

CodeDance: A Dynamic Tool-integrated MLLM for Executable Visual Reasoning

Qi Song^{1,4*} Honglin Li^{2,4*} Yingchen Yu³ Haoyi Zhou¹ Lin Yang²

Song Bai³ Qi She⁴ Zilong Huang^{3†} Yunqing Zhao^{3†}

¹Beihang University ²Westlake University

³ByteDance Singapore ⁴ByteDance China

zilong.huang2020@gmail.com yunqing.z.0817@gmail.com

[CodeDance-VL.github.io](https://github.com/ByteDance/CodeDance-VL)

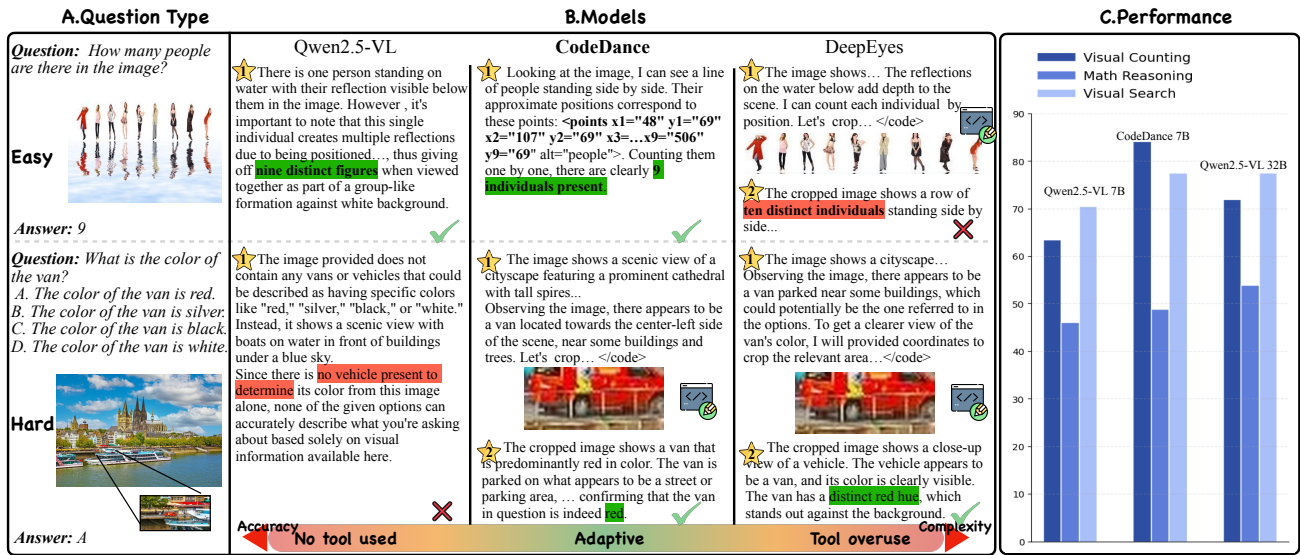


Figure 1. Motivation and effectiveness of dynamic tool invocation in CodeDance. **Left:** qualitative examples show that both tool underuse (unable to invoke tools on challenging tasks) and tool overuse (redundant calls on easy tasks) lead to hallucinated reasoning, incorrect answers, and unnecessary complexity (more reasoning turns and longer rollout time), whereas CodeDance dynamically invokes tools according to task difficulty to obtain correct solutions. **Right:** quantitative results show CodeDance-7B consistently surpasses all Qwen2.5-VL 7B baselines and exceeds the 32B version on several tasks.

Abstract

Recent releases such as o3 highlight human-like “thinking with images” reasoning that combines tool use with step-wise verification, yet most open-source approaches still rely on text-only chains, rigid visual schemas, or single-step pipelines, limiting flexibility, interpretability, and transferability on complex tasks. We introduce **CodeDance**, which explores executable code as a general solver for visual reasoning. Unlike fixed-schema calls (e.g., only predicting bounding-box coordinates), CodeDance defines, composes, and executes code to orchestrate multiple tools, compute intermediate results, and render visual artifacts (e.g., boxes, lines, plots) that support transparent, self-checkable rea-

soning. To guide this process, we introduce a reward for balanced and adaptive tool calling, which balances exploration with efficiency and mitigates tool overuse. Interestingly, beyond the expected capabilities taught by atomic supervision, we empirically observe novel emergent behaviors during RL training: CodeDance demonstrates novel tool invocations, unseen compositions, and cross-task transfer. These behaviors arise without task-specific fine-tuning, suggesting a general and scalable mechanism for executable visual reasoning. Extensive experiments across reasoning benchmarks (e.g., visual search, math, chart QA) show that CodeDance not only consistently outperforms schema-driven and text-only baselines, but also surpasses closed models such as GPT-4o and larger open-source models.

*Equal contribution. †Corresponding authors.

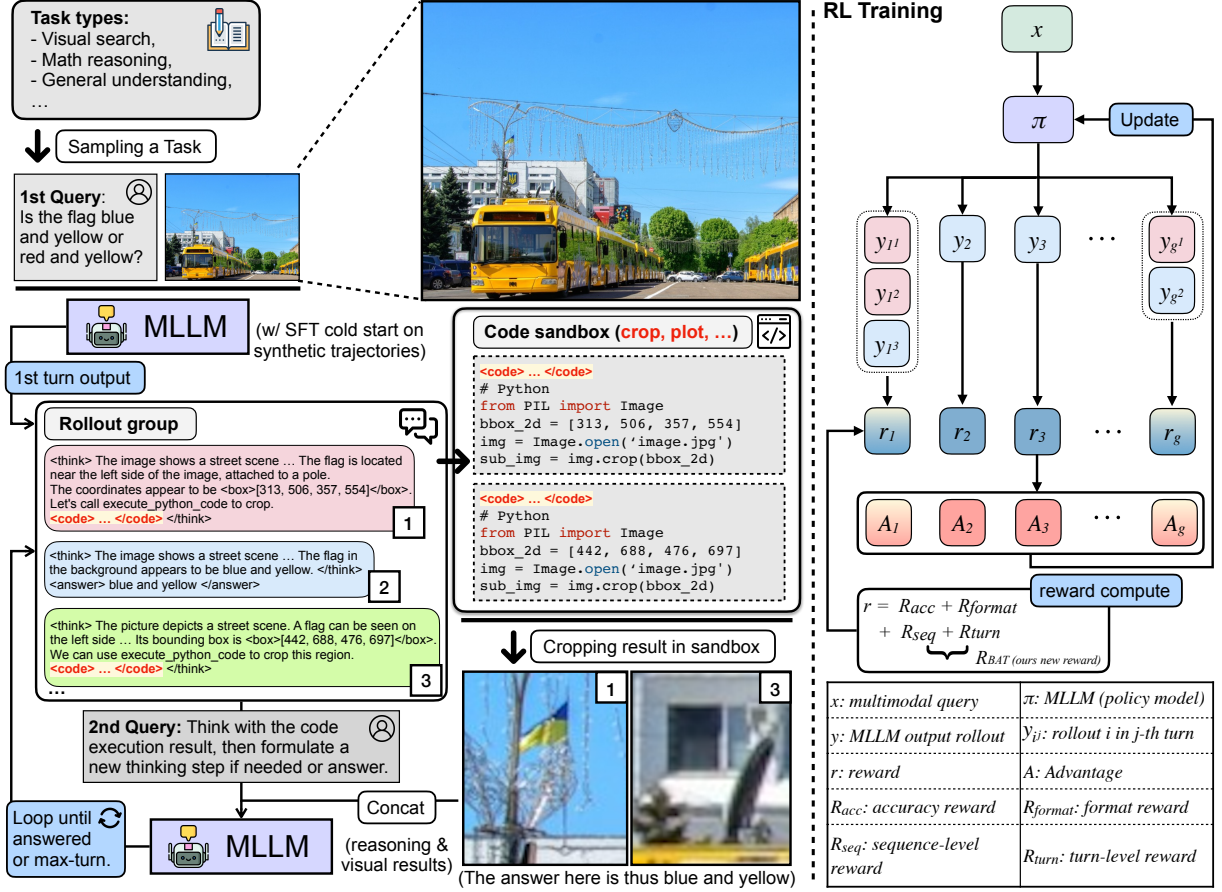


Figure 2. **Overview.** CodeDance enables executable visual reasoning and leverages tool calling adaptively based on task difficulty.

1. Introduction

Multimodal Large Language Models (MLLMs) have made rapid progress, showing strong capabilities in both visual perception and reasoning. By leveraging the language-centric chain-of-thought (CoT) mechanism [2, 27], models can decompose complex problems into intermediate steps, thereby improving performance on challenging tasks. However, when extended to visual tasks, traditional CoT relies on static context, preventing models from dynamically interacting with visual inputs or incorporating new observations during intermediate reasoning [4, 44], creating an information bottleneck that hinders multi-round focusing and validation. To address this, the o3 system [12] integrates the ability to actively seek new information through multiple tool invocations, supporting iterative reasoning over visual inputs and demonstrating strong perception and analysis.

Research gaps. While recent models have made notable progress, fundamental gaps remain unresolved. (1) Current approaches largely extend CoT into multimodal reasoning via text-only templates, failing to incorporate new observations, refine intermediate steps, or validate its reasoning against visual evidence [7, 8]. (2) In addition, o3 remains a

proprietary black-box system: its internal mechanisms are inaccessible, its reasoning process is less transparent, and its outputs cannot be systematically studied or reproduced. (3) Most open-source systems incorporating visual reasoning remain restricted to predefined visual workflows, or rigid and schema-based pipelines (e.g., predicting bounding box coordinates for cropping operations), which are inherently inflexible and task-specific, limiting transfer to new tools and tasks [20, 21, 39, 43]. (4) Existing methods [41, 43] do not consider when a model should invoke tools, leading to tool underuse or overuse, as shown in Figure 1. Consequently, the field still lacks an open and verifiable medium that is general across tools and tasks for multimodal reasoning, allowing MLLMs to dynamically compose tools, produce intermediate artifacts, and self-check their outputs in a transparent and reproducible manner. Addressing this gap is crucial for achieving task-adaptive, explainable, and transferable reasoning across complex real-world tasks.

In this work, we introduce **CodeDance**, a multimodal reasoning framework that leverages executable code as a unified medium for visual reasoning. Unlike prior schema-based pipelines with fixed operation templates, code en-

Table 1. Experiment results on benchmarks including Counting, Visual Search, and General datasets. [†] denotes reproduced results.

Model	Visual Counting		Visual Search			General	
	CountBench	PixmoCount	V* Bench	HR-Bench 4K	HR-Bench 8K	ChartQA	CharXiv
Closed-Source MLLMs							
GPT-4o	87.9	-	67.5	65.0	59.6	86.7	47.1
Open-Source MLLMs							
Llava-OneVision-7B	82.3	54.4	72.7	68.5	60.0	80.4	27.1
Llava-OneVision-72B	-	60.7	73.8	66.3	60.9	83.7	-
InternVL2.5-8B	55.9	-	73.7	72.0	65.5	82.8	37.2
InternVL3-8B	80.3	-	70.2	70.5	70.0	86.1	38.3
InternVL3-78B	-	-	76.4	75.5	67.3	89.7	46.0
Qwen2.5-VL-72B	93.6	62.3	84.8	79.4	76.3	89.5	49.7
Qwen2.5-VL-32B	87.8	56.0	85.9	74.8	71.6	-	47.6
Qwen2.5-VL-7B	76.5	50.4	76.4	69.0	66.0	86.3	42.1
Tool-Integrated Open-Source MLLM							
Pixel Reasoner-7B	-	-	84.3	72.9	66.9	-	-
Deepeyes-7B [†]	80.4	57.2	90.4	74.8	71.9	78.2	-
Thyme-VL-7B	84.8 [†]	-	82.2	77.0	72.0	86.1	44.2 [†]
CodeDance-7B	91.2	77.1	84.8	75.2	72.3	87.5	44.1
Δ vs. Qwen2.5-VL-7B	↑19.2%	↑53.0%	↑11.0%	↑9.0%	↑9.5%	↑1.4%	↑4.7%

ables the model to define, compose, and execute diverse visual-symbolic operations, producing both intermediate artifacts (e.g., cropped regions, plots, annotations) and final answers within a unified, verifiable reasoning process. To equip the model with fundamental skills, we curate a high-quality trajectory dataset and use supervised fine-tuning (SFT) to teach atomic capabilities such as counting, spatial grounding, and image annotation, enabling an iterative exploration-reflection reasoning process. Building on this foundation, we employ reinforcement learning (RL) to further enhance tool-based reasoning. A central challenge we identify is a *trade-off between exploration and selectivity*: naïve policies often overuse tools, incurring unnecessary steps, or underuse them, failing to leverage visual interactions when needed. To address this, we design a difficulty-adaptive tool-reward mechanism that adjusts incentives based on task complexity, encouraging longer operation chains for hard problems while discouraging unnecessary calls for simple ones. This aligns learning with the structure of multimodal tasks, resulting in more adaptive and transparent reasoning. Together, these components enable CodeDance to advance beyond rigid schema-based methods and offer an open, generalizable medium for executable visual reasoning.

Empirical observation of emergent behaviors. Although trained only on atomic operations, the framework exhibits behaviors beyond direct supervision during RL, including composing operations (e.g., localization followed

by counting) and generating new procedural code for visual analysis. These behaviors arise from the base model’s pre-trained knowledge and are driven by adaptive rewards rather than explicit instruction, highlighting the scalability of dynamic invocation for more flexible and general reasoning.

Our contributions are summarized as follows: (1) We introduce CodeDance, a multimodal agent that can “think with images” by planning and composing visual-symbolic operations through executable code as a unified medium. To this end, we curate a 34K high-quality SFT dataset covering diverse atomic code capabilities (e.g., cropping, drawing, point plotting), and additionally design a difficulty-adaptive reward mechanism for RL, enabling multi-turn reasoning and balanced tool use. (2) We evaluate CodeDance on more than 10 multimodal benchmarks, spanning both general perception and complex reasoning (e.g., visual search, math reasoning). Across multiple benchmarks, it outperforms advanced closed models (e.g., GPT-4o) and larger open-source baselines (e.g., Qwen2.5-VL-32B), demonstrating strong perception and reasoning capabilities, and broad generalizability. (3) Despite being trained only on atomic operations, CodeDance empirically exhibits emergent behaviors during RL training, including spontaneous novel tool routines, unseen operation compositions, and cross-task transfer to novel tasks. These promising observations highlight the scalability and generality of our framework, and we empirically verify them in ablation studies.

2. Related Works

Multimodal reasoning and tool invocation. Building upon text-based chain-of-thought (CoT) reasoning [27, 33], researchers have extended intermediate reasoning steps to multimodal settings [34, 43] including counting [38], localization [29], charts [9], and visual math [3]. To enhance reasoning capabilities, recent works integrate external tools through reasoning-and-acting frameworks [32, 33], learned API usage [15], and multimodal agents that orchestrate OCR, detection, and editors [17, 28]. ViperGPT [22] compiles queries into executable programs. Recent models like OpenAI’s o3 [12] integrate comprehensive tool capabilities directly into reasoning chains, trained via RL on large-scale CoT data. Other approaches include RL-based tool invocation [20, 43] and SFT-based methods [26]. However, challenges remain such as ad-hoc operations, sparse supervision, limited task coverage and lack of comprehensive evaluation. We diverge from prior work by pursuing code as a general medium to execute multimodal reasoning across diverse atomic abilities.

Adaptive thinking capability of MLLMs. Recent work on adaptive reasoning in LLMs explores how models can dynamically decide when and how much to think [36] instead of relying on fixed chain-of-thought steps. Approaches such as adaptive thinking [23, 30], concise reasoning [19], and learn-to-switch frameworks [40] show that LLMs can selectively perform deeper reasoning only when necessary, improving efficiency while preserving accuracy. Meanwhile, MLLMs can extend this: recent fast-slow vision-language reasoning demonstrates that models can also adjust reasoning depth based on visual-textual difficulty [31]. However, these methods focus mainly on adjusting internal reasoning depth rather than deciding when visual operations or tools should be invoked. This limitation motivates the investigation of adaptive tool calling for multimodal reasoning, where MLLMs must think with visual inputs effectively.

3. Methodology

3.1. Overview

Think-execute-feedback as reasoning unit. An overview of our framework is shown in Figure 2. Given a multimodal user query (including a text prompt and an image), the policy model (an MLLM) produces rollouts interleaving natural-language reasoning with executable code (e.g., cropping, plotting). Code is executed in a separate sandbox, and the resulting visual evidence (e.g., a cropped region) is concatenated with text to refine reasoning in the next turn or yield the final answer (e.g., “blue and yellow” in Figure 2).

We define a *think-execute-feedback* cycle as the minimal reasoning unit under a policy model π , where each turn comprises (i) the current query and reasoning trace, (ii) a candidate action, and (iii) the resulting observation after

Table 2. Comprehensive results across math reasoning benchmarks. [†] denotes results from their official papers.

Model	Math-Benchmark			
	MathVision	MathVista	MathVerse	WeMath
Larger MLLMs without Reasoning				
GPT-4o	36.5	63.4	35.3	44.2
Qwen2.5-VL-72B	38.1	74.8	57.6	-
Open-source Reasoning MLLMs				
R1-Onevision-7B [†]	29.9	64.1	40.0	-
R1-VL-7B [†]	24.7	63.5	40.0	-
Open-source General MLLMs				
InternVL2.5-8B	22.0	64.4	39.5	23.9
Llava-OV-7B	18.4	63.2	26.2	17.3
Qwen2.5-VL-7B	25.0	68.1	45.1	35.4
DeepEyes-7B	26.6	70.1	47.3	38.9
CodeDance-7B (Ours)	29.6	70.3	46.8	39.6

code execution. Formally, a trajectory is:

$$\tau = ((s_{t_1}, a_{t_1}, s'_{t_1}), \dots, (s_{t_{M-1}}, a_{t_{M-1}}, s'_{t_{M-1}}), (s_{t_M}, a_{\text{ans}})),$$

where t is the time step. $s_t = (x, o_t, \epsilon_t)$ contains the original query x , the accumulated reasoning trace o_t , and interpreter feedback ϵ_t . Actions a_t are drawn from a space including tool calls (code snippets) and a terminal answer; executing code yields an observation and updates the state to s'_t . By iterating $a_t \sim \pi(\cdot | s_t)$ until a final answer is produced or a maximum turn budget M is reached, each turn becomes an executable and verifiable reasoning unit. Building on this formulation, Section 3.2 details the curation of a high-quality trajectory dataset encompassing diverse atomic abilities. This dataset provides the foundation for initializing the policy model through SFT, before advancing to RL.

Policy optimization via verifiable evidence. In the RL stage, we require a policy optimization method that can compare multiple rollouts and update the model accordingly. In our case, Group Relative Policy Optimization (GRPO) [16] provides a natural baseline, as it directly normalizes rewards across sampled trajectories without relying on a separate value network. However, standard GRPO assigns a uniform advantage to all tokens within a trajectory, which limits its effectiveness for multi-turn tool reasoning requiring intermediate correction. To address this, we extend the reward design with sequence-level and turn-level components. In particular, each rollout is evaluated with a composite reward r that integrates outcome and tool-related signals (also see Figure 2, right side):

$$r(\tau) = R_{\text{acc}}(\tau) + R_{\text{format}}(\tau) + R_{\text{BAT}}(\tau),$$

where R_{acc} denotes final-answer correctness and R_{format} enforces format compliance, respectively. In our design, we introduce a two-level reward R_{BAT} (**Reward for Balanced Adaptive Tool Calling**) that advocates adaptive tool calling based on the task difficulty. In particular, it decomposes

into a sequence-level R_{seq} and a turn-level R_{turn} , balancing task difficulty with step-wise tool calling correctness. Subsequently, the advantage A is written as:

$$A(\tau) = A_{\text{seq}}(R_{\text{acc}}, R_{\text{format}}, R_{\text{seq}}) + A_{\text{turn}}(R_{\text{turn}}).$$

Next, we show that this formulation combines global, step-level trajectory outcomes with local, turn-level execution feedback, producing more adaptive tool invocation.

Sequence-level adaptive reward for code-invocation.

Simply rewarding every successful tool call can lead to degenerate behaviors such as tool spamming or reward hacking on trivial problems (see Appendix Figure 7), which may hinder the reasoning performance (ablation studies in Table 3). To address this, we design an adaptive reward that conditions tool incentives on the group-level accuracy μ_{acc} : when most rollouts already solve the task correctly (indicating that the problem is relatively easy or solvable without additional tool assistance), further invocations are discouraged. In contrast, low μ_{acc} encourages additional exploration. Formally, the sequence-level reward is as follows:

$$R_{\text{seq}} = \left(0.5 + 0.5 \cdot \mathbb{I}_{R_{\text{acc}}(\tau) > 0}\right) \cdot d \cdot \frac{N_{\text{succ}}(\tau)}{N_{\text{total}}(\tau)}, \quad (1)$$

$$d = \sigma(\gamma(0.5 - \mu_{\text{acc}})) - \delta, \sigma(z) = \frac{1}{1 + e^{-z}},$$

where $N_{\text{succ}}(\tau)$ and $N_{\text{total}}(\tau)$ denote the numbers of successful and total tool calls in trajectory τ , and d is a group-accuracy-dependent scaling factor that suppresses rewards for unnecessary tool calls on easy queries (high μ_{acc}) and amplifies rewards for tool-assisted solutions on hard queries (low μ_{acc}), thereby enabling adaptive tool invocation. Here, γ and δ are hyper-parameters controlling the strength: higher μ_{acc} reduces d (discouraging redundant calls), while lower μ_{acc} increases d (promoting exploration). We set $\gamma = 4$ and $\delta = 0.2$ in our main experiments. See our Appendix 3.2 for discussion of these hyper-parameters.

Turn-level execution reward. To penalize failed executions and provide dense correction signals, we introduce a turn-level reward. For each turn m , an immediate penalty $r_{\text{turn},m} = -0.5$ is assigned if the code execution fails, and 0 otherwise. To capture long-term effects, we recursively redefine $r_{\text{turn},m}$ as the accumulated discounted return:

$$R_{\text{turn}}^m = r_{\text{turn}}^m + \beta \cdot R_{\text{turn}}^{m+1}, \quad A_{\text{turn}}^m = \frac{R_{\text{turn}}^m - \mu_{\text{batch}}}{\sigma_{\text{batch}}}. \quad (2)$$

Here, R_{turn}^m denotes the discounted cumulative turn-level return, β is a discount factor, and $\mu_{\text{batch}}, \sigma_{\text{batch}}$ denote the batch-wise mean and standard deviation of R_{turn} , respectively. This turn-level reward discourages credit assignment to incorrect intermediate reasoning steps, even when the final answer is correct, and it helps mitigate entropy collapse during training (Figure 5). The final advantage is obtained by combining the resulting A_{turn} with the sequence-level advantage A_{seq} (from outcome-level rewards, see Appendix).

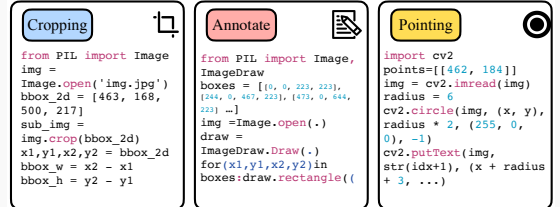
Together, the group-adaptive R_{seq} evaluates the quality of an *entire* trajectory, while R_{turn} assesses the correctness of *individual* tool calls. This complementary design, which we term $R_{\text{BAT}} = R_{\text{seq}} + R_{\text{turn}}$, mitigates reward hacking, balances efficiency with necessary exploration, and yields more robust multimodal reasoning policies. We discuss more design choices and results in Section 4.4.

3.2. Data Engineering

In our design, the lack of high-quality, multi-turn, multimodal reasoning datasets that advocate adaptive tool calling with interleaved code trajectories remains a major barrier for training. To address this, we construct a 34k dataset composed of questions with *varying difficulty levels* for cold start, featuring (1) the reasoning trajectories that include code execution results appended as **visual evidence** in training, and (2) **adaptive tool invocation** based on the level of questions. For easy questions, the model is encouraged to answer directly without any reasoning process. For hard questions, we enable code-based tool integration for executable reasoning. Practically, to mitigate the issue of tool spamming, we ensure that the pipeline follows a two-step design: (i) **weak-to-strong filtering**, where public resources (e.g., SA1B, GEOqa-plus, MMK12) are automatically filtered and stratified by difficulty using Qwen2.5-VL-7B; and (ii) **multi-turn atomic supervision**, where hard cases are decomposed into trajectories covering three categories: fundamental image transforms (crop, resize), mathematical computation (measurement, algebra, aggregation), and open-ended visual editing (drawing, annotation, etc.). Each trajectory is further verified by a stronger MLLM (e.g., Qwen2.5-VL-72B) to cross-validate the correctness. The query, code, and response are embedded as follows:

Multimodal CoT trajectories for cold-start

Query: <IMAGE> Is the flag blue and yellow or red and yellow?
Response: The image shows..., Let's call
 execute_python_code: \n
 <code>from PIL import Image \n img =
 Image.open('img.jpg') ...</code>.
 Appending compiling results... \n
 <answer>blue and yellow</answer>
Example atomic abilities (<code></code>):



These trajectories provide verifiable supervision of the atomic skills used for SFT as cold-start. In Appendix 2.1, we provide the comprehensive recipe on trajectory synthe-

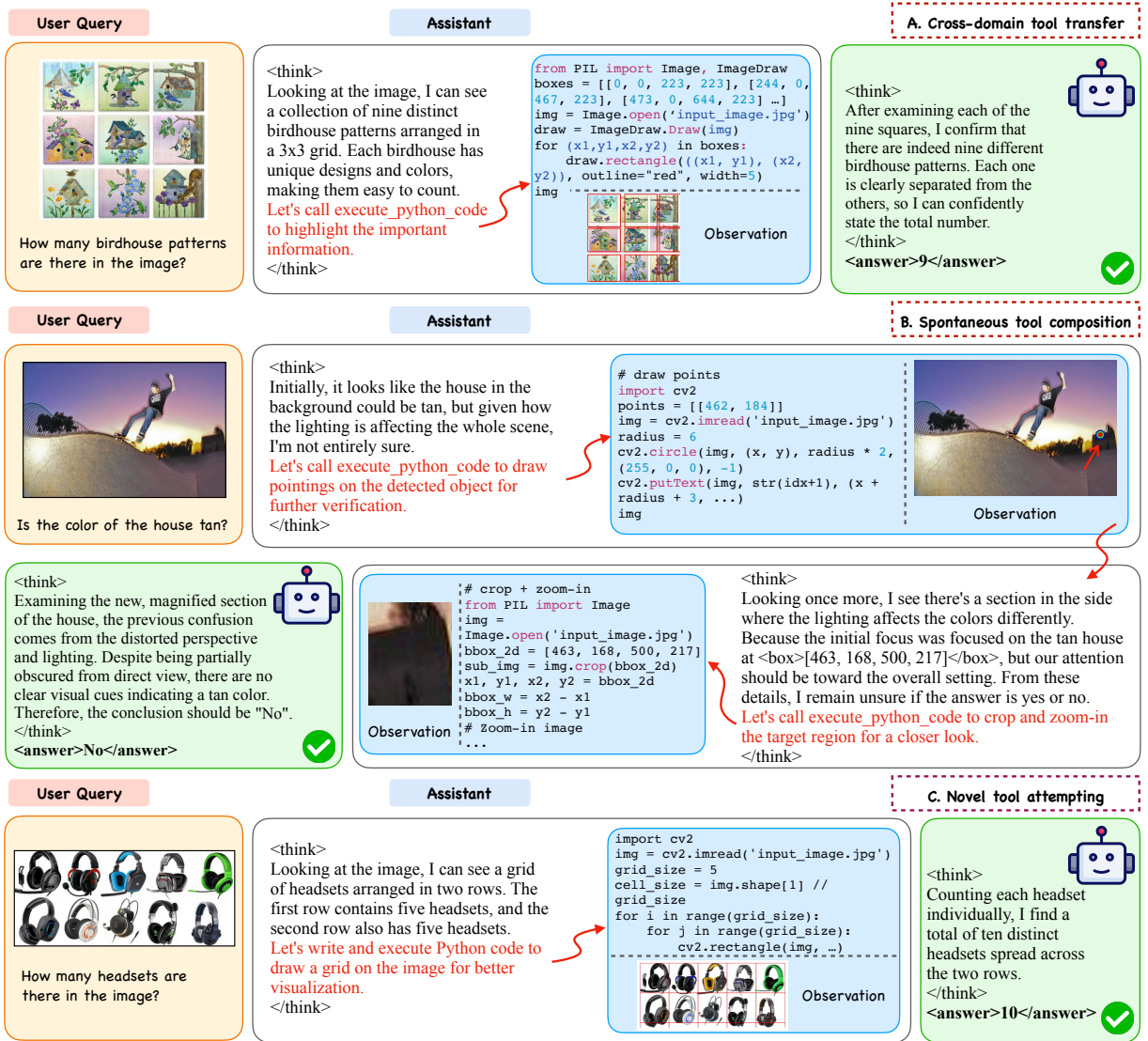


Figure 3. **Intriguing and novel reasoning trajectories emerge during RL.** These behaviors are absent from the SFT data and arise from pretrained knowledge further shaped by RL, reflecting our design of adaptive tool invocation. These emergent patterns motivate our scaling study in Figure 4, where we further examine whether these capabilities strengthen with larger data, longer training, and bigger models.

sis, filtering and code integration for SFT data, with additional examples for RL training.

4. Experiments

Implementation Details. Our method can be seamlessly integrated into popular MLLM architectures. In our experiments, we mainly build on Qwen2.5-VL-7B [1] as the base model, and compare against both open-source reasoning MLLMs (e.g., DeepEyes [43]) and advanced closed models (e.g., GPT-4o) across four benchmark categories: math reasoning (MathVista [10], MathVision [24], MathVerse [37], WeMath [14], general reasoning (ChartQA [11]), counting (PixmoCount [5], CountBench [13]), and visual search (V* Bench [29], HRBench [25]). For training, we adopt SWIFT [42] for SFT and VerL [18] for RL. We empiri-

cally set $\gamma = 4$, $\delta = 0.2$, and $\beta = 0.2$. To ensure a fair comparison, we adopt VLMEvalKit [6] as the evaluation framework. The max-turn is set to 6 in training and 10 in evaluation. We include additional details in Appendix.

4.1. Visual Reasoning Tasks

As Table 1, CodeDance attains strong results across diverse visual reasoning benchmarks, including counting, visual search, and chart understanding. Notably, it achieves state-of-the-art performance on Counting and ChartQA, outperforming the baseline by a large margin and surpassing even larger models. These improvements highlight the advantage of executable code as a reasoning medium: by delegating fine-grained visual analysis to code-based tools, it extends beyond the raw perceptual capacity of the base

Table 3. Ablation study on reward design. We report accuracy, average turns, and execution success rate on different reasoning benchmarks.

Components	CountBench	Pixmo	V* Bench	HR 4K	HR 8K	MVision	MVerse	Avg.	Succ.
	Acc. / Turns	Acc. / Turns	Acc. / Turns	Acc. / Turns	Acc. / Turns	Acc. / Turns	Acc. / Turns	Acc. / Turns	
SFT Cold-Start (w/o RL)	85.3 1.27	66.9 1.39	82.7 2.01	72.1 1.17	67.1 1.09	23.0 2.84	41.4 2.19	62.6 1.71	0.96
RL with $R_{\text{acc}} + R_{\text{format}}$	<u>89.0</u> 1.22	72.9 1.54	<u>84.8</u> 1.00	74.5 1.03	<u>71.1</u> 1.04	<u>28.6</u> 1.97	<u>46.5</u> 2.17	66.8 <u>1.42</u>	<u>0.97</u>
+ R_{DeepEyes} [43]	83.1 2.35	70.6 2.25	86.4 2.01	75.8 2.07	70.3 2.12	26.6 2.65	46.3 2.37	65.6 2.26	<u>0.97</u>
+ R_{BAT} -w/o Turn-level Reward	87.4 1.01	78.8 1.00	86.4 1.12	75.1 1.07	70.1 1.08	<u>28.6</u> 1.97	46.3 2.41	<u>67.5</u> 1.38	0.91
+ R_{BAT} (CodeDance)	91.2 1.00	<u>77.1</u> 1.00	<u>84.8</u> 1.00	<u>75.2</u> 1.06	72.3 1.06	29.6 2.19	46.8 2.35	68.1 1.38	0.99

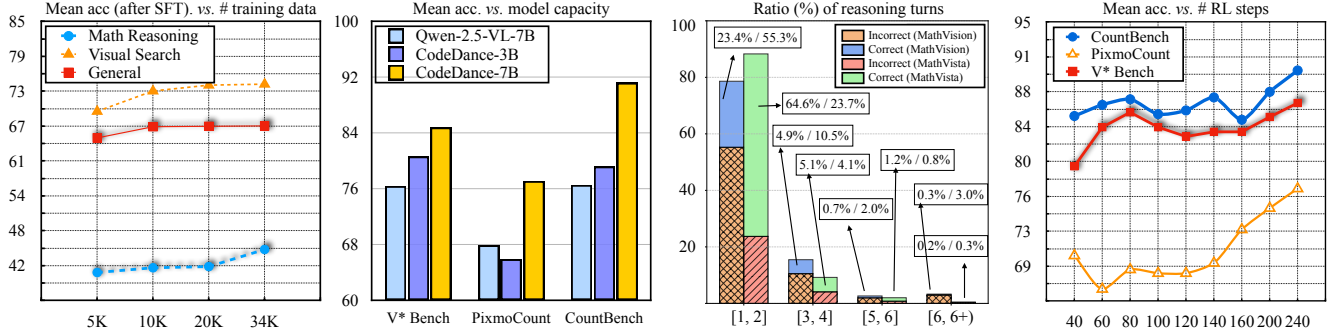


Figure 4. Scaling up compute budget on four dimensions: dataset size for SFT, model capacity, max-turns during inference and RL steps.

model, yielding gains that cannot be achieved through scaling alone, particularly on perception-heavy tasks.

4.2. Math Reasoning Tasks

In mathematical reasoning, CodeDance shows consistent gains over open-source baselines (see Table 2). For instance, it improves accuracy on MathVision from 25.0 to 29.6 (+18.4%) and on WeMath from 35.4 to 39.6 (+11.9%), while maintaining competitive results on other benchmarks. These tasks require precise symbolic manipulation and stepwise calculations, which are naturally supported by executable code. By externalizing intermediate steps into verifiable scripts, CodeDance demonstrate strong accuracy and reliability than relying solely on internal approximation.

4.3. Key Findings: Emergent Behaviors during RL

Throughout the RL process, we observe novel and surprising empirical findings (shown in Figure 3) that go beyond the atomic supervision provided during SFT. These findings point toward the scalability of code as a general reasoning medium, and we empirically study the potential in Figure 4.

Cross-domain tool transfer. We observe an emergent generalization ability in our CodeDance, where visual operations defined for a specific task can be repurposed in other contexts. For example, the bounding-box operation was initially designed to highlight particular results within chart tasks in our SFT data. However, the model demonstrates the ability to adapt this operation for counting tasks during RL training: e.g. In Figure 3A, the MLLM assistant first

localizes all candidate objects by drawing bounding boxes, then validates the correctness of each localization, and subsequently derives the final count. Additional tool transfer trajectories can be found in our Appendix. Such behavior indicates that task-specific visual operations are not rigidly bound to their original purpose, but can be flexibly generalized to support broader multimodal reasoning scenarios. This suggests that visual operations such as bounding boxes can function as general *reasoning primitives*, serving as transferable building blocks across heterogeneous tasks.

Novel tool composition of learnt capabilities. Although during SFT data curation and collection, each task was restricted to a single predefined tool or coding operation, we observe that after post-training the model develops the ability to compose multiple atomic operations to address more complex tasks beyond the training coverage: In Figure 3B, to validate the color of the house, the MLLM assistant first applies a pointing operation to check the house position correctness, then use crop with zoom-in to focus on fine-grained details. Similarly, we show example in Appendix Figure 10 that the bounding-box drawing and crops are combined to focus and solve the chart reasoning task. These observations highlight the emergence of novel tool compositions, where elementary visual operations are flexibly combined to form higher-level reasoning strategies.

Incentivizing emergence of novel unseen capabilities. Interestingly, we also find that the model exhibits a certain potential to generate tool codes not explicitly defined in the SFT data. These codes appear to be drawn from the model’s

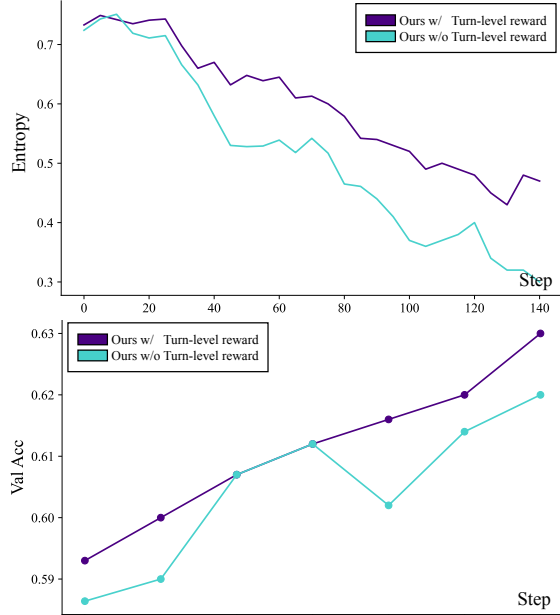


Figure 5. Entropy and validation accuracy of model’s generation.

pretraining knowledge and are occasionally activated during the post-training stage. For example, when asked to count the number of headsets in an image (Figure 3C), the MLLM does not directly respond with a number, but instead attempts to write Python code with OpenCV functions (e.g., using `cv2.rectangle` to overlay a grid for better visualization). This observation suggests that, beyond reproducing SFT-defined behaviors, the model attempts to reuse and adapt pretrained capabilities (e.g. complex OpenCV operations) to support reasoning tasks, indicating a certain potential for more flexible tool usage.

4.4. Ablation Studies

Validation of reward design. We compare three reward designs for guiding tool usage (Table 3): (i) GRPO-reward (*Outcome-level reward*) focuses only on final-answer correctness. While it shortens interaction turns, it discourages tool usage and underperforms on complex tasks. (ii) *DeepEyes reward* grants positive signals for every successful tool execution upon accurate answer. Although this encourages exploration, it also leads to tool overuse on trivial problems, increasing turns without consistent accuracy gains. A qualitative example is provided in Appendix Figure 7. (iii) *Our reward R_{BAT}* for adaptive tool-call balances the two extremes by penalizing redundant calls and rewarding selective, high-impact interactions. As Table 3, outcome-level GRPO under-utilizes tools, and the *DeepEyes reward* inflates turns without reliable accuracy improvement. Furthermore, removing the turn-level reward severely degrades the execution success rate. Overall, the full R_{BAT} achieves the best overall accuracy while avoiding unnecessary tool use, consistently surpassing both baselines.

Validation on scaling up the compute budget. Motivated by the emergent behaviors observed in Figure 3, we further examine whether CodeDance continues to benefit from scaling along four axes, as summarized in Figure 4. (i) **Training dataset size** for SFT, including Math Reasoning (MathVision, MathVerse, MathVista), Visual Search (V*, HR 4K, HR 8K), and General (ChartQA, MMVet, MMStar). We consistently observe that enlarging the SFT dataset from 5K to 34K yields steady accuracy gains, showing that both tool selection and symbolic planning benefit from broader coverage. (ii) **Increasing model capacity** from 3B to 7B substantially boosts reasoning benchmarks such as counting and search, where CodeDance-3B even outperforms a stronger 7B model. (iii) **Extending RL training** up to 240 steps further improves accuracy without overfitting, supported by our reward design R_{BAT} . (iv) **Increase the inference turn budget.** Although we set a maximum of 6 turns of the RL training, we observe that allowing more turns at inference (10) continues to improve reasoning performance. In experiments, the model achieves additional gains even beyond 6 turns (0.3%), suggesting that the learned policy can generalize to longer reasoning horizons.

Empirical study on inference entropy and accuracy.

In Figure 5, we evaluate the impact of incorporating turn-level reward (R_{turn}) on training dynamics and generalization. We study the impact of turn-level reward in R_{BAT} on several benchmarks, mainly including visual search (V* Bench), math reasoning (MathVista, MathVerse, MathVision), and counting benchmarks (PixmoCount and CountBench). (a) *Entropy*: Without R_{turn} , policy entropy collapses quickly because flawed intermediate steps may still lead to correct final answers, reinforcing shortcuts and limiting exploration [35]. With R_{turn} , intermediate penalties delay collapse, sustaining exploration. (b) *Validation Accuracy*: The additional corrective signals prevent premature convergence and translate into consistently higher accuracy, showing that local feedback improves global generalization.

5. Discussion

We present CodeDance, a task-aware, difficulty-adaptive MLLM that uses executable code for tool-integrated visual reasoning. By allowing models to define and execute code, CodeDance enables flexible reasoning with interpretable intermediate evidence, while promoting adaptive tool calls and discouraging tool spamming. During RL, CodeDance exhibits emergent behaviors beyond supervised skills, including novel tool routines, compositional strategies, and cross-domain transfer. We further verify its scalability, observing clear and consistent gains with increased compute budget. Even at the 7B scale, CodeDance achieves competitive performance across diverse benchmarks, highlighting code-based, difficulty-adaptive reasoning as a scalable and transferable paradigm for multimodal and agentic AI.

Acknowledgements

This work was supported by the grants from the Natural Science Foundation of China (62202029), and Young Elite Scientists Sponsorship Program by CAST (No. 2023QNRC001).

References

- [1] Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. Qwen-vl: A versatile vision-language model for understanding, localization, text reading, and beyond. *arXiv preprint arXiv:2308.12966*, 2023. 6
- [2] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020. 2
- [3] Xinyan Chen, Renrui Zhang, Dongzhi Jiang, Aojun Zhou, Shilin Yan, Weifeng Lin, and Hongsheng Li. Mint-cot: Enabling interleaved visual tokens in mathematical chain-of-thought reasoning. *arXiv preprint arXiv:2506.05331*, 2025. 4
- [4] Jiwan Chung, Junhyeok Kim, Siyeol Kim, Jaeyoung Lee, Min Soo Kim, and Youngjae Yu. Don’t look only once: Towards multimodal interactive reasoning with selective visual revisitation. *arXiv preprint arXiv:2505.18842*, 2025. 2
- [5] Matt Deitke, Christopher Clark, Sangho Lee, Rohun Tripathi, Yue Yang, Jae Sung Park, Mohammadreza Salehi, Niklas Muennighoff, Kyle Lo, Luca Soldaini, et al. Molmo and pixmo: Open weights and open data for state-of-the-art vision-language models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 91–104, 2025. 6
- [6] 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. 6
- [7] 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. 2
- [8] Jongwoo Ko, Sungnyun Kim, Sungwoo Cho, and Se-Young Yun. Flex-judge: Think once, judge anywhere. *arXiv preprint arXiv:2505.18601*, 2025. 2
- [9] Qianlong Li, Chen Huang, Shuai Li, Yuanxin Xiang, Deng Xiong, and Wenqiang Lei. Graphotter: Evolving llm-based graph reasoning for complex table question answering. *arXiv preprint arXiv:2412.01230*, 2024. 4
- [10] 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. 6
- [11] 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. 6
- [12] OpenAI. Openai o3 and o4-mini system card. <https://cdn.openai.com/pdf/2221c875-02dc-4789-800b-e7758f3722c1/o3-and-o4-mini-system-card.pdf>, 2025. System Card. 2, 4
- [13] Roni Paiss, Ariel Ephrat, Omer Tov, Shiran Zada, Inbar Mosseri, Michal Irani, and Tali Dekel. Teaching CLIP to Count to Ten. *arXiv preprint arXiv:2302.12066*, 2023. 6
- [14] 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. 6
- [15] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36:68539–68551, 2023. 4
- [16] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024. 4
- [17] Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face. *Advances in Neural Information Processing Systems*, 36:38154–38180, 2023. 4
- [18] Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv: 2409.19256*, 2024. 6
- [19] Vaishnavi Shrivastava, Ahmed Awadallah, Vidhisha Balachandran, Shivam Garg, Harkirat Behl, and Dimitris Papailiopoulos. Sample more to think less: Group filtered policy optimization for concise reasoning. *arXiv preprint arXiv:2508.09726*, 2025. 4
- [20] 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. 2, 4
- [21] Zhaochen Su, Linjie Li, Mingyang Song, Yunzhuo Hao, Zhengyuan Yang, Jun Zhang, Guanjie Chen, Jiawei Gu, Juntao Li, Xiaoye Qu, et al. Openthinking: Learning to think with images via visual tool reinforcement learning. *arXiv preprint arXiv:2505.08617*, 2025. 2
- [22] Dídac Surís, Sachit Menon, and Carl Vondrick. Vipergpt: Visual inference via python execution for reasoning. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 11888–11898, 2023. 4
- [23] Xu Wan, Wei Wang, Wenyue Xu, Wotao Yin, Jie Song, and Mingyang Sun. Adapthink: Adaptive thinking preferences for reasoning language model. *arXiv preprint arXiv:2506.18237*, 2025. 4

- [24] Ke Wang, Junting Pan, Weikang Shi, Zimu Lu, Houxing Ren, Aojun Zhou, Mingjie Zhan, and Hongsheng Li. Measuring multimodal mathematical reasoning with math-vision dataset. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024. 6
- [25] Wenbin Wang, Liang Ding, Minyan Zeng, Xiabin Zhou, Li Shen, Yong Luo, and Dacheng Tao. Divide, conquer and combine: A training-free framework for high-resolution image perception in multimodal large language models. *arXiv preprint*, 2024. 6
- [26] Ye Wang, Qianglong Chen, Zejun Li, Siyuan Wang, Shijie Guo, Zhirui Zhang, and Zhongyu Wei. Simple o3: Towards interleaved vision-language reasoning. *arXiv preprint arXiv:2508.12109*, 2025. 4
- [27] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022. 2, 4
- [28] Chenfei Wu, Shengming Yin, Weizhen Qi, Xiaodong Wang, Zecheng Tang, and Nan Duan. Visual chatgpt: Talking, drawing and editing with visual foundation models. *arXiv preprint arXiv:2303.04671*, 2023. 4
- [29] 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. 4, 6
- [30] Siye Wu, Jian Xie, Yikai Zhang, Aili Chen, Kai Zhang, Yu Su, and Yanghua Xiao. Arm: Adaptive reasoning model. *arXiv preprint arXiv:2505.20258*, 2025. 4
- [31] Wenyi Xiao and Leilei Gan. Fast-slow thinking grpo for large vision-language model reasoning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. 4
- [32] Zhengyuan Yang, Linjie Li, Jianfeng Wang, Kevin Lin, Ehsan Azarnasab, Faisal Ahmed, Zicheng Liu, Ce Liu, Michael Zeng, and Lijuan Wang. Mm-react: Prompting chatgpt for multimodal reasoning and action. *arXiv preprint arXiv:2303.11381*, 2023. 4
- [33] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022. 4
- [34] Edward Yeo, Yuxuan Tong, Morry Niu, Graham Neubig, and Xiang Yue. Demystifying long chain-of-thought reasoning in llms. *arXiv preprint arXiv:2502.03373*, 2025. 4
- [35] Qiyang Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gao-hong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025. 8
- [36] Jiajie Zhang, Nianyi Lin, Lei Hou, Ling Feng, and Juanzi Li. Adapthink: Reasoning models can learn when to think. *arXiv preprint arXiv:2505.13417*, 2025. 4
- [37] 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. 6
- [38] Xiang Zhang, Juntao Cao, and Chenyu You. Counting ability of large language models and impact of tokenization. *arXiv preprint arXiv:2410.19730*, 2024. 4
- [39] 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. 2
- [40] Xiaoyun Zhang, Jingqing Ruan, Xing Ma, Yawen Zhu, Haodong Zhao, Hao Li, Jiansong Chen, Ke Zeng, and Xunliang Cai. When to continue thinking: Adaptive thinking mode switching for efficient reasoning. *arXiv preprint arXiv:2505.15400*, 2025. 4
- [41] 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. 2
- [42] Yuze Zhao, Juntao Huang, Jinghan Hu, Xingjun Wang, Yunlin Mao, Daoze Zhang, Zeyinzi Jiang, Zhikai Wu, Baole Ai, Ang Wang, Wenmeng Zhou, and Yingda Chen. Swift: a scalable lightweight infrastructure for fine-tuning, 2024. 6
- [43] Ziwei Zheng, Michael Yang, Jack Hong, Chenxiao Zhao, Guohai Xu, Le Yang, Chao Shen, and Xing Yu. Deep-eyes: Incentivizing “thinking with images” via reinforcement learning. *arXiv preprint arXiv:2505.14362*, 2025. 2, 4, 6, 7
- [44] Xin Zou, Yizhou Wang, Yibo Yan, Yuanhuiyi Lyu, Ken-ting Zheng, Sirui Huang, Junkai Chen, Peijie Jiang, Jia Liu, Chang Tang, et al. Look twice before you answer: Memory-space visual retracing for hallucination mitigation in multimodal large language models. *arXiv preprint arXiv:2410.03577*, 2024. 2