benchmarks contain non-negligible effective correction rates, suggesting that tool calls can help on particular subsets. However, these gains are not the dominant pattern across the full benchmark suite. Together with the solved-set coverage in Table 7, this table provides a more detailed account of why the aggregate tool-enabled performance should not be interpreted as direct evidence of broad capability expansion.

E Process-Level Attribution Details

To complement the correctness-based decompositions, we further annotate the role played by each tool call inside the reasoning trajectory. This analysis is conducted on the full tool-enabled outputs of DeepEyesV2 and Thyme across all 15 benchmarks. We parse each model output, identify whether the agent actually invokes a tool, and keep only trajectories with observed tool-use behavior. The annotation therefore focuses on what happens when the tool-use mechanism is explicitly activated, rather than on all evaluated samples.

We use Qwen3-VL-30B as the judge model for process-level attribution. For each tool-using trajectory, the judge is given the question, the model's reasoning context, the tool-call content, the returned tool output, and the final answer. The full judge prompt is provided below. The prompt is designed to avoid judging a tool call only by the final answer. In particular, it explicitly instructs the judge not to mark a tool output as useful merely because the final prediction is correct. For example, a tool call that only re-displays the original image, such as a simple `plt.imshow(image_1)` operation without cropping, zooming, OCR, enhancement, measurement, or parsing, should not be treated as providing novel information. Such cases are usually confirmatory or non-contributory rather than genuine evidence of tool-enabled reasoning.

The judge assigns labels along three strictly separated dimensions: *Information Gain*, *Tool Output Quality*, and *Integration Status*. For *Information Gain*, the allowed labels are *Novel*, *Confirmatory*, and *Irrelevant*. For *Tool Output Quality*, the allowed labels are *Useful/Correct*, *Partially useful*, and *Wrong/Failed*. For *Integration Status*, the allowed labels are *Used correctly*, *Ignored*, and *Misused/Misinterpreted*. During post-processing, we normalize *Useful/Correct* to *Correct*, and *Partially useful* to *Partially correct*. The prompt also emphasizes that *Novel* should be assigned conservatively, since many tool calls only reformat, restate, or visually redisplay information already available to the model.

We then map the three-dimensional labels into mutually exclusive trajectory roles using a priority-based rule. If the integration label is *Misused / Misinterpreted*, the trajectory is assigned to *Misuse / other*, regardless of the information gain or

Table 8: Full benchmark-level tool-call outcome decomposition for DeepEyesV2 and Thyme. TRR, TFR, TER, and TIR are computed on samples where the Tool-Enabled Agent calls tools, using either the Tool-Free Agent or the Pure-Text Reasoner as the non-tool baseline. Cell background colors follow the same blue colormap used in Table 3, with darker shades indicating larger values.

Model	Task	Benchmark	vs Tool-Free Agent				vs Pure-Text Reasoner			
			TRR	TFR	TER	TIR	TRR	TFR	TER	TIR
DeepEyesV2	Real-World Understanding	V* Bench	74.9	13.1	6.8	5.2	77.0	12.6	4.7	5.8
		HRBench 4K	68.0	17.5	7.8	6.8	68.6	16.1	7.1	8.1
		HRBench 8K	65.0	23.4	5.5	6.1	62.9	20.9	7.6	8.6
		Tree Bench	22.3	47.5	19.3	10.9	28.7	44.6	12.9	13.9
		Avg	59.1	24.8	9.0	7.2	59.9	22.8	8.2	9.2
	OCR	OCR Bench	80.2	12.0	1.3	6.5	78.5	8.7	3.0	9.8
		SEED 2 PLUS	65.1	24.1	4.7	6.1	63.4	23.7	6.4	6.5
		Avg	69.7	20.4	3.7	6.2	68.0	19.1	5.3	7.5
	Chart	CharXiv Descriptive	70.1	17.4	5.6	6.9	69.3	15.3	6.4	9.0
		CharXiv Reasoning	32.4	42.5	11.3	13.8	30.7	42.0	13.0	14.3
		Chart QA	79.3	8.1	8.8	3.8	76.8	7.7	11.3	4.2
		Avg	68.1	17.6	7.4	6.8	66.6	16.3	8.9	8.1
	Reasoning	MathVista	66.4	20.1	6.1	7.4	64.2	19.4	8.3	8.1
		MathVerse	37.8	37.5	13.9	10.8	39.8	35.4	12.0	12.9
		MathVision	15.1	60.2	11.3	13.4	15.1	60.0	11.3	13.6
		WeMath	54.0	23.6	9.4	13.0	54.0	20.1	9.4	16.5
		DynaMath	46.7	31.2	10.9	11.2	46.7	31.9	10.9	10.5
		LogicVista	32.9	35.6	15.2	16.3	30.9	38.9	17.2	13.0
		Avg	39.8	37.1	11.4	11.6	40.1	36.5	11.1	12.3
	Overall		52.3	29.0	9.3	9.4	52.0	28.0	9.6	10.4
Thyme	Real-World Understanding	V* Bench	67.7	16.1	3.2	12.9	64.5	19.4	6.5	9.7
		HRBench 4K	73.0	17.0	2.0	8.0	67.0	16.0	8.0	9.0
		HRBench 8K	74.3	19.3	3.7	2.8	71.6	19.3	6.4	2.8
		Tree Bench	40.0	53.3	0.0	6.7	33.3	33.3	6.7	26.7
		Avg	71.0	20.0	2.7	6.3	66.7	18.8	7.1	7.5
	OCR	OCR Bench	85.7	11.8	0.3	2.2	82.9	8.0	3.0	6.1
		SEED 2 PLUS	68.1	23.4	2.1	6.4	66.0	25.5	4.3	4.3
		Avg	83.7	13.2	0.5	2.7	81.0	10.0	3.2	5.9
	Chart	CharXiv Descriptive	40.0	40.0	0.0	20.0	40.0	20.0	0.0	40.0
		CharXiv Reasoning	0.0	100.0	0.0	0.0	0.0	50.0	0.0	50.0
		Chart QA	88.9	11.1	0.0	0.0	44.4	0.0	44.4	11.1
		Avg	62.5	31.2	0.0	6.2	37.5	12.5	25.0	25.0
	Reasoning	MathVista	0.0	100.0	0.0	0.0	0.0	100.0	0.0	0.0
		MathVerse	37.5	39.6	22.9	0.0	39.6	35.4	20.8	4.2
		MathVision	12.5	75.0	0.0	12.5	12.5	62.5	0.0	25.0
		WeMath	50.0	50.0	0.0	0.0	0.0	0.0	50.0	50.0
		DynaMath	29.7	40.6	25.0	4.7	28.1	35.9	26.6	9.4
		Avg	31.7	43.1	22.0	3.3	30.9	37.4	22.8	8.9
	Overall		71.3	20.3	4.5	4.0	67.9	17.0	7.8	7.2

output quality labels. A trajectory is assigned to *Genuine contribution* only when the tool output is *Novel*, *Correct*, and *Used correctly*. If the tool output is *Novel* and *Correct* or *Partially correct*, but the model ignores it, the trajectory is assigned to *Missed opportunity*. If the tool output is *Confirmatory*, has *Correct* or *Partially correct* quality, and is either used or ignored, the trajectory is assigned to *Redundant confirmation*. If the tool output is *Wrong / Failed* and is either ignored or used, the trajectory is assigned to *Failed or non-contributory call*. Remaining rare label combinations are assigned to *Other*.

For visualization, we merge these fine-grained roles into four display categories. *Redundant confirmation* and *Genuine contribution* are kept as separate categories. *Failed or non-contributory call* is displayed as *Failed / non-contributory*. The remaining roles, including *Misuse or harmful integration*, *Missed opportunity*, and *Other*, are grouped as *Misuse / other*. Figure 3 shows the resulting task-family-level role distributions for DeepEyesV2 and Thyme.

Tables 9 and 10 provide the corresponding benchmark-level breakdowns. These tables expand Figure 3 by reporting the role distribution for each individual benchmark rather than only for each task family. They make it possible to see whether a task-family trend is broadly shared across benchmarks or driven by a small number of datasets. For example, they show which OCR or chart benchmarks contribute most to redundant confirmation, and which reasoning benchmarks contribute more to failed or non-contributory tool calls. Thus, the tables serve as the fine-grained audit trail behind the process-level summary in the main text.

The first two subfigures in Figure 3 show 100%-stacked role distributions over four task families: real-world understanding, OCR, chart understanding, and mathematical reasoning. The third subfigure further examines the subset labeled as *Genuine tool contribution*. For this subset, we check whether the same agent can answer the sample correctly when tool use is disabled at inference time. This tests whether a trajectory that appears genuinely tool-contributive at the process level also corresponds to an example that the agent could not solve without tools.

The results show that even the process-level "genuine" subset contains substantial overlap with tool-free correctness. Among trajectories labeled as *Genuine contribution*, 59.7% of DeepEyesV2 samples and 79.4% of Thyme samples are still answered correctly by the corresponding Tool-Free Agent. Conversely, only 40.3% and 20.6% of these subsets, respectively, correspond to samples that the Tool-Free Agent gets wrong. This indicates that even when the tool output is judged to be novel, correct, and integrated, the tool call often does not uniquely expand the agent's solvable set.

To assess the reliability of the judge-based attribution, we further conduct a manual audit on a random 40% subset of the judged tool-using trajectories. Human inspection shows broad agreement with the judge labels on the major role categories, especially the distinctions among redundant confirmation, genuine contribution, and failed or non-contributory calls. Most disagreements occur in borderline cases where the tool output is partially useful, or where the trajectory makes it difficult to determine whether the final answer actually depends on the tool output. No systematic disagreement was found that would materially alter the dominant role trends observed in Figure 3 and Tables 9 and 10. We therefore use the judge annotations as a process-level diagnostic signal rather than as ground-truth causal attribution.

Judge prompt for tool-use process attribution. The following prompt is used to annotate tool-called trajectories in the tool-use process analysis.

```
You are an expert judge for multimodal agent
↪  tool-use analysis.

Your task is to inspect an agent trajectory and
↪  label the role of tool use using three
↪  dimensions.

Judge only from observable evidence in the
↪  trajectory. Do NOT give credit to tool use
↪  just because:
- the final answer is correct,
- the model says things became "clearer",
- or a tool call happened.

You must distinguish between truly new evidence
↪  and mere repetition / self-confirmation.
You must also keep the three dimensions strictly
↪  separate. Never reuse a label from one
↪  dimension in another dimension.

Definitions:
1. Information Gain
- Novel: the tool provides new task-relevant
↪  information not already available from the
↪  model's unaided reasoning.
- Confirmatory: the tool mainly confirms or
↪  repeats an already available conclusion.
- Irrelevant: the tool output is not materially
↪  useful for solving the task.

2. Tool Output Quality
```

```
- Useful/Correct: the tool output is correct and
↪  meaningfully usable.
- Partially useful: the tool output is partially
↪  correct, incomplete, or only weakly useful.
- Wrong/Failed: the tool output is wrong, failed,
↪  or unusable.

3. Integration Status
- Used correctly: the final reasoning
↪  meaningfully uses the tool output in a
↪  correct way.
- Ignored: useful or potentially useful tool
↪  output is not actually used.
- Misused/Misinterpreted: the model uses the tool
↪  output incorrectly or is misled by it.

Important decision rules:
1. Novel should be rare. Only assign Novel if the
↪  tool output clearly reveals or computes
↪  information
   that was not already available from the
   ↪  pre-tool reasoning.
2. If the tool only redisplays the original image
↪  without crop, zoom, OCR, enhancement,
↪  measurement,
   parsing, or other transformation, do NOT label
   ↪  it as Novel.
3. If the tool output is empty, trivial, or only
↪  repeats the original image view, it should
↪  usually be
   Confirmatory or Irrelevant, not Novel.
4. Do not infer tool usefulness from final
↪  correctness. A correct final answer can still
↪  correspond to
   Confirmatory, Partially useful, or even
   ↪  Irrelevant tool use.
5. Useful/Correct requires that the tool output
↪  itself is materially informative and correct.
↪  If it adds
   little or no new evidence, prefer Partially
   ↪  useful or Wrong/Failed as appropriate.
6. If the post-tool answer mostly repeats the
↪  same conclusion already stated before the
↪  tool call, that is
   evidence against Novel information gain.
7. Irrelevant is ONLY valid for information_gain.
↪  It is NOT a valid value for
↪  tool_output_quality or
   integration_status.
8. If the tool output is empty, blank, trivial,
↪  or unusable, then tool_output_quality should
↪  usually be
   Wrong/Failed.
9. If the tool output is merely the same
↪  unprocessed image view, with no crop / zoom /
↪  OCR / enhancement /
   measurement / parsing, then
   ↪  tool_output_quality should usually be
   ↪  Partially useful at most, and often
   Wrong/Failed if the execution adds nothing.
10. If the tool output adds nothing and the final
↪  answer comes from the same pre-tool
↪  reasoning, prefer
    integration_status = Ignored, not Used
    ↪  correctly.

Field-specific allowed values:
- information_gain must be exactly one of: Novel,
↪  Confirmatory, Irrelevant
- tool_output_quality must be exactly one of:
↪  Useful/Correct, Partially useful,
↪  Wrong/Failed
- integration_status must be exactly one of: Used
↪  correctly, Ignored, Misused/Misinterpreted

Return ONLY a JSON object with exactly these
↪  keys:
{
  "information_gain": "...",
  "tool_output_quality": "...",
  "integration_status": "...",
  "confidence": 0-100,
  "rationale": "short explanation"
}

User prompt template:
Analyze the following multimodal agent
↪  trajectory.

Metadata:
- Model setting: {model_key}
- Benchmark: {benchmark}
- Sample index: {sample_index}
- Tool use times: {tool_use_times}
- Final score: {score}

Trajectory:
<trajectory>
{trajectory_text}
</trajectory>

Return only the JSON object.
```

F Generated-Token Statistics

In addition to accuracy and solved-set coverage, we also examine whether tool-enabled inference brings a compensating efficiency benefit. Even if tool access does not consistently improve final-answer correctness, it might still be useful if external tool execution reduces the amount of model generation required to reach an answer. The main text summarizes this analysis at the task-family level; here we provide the detailed token statistics.

Figure 4 shows the average number of generated tokens across task families for the Tool-Enabled Agent, the Tool-Free Agent, and the Pure-Text Reasoner. Table 11 further reports the benchmark-level token counts. We count tokens generated by the model, including reasoning text and tool-call messages. External tool outputs are not treated as model-generated tokens, since they are produced by the execution environment rather than by the agent itself.

The results show that tool-enabled inference does not reliably reduce generation cost. For DeepEyesV2, the Tool-Enabled Agent often produces more tokens than both non-tool settings, and its overall average remains higher than the Tool-Free Agent and the Pure-Text Reasoner. Thyme shows

Table 9: Benchmark-level tool-use role distribution for DeepEyesV2. Percentages are computed over samples where the tool-enabled agent calls tools within each benchmark. Entries marked "–" indicate that the agent did not call tools on that benchmark.

Task	Benchmark	N	Redundant confirmation	Genuine contribution	Failed / non-contributory	Misuse / other
RWU	V*	191	23.6	11.5	52.9	12.0
RWU	HRBench-4K	800	24.8	7.9	59.9	7.5
RWU	HRBench-8K	800	22.5	7.9	62.3	7.4
RWU	TreeBench	404	13.9	3.2	79.2	3.7
OCR	OCRBench	999	54.3	3.8	40.8	1.1
OCR	SEED2 PLUS	2276	67.7	7.0	22.4	3.0
Chart	CharXiv-Desc	3999	44.8	16.7	37.6	0.9
Chart	CharXiv-Reason	1000	48.9	11.5	36.7	2.9
Chart	ChartQA	2500	78.7	15.2	5.1	1.0
Reasoning	MathVista	1000	57.5	26.9	12.6	3.0
Reasoning	MathVerse	3940	43.9	50.3	0.6	5.2
Reasoning	MathVision	3039	54.3	31.1	8.2	6.4
Reasoning	WeMath	1740	50.0	44.4	3.6	2.0
Reasoning	DynaMath	5009	42.4	41.3	13.4	2.9
Reasoning	LogicVista	447	63.5	21.0	13.0	2.5

Table 10: Benchmark-level tool-use role distribution for Thyme. Percentages are computed over samples where the tool-enabled agent calls tools within each benchmark. Entries marked "–" indicate that the agent did not call tools on that benchmark.

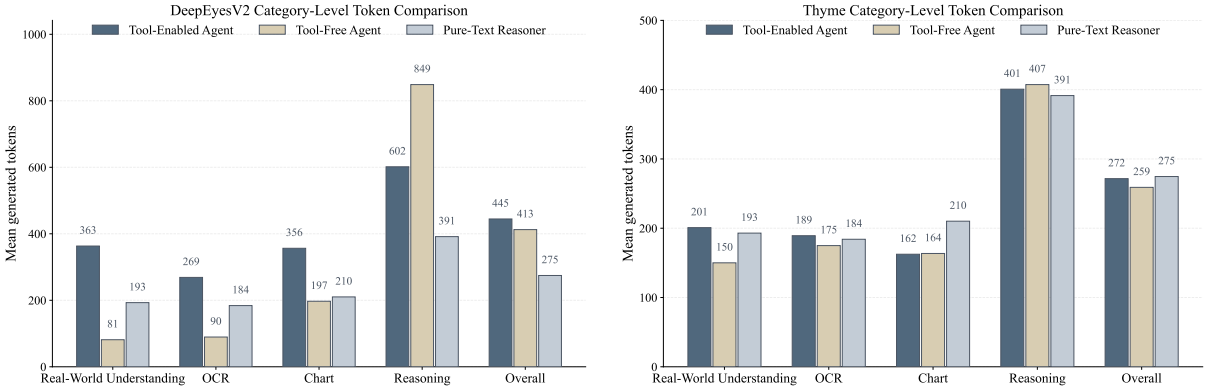
Task	Benchmark	N	Redundant confirmation	Genuine contribution	Failed / non-contributory	Misuse / other
RWU	V*	31	0.0	100.0	0.0	0.0
RWU	HRBench-4K	100	0.0	100.0	0.0	0.0
RWU	HRBench-8K	109	0.0	93.6	5.5	0.9
RWU	TreeBench	15	0.0	86.7	6.7	6.7
OCR	OCRBench	363	0.0	95.0	3.3	1.7
OCR	SEED2 PLUS	47	0.0	100.0	0.0	0.0
Chart	CharXiv-Desc	5	0.0	100.0	0.0	0.0
Chart	CharXiv-Reason	2	0.0	50.0	0.0	50.0
Chart	ChartQA	9	100.0	0.0	0.0	0.0
Reasoning	MathVista	1	100.0	0.0	0.0	0.0
Reasoning	MathVerse	48	25.0	75.0	0.0	0.0
Reasoning	MathVision	8	37.5	62.5	0.0	0.0
Reasoning	WeMath	2	100.0	0.0	0.0	0.0
Reasoning	DynaMath	64	15.6	73.4	0.0	10.9
Reasoning	LogicVista	0	–	–	–	–

smaller differences, but the same qualitative pattern holds: enabling tools does not lead to a consistent reduction in generated tokens across benchmarks. In several task families, tool-enabled inference introduces additional reasoning and tool-call overhead without a corresponding decrease in final generation length.

The benchmark-level results in Table 11 make this pattern more explicit. Token counts vary across datasets and agents, but the tool-enabled setting is not systematically more efficient. This complements the accuracy and solved-set analyses: in the settings we study, tool access neither consistently expands the set of solved examples nor provides a reliable token-saving benefit.

G Use of AI Assistants

We used ChatGPT for minor language polishing and readability improvements. The authors retained full responsibility for all scientific claims, analyses, and conclusions in the paper.



(a) DeepEyesV2 token totals comparison.

(b) Thyme token totals comparison.

Figure 4: Average generated-token counts across task families for the Tool-Enabled Agent, the Tool-Free Agent, and the Pure-Text Reasoner.

Table 11: Benchmark-level token cost comparison. Values report the average number of generated tokens per sample. Score gap is computed as the Tool-Enabled Agent score minus the Pure-Text Reasoner score on the same benchmark.

Model	Benchmark	Pure-Text Reasoner	Tool-Free Agent	Tool-Enabled Agent	Score gap
DeepEyesV2	V* Bench	141.4	55.6	354.4	-1.1
	HRBench 4K	215.3	66.0	324.2	-1.1
	HRBench 8K	209.8	69.5	340.9	-0.3
	Tree Bench	205.3	134.5	433.3	-0.7
	OCR Bench	163.6	67.3	212.8	-2.4
	SEED 2 PLUS	204.5	111.7	324.9	-0.1
	CharXiv descriptive	208.0	167.3	325.9	3.2
	CharXiv reasoning	243.7	305.1	481.3	-1.3
	Chart QA	178.8	118.9	262.1	8.8
	MathVista	277.1	378.1	444.0	0.2
	MathVerse	404.1	916.8	572.8	-0.3
	MathVision	541.1	1626.0	904.0	-1.8
	WeMath	371.8	773.7	483.0	-7.1
	DynaMath	374.1	644.2	588.1	3.1
	LogicVista	380.6	753.7	617.3	4.3
Thyme	V* Bench	141.4	140.4	214.8	0.0
	HRBench 4K	215.3	147.8	197.3	0.5
	HRBench 8K	209.8	148.6	203.7	1.0
	Tree Bench	205.3	163.5	187.7	-3.7
	OCR Bench	163.6	204.4	218.2	2.7
	SEED 2 PLUS	204.5	145.5	160.2	0.4
	CharXiv descriptive	208.0	137.4	135.9	0.7
	CharXiv reasoning	243.7	208.2	206.0	-1.2
	Chart QA	178.8	145.1	145.6	8.0
	MathVista	277.1	278.9	265.2	0.2
	MathVerse	404.1	362.0	359.6	-4.8
	MathVision	541.1	855.0	820.3	-3.4
	WeMath	371.8	380.3	384.7	-5.2
	DynaMath	374.1	269.1	273.4	-0.4
	LogicVista	380.6	298.7	300.9	4.5