



DeepEyesV2: Toward Agentic Multimodal Model

Jack Hong^{*}, Chenxiao Zhao^{*}, ChengLin Zhu^{*}, Weiheng Lu, Guohai Xu[†], XingYu

Xiaohongshu Inc.

Project Homepage

jaaackhong@gmail.com, {chenxiao2, xuguohai}@xiaohongshu.com

Abstract

Agentic multimodal models should not only comprehend text and images, but also actively invoke external tools, such as code execution environments and web search, and integrate these operations into reasoning. In this work, we introduce **DeepEyesV2** and explore how to build an agentic multimodal model from the perspectives of data construction, training methods, and model evaluation. We observe that direct reinforcement learning alone fails to induce robust tool-use behavior. This phenomenon motivates a two-stage training pipeline: a cold-start stage to establish tool-use patterns, and reinforcement learning stage to further refine tool invocation. We curate a diverse, moderately challenging training dataset, specifically including examples where tool use is beneficial. We further introduce RealX-Bench, a comprehensive benchmark designed to evaluate real-world multimodal reasoning, which inherently requires the integration of multiple capabilities, including perception, search, and reasoning. We evaluate **DeepEyesV2** on RealX-Bench and other representative benchmarks, demonstrating its effectiveness across real-world understanding, mathematical reasoning, and search-intensive tasks. Moreover, **DeepEyesV2** exhibits task-adaptive tool invocation, tending to use image operations for perception tasks and numerical computations for reasoning tasks. Reinforcement learning further enables complex tool combinations and allows model to selectively invoke tools based on context. We hope our study can provide guidance for community in developing agentic multimodal models.

1 Introduction

An agentic multimodal model should not only be capable of understanding text and images, but can also actively invoke tools (*e.g.*, a code execution environment or a web search interface) and seamlessly integrate these operations into its advanced reasoning process. For example, as illustrated in Figure 1 (b), when asked to identify the species of a flower in an image, an agentic multimodal model first crops the region containing that flower, then uses the cropped image to search and determine the species. Although existing multimodal models demonstrate strong perception and interpretation abilities, they remain largely passive and lack the ability to autonomously invoke external tools, which is essential for agentic multimodal models. Tools enable explicit, verifiable operations on inputs (*e.g.*, cropping, measuring, computation) and provide access to up-to-date, source-grounded knowledge, thereby improving accuracy, reducing hallucinations, and supporting traceable reasoning.

^{*}Equal contribution.

[†]Corresponding author.

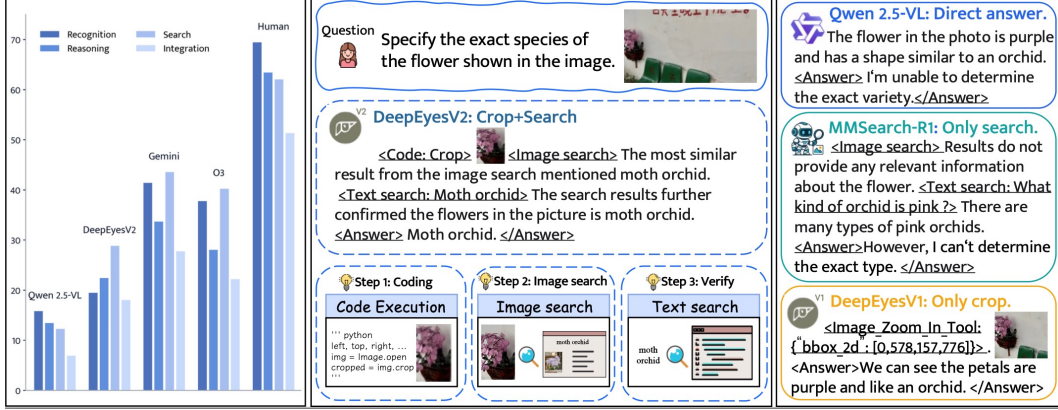


Figure 1: **Illustration of agentic multimodal models.** (a) Existing models show unsatisfactory performance in real-world scenarios, showing clear limitations especially when perception, reasoning, and search must be tightly integrated. (b) A multi-step visual reasoning example requiring coordinated perception, search, and reasoning.

These tool-use capabilities can be categorized into two types: (i) Operation tools: Current models cannot perform complex operations on visual or numerical data, including fine-grained image manipulations (e.g., cropping, measuring) and quantitative computations. This limits their capacity to reason about detailed visual content or solve mathematical problems. (ii) Information retrieval tools: Models cannot proactively access up-to-date external knowledge, which often leads to outdated conclusions or statements without verifiable sources. Some recent works attempt to rely on a single tool. For example, as shown in Figure 1 (b), DeepEyes [62] uses cropping to achieve fine-grained perception, but due to the lack of information retrieval capability, DeepEyes cannot correctly determine the category based solely on its internal knowledge. In contrast, although MMSearch-R1 [52] can perform search, it lacks fine-grained perception, leading to retrieval failures. A substantial gap remains between existing approaches and truly agentic multimodal models. While o3 [38] has explored “thinking with image” reasoning pattern that combines operations and search, how to realize such capabilities remains unclear.

To explore how to construct such agentic multimodal models, we introduce *DeepEyesV2*, which seamlessly integrates tool invocation within the dynamic reasoning loop. *DeepEyesV2* actively decides when and how to invoke tools, enabling a dynamic process of evidence acquisition and verification. Then, tool outputs are iteratively incorporated into reasoning process, allowing model to refine its hypotheses, validate intermediate results, and ultimately arrive at more reliable and interpretable conclusions. In this work, we systematically investigate key aspects of building an agentic MLLM, including model training strategies, dataset curation, and evaluation protocols.

We first follow the setup of DeepEyes [62] and apply reinforcement learning directly on Qwen2.5-VL [4], but find that limited inherent tool-use capability prevents stable tool invocation. This highlights the need for a cold-start stage to establish reliable tool-use patterns. Thus, we curate a high-quality dataset that spans diverse scenarios, including perception, reasoning, and search tasks. After cleaning, we apply two filters: (i) difficulty filtering, retaining only questions unsolvable by the base model, and (ii) tool-benefit classification, keeping cases where tool use improves accuracy. Data are split into two subsets: tool-solvable examples for RL and harder unsolved cases for cold start, further augmented with long chain-of-thought trajectories. Supervised fine-tuning on this cold-start dataset enables the model to acquire basic tool-use patterns and deeper reasoning, after which RL further strengthens tool invocation. Notably, we rely only on two simple rewards, accuracy and format, without complex reward engineering [41].

DeepEyesV2 demonstrates strong synergistic capabilities across perception, search, and reasoning. However, existing benchmarks mostly focus on just one of these abilities and lack an integrated, cross-capability benchmark that can comprehensively evaluate all three. Therefore, we propose a new benchmark, called RealX-Bench. RealX-Bench emphasizes cross-capability integration, requiring models to attend to fine-grained visual regions, retrieve external evidence, and reason over multimodal context. As shown in Figure 1 (a), current models perform well below human

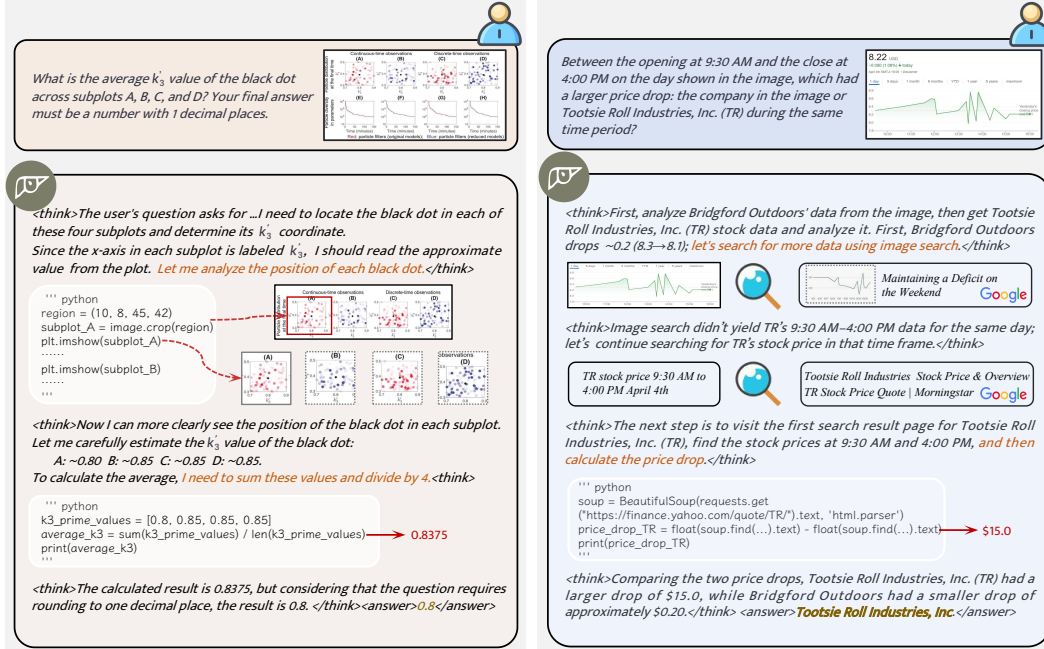


Figure 2: Case reasoning trajectory of DeepEyesV2. DeepEyesV2 seamlessly integrates code execution and web search within its iterative reasoning process. Notably, in the right case, the behavior of accessing webpages via code does not exist in cold start data and is spontaneously acquired during reinforcement learning.

performance on RealX-Bench, revealing a substantial performance gap and underscoring RealX-Bench’s difficulty. Compared with current open-source models and models limited to using a single tool, DeepEyesV2 demonstrates powerful coordination across the three capabilities.

Besides, we evaluate DeepEyesV2 on benchmarks covering real-world understanding, mathematical reasoning, and search-intensive tasks. DeepEyesV2 outperforms both general-purpose MLLMs and prior specific reasoning approaches. Specifically, on real-world understanding benchmarks, DeepEyesV2 surpasses even Qwen2.5-VL-32B in some benchmarks through effective tool use. On reasoning tasks, DeepEyesV2 shows performance gains across multiple benchmarks, including +7.1 on MathVerse (52.7% accuracy). On search benchmarks, DeepEyesV2 delivers strong advantages, reaching 63.7% on MMSearch [21], far beyond the MMSearch-R1 [52] (53.8%). These results demonstrate that by reliably invoking tools, DeepEyesV2 extends its comprehensive capabilities, achieving accurate and advanced reasoning.

We observe the task-dependent tool invocation patterns in DeepEyesV2. For perception tasks, DeepEyesV2 primarily uses image operations, such as cropping, to extract fine-grained visual details, whereas for reasoning tasks, DeepEyesV2 favors numerical analysis. Moreover, reinforcement learning can further enhances tool-use behavior, enabling more complex tool combinations and adaptive decision-making. DeepEyesV2 learns to selectively invoke tools based on the problem context, reflecting the emergence of autonomous, agentic reasoning.

The main contributions are summarized as follows: (i) We introduce DeepEyesV2, an agentic multimodal model that unifies code execution and web search within a single reasoning loop, enabling reliable and complex reasoning. (ii) We construct a carefully curated training corpus through rigorous data filtering and cleaning. The resulting dataset is diverse in task types, of appropriate difficulty, and explicitly designed to ensure the beneficial integration of tools. Based on this, we build both cold-start SFT data and RL data that complement each other. (iii) Extensive experiments across real-world understanding, mathematical reasoning, and search-intensive benchmarks demonstrate the strong reasoning and tool-usage ability of DeepEyesV2. (iv) We propose RealX-Bench, a comprehensive benchmark designed to evaluate real-world multimodal reasoning involving perception, search, and reasoning integration, providing a rigorous platform for assessing agentic multimodal intelligence. (v) We analyze the dynamics of tool-use behavior in DeepEyesV2, revealing task-adaptive patterns.

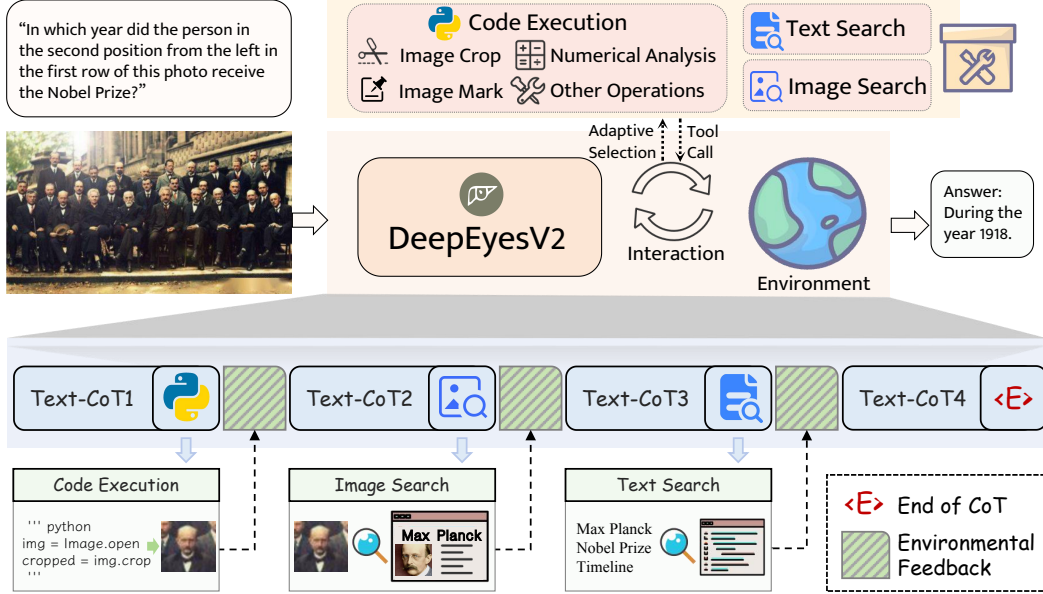


Figure 3: **Pipeline of DeepEyesV2.** DeepEyesV2 invokes tools and incorporates execution results into subsequent reasoning steps, enabling iterative and tool-augmented multimodal inference.

Besides, we also find reinforcement learning can enable more complex tool combinations and adaptive, context-aware tool invocation.

2 Related Works

Multimodal Large Language Models. The field of multimodal large language models (MLLMs) has witnessed rapid progress in recent years. Early efforts mainly focus on combining pretrained visual encoders with large language models through lightweight adapters or projection layers, enabling basic vision–language alignment and simple multimodal understanding [26, 31, 30, 3, 8]. Subsequently, more powerful architectures such as Qwen2.5-VL [4], LLaVA-OneVision [24], and InternVL3 [64], expand the training scale and integrated more diverse visual data, significantly improving performance on benchmarks of visual question answering, captioning, and general perception tasks. Recently, some OmniMLLMs [28, 60, 15, 19, 17] are capable of processing a mix of modalities like speech, video, and images simultaneously. However, existing MLLMs remain largely passive: they can interpret multimodal inputs and generate answers, but lack the ability to actively invoke external tools for computation or knowledge retrieval, which limits their reliability in complex reasoning tasks.

Thinking with Images. The paradigm of “Think with Image” is first introduced by o3 [38], which demonstrated that multimodal models can interleave reasoning with iterative visual analysis, actively manipulating images to support step-by-step problem solving. Many works attempt to reproduce such capabilities. Most approaches [41, 23, 33, 13, 20, 57] adopt a two-stage training pipeline, where a cold-start phase is followed by reinforcement learning. In contrast, DeepEyes [62] only adopts reinforcement learning alone and incentivizes the “Think with Image” behaviors, leading to strong reasoning performance. However, the majority of these efforts employ a rather limited tool set, typically restricted to region cropping for fine-grained perception. To improve generality, PyVision [61] and Thyme [58] utilize code execution to enable more flexible visual operations. Despite this progress, these models remain constrained to image manipulation only, and are unable to handle knowledge-intensive questions where access to up-to-date external information is essential.

Search-oriented Reasoning. To mitigate the inherent knowledge limitations of large multimodal language models, a growing line of work explores augmenting them with external knowledge acquisition. Early approaches commonly adopt the retrieval-augmented generation (RAG) paradigm [40, 22], where relevant information is retrieved from a pre-constructed knowledge base and fed into the model. While effective, this paradigm remains constrained by the static and finite nature of the underlying corpus. To overcome these limitations, more recent studies attempt to leverage online

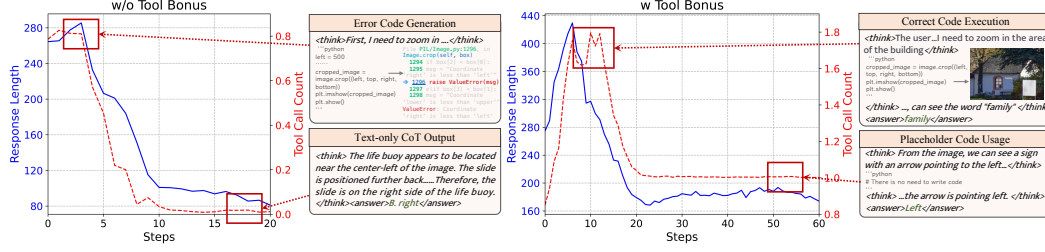


Figure 4: **Pioneer Experiments** reveal that existing multimodal models cannot directly acquire reliable tool use ability through RL, demonstrating the necessity of a cold start phase. The **red dashed line** represents tool calls number in a single rollout, and the **blue solid line** represents the average response length.

search to dynamically access broader and up-to-date information [63]. Beyond purely textual queries, some efforts extend search into the multimodal domain, enabling retrieval of not only documents but also images, charts, or other media forms relevant to the task [52, 48]. These advances highlight the potential of search-augmented reasoning to complement perception and tool-use capabilities, ultimately broadening the scope of problems that multimodal models can effectively address.

3 DeepEyesV2

We explore how to construct agentic multimodal models from the perspectives of training strategy, dataset design, and evaluation. We begin in Section 3.1 by presenting the overall pipeline of *DeepEyesV2*, which integrates tool invocation into the reasoning loop. Then, we conduct pioneer experiments in Section 3.2 to reveal the limitations of existing models in reliably using tools, underscoring the necessity of a cold-start stage. After that, we describe the curation of a high-quality training dataset and the principles behind its construction in Section 3.3. Finally, building on the cold-start foundation, we apply a reinforcement learning stage to further enhance the efficiency and flexibility of tool-use behavior, which is described in Section 3.4.

3.1 Overall Pipeline

Similar to DeepEyes [62], *DeepEyesV2* is an agentic multimodal model, but with extended tool-use capabilities beyond simple cropping. In *DeepEyesV2*, programmatic code execution and web retrieval are treated as complementary and interleavable tools inside a single reasoning trajectory (see Figure 3). Given an image input and the corresponding user query, *DeepEyesV2* first generates an initial reasoning plan, and explicitly determines whether this question can be solved directly through internal reasoning or requires tool invocation. If tool use is necessary, *DeepEyesV2* emits executable Python code or issues web search queries. Code execution is carried out in a sandboxed environment and can produce structured outputs such as transformed images, numerical measurements, computed arrays, plots, or execution logs. Image queries are submitted via SerpAPI and return the top five visually matched webpages (each with a thumbnail and title). Text queries return the five most relevant webpages, along with titles and snippets. All tool outputs are converted into observations and appended to model’s context. *DeepEyesV2* then thinks further in light of these observations and may plan further tool invocations (either additional code, further searches, or both), iterating this reasoning-tool-integration loop until a conclusive answer is produced.

DeepEyesV2 can dynamically choose, combine, and use tools as reasoning unfolds. This integration yields three main advantages: (i) it allows **expanded and enhanced analytical capability** through executable code; (ii) it enables **active and real-time knowledge seeking** by retrieving multimodal evidence from the web; and (iii) it supports **iterative, interleaved multi-tool reasoning**, in which code execution and search can be dynamically combined within a single trajectory, rather than being isolated modules. Together, these features position *DeepEyesV2* as a more general, reliable, and extensible framework for multimodal reasoning.

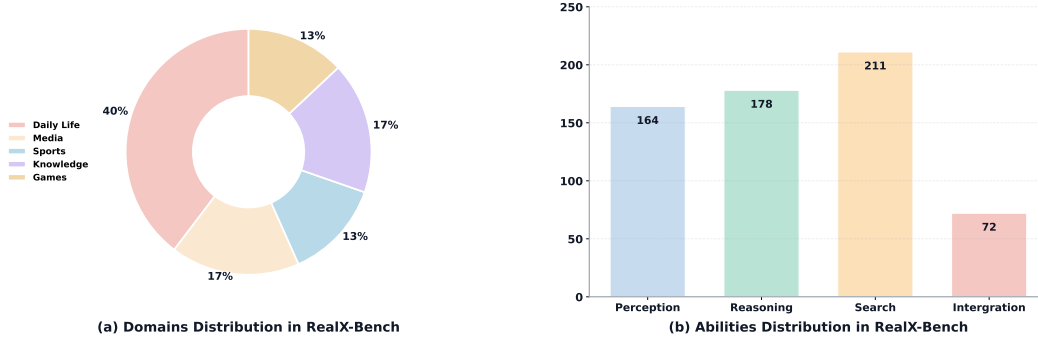


Figure 5: **Statistics of RealX-Bench.** (a) Domain distribution across five representative categories: Daily Life, Media, Sports, Knowledge, and Games. (b) Distribution of subsets classified by required abilities: perception, reasoning, search, and integration. These numbers may overlap because the challenges are not mutually exclusive. Integration denotes questions that are difficult across all three abilities simultaneously.

3.2 Pioneer Experiments

To investigate whether MLLMs can directly acquire tool-use ability through reinforcement learning, we first conduct a pioneer experiment on Qwen2.5-VL [4] following DeepEyes [62]. As shown in Figure 4, during training, we observe that in the early stages model occasionally attempts to produce Python code, but these outputs are often buggy or fail to execute, indicating that existing MLLMs struggle to generate stable and reliable code. As training continues, model gradually abandons code generation and converges to producing only short reasoning chains followed by direct answers, thereby bypassing tool use. Then, to encourage tool invocation, we incorporate the tool usage bonus mechanism from DeepEyes, which explicitly rewards the generation of code. With this additional signal, model is indeed able to produce correct and runnable code in the early stages, suggesting that the mechanism can enforce coding ability. However, with continued training a new degeneration emerges: model’s behavior converged to emitting exactly one code block per query, and this single block typically consists of non-executable, placeholder comments rather than meaningful code, revealing the phenomenon of reward hacking. This pioneer experiment highlights that existing MLLMs cannot reliably learn complex tool use through direct RL alone, motivating the need for a cold start to bootstrap model’s tool invocation ability.

3.3 Training Data Curation

Data Collection. Pioneer experiments have highlighted the necessity of constructing a high-quality dataset for supervised fine-tuning to explicitly guide model to learn how to generate executable code and perform tool invocations. Following DeepEyes [62], we collect data in accordance with the following principles: (i) **Diverse tasks and image distribution.** We incorporate varied data to cover a wide range of multimodal challenges and visual components. (ii) **Verifiability and structured format.** All questions are reformulated into a structured, open-ended QA format to facilitate objective evaluation. We exclude examples that cannot be reliably verified, such as those with incorrect answers, ambiguous phrasing, or poor readability. (iii) **Appreciate difficulty.** We exclude examples that the base model can easily solve and prioritize questions that remain challenging. (iv) **Beneficial integration of tools.** We categorize examples based on whether tool usage leads to correct answers. Cases where model can solve correctly using additional tool calls are reserved for reinforcement learning, whereas examples that remain unsolved even with tool assistance are used for cold start.

Specially, we curate data from three major categories: perception, reasoning, and search. Besides, we also include long Chain-of-Cot (CoT) reasoning data in cold start subset. Please refer to Appendix A.1 for more details on data sources. All datasets are carefully cleaned, reformatted, and divided into subsets for cold start or reinforcement learning subsets. To ensure sufficient difficulty, we employ Qwen2.5-VL-7B [4] as a baseline evaluator. For each question, model is prompted to generate 8 responses, and we retain only those instances where it answers correctly at most two times, thereby filtering out trivial cases. To further assess tool-use effectiveness, we prompt model to solve each

question with tool invocation, again collecting 8 responses per instance, and categorize examples according to their success rate.

Trajectories Synthesis. We construct cold start datasets by eliciting step-by-step trajectories from models (*e.g.*, Gemini 2.5 Pro [10], GPT-4o [18], and Claude Sonnet 4 [2]). For each prompt, these models are prompted to produce detailed reasoning traces that explicitly include tool-invocation markers (*e.g.*, code snippets). Each declared tool call is executed, and the returned outputs are fed back to the originating model, and model continues reasoning, potentially issuing further tool calls, until it produces a final answer. The entire interaction is recorded as a single trajectory. Only trajectories with correct final answers and error-free code are retained for high-quality cold-start data.

3.4 Agentic Reinforcement Learning

After cold-start training has equipped model with basic tool-use patterns, we adopt reinforcement learning to further enhance its ability to integrate tools in dynamic environment. Unlike SFT, which relies on learning from static trajectories, agentic RL places the model in an interactive environment where it must dynamically decide when and how to invoke tools in order to solve tasks. Following DeepEyes [62], we employ a sparse and outcome-driven reward. The overall reward consists of two components: (i) accuracy reward R_{acc} , which evaluates whether the final answer matches the ground truth, and (ii) format reward R_{format} , which penalizes outputs that violate required formats. The total reward is defined as: $R = R_{acc} + R_{format}$.

4 RealX-Bench

Existing multimodal benchmarks such as MME-RealWorld [59], SEED-Bench [29], and MM-Search [21] primarily evaluate isolated capabilities, for instance, perception, retrieval, or reasoning. However, real-world multimodal understanding often demands coordination across multiple abilities. Thus, we introduce RealX-Bench, a comprehensive benchmark that evaluates the coordinated interplay of perception, search, and reason in complex real-world scenarios.

4.1 Design Principles.

We define three core abilities, perception, search, and reasoning, as follows: perception is the capacity to recognize and locate relevant visual elements; search means finding the needed information from the web or provided resources; reasoning means combining evidence to reach the correct answer through clear, multi-step logic. To comprehensively evaluate model’s coordinated interplay of these abilities, we construct RealX-Bench adheres to the following design principles. (i) **Challenging.** Each question is deliberately difficult. For perception, challenge means precise localization of subtle targets under clutter or occlusion. For search, it requires multi-hop evidence gathering. For reasoning, it involves multi-step logical composition with intermediate consistency checks. Each question is constructed to exhibit at least one difficulty dimension. (ii) **Real-World.** All questions are grounded in real-world scenarios and realistic content distributions, and are refined for semantic fidelity and practical relevance. (iii) **Objectivity.** Every question has a short, unique answer in a standardized format and can be automatically verified via programmatic checks, enabling efficient, reproducible, and scalable evaluation.

4.2 Benchmark Construction

The construction follows a four-stage workflow: data collection, QA annotation, difficulty and category labeling, and quality control. First, We collect openly available images and their corresponding user questions from the internet, which faithfully reflect real-world scenarios. These questions fully reflect real-world scenarios. We filter them for visual quality and content diversity to ensure high quality and broad coverage. Then, experts refine each question and answer to better suit formal contexts and to ensure fluent, coherent language. After that, annotators assign a difficulty label to each question (*e.g.*, whether it is perception challenging) and tag the corresponding image category. Finally, quality control checks verify answer correctness and uniqueness for every QA pair.

Table 1: **Results on RealX-Bench.**

Model	Text Search	Image Search	Average	Perception	Reasoning	Search	Integration
Proprietary & Open-source Models							
GPT4o [18]	✓	✓	32.3	29.9	22.5	29.4	16.7
			32.0	29.3	23.0	29.4	16.7
	✓	✓	36.3	29.3	25.8	36.5	16.7
			38.7	30.5	27.5	36.5	15.3
Gemini 2.5 Pro [10]	✓	✓	39.3	34.8	24.2	36.0	16.7
			41.7	39.0	28.7	38.4	23.6
	✓	✓	45.0	37.8	33.2	43.1	25.0
			46.0	41.5	33.7	43.6	27.8
o3 [38]	✓	✓	35.0	31.7	23.0	30.8	11.1
			41.0	34.8	28.7	37.9	19.4
	✓	✓	41.3	37.8	28.1	40.3	22.2
			39.3	38.4	25.3	37.0	20.8
Qwen2.5-VL-7B [4]	✓	✓	17.0	15.9	13.5	12.3	6.9
			21.7	17.7	15.7	18.5	7.6
	✓	✓	19.7	16.6	14.4	15.9	8.3
			22.3	17.1	16.3	19.9	9.7
Qwen2.5-VL-32B [4]	✓	✓	25.0	21.3	19.7	19.9	12.5
			25.7	25.6	20.2	19.0	16.7
	✓	✓	30.7	27.4	23.0	26.1	15.3
			32.0	27.4	29.2	31.8	23.6
Qwen2.5-VL-72B [4]	✓	✓	25.3	23.1	17.4	17.5	9.7
			26.3	28.7	19.7	20.4	16.7
	✓	✓	28.0	28.7	20.2	20.4	15.3
			31.0	35.4	25.8	25.6	23.6
Grounded Reasoning Models							
Thyme [58]			21.0	18.3	14.6	12.8	4.2
DeepEyes [62]			19.0	19.5	14.6	12.8	9.7
Agentic Multimodal Model							
DeepEyesV2	✓	✓	28.3	19.5	22.5	28.9	18.1
Δ (vs Qwen2.5-VL-7B)			+6.0	+2.4	+6.2	+10.0	+8.4
Human Performance							
Human	✓	✓	70.0	69.5	63.5	62.1	51.4

4.3 Data Statistics.

RealX-Bench consists of 300 question-answer pairs spanning five representative real-world domains, and we show the data statics in Figure 5. Along the difficulty dimension, each question is annotated on three ability axes, perception, search, and reasoning, with non-mutually exclusive labels. Because difficulty can be coupled (e.g., a question may be both perception-challenging and require multi-hop search), these counts overlap. Notably, 24% questions are simultaneously challenging across all three abilities. Compared with prior benchmarks that mainly assess a single capability in isolation, RealX-Bench enables evaluation of integrated performance across perception, search, and reasoning.

5 Experiments

5.1 Implementation Details

We conduct training in two stages: cold start SFT and reinforcement learning. The backbone model is Qwen2.5-VL-7B [4]. For SFT, we train with a batch size of 128 and a learning rate of 1×10^{-5} . Model is optimized for 3 epochs using AdamW [34] optimizer with cosine learning rate decay. For RL, we

Table 2: Results on real-world & OCR & chart understanding Benchmarks.

Model	Tool	Param Size	Real-World Understanding					OCR		Chart		
			V* Bench	HRBench 4K	HRBench 8K	MME-RealWorld	Tree Bench	OCR Bench	SEED 2 Plus	CharXiv descriptive	CharXiv reasoning	Chart QA
Open-source Models												
LLaVA-OV	✗	7B	75.4	63.0	59.8	57.4	37.3	-	-	-	-	80.0
Qwen2.5-VL	✗	7B	78.5	71.6	67.9	57.3	37.0	864	70.4	72.7	40.2	86.2
Qwen2.5-VL	✗	32B	80.6	74.1	69.9	-	42.5	-	72.4	83.2	48.0	-
InternVL3	✗	8B	81.2	70.0	69.3	-	38.8	880	69.7	73.6	37.6	86.6
Grounded Reasoning Models												
Pixel-Reasoner	Crop	7B	84.3	74.0	66.9	64.4	39.0	-	-	-	-	-
DeepEyes	Crop	7B	85.6	75.1	72.6	-	37.5	-	-	-	-	-
Thyme	Code	7B	82.2	77.0	72.0	64.8	-	863	-	-	-	86.1
Agentic Multimodal Model												
DeepEyesV2	General	7B	81.8	77.9	73.8	64.9	42.5	882	70.5	78.6	48.9	88.4
Δ (vs Qwen2.5-VL-7B)			+3.3	+6.3	+5.9	+7.6	+5.5	+18	+0.1	+5.9	+8.7	+2.2

Table 3: Results on multimodal reasoning benchmarks.

Model	Tool	Param Size	MathVista	MathVerse	MathVision	WeMath	DynaMath	LogicVista
<i>Open-source Models</i>								
LLaVA-OV	✗	7B	58.6	19.3	18.3	20.9	-	33.3
Qwen-2.5-VL	✗	7B	68.3	45.6	25.6	34.6	53.3	45.9
InternVL3	✗	8B	71.6	39.8	29.3	37.1	-	44.1
<i>Text-only Reasoning Models</i>								
MM-Eureka	✗	7B	72.6	-	28.1	21.8	-	46.3
ThinkLite	✗	7B	71.6	-	24.6	41.8	-	42.7
VL-Rethinker	✗	7B	73.7	-	28.4	36.3	-	42.7
VLAA-Thinker	✗	7B	71.7	-	24.2	35.7	-	45.9
<i>Grounded Reasoning Models</i>								
DeepEyes	Crop	7B	70.1	47.3	26.6	38.9	55.0	47.7
Thyme	Code	7B	70.0	-	27.6	39.3	-	49.0
<i>Agentic Multimodal Model</i>								
DeepEyesV2	General	7B	71.9	52.7	28.9	38.1	57.2	48.7
Δ (vs Qwen2.5-VL-7B)			+3.6	+7.1	+3.3	+3.5	+3.9	+2.8

Table 4: Results on search-oriented benchmarks.

Model	Tool	Model Size	FVQA-test	InfoSeek	MMSearch	SimpleVQA
<i>Open-source & Proprietary Models</i>						
GPT4o	✗	-	41.7	42.7	22.2	46.6
Gemini 2.5 Pro	✗	-	37.2	37.0	26.9	53.4
Qwen-2.5-VL	✗	7B	20.3	20.1	12.8	38.4
<i>Search Models</i>						
Qwen-2.5-VL	Search	7B	52.9	53.7	52.2	51.6
MMSearch-R1	Search	7B	58.4	55.1	53.8	57.4
WebWatcher	Search	7B	-	-	49.1	54.3
<i>Agentic Multimodal Model</i>						
DeepEyesV2	General	7B	60.6	51.1	63.7	59.4
Δ (vs Qwen2.5-VL-7B Search)			+7.7	-2.6	+11.5	+7.8

Table 5: **Ablation study on cold start data.** Perception and reason represent multi-turn agent data with code execution, respectively, while Long CoT refers to single-turn, purely text-based reasoning data. Long CoT refers to text-only reasoning data. For more details, please refer to Appendix A.1.

Perception	Reason	Long CoT	V* Bench	SEED 2 Plus	CharXiv descriptive	CharXiv reasoning	Math Vista	Math Verse
Qwen-2.5-VL-7B			63.9	69.2	68.9	35.7	65.3	36.2
✓			78.0	68.2	70.6	40.8	66.8	38.4
	✓		76.9	66.3	68.1	38.7	63.6	36.7
	✓	✓	75.9	68.7	72.0	43.1	68.2	47.6
✓	✓	✓	78.5	69.6	73.4	44.3	68.3	47.1

Table 6: **Ablation study on reinforcement learning data.** DeepEyesV2-SFT denotes the model after cold start. For more details about reinforcement learning data, please refer to Appendix A.1.

Perception	Reason	Search	V* Bench	SEED 2 Plus	CharXiv descriptive	CharXiv reasoning	Math Vista	Math Verse	Info Seek	MM Search
DeepEyesV2-SFT			78.5	69.6	73.4	44.3	68.3	47.1	47.9	56.8
✓			79.3	70.2	76.0	45.6	69.5	47.6	44.6	52.6
	✓		77.4	69.3	72.3	45.2	70.4	49.8	43.0	53.7
✓	✓		80.9	70.4	78.2	48.7	71.2	52.0	44.2	55.0
✓	✓	✓	81.8	70.5	78.6	48.9	71.9	52.7	51.1	63.7

adopt DAPO [55] as the optimization algorithm, with a batch size of 256 and 16 rollouts per prompt. The KL coefficient is set to 0.0, and the maximum response length is capped at 16,384 tokens. The learning rate is 1×10^{-6} , and the upper and lower clip ratios are 0.30 and 0.20, respectively. We utilize VLMEvalKit [12] to conduct all the evaluation, except for RealX-Bench, so the performance of DeepEyes may be a little different from [62].

5.2 Evaluation on RealX-Bench

We evaluate existing models and *DeepEyesV2* on RealX-Bench to assess their ability to integrate perception, search, and reasoning, and results are shown in Table 1.

Struggling to Integrate Perception, Search, and Reasoning. Even the best proprietary model achieves only 46.0% accuracy, far below human performance. Moreover, current models exhibit severe limitations in coordinating all three skills; for example, Gemini’s accuracy on subsets (27.8%) that require combining all three skills is much lower than its average accuracy (46.0%).

Search Benefits. Incorporating search tools effectively improves accuracy, especially in scenarios that require search. Using both text and image search yields substantial performance gains. However, text-only search provides larger improvements than image-only search, suggesting that current models still have limited ability to integrate image-search results effectively.

DeepEyesV2 demonstrates better coordination. Compared with other open-source models and models that incorporate zooming tools, *DeepEyesV2* achieves superior performance. In particular, on tasks that require coordination of all three capabilities, *DeepEyesV2* far outperforms other models, highlighting its strong multi-skill coordination.

5.3 Results on Other Benchmarks

Real-World & OCR & Chart Understanding. We evaluate *DeepEyesV2* across three categories of benchmarks: real-world understanding, OCR, and chart understanding. For comparison, we include two kinds of models: (i) open-source general-purpose MLLMs, including LLaVA-OneVision [24], Qwen2.5-VL [4], and InternVL3 [64]; and (ii) grounded reasoning models, such as DeepEyes [62] and Thyme [58]. DeepEyes performs fine-grained perception by cropping the target region, while Thyme manipulates images through executable code. Compared to base model Qwen2.5-VL-7B, *DeepEyesV2* demonstrates substantial performance gains, and even surpasses Qwen2.5-VL-32B in some benchmarks (Table 2), highlighting the effectiveness of tool-augmented reasoning. Moreover, it

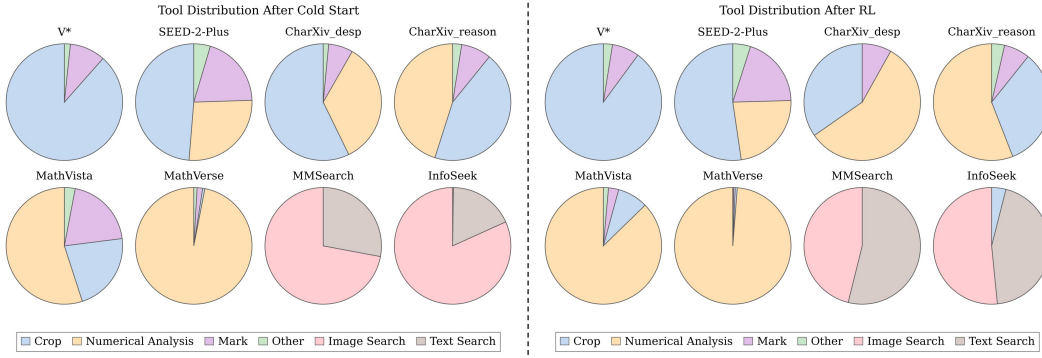


Figure 6: **Tool distribution comparison.** *DeepEyesV2* demonstrates the task-specific tool-calling distribution across different tasks. Reinforcement learning leads to a distribution shift.

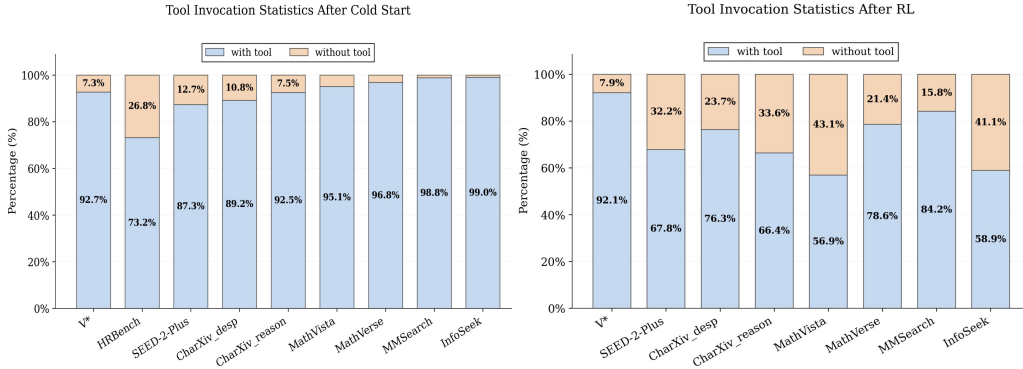


Figure 7: **Tool invocation statics.** After reinforcement learning, *DeepEyesV2*'s tool-calling frequency decreases, which enhances *DeepEyesV2*'s tool-calling flexibility and allow it to decide dynamically whether to invoke tools.

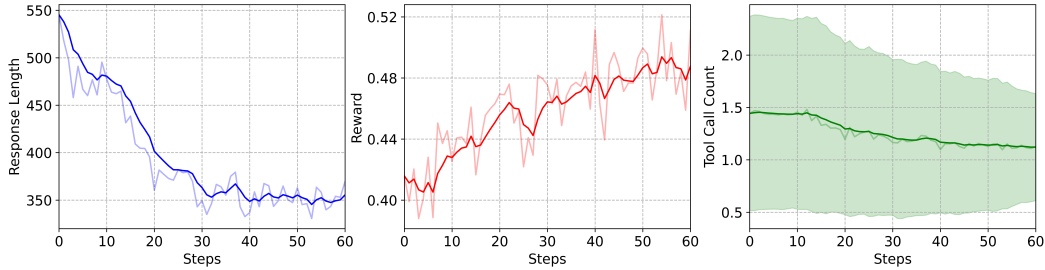


Figure 8: **Training dynamics of RL.** On the right, the green parts indicate the mean and standard deviation of the number of tool calls. During training, although the average response length steadily declines, the variance in tool-call counts remains high, indicating that the model can still perform complex tool-usage reasoning. Overall, reinforcement learning improves the efficiency of *DeepEyesV2*'s reasoning and tool usage.

consistently outperforms existing grounded reasoning models. These results indicate that dynamic tool invocation enables model to extract fine-grained details, thereby improving real-world scene comprehension.

Multimodal Reasoning. We further evaluate *DeepEyesV2* on mathematical reasoning benchmarks to assess its strong reasoning capability. As shown in Table 3, we compare *DeepEyesV2* against existing open-source MLLMs, such as Qwen2.5-VL [4], text-only multimodal reasoning models, including MM-Eureka [37], and grounded reasoning models, such as DeepEyes [62] and Thyme [58]. *DeepEyesV2* consistently outperforms these alternatives, and notably achieves stronger results than

text-only multimodal reasoning models, underscoring the benefit of tool use for enhancing mathematical reasoning.

Online Searching. To further examine the effectiveness of external information acquisition, we evaluate *DeepEyesV2* on search-oriented benchmarks. These datasets encompass knowledge-intensive visual question answering, fact verification, and multimodal retrieval-based reasoning, all of which require models to go beyond perceptual understanding and actively retrieve external evidence. For comparison, we benchmark *DeepEyesV2* against both general-purpose MLLMs such as Qwen2.5-VL, Gemini 2.5 Pro, and GPT4o, as well as models where search capability is incorporated [21, 16]. As shown in Table 4, *DeepEyesV2* demonstrates superior search capabilities, achieving consistently higher accuracy across all benchmarks.

5.4 Analysis

Training Data. To understand how training data influences the development of tool-use ability, we investigate the impact of different dataset compositions.

- **Cold Start Data.** We conduct ablations on the SFT dataset (Table 5). **Perception** and **reason** represent multi-turn agent data with code execution, respectively, while **Long CoT** refers to single-turn, purely text-based reasoning data. Directly evaluating Qwen2.5-VL-7B brings a great performance drop and confirms that existing MLLMs lack robust tool-use ability. Training only on perception data helps perception benchmarks but not reasoning; training only on reasoning data yields limited or negative gains, showing perception and reasoning rely on distinct tool-use patterns, with reasoning being more complex and harder to master. Adding long CoT trajectories substantially enhances reasoning and tool use, demonstrating that stronger thinking ability directly facilitates better tool use. Combining perception, reasoning, and CoT data achieves the best overall results, highlighting the complementary benefits of diverse supervision and the value of long CoT for complex reasoning.

Overall, these results highlight two key factors of cold start data: (i) **diversity**, as perception and reasoning rely on different tool-use patterns and data with diverse tasks should be involved to improve generalization; and (ii) **the inclusion of long CoT data**, which strengthens reasoning and substantially improves tool use on complex tasks.

- **RL Data.** We further conduct ablation studies on different subsets of RL data. Results are shown in Table 6. When training with only perception data, model achieves clear improvements on image-understanding benchmarks, but its performance on mathematics and search tasks declines. A similar trend is observed when using only reasoning data, where reasoning-related benchmarks improve, but perception and search tasks degrade. In contrast, combining perception and reasoning data yields consistent gains across both categories, demonstrating their complementary nature. Finally, incorporating search data leads to significant improvements on retrieval-oriented benchmarks, resulting in balanced and robust overall performance. These results emphasize that **data diversity** is critical for reinforcement learning in agentic multimodal models.

In-Deep Analysis. Then, we conduct an in-depth analysis of *DeepEyesV2*’s tool-use behavior after cold start and RL, comparing the two stages to better understand how training shapes and alters the model’s strategies for invoking tools.

- **Tool Distribution.** To understand how the model leverages tools across various scenarios, we analyze tool-use distributions over eight benchmarks before and after reinforcement learning (Figure 6). *DeepEyesV2* exhibits clear **task-dependent preferences**: in real-world perception tasks (V^*), model mainly uses cropping to obtain fine-grained visual details; in OCR tasks (SEED-Bench-2-Plus), it additionally performs region marking and numerical computations; chart-related tasks (CharXiv) involve more arithmetic operations; reasoning benchmarks (MathVista, MathVerse) are dominated by mathematical computations for intermediate verification and final answers; and search tasks (MMSearch, InfoSeek) primarily invoke search tools.

Moreover, when comparing behaviors before and after RL, we observe a notable **shift**. After reinforcement learning, model tends to perform more numerical operations across multiple tasks, and begins to integrate image manipulation tools (*e.g.*, cropping) with search in search benchmarks, indicating that RL helps model develop a more synergistic use of heterogeneous tools to solve complex queries.

- **Adaptive Thinking.** We further investigate tool-use efficiency by measuring the proportion of questions where model invokes tools before and after RL. As shown in Figure 7, prior to RL, model over-relies on tools, using them for most questions. After RL, however, tool invocation rate decreases significantly, showing that model learns adaptive reasoning: it solves problems directly when tools are unnecessary while still leveraging them when beneficial. Combined with Figure 9, these results highlight that reinforcement learning improves both **efficiency and flexibility**, enabling the balance between textual reasoning and tool calls.

- **Training Dynamic.** We further analyze model dynamics during RL by tracking response length, reward, and tool invocation frequency throughout training (Figure 8). The average number of tool calls steadily decreases over time; however, the variance remains large, indicating that model does not simply converge to a fixed number of tool invocations (e.g., one per query). Instead, model learns adaptive thinking: it selectively invokes tools when necessary, while handling simpler problems with minimal or no tool use. For more challenging queries, the number and complexity of tool calls remain high, reflecting flexible and task-aware strategies. Shorter response lengths further indicate more efficient reasoning, allocating detailed tool-based steps only when beneficial. Together, these findings highlight that reinforcement learning not only enhances tool-use **effectiveness**, while fostering **diversity, complexity, and efficiency** in reasoning.

6 Conclusion

In this work, we explore how to construct agentic multimodal models that can actively invoke tools and integrate them into reasoning, from the perspectives of training, dataset design, and evaluation. We introduce *DeepEyesV2* and conduct a practical two-stage training pipeline: supervised fine-tuning on a curated dataset to establish robust tool-use patterns, followed by reinforcement learning to strengthen and adapt tool invocation. Our analysis reveals task-dependent tool-use behaviors, and reinforcement learning enables more complex, context-aware tool combinations. Extensive experiments across perception, reasoning, and search benchmarks demonstrate the strong reasoning ability of *DeepEyesV2*, highlighting the advantages of combining tool invocation with reasoning.

References

- [1] Manoj Acharya, Kushal Kafle, and Christopher Kanan. Tallyqa: Answering complex counting questions. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 8076–8084, 2019.
- [2] Anthropic. Claude 4. <https://www.anthropic.com/news/claude-4>, 2025.
- [3] Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.
- [4] Junjie Bai, Jiayi Wei, Zhiwei Guo, Ziyu Zhou, et al. Qwen2.5-vl: A family of vision-language models from 7b to 72b. *arXiv preprint arXiv:2502.04567*, 2025.
- [5] Hardy Chen, Haoqin Tu, Fali Wang, Hui Liu, Xianfeng Tang, Xinya Du, Yuyin Zhou, and Cihang Xie. Sft or rl? an early investigation into training rl-like reasoning large vision-language models. *arXiv preprint arXiv:2504.11468*, 2025.
- [6] Shuang Chen, Yue Guo, Zhaochen Su, Yafu Li, Yulun Wu, Jiacheng Chen, Jiayu Chen, Weijie Wang, Xiaoye Qu, and Yu Cheng. Advancing multimodal reasoning: From optimized cold start to staged reinforcement learning. *arXiv preprint arXiv:2506.04207*, 2025.
- [7] Yang Chen, Hexiang Hu, Yi Luan, Haitian Sun, Soravit Changpinyo, Alan Ritter, and Ming-Wei Chang. Can pre-trained vision and language models answer visual information-seeking questions? *arXiv preprint arXiv:2302.11713*, 2023.
- [8] Zhe Chen, Jiannan Wu, Wenhai Wang, Weijie Su, Guo Chen, Sen Xing, Muyan Zhong, Qinglong Zhang, Xizhou Zhu, Lewei Lu, et al. Internvl: Scaling up vision foundation models and aligning for generic visual-linguistic tasks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 24185–24198, 2024.

- [9] Xianfu Cheng, Wei Zhang, Shiwei Zhang, Jian Yang, Xiangyuan Guan, Xianjie Wu, Xiang Li, Ge Zhang, Jiaheng Liu, Yuying Mai, et al. Simplevqa: Multimodal factuality evaluation for multimodal large language models. *arXiv preprint arXiv:2502.13059*, 2025.
- [10] Gabriel Comanici, Aakanksha Chowdhery, Richard Sutton, et al. Gemini 2.5 pro: Scaling agentic multimodal reasoning with retrieval and code execution. *arXiv preprint arXiv:2502.07012*, 2025.
- [11] 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 multimodal models. *arXiv e-prints*, pages arXiv–2409, 2024.
- [12] Haodong Duan, Junming Yang, Yuxuan Qiao, Xinyu Fang, Lin Chen, Yuan Liu, Xiaoyi Dong, Yuhang Zang, Pan Zhang, Jiaqi Wang, et al. Vlmevalkit: An open-source toolkit for evaluating large multi-modality models. In *Proceedings of the 32nd ACM international conference on multimedia*, pages 11198–11201, 2024.
- [13] Yue Fan, Xuehai He, Diji Yang, Kaizhi Zheng, Ching-Chen Kuo, Yuting Zheng, Sravana Jyothi Narayanaraju, Xinze Guan, and Xin Eric Wang. Grit: Teaching mllms to think with images. *arXiv preprint arXiv:2505.15879*, 2025.
- [14] Jiazhan Feng, Shijue Huang, Xingwei Qu, Ge Zhang, Yujia Qin, Baoquan Zhong, Chengquan Jiang, Jinxin Chi, and Wanjun Zhong. Retool: Reinforcement learning for strategic tool use in llms. *arXiv preprint arXiv:2504.11536*, 2025.
- [15] Chaoyou Fu, Haojia Lin, Zuwei Long, Yunhang Shen, Yuhang Dai, Meng Zhao, Yi-Fan Zhang, Shaoqi Dong, Yangze Li, Xiong Wang, et al. Vita: Towards open-source interactive omni multimodal llm. *arXiv preprint arXiv:2408.05211*, 2024.
- [16] Xinyu Geng, Peng Xia, Zhen Zhang, Xinyu Wang, Qiuchen Wang, Ruixue Ding, Chenxi Wang, Jialong Wu, Yida Zhao, Kuan Li, et al. Webwatcher: Breaking new frontiers of vision-language deep research agent. *arXiv preprint arXiv:2508.05748*, 2025.
- [17] Jack Hong, Shilin Yan, Jiayin Cai, Xiaolong Jiang, Yao Hu, and Weidi Xie. Worldsense: Evaluating real-world omnimodal understanding for multimodal llms. *arXiv preprint arXiv:2502.04326*, 2025.
- [18] Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.
- [19] Jitesh Jain, Zhengyuan Yang, Humphrey Shi, Jianfeng Gao, and Jianwei Yang. Ola-vlm: Elevating visual perception in multimodal llms with auxiliary embedding distillation. *arXiv preprint arXiv:2412.09585*, 2024.
- [20] Chaoya Jiang, Yongrui Heng, Wei Ye, Han Yang, Haiyang Xu, Ming Yan, Ji Zhang, Fei Huang, and Shikun Zhang. Vlm-r3: Region recognition, reasoning, and refinement for enhanced multimodal chain-of-thought. *arXiv preprint arXiv:2505.16192*, 2025.
- [21] Dongzhi Jiang, Renrui Zhang, Ziyu Guo, Yanmin Wu, Jiayi Lei, Pengshuo Qiu, Pan Lu, Zehui Chen, Chaoyou Fu, Guanglu Song, et al. Mmsearch: Benchmarking the potential of large models as multi-modal search engines. *arXiv preprint arXiv:2409.12959*, 2024.
- [22] 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.
- [23] Xin Lai, Junyi Li, Wei Li, Tao Liu, Tianjian Li, and Hengshuang Zhao. Mini-o3: Scaling up reasoning patterns and interaction turns for visual search. *arXiv preprint arXiv:2509.07969*, 2025.

- [24] 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.
- [25] Bohao Li, Yuying Ge, Yi Chen, Yixiao Ge, Ruimao Zhang, and Ying Shan. Seed-bench-2-plus: Benchmarking multimodal large language models with text-rich visual comprehension. *arXiv preprint arXiv:2404.16790*, 2024.
- [26] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International conference on machine learning*, pages 19730–19742. PMLR, 2023.
- [27] Lei Li, Yuqi Wang, Runxin Xu, Peiyi Wang, Xiachong Feng, Lingpeng Kong, and Qi Liu. Multimodal arxiv: A dataset for improving scientific comprehension of large vision-language models. *arXiv preprint arXiv:2403.00231*, 2024.
- [28] Yadong Li, Jun Liu, Tao Zhang, Song Chen, Tianpeng Li, Zehuan Li, Lijun Liu, Lingfeng Ming, Guosheng Dong, Da Pan, et al. Baichuan-omni-1.5 technical report. *arXiv preprint arXiv:2501.15368*, 2025.
- [29] Zhenzhi Li, Yichi Zhang, Haoran Duan, Yizhou Zhang, et al. Seed-bench: Benchmarking multimodal llms with generative comprehension. *arXiv preprint arXiv:2307.16125*, 2023.
- [30] Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 26296–26306, 2024.
- [31] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36:34892–34916, 2023.
- [32] Yuliang Liu, Zhang Li, Mingxin Huang, Biao Yang, Wenwen Yu, Chunyuan Li, Xu-Cheng Yin, Cheng-Lin Liu, Lianwen Jin, and Xiang Bai. Ocrbench: on the hidden mystery of ocr in large multimodal models. *Science China Information Sciences*, 67(12):220102, 2024.
- [33] Ziyu Liu, Yuhang Zang, Yushan Zou, Zijian Liang, Xiaoyi Dong, Yuhang Cao, Haodong Duan, Dahua Lin, and Jiaqi Wang. Visual agentic reinforcement fine-tuning. *arXiv preprint arXiv:2505.14246*, 2025.
- [34] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.
- [35] 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. *arXiv preprint arXiv:2310.02255*, 2023.
- [36] Ahmed Masry, Do Xuan Long, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. Chartqa: A benchmark for question answering about charts with visual and logical reasoning. *arXiv preprint arXiv:2203.10244*, 2022.
- [37] Fanqing Meng, Lingxiao Du, Zongkai Liu, Zhixiang Zhou, Quanfeng Lu, Daocheng Fu, Botian Shi, Wenhai Wang, Junjun He, Kaipeng Zhang, et al. Mm-eureka: Exploring visual aha moment with rule-based large-scale reinforcement learning. *CoRR*, 2025.
- [38] OpenAI. Thinking with images. <https://openai.com/index/thinking-with-images/>, 2025.
- [39] Runqi Qiao, Qiuna Tan, Guanting Dong, Minhui Wu, Chong Sun, Xiaoshuai Song, Zhuoma GongQue, Shanglin Lei, Zhe Wei, Miaoxuan Zhang, et al. We-math: Does your large multimodal model achieve human-like mathematical reasoning? *arXiv preprint arXiv:2407.01284*, 2024.
- [40] Huatong Song, Jinhao Jiang, Yingqian Min, Jie Chen, Zhipeng Chen, Wayne Xin Zhao, Lei Fang, and Ji-Rong Wen. R1-searcher: Incentivizing the search capability in llms via reinforcement learning. *arXiv preprint arXiv:2503.05592*, 2025.

- [41] Alex Su, Haozhe Wang, Weiming Ren, Fangzhen Lin, and Wenhui Chen. Pixel reasoner: Incentivizing pixel-space reasoning with curiosity-driven reinforcement learning. *arXiv preprint arXiv:2505.15966*, 2025.
- [42] Kaibin Tian, Zijie Xin, and Jiazhen Liu. SeekWorld: Geolocation is a natural RL task for o3-like visual clue-tracking. <https://github.com/TheEighthDay/SeekWorld>, 2025. GitHub repository.
- [43] Haochen Wang, Xiangtai Li, Zilong Huang, Anran Wang, Jiacong Wang, Tao Zhang, Jiani Zheng, Sule Bai, Zijian Kang, Jiashi Feng, et al. Traceable evidence enhanced visual grounded reasoning: Evaluation and methodology. *arXiv preprint arXiv:2507.07999*, 2025.
- [44] Haozhe Wang, Chao Qu, Zuming Huang, Wei Chu, Fangzhen Lin, and Wenhui Chen. V1-rethinker: Incentivizing self-reflection of vision-language models with reinforcement learning. *arXiv preprint arXiv:2504.08837*, 2025.
- [45] Jiacong Wang, Zijian Kang, Haochen Wang, Haiyong Jiang, Jiawen Li, Bohong Wu, Ya Wang, Jiao Ran, Xiao Liang, Chao Feng, et al. Vgr: Visual grounded reasoning. *arXiv preprint arXiv:2506.11991*, 2025.
- [46] Ke Wang, Juntong Pan, Weikang Shi, Zimu Lu, Houxing Ren, Aojun Zhou, Mingjie Zhan, and Hongsheng Li. Measuring multimodal mathematical reasoning with math-vision dataset. *Advances in Neural Information Processing Systems*, 37:95095–95169, 2024.
- [47] Ke Wang, Houxing Ren, Aojun Zhou, Zimu Lu, Sichun Luo, Weikang Shi, Renrui Zhang, Linqi Song, Mingjie Zhan, and Hongsheng Li. Mathcoder: Seamless code integration in llms for enhanced mathematical reasoning. *arXiv preprint arXiv:2310.03731*, 2023.
- [48] Qiuchen Wang, Ruixue Ding, Yu Zeng, Zehui Chen, Lin Chen, Shihang Wang, Pengjun Xie, Fei Huang, and Feng Zhao. Vrag-rl: Empower vision-perception-based rag for visually rich information understanding via iterative reasoning with reinforcement learning. *arXiv preprint arXiv:2505.22019*, 2025.
- [49] Wenbin Wang, Liang Ding, Minyan Zeng, Xiabin Zhou, Li Shen, Yong Luo, Wei Yu, and Dacheng Tao. Divide, conquer and combine: A training-free framework for high-resolution image perception in multimodal large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 7907–7915, 2025.
- [50] Xiyao Wang, Zhengyuan Yang, Chao Feng, Hongjin Lu, Linjie Li, Chung-Ching Lin, Kevin Lin, Furong Huang, and Lijuan Wang. Sota with less: Mcts-guided sample selection for data-efficient visual reasoning self-improvement. *arXiv preprint arXiv:2504.07934*, 2025.
- [51] 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.
- [52] Jinming Wu, Zihao Deng, Wei Li, Yiding Liu, Bo You, Bo Li, Zejun Ma, and Ziwei Liu. Mmsearch-rl: Incentivizing llms to search. *arXiv preprint arXiv:2506.20670*, 2025.
- [53] Penghao Wu and Saining Xie. V?: Guided visual search as a core mechanism in multimodal llms. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13084–13094, 2024.
- [54] Yijia Xiao, Edward Sun, Tianyu Liu, and Wei Wang. Logicvista: Multimodal llm logical reasoning benchmark in visual contexts. *arXiv preprint arXiv:2407.04973*, 2024.
- [55] Qiyang Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.
- [56] Renrui Zhang, Dongzhi Jiang, Yichi Zhang, Haokun Lin, Ziyu Guo, Pengshuo Qiu, Aojun Zhou, Pan Lu, Kai-Wei Chang, Yu Qiao, et al. Mathverse: Does your multi-modal llm truly see the diagrams in visual math problems? In *European Conference on Computer Vision*, pages 169–186. Springer, 2024.

- [57] Xintong Zhang, Zhi Gao, Bofei Zhang, Pengxiang Li, Xiaowen Zhang, Yang Liu, Tao Yuan, Yuwei Wu, Yunde Jia, Song-Chun Zhu, et al. Chain-of-focus: Adaptive visual search and zooming for multimodal reasoning via rl. *arXiv preprint arXiv:2505.15436*, 2025.
- [58] Yi-Fan Zhang, Xingyu Lu, Shukang Yin, Chaoyou Fu, Wei Chen, Xiao Hu, Bin Wen, Kaiyu Jiang, Changyi Liu, Tianke Zhang, et al. Thyme: Think beyond images. *arXiv preprint arXiv:2508.11630*, 2025.
- [59] Yi-Fan Zhang, Huanyu Zhang, Haochen Tian, Chaoyou Fu, Shuangqing Zhang, Junfei Wu, Feng Li, Kun Wang, Qingsong Wen, Zhang Zhang, et al. Mme-realworld: Could your multimodal llm challenge high-resolution real-world scenarios that are difficult for humans? *arXiv preprint arXiv:2408.13257*, 2024.
- [60] Jiaxing Zhao, Xihan Wei, and Liefeng Bo. R1-omni: Explainable omni-multimodal emotion recognition with reinforcement learning. *arXiv preprint arXiv:2503.05379*, 2025.
- [61] Shitian Zhao, Haoquan Zhang, Shaoheng Lin, Ming Li, Qilong Wu, Kaipeng Zhang, and Chen Wei. Pyvision: Agentic vision with dynamic tooling. *arXiv preprint arXiv:2507.07998*, 2025.
- [62] Yifan Zheng, Ziyu Wang, Ming Li, Jialiang Zhang, et al. Deepeyes: Towards agentic multimodal reasoning via tool-augmented vision models. *arXiv preprint arXiv:2501.01234*, 2025.
- [63] Yuxiang Zheng, Dayuan Fu, Xiangkun Hu, Xiaojie Cai, Lyumanshan Ye, Pengrui Lu, and Pengfei Liu. Deepresearcher: Scaling deep research via reinforcement learning in real-world environments. *arXiv preprint arXiv:2504.03160*, 2025.
- [64] Jinguo Zhu, Weiyun Wang, Zhe Chen, Zhaoyang Liu, Shenglong Ye, Lixin Gu, Hao Tian, Yuchen Duan, Weijie Su, Jie Shao, et al. Internv13: Exploring advanced training and test-time recipes for open-source multimodal models. *arXiv preprint arXiv:2504.10479*, 2025.

A Appendix

A.1 Training Data

For perception-oriented tasks, we include V* [53], ArxivQA [27], Pixmo Counting [11], TallyQA [1], and SeekWorld [42], covering a wide range of scenarios such as object recognition, visual counting, and chart interpretation. For reasoning tasks, we adopt ReVisual [6] to provide complex reasoning problems, and additionally incorporate MathCoder [47] and Retool [14] to supplement with executable code-based problem-solving examples. Besides, we also include long Chain-of-Cot (CoT) reasoning data in cold start subset. For search-related tasks, we employ MMSearch-R1 [52], which includes both image-based and text-based retrieval questions. We further include data from VGR [45], Chain-of-Focus [57], and VLM-R³ [20] to strengthen the reinforcement learning corpus.

We present the distributions of our cold start and RL data in Figure 9. The cold start data is divided into four parts: perception, reasoning, search, and Long CoT, while the RL data includes perception, reasoning, and search.

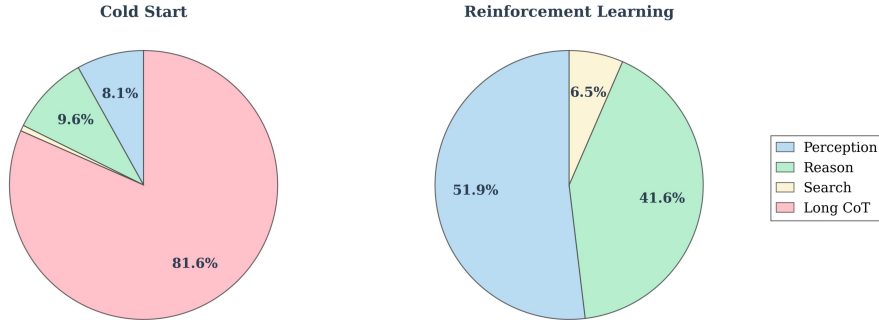


Figure 9: **Distribution of cold start and reinforcement learning data.**

A.2 Evaluation Protocol

we summarize the benchmarks and models we compare across different kinds of tasks. Benchmarks cover three main categories: real-world understanding, mathematical reasoning, and search-intensive tasks, capturing the diversity of challenges faced by agentic multimodal models.

Real-World & OCR & Chart Understanding. For real-world understanding, we adopt V* [53], HRBench [49], MME-RealWorld [59], and TreeBench [43]; for OCR, we use OCRBench [32] and Seed-Bench-2-Plus [25]; and for chart reasoning, we evaluate on CharXiv [51] and ChartQA [36]. For comparison, we include two kinds of models: (i) open-source general-purpose MLLMs, including LLaVA-OneVision [24], Qwen2.5-VL [4], and InternVL3 [64]; and (ii) grounded reasoning models, such as Pixel-Reasoner [41], DeepEyes [62] and Thyme [58].

Multimodal Reasoning. We include MathVista [35], MathVerse [56], MathVision [46], WeMath [39], and LogicVista [54]. We compare *DeepEyesV2* against existing open-source MLLMs, such as Qwen2.5-VL [4], text-only multimodal reasoning models, including MM-Eureka [37], ThinkLite [50], VL-Rethinker [44], and VLAA-Thinker [5], and grounded reasoning models, such as DeepEyes [62] and Thyme [58].

Online Searching. We compare *DeepEyesV2* on FVQA-test [52], InfoSeek [7], MMSearch [21], and SimpleVQA [9]. We benchmark *DeepEyesV2* against both general-purpose MLLMs such as Qwen2.5-VL [4], Gemini 2.5 Pro [10], and GPT4o [18], as well as models where search capability is incorporated [21, 16].

A.3 Tool Taxonomy

The tools can be categorized into three major classes:

1. Code Execution. Code execution covers a set of operations that require Python-based execution. We further divide it into four subtypes:

- **Crop:** extract a specific region of the input image for fine-grained analysis.

```
cropped = image_1.crop((top, left, right, bottom))

plt.imshow(cropped)
plt.axis('off')
plt.show()
```

- **Numerical Analysis:** perform numerical computations, formula evaluation, or quantitative reasoning.

```
import math
height = 68

w = height / math.tan(math.radians(37))

x = w / math.tan(math.radians(46))
print(f"w = {w}")
print(f"x = {x}")
```

- **Mark:** annotate or highlight regions of interest in the image to support reasoning.

```
from PIL import ImageDraw

draw = ImageDraw.Draw(image_1)

box = (50, 50, 300, 200)
color = (255, 0, 0)
thickness = 8

draw.rectangle(box, outline=color, width=thickness)

plt.imshow(image_1)
plt.show()
```

- **Other:** other manipulation operations such as rotation, enhancement, or resizing.

```
from PIL import ImageEnhance

enhancer = ImageEnhance.Brightness(image_1)

factor = 1.5

bright_img = enhancer.enhance(factor)

plt.imshow(bright_img)
plt.axis('off')
plt.show()
```

2. Image Search. Given an image query, we utilize SerpAPI to retrieve visually similar results from the web, returning candidate images with thumbnails.

3. Text Search. Based on a textual query, we retrieve relevant webpages and provides both titles and snippets of content.

A.4 Error Analysis

We categorize the errors made by *DeepEyesV2* into three main types (Figure 10). First, tool execution errors occur when the model generates a correct reasoning trajectory but fails during tool operation, such as cropping the wrong region or using incorrect search keywords. Second, tool selection

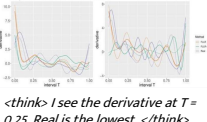
Tool Execution Error	Tool Selection Error
 Q: What color is the bicycle frame in the image? A: Brown GT: Dark blue <i><think> I need to zoom in the bicycle frame. </think></i> <pre>python subplot = image.crop(region) plt.imshow(subplot) </pre> <i>Error Reason:</i> The model selected the correct tool (image cropping), but the execution went wrong as the target region was mislocalized.	 Q: What is the specific model of the car in the image? A: Unknown GT: Dongfeng Honda <i><think> Let's search the model of this car. </think></i> Model of the dark blue car <i>Error Reason:</i> Model called the wrong tool (Text Search instead of Image Search)
 Q: When was the phone model in this image released? A: 2007 GT: 2022 <i><think> Let's search the release time of iPhone 14 pro. </think></i> iPhone release date <i>Error Reason:</i> The issue is that the model failed to specify "iPhone 14 Pro" in the query, retrieving a generic result for "iPhone"	 Q: Which one has the lowest derivative at $T = 0.25$ in plot 1? A: Real GT: FLLR-R <i><think> I see the derivative at $T = 0.25$. Real is the lowest. </think></i> <pre>python subplot = image.crop(region) plt.imshow(subplot) </pre> <i>Error Reason:</i> The model called the Image Crop tool and got the relevant region, but misparsed the result.

Figure 10: Error analysis.

errors arise when the model chooses an inappropriate tool for the task, for example selecting text search when an image search is required. Third, tool result analysis errors happen when the model correctly selects and executes a tool, but misinterprets or incorrectly analyzes the returned outputs. This categorization helps to identify the main sources of failure and guides future improvements in tool-invoked reasoning.

A.5 Prompt

SYSTEM_PROMPT

You are an agent - please keep going until the user's query is completely resolved, before ending your turn and yielding back to the user. Only terminate your turn when you are sure that the problem is solved.

Solve the following problem step by step. In your reasoning process, if the answer cannot be determined, you can write Python code in a Jupyter Notebook to process the image and extract more information from it. The stdout and stderr content, along with the images generated by "plt.show()" will be returned to better assist with the user query.

You MUST use the python tool to analyze or transform images whenever it could improve your understanding. This includes but is not limited to zooming in, rotating, adjusting contrast, computing statistics, or isolating features.

If you find you sufficient knowledge to confidently answer the question, you MUST conduct search to thoroughly seek the internet for information. No matter how complex the query, you will not give up until you find the corresponding information.

You can conduct image search, which will trigger a Google Lens search using the original image to retrieve relevant information that can help you confirm the visual content, and text search, which will use Google Search to return relevant information based on your query.

You MUST plan extensively before each function call, and reflect extensively on the outcomes of the previous function calls. DO NOT do this entire process by making function calls only, as this can impair your ability to solve the problem and think insightfully.

Additionally, you can combine python tool with search to assist in answering questions. Python tool can help enhance your understanding of images, while search tools can provide the knowledge you lack. Please use python tool and search flexibly. However, you can only call one type of tool in a single round; you cannot use a python tool and perform a search simultaneously.

For all the provided images, in order, the i-th image has already been read into the global variable "image_i" using the "PIL.Image.open()" function. For example, the first image can be accessed as "image_1". When writing Python code, you can directly use these variables without needing to read them again.

Tools

python

Your python code should be enclosed within `<code>` `</code>` tag.

Example for calling Python code in Jupyter Notebook:

```
<code>
```python
\# python code here
```
</code>
```

Note:

1. ****python**** can be called to analyze the image. ****python**** will respond with the output of the execution or time out after 300.0 seconds.
2. Like jupyter notebook, you can use Python code to process the input image and use "plt.show()" to visualize processed images in your code.
3. All python code are running in the same jupyter notebook kernel, which means the functions and variables are automatically stored after code execution.
4. Your program should always return in finite time. Do not write infinite loop in your code.
5. Writing file to disk is not allowed.

search

You are provided with function signatures within `<tools>``</tools>` XML tags:

```
<tool_call>
{"type": "function", "function":
{
  "name": "image_search",
  "description": "Retrieves top 10 images and descriptions
from Google's image search using the original image.
Should only be used once.",
},
{
  "name": "search",
  "description": "Performs batched web searches: supply an
array 'query'; the tool retrieves the top 10 results for
each query in one call.",
  "parameters": {
    "type": "object",
    "properties": {
      "query": {
        "type": "string",
        "description": "Search query to find
relevant information."
      }
    }
  }
}
```

```

    }
  },
  "required": [
    "query"
  ]
}
}
</tool_call>

```

Example for calling search: Return a json object with function name and arguments within <tool_call></tool_call> XML tags:

```

<tool_call>
{"name": "image_search"}
</tool_call>
<tool_call>
{"name": "search", "arguments": {"query": "Does Cloudflare
analyze submitted data to block attacks"}}
</tool_call>

```

Note:

1. You MUST engage in many interactions, delving deeply into the topic to explore all possible aspects until a satisfactory answer is found.
2. Before presenting a Final Answer, you will ****cross-check**** and ****validate the information**** you've gathered to confirm its accuracy and reliability.
3. You will carefully analyze each information source to ensure that all data is current, relevant, and from credible origins.
4. Please note that you can ****only**** call search once at a time. If you need to perform multiple searches, please do so in the next round.
5. You can ****only**** conduct image search once.

USER_PROMPT

```

{Question}
You must put your answer inside <answer> </answer> tags, i.e.,
<answer> answer here </answer>. Please reason step by step. Use Python
code to process the image if necessary. You can conduct search to seek the Inter-
net. Format strictly as <think> </think> <code> </code>(if code is needed)
or <think> </think> <tool_call> </tool_call>(if function call is needed) or
<think> <think> <answer> </answer>.

```

RETURN_CODE_USER_PROMPT

Code execution result:

```

stdout:
'''
{stdout}
'''

stderr:
'''
{stderr}
'''

Image:
{image}

```

RETURN_IMAGE_SEARCH_USER_PROMPT

A Google image search for the image found 5 results:

Web Results

1. <image> [{title}]
2. <image> [{title}]

RETURN_TEXT_SEARCH_USER_PROMPT

A Google search for '{query}' found 5 results:

Web Results

1. [{title}] ({link}) {snippet}
2. [{title}] ({link}) {snippet}

A.6 More Cases

We show more cases in Figure 11, 12, 13.



Figure 11: Case Study 1.



Where is the card of Two-Sided Style located in the image?





<think>The image shows a collection of cards with various characters. The question asks for the location of the card of "Two-Sided Style." However, the image does not provide any text or labels that identify the characters or their styles. *Without additional context or information, it's not possible to determine which card corresponds to "Two-Sided Style."*</think>

Two-Sided Style card



Fate/Grand Order Wiki | Fandom is of Two-Sided Style.
 

<think>The text search results suggest that the specific card for "Two-Sided Style" is related to "Fate/Grand Order," a popular mobile game.. *Therefore, a image search is necessary to find the location of the card.*</think>






Fate/Grand Order
 

<think>Given the context of the image and the search results, it is likely that the card for "Two-Sided Style" is located in the second of bottom row of the image.</think><answer>**The second in bottom row**</answer>

Figure 12: Case Study 2.

What is the name of the one in the first position from the left in the third row of the picture?

<think>The user's question is to identify the character in the first position of the third row in the image. First, I need to locate the specific position of the third row in the image, then identify the character in the first position of that row. The image contains multiple rows and columns of cartoon characters. To further confirm the identity of this character, I can crop this area and perform an image search. </think>

```
''' python
width, height = image_1.size
cell_width = width // 4
.....
cropped_image = image_1.crop
((left, top, right, bottom))
plt.show()
.....
'''
```

<think>The cropped image shows a yellow character. Based on its appearance, this character resembles "Capybara". To further verify, an image search can be conducted using this cropped image to check. </think>

Capybara

<think>The image search results show that the yellow pumpkin-shaped character in the upper left corner of the first image is very similar to the character in the cropped image. Therefore, it can be confirmed that this character is "Capybara". </think> <answer>**Capybara**</answer>

Figure 13: Case Study 3.