Deliberate Thought in LLMs: A Study of Reasoning Depth Across Tasks and Frameworks

Jeremiah Giordani
Princeton University
jg0037@princeton.edu

Abstract

As large language models have become increasingly integrated in real-world applications, a core challenge remains: evaluating and improving their capacity for reasoning on complex tasks. Evaluation benchmarks often focus on final-answer correctness, and sometimes fail to capture how intermediate reasoning steps contribute to LLM performance. This is particularly true in multi-stage tasks, such as coding and mathematic problem solving. In this paper, we propose a general framework for evaluating reasoning in LLMs. We focus on understanding how the structure and number of reasoning steps influences outcomes. Specifically, we study tasks across mathematics and code generation, varying the number of reasoning steps and comparing outputs across a set of widely used agentic libraries. These libraries are commonly deployed in production environments, and provide different levels of abstractions for managing step-wise reasoning. Our results indicate that reasoning steps significantly affect the quality of LLM outputs, but plateau after a certain number of steps. We also find that different agentic frameworks can amplify reasoning gains. This work contributes a methodology and evaluation toolkit for understanding and improving LLM reasoning in applied systems.

1 Introduction

As large language models (LLMs) are increasingly deployed in real-world applications, a growing emphasis has been placed on their ability to perform multi-step reasoning. Use cases such as mathematic problems, code generation, or planning actions often require more than a single pass response. Instead, they benefit from iterative design, intermediate thought, and planning.

While LLMs have shown impressive capabilities on such tasks, open questions remain about how to improve performance. For example, it's unclear if breaking down a task into intermediate steps would improve outcomes. Further, the optimal number of reasoning steps is not clear, and at what point additional reasoning steps provide diminishing returns. More broadly, it's not clear what architectures and agentic frameworks most effectively support reasoning.

In this work, we conduct a systematic empirical study of these questions. We evaluate how varying the number and structure of reasoning steps influences the performance of LLMs across two key domains: mathematical problem solving and code generation. The goal is to assess how reasoning strategies impact outcomes by comparing different prompting styles, ranging from one-shot completion to step-by-step plans, reflections, and revisions.

We also compare the behavior of several widely used agentic frameworks, including LangChain, Pydantic, CrewAI, and AutoGen. These frameworks are commonly used to manage multi-step interactions with LLMs in production. While these libraries aim to support reasoning and complex task completion, they differ in how they manage state, memory, and control flow. We evaluate how these differences affect agent performance across reasoning tasks.

Through this analysis, we aim to understand not only whether reasoning improves performance, but also how the structure and implementation of reasoning impacts the quality of LLM outputs. Our findings offer practical insights for developers building agentic architectures optimized for reasoning.

2 Related Work

2.1 Reasoning in Language Models

Recent advances in large language models (LLMs) have substantially improved their ability to perform complex reasoning tasks. Prior work has shown that models like OpenAI's GPT-4 and other

similar models have developed useful, but sometimes limited abilities to reason. More recent models designed for reasoning, such as GPT-o1 and o3 have significantly outperformed early models on tasks requiring multi-step problem solving (Li et al., 2024). For example, GPT-o1 achieved up to 93% accuracy on the 2024 AIME competition problems, an exam that's used to evaluate advanced human mathematical aptitude. This vastly outperforms the 12% score from GPT-4 (OpenAI, 2024). These gains are largely attributed to new training strategies, including reinforcement learning, "Mixture-of-Experts" architectures, and long-chain-of-thought optimizations. Such techniques enable models to emulate "System 2" reasoning, including planning, self-checking, and correction, during generation (de Winter et al., 2024). This progress demonstrates that LLMs can internalize reasoning behaviors previously implemented via manual multi-step or tool-augmented settings.

2.2 Agentic Frameworks for Structured Reasoning

While powerful LLMs can reason independently to a degree, agentic frameworks aim to structure reasoning by coordinating model interactions and tool use. The most common libraries include Pydantic-AI, LangChain, CrewAI, and AutoGen, each offering different approaches to this orchestration.

- Pydantic-AI enforces schema-constrained reasoning. This allows developers to tightly control the format and content of LLM outputs using schema validation (Pydantic AI, 2024). This ensures correctness and predictability in domains where structured output is critical.

- LangChain provides a flexible infrastructure for chaining LLMs with external tools and memory (LangChain AI, 2024). It's modular design makes it an extremely popular choice for building custom agents, and enforcing process-based generation.

- CrewAI provides multi-agent collaboration through the emulation of role-based delegation (CrewAI Inc., 2024). Each "crew member" performs a specialized task, creating modular and clear reasoning chains. While its architecture is ideal for structured, complex workflows, it can struggle with open-ended and ambiguous tasks (Winland et al., 2024).

- AutoGen mediates inter-agent conversations and interactive refinements (Microsoft, 2023). It's designed to allow agents to critique and build on each other's responses to form a better solution. This is especially effective in domains like mathematics and coding, when intermediate errors can be self-corrected (Wu, 2023).

Across these libraries, a common goal is to externalize and structure the reasoning process, often yielding some performance gains. For example, frameworks like MathChat (built on AutoGen), improved math accuracy by 6-15% compared to naive prompting (Wu, 2023). Similarly, OctoTools, a recently developed meta-framework, showed 9.3% accuracy improvements across diverse reasoning tasks by combining planning with tool usage (Lu et al., 2025).

2.3 Benchmarks for Evaluating Reasoning

To measure progress in LLM reasoning, evaluation tends to focus on two key domains - mathematical problem solving and code generation. Within these domains, there are a few common benchmarks:

- Mathematics (AIME Jia (2024)): The American Invitational Mathematics Examination (AIME) serves as a rigorous benchmark for logical reasoning. Studies have tested both the base models, as well as agentic systems on AIME tasks. Agentic frameworks, such as MathChat (Wu, 2023) and PAL (Greyling, 2023) have demonstrated incremental improvements on mathematic problem solving more generally, but the most dramatic improvements have come from newer, reasoning optimized models, such as GPT-o1 and DeepSeek-V3 (OpenAI, 2024), (DeepSeek-AI et al., 2025). These models have achieved up to 80-93% accuracy, nearly matching human experts.

- Programming (HumanEval Chen et al. (2021)): HumanEval is a widely used benchmark for evaluating functional correctness in code generation. It consists of natural language prompts, and extensions have introduced comprehensive unit testing to measure correctness (CodeParrot, 2024).

3 Methodology

Our experiments are designed to evaluate how multi-step reasoning and agentic orchestration affect performance across a range of domains, from logical problem solving to creative generation. We study three core tasks: mathematical reasoning, code generation, and visual composition via SVG design.

3.1 Tasks and Evaluation Domains

We evaluate agents on the following representative tasks.

Mathematical Reasoning: We use two math benchmarks to measure mathematic problem solving: AIME (Jia, 2024) and GSM8K (OpenAI, 2021). AIME consists of competition level problems that require multiple steps, while GSM8K is a grade-school arithmetic dataset. Each problem has a numeric answer and performance is measured on exact match accuracy.

Code Synthesis: To assess agents' ability to synthesize functional code, we use the HumanEval benchmark - a collection of Python programming problems. (CodeParrot, 2024). We measure accuracy based on passing a series of comprehensive unit tests.

Creative SVG Design (Space Scene Generation): To explore generative reasoning on creative tasks, we draw on the procedure outlined in Bubeck et al. (2023), tasking agents with creating a SVG image of a space scene. The prompt instructs agents to design a visually compelling depiction of outer space using only vector graphics. Importantly, this task has no single correct answer. Instead, we qualitatively assess how reasoning steps impact stylistic and structural variation in generated scenes.

3.2 Agentic Frameworks

We implement the same set of reasoning protocols using five different frameworks

- **Baseline (GPT-4o)**
- **Pydantic-AI**
- **LangChain**
- **CrewAI**
- **AutoGen**

All agents use GPT-4o via the Azure OpenAI service as the underlying language model. We use a baseline direct LLM call to compare the effect of agentic frameworks compared to a control group. Similarly, we use GPT-o3-mini, a top-tier reasoning model from OpenAI, to evaluate how the standard language model performance compares to reasoning language models.

3.3 Reasoning Paradigms

To isolate the impact of reasoning depth, we evaluate each agent under four distinct prompting strategies.

- **Structured Output:** A minimal instruction, directing the model to return a single answer in structured form, with no intermediate reasoning.
- **One-shot:** A conventional instruction that allows the model to internally reason, but without explicit decomposition in reasoning.
- **Two-step Reasoning:** The model first produces a structured plan for execution, then generates a final output based on that plan.
- **Three-step Reasoning:** The model plans, reflects on its plan, and then produces the final output. This setup encourages internal revision before committing to a solution.

4 Results

4.1 Math Reasoning Performance

Table 1 presents accuracy scores on two math reasoning benchmarks: AIME and GSM8K. It also lists accuracy across different reasoning paradigms and agentic frameworks.

Across all frameworks, the addition of a single reasoning step led to a substantial increase in performance. For example, the baseline GPT-4o improved from 0% to 11.3% on AIME and from 56.3% to 84.2% on GSM8K when moving from structured output to one-shot prompting. These results highlight the critical role of intermediate reasoning in enabling models to solve nontrivial problems.

The progress from one-shot to multi-step reasoning (2 Step and 3 Step) produces some further gains, but they are generally modest, if at all. Some agentic frameworks, particularly CrewAI and AutoGen benefit from deeper reasoning chains. On AIME, from One-shot to 3 Step reasoning, CrewAI and AutoGen increased from 12.1% to 17.5% and

Table 1: Accuracy (%) on AIME and GSM8K Benchmarks Across Reasoning Conditions

Dataset	Agent	Structured Output	One-shot	2 Step	3 Step
AIME	Baseline (GPT-4o)	0.0	11.3	12.0	11.3
	Pydantic-AI	0.0	10.4	12.1	12.8
	LangChain	0.0	9.9	12.6	11.0
	CrewAI	0.0	11.5	14.4	15.7
	AutoGen	0.0	12.1	17.5	14.2
	Reasoning (o3-mini)	67.9	-	-	-
GSM8K	Baseline (GPT-4o)	56.3	84.2	87.0	87.4
	Pydantic-AI	48.7	79.1	83.2	81.0
	LangChain	52.9	86.0	80.1	82.7
	CrewAI	51.1	77.8	84.4	85.5
	AutoGen	50.5	81.3	85.0	87.3
	Reasoning (o3-mini)	100.0	-	-	-

Table 2: Accuracy (%) on HumanEval Coding Dataset

Dataset	Agent	Structured Output	One-shot	2 Step	3 Step
HumanEval	Baseline (GPT-4o)	63.7	82.3	83.6	83.9
	Pydantic-AI	65.5	84.1	81.2	79.6
	LangChain	59.0	80.4	83.5	82.3
	CrewAI	61.3	79.3	85.6	87.9
	AutoGen	58.8	77.2	83.2	85.0
	Reasoning (o3-mini)	98.4	-	-	-

11.5% to 15.7% respectively. These agentic frameworks are designed to support internal dialogue and refinement, indicating that such frameworks may be more efficient at leveraging additional reasoning steps. In contrast, tools like Pydantic-AI and LangChain see smaller improvements, with returns often diminishing between 2 Step and 3 Step reasoning.

The GPT-o3-mini model (o3-mini), a reasoning optimized LLM trained with longer chain of thoughts, achieves dramatically higher performance in both domains, scoring 67.9% on AIME and 100.0% on GSM8K. These results highlight the continued advantage of architectural and training improvements in LLMs. It suggests that even the best prompting and agentic orchestration may not be enough to fill the gap for more general-purpose LLMs, like GPT-4o.

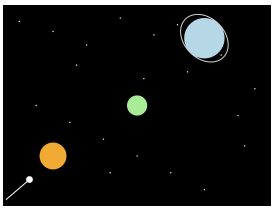
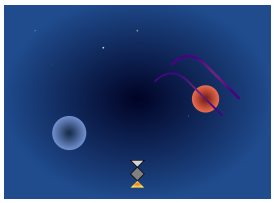
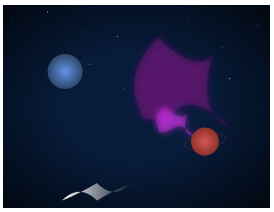
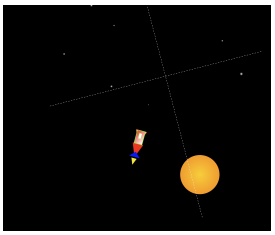
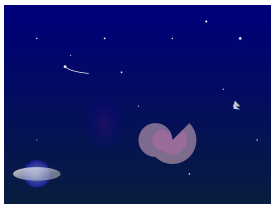
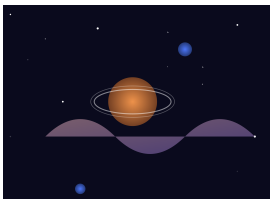
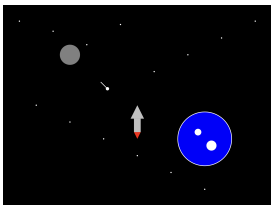
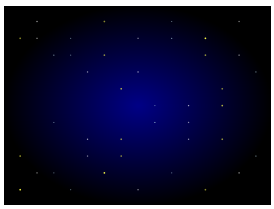
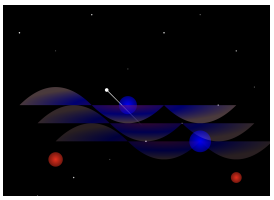
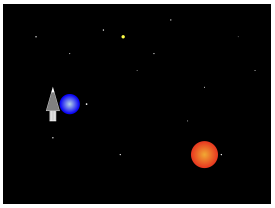
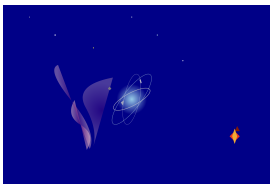
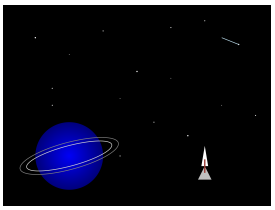
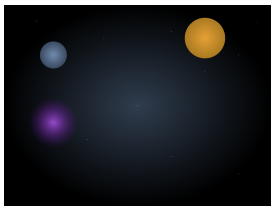
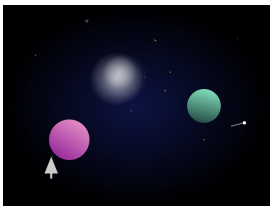
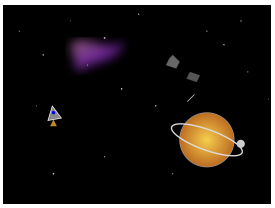
4.2 Results: Code Generation on HumanEval

Next, we evaluated each agent on the HumanEval dataset, which consists of Python programming problems, with accompanying test cases to assess correctness. As shown in Table 2, all agents benefit from reasoning aware prompting, compared to the structured output baseline. The baseline GPT-4o improves from a pass rate of 63.7% to 82.3% in one-shot prompting, and achieves less significant, but slight improvements beyond that. This overall trend confirms prior observations: reasoning enhances performance, but benefits diminish after around 2 steps.

Notably, we see the same trend as with the mathematical reasoning experiments - more explicitly conversational agentic frameworks, CrewAI and AutoGen demonstrate the strongest gains from multi-step prompting, with CrewAI reaching 87.9% pass rate in the three step setting, outperforming other agentic orchestration methods and reasoning paradigms. This again reinforces the idea that agentic dialogue and structured self reflection offer additional benefit in multi-step code synthesis tasks.

However, once again, the reasoning optimized o3-mini model dramatically surpasses all other methods, with a 98.4% pass rate, highlighting the growing gap between standard prompting and pretrained reasoning capabilities. These results further

Table 3: Rendered Space Scenes by Agent and Reasoning Strategy

Agent	One-shot	2-Step Reasoning	3-Step Reasoning
Baseline (GPT-4o)			
Pydantic-AI			
LangChain			
CrewAI			
AutoGen			
Reasoning (o3-mini)			

support the conclusion that while prompting techniques and agentic scaffolding help, model architecture and training remain the most critical factors in performance on complex reasoning tasks like code generation.

4.3 Rendered Space Scenes: Visualizing Reasoning Effects

To complement our quantitative evaluations, we include a creative task with no objectively correct output: rendering a space-themed SVG image. Drawing off of related work that introduced SVG generation as an LLM evaluation methodology, (Bubeck et al., 2023), this test serves as a qualitative probe into how different agentic architectures and reasoning strategies influence the generation process. Unlike math or coding, where correctness is binary, the space scenes allow us to observe how the nature of reasoning manifests visually. Using the same prompts, but varying the number of reasoning steps, we can assess how structured reasoning affects the creativity, complexity and abstraction of the out-

puts. The resulting scenes are visual artifacts of the model's internal processes, providing an interpretable window into reasoning dynamics. These results are shown in Table 3.

Notably, we observe a clear and interesting trend: as the number of reasoning steps increases, the outputs of the model tend to become more unconventional and surreal. In the one-shot setting, most agents produce relatively grounded and conventional depictions of space scenes, including planets, stars, and sometimes simple rockets. However, with additional reasoning steps, the scenes evolve to include more abstract shapes, layered gradients, and unfamiliar objects. This pattern does not seem to vary across agentic architecture. In general, as reasoning steps increase, the models introduce new ambiguous forms and artistic elements, suggesting that reasoning not only affects factual correctness, but also shifts the model's creative tendencies. These results highlight that multi-step reasoning can push LLMs beyond standard completions into more exploratory and stylistically varied outputs.

5 Discussion

5.1 Findings on Agentic Reasoning

Our results offer several insights into the role of multi-step reasoning and agentic frameworks in LLM performance. First, across all tasks, including structured math problems, open ended code generation and creative visual design, we observe that reasoning makes a substantial impact on output. For quantitative tasks, including mathematical problem solving and code generation, we observe that it tends to improve accuracy. This confirms that giving LLMs space to simulate deliberation enables stronger outcomes. For creative tasks, we see that additional reasoning tends to increase abstract generations and non-standard outputs.

Second, we find that agentic frameworks that explicitly optimize for multi-step internal deliberation, such as AutoGen and CrewAI, benefit the most from additional reasoning steps. These agents not only achieve higher quantitative performance on reasoning intensive benchmarks like AIME, but they also exhibit more consistent gains when increasing reasoning steps, compared to less complex frameworks. Even though these frameworks do show improvements with more reasoning steps, the performance increases tend to be marginal. Independent of agentic framework, we find a trend of diminishing returns beyond two or three reasoning steps in most cases, with agent performance often plateauing. This is more pronounced in more simple agentic frameworks (Pydantic-AI, LangChain), but remains notable in more complex ones (CrewAI, AutoGen). This suggests that longer reasoning chains may require better error correction or memory mechanisms to be consistently beneficial.

Finally, our results show that, while reasoning tends to improve the capabilities of LLMs, and agentic architectures may further benefit performance, this does not come close to closing the gap between standard language models, and reasoning optimized models, like o3-mini. This suggests that improvements on complex tasks that require planning are likely to come from reasoning models, not better frameworks or reasoning paradigms.

5.2 Future Work

Our experiments point toward several directions for future research. First, additional granularity in reasoning supervision and scaffolding may further improve agent reliability. A key component missing from this research is tool use, which is becoming more prevalent in agentic deployments. Enabling the use of math and coding tools may further benefit performance.

Second, in our study, we study fixed reasoning depth (1, 2, or 3 steps). However, current top-tier reasoning models enable the LLMs to dynamically set the depth of reasoning. Future work could evaluate not only performance on logical baselines, but also how frameworks might benefit from adjusting the number of reasoning steps, or invoking various stages only when needed.

5.3 Implications for Industry Deployments

From a deployment perspective, our findings offer practical guidance for teams building LLM-based applications in industry. First, reasoning scaffolds such as chain of thought, structured planning, and agentic orchestration consistently improve performance. This is especially true for tasks that require logic, creativity, or long-horizon generation. These scaffolds offer a low-cost way to enhance weaker base models, if reasoning optimized models are not viable.

Second, agentic frameworks differ in how well they leverage reasoning. AutoGen and CrewAI provide the most support for internal dialog and modular task planning, making them well-suited for applications where multi-step reasoning is essential. In contrast, lighter frameworks like Pydantic-AI

may be preferable in cases where strict formatting and schema adherence are more important than flexible reasoning.

Finally, developers should be mindful that reasoning depth introduces latency and cost. Our findings suggest that two or three reasoning steps are often sufficient to reap most of the benefits, and it's likely that early stopping may help manage tradeoffs. Ultimately, reasoning-aware agent design should be seen not only as a performance tool, but a design tool to create more interpretable and controllable AI systems.

6 Conclusion

In this work, we present an empirical investigation into the role of reasoning in large language model performance across math, coding, and creative generation tasks. Our procedure involved varying the number of reasoning steps and evaluating across a diverse set of agentic frameworks. We show that multi step reasoning consistently improves quantitative outcomes. Agentic systems that support internal deliberation, such as CrewAI and AutoGen demonstrate especially strong gains, suggesting that structured orchestration enhances the utility of general-purpose LLMs. However, we find that no agentic framework or level of reasoning seems sufficient to bridge the gap between base LLMs and reasoning optimized models, such as o3-mini. Our findings provide actionable insights for both researchers and developers, emphasizing the importance of reasoning aware design in building more capable, interpretable, and adaptable AI systems.

References

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, Harsha Nori, Hamid Palangi, Marco Tulio Ribeiro, and Yi Zhang. 2023. Sparks of artificial general intelligence: Early experiments with gpt-4.

Harsha Chaitanya. 2024. Comparing ai agent platforms: Crewai, autogen, langchain and pydantic ai. https://medium.com/@harshachaitanya27/. Accessed: 2025-04-22.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021. Evaluating large language models trained on code.

CodeParrot. 2024. Instructhumaneval dataset. https://huggingface.co/datasets/codeparrot/instructhumaneval. Accessed: 2025-04-22.

CrewAI Inc. 2024. crewai: A framework for multi-agent orchestration. https://github.com/crewAIInc/crewAI. Accessed: 2025-04-22.

Joost C. F. de Winter, Dimitra Dodou, and Yke Bauke Eisma. 2024. System 2 thinking in openai's o1-preview model: Near-perfect performance on a mathematics exam. *Computers*, 13(11).

DeepSeek-AI et al. 2025. Deepseek-v3 technical report.

Dani Fuyà. 2024. Best 4 ai agent frameworks in 2025: How to choose the right one for your project. https://bestaiagents.ai/blog/best-4-ai-agent-frameworks-2025. Accessed: 2025-04-22.

Cobus Greyling. 2023. Pal: Program-aided large language models. https://cobusgreyling.medium.com/pal-program-aided-large-language-models-30db3e59f796. Accessed: 2025-05-02.

Maxwell Jia. 2024. Aime 2024: A benchmark for evaluating mathematical reasoning in large language models. https://huggingface.co/datasets/Maxwell-Jia/AIME_2024. Accessed: 2025-04-22.

LangChain AI. 2024. Langgraph: A library for building multi-actor applications with memory and control flow. https://github.com/langchain-ai/langgraph. Accessed: 2025-04-22.

Leo Li, Ye Luo, and Tingyou Pan. 2024. Openai-o1 ab testing: Does the o1 model really do good reasoning in math problem solving?

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. 2023. Is your code generated by chatGPT really correct? rigorous evaluation of large language models for code generation. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Pan Lu, Bowen Chen, Sheng Liu, Rahul Thapa, Joseph Boen, and James Zou. 2025. Octotools: An agentic framework with extensible tools for complex reasoning.

Microsoft. 2023. Autogen: Enabling next-gen llm applications via multi-agent conversation framework. https://github.com/microsoft/autogen. Accessed: 2025-04-22.

OpenAI. 2021. Gsm8k: A benchmark for grade school math word problems. https://huggingface.co/datasets/openai/gsm8k. Accessed: 2025-04-22.

OpenAI. 2024. Learning to reason with llms. https://openai.com/index/learning-to-reason-with-llms/. Accessed: 2025-04-22.

OpenAI et al. 2024. Gpt-4o system card.

Pydantic AI. 2024. Pydantic ai: An agentic framework built around pydantic schemas. https://github.com/pydantic/pydantic-ai. Accessed: 2025-04-22.

Shanghaoran Quan, Jiaxi Yang, Bowen Yu, Bo Zheng, Dayiheng Liu, An Yang, Xuancheng Ren, Bofei Gao, Yibo Miao, Yunlong Feng, Zekun Wang, Jian Yang, Zeyu Cui, Yang Fan, Yichang Zhang, Binyuan Hui, and Junyang Lin. 2025. Codeelo: Benchmarking competition-level code generation of llms with human-comparable elo ratings.

Vanna Winland, Meredith Syed, and Anna Gutowska. 2024. What is crewai? https://www.ibm.com/think/topics/crew-ai. Accessed: 2025-04-22.

Yiran Wu. 2023. Mathchat - a conversational framework to solve math problems. https://microsoft.github.io/autogen/0.2/blog/2023/06/28/MathChat/. Accessed: 2025-04-22.