MolViBench: Evaluating LLMs on Molecular Vibe Coding

Jiatong Li [1,2] **Yatao Bian** [1]

[1]*School of Computing, National University of Singapore* [2]*Department of Computing, Hong Kong Polytechnic University*. Correspondence to: Yatao Bian ybian@nus.edu.sg.

1. Introduction

Large Language Models (LLMs) have demonstrated remarkable potential in scientific research [1, 2], especially serving as intelligent assistants that provide insights and recommendations for scientists [3, 4]. Although Chemical agents [5, 6] show promise, they normally adopt fixed function calls with poor extension to flexible operations and emerging requirements. Recently, a new interaction paradigm, which we refer to as *Molecular Vibe Coding*, has emerged, where researchers articulate their intent through natural language, and LLMs create executable code to fulfill these requests, especially the molecular editing [7]. This paradigm is particularly prevalent in molecular discovery, where chemists and computational biologists could prompt LLMs to *compute molecular descriptors* [8], *filter compounds by Lipinski's Rule of Five* [9], or *build a virtual screening pipeline* [10], expecting functional programs and chemically correct results in return.

Despite the rapid adoption of this workflow in drug discovery and cheminformatics research, no existing benchmark directly evaluates it. Current evaluation resources fall into two disconnected categories. On one hand, general code generation benchmarks such as HumanEval [11], MBPP [12], and SWE-bench [13] measure programming competence through domain-agnostic function synthesis or repository-level bug fixing, but their questions require no chemistry knowledge and impose no chemical correctness constraints. On the other hand, chemistry-focused benchmarks such as S^2-Bench [3] and ChemCoTBench [14] directly evaluate knowledge recall or predictive modeling performance, but do not test whether an LLM can produce code that implements molecular computations. This leaves a critical blind spot that the intersection of programming ability, domain-specific chemical reasoning, and output-level chemical correctness remains unevaluated.

This gap matters in practice. When an LLM generates code for molecular tasks, the code may execute without errors yet produce chemically invalid results, a failure mode we term *runnable but chemically wrong*. For example, a model might correctly call the functions of RDKit [15], but silently mishandle stereochemistry, apply an incorrect reaction template, or yield a molecule that fails to meet the given constraints. Such errors are particularly dangerous in scientific workflows because they are difficult to detect without domain expertise, and they can propagate silently into downstream decisions such as compound selection or lead optimization.

Therefore, we introduce **MolViBench**, the first benchmark tailored for *Molecular Vibe Coding* with 358 real-world molecular coding questions. MolViBench is guided by three design principles:

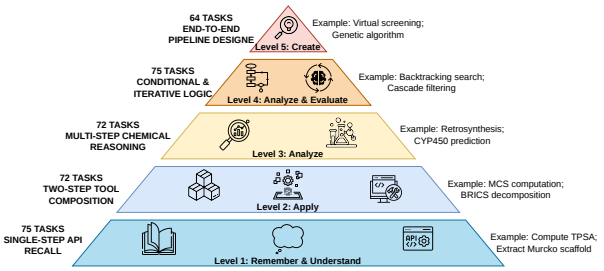

Fig. 1: Task Composition of MolViBench organized by Bloom's taxonomy [16].

Cognitive progression. Tasks are organized into five levels following Bloom's taxonomy [16], ranging from basic API recall to end-to-end pipeline design. This enables fine-grained analysis of where model capability degrades, including memory-intensive API usage, multi-step chemical reasoning, or system-level orchestration.

Deterministic evaluability. Each task is constructed so that, given valid input, the expected output is deterministically verifiable. We enforce a constrained library setting centered on RDKit to eliminate cross-toolkit inconsistencies, and provide gold-standard reference solutions for all 358 tasks under a unified function interface. Building on these solutions, we propose a multi-layered evaluation protocol: type-aware output comparison as the primary check, structural validation for tasks with non-deterministic components, and AST-based API-semantic fallback analysis that verifies whether the generated code invokes the correct domain functions even when output values diverge.

Workflow realism. Our questions span 12 categories of real-world drug discovery operations, including molecular characterization, ADMET screening, virtual screening, combinatorial chemistry, lead optimization, reaction simulation, clustering, QSAR modeling, and more. This breadth ensures that benchmark outcomes reflect practical utility rather than artificial difficulty.

We systematically evaluate 9 frontier LLMs and compare three inference paradigms: Direct Generation, Incremental Repair, and Agent Collaboration. Our experiments yield several notable findings. First, Claude Opus 4.6 with Incremental Repair achieves the strongest overall performance (Pass@1 = 39.7%). Second, performance consistently degrades with increasing Bloom's taxonomy level across all models, con-

Table 1: Task distribution across Bloom's taxonomy levels and output types. "Other" includes `set`, `Image`, and `DataFrame` returns.

Level	Tasks	int	float	bool	str	list	dict	tuple	Other
L1	75	19	12	21	8	5	6	1	3
L2	72	1	5	3	29	24	7	1	2
L3	72	0	0	6	7	34	25	0	0
L4	75	0	0	0	0	0	75	0	0
L5	64	0	0	0	0	42	22	0	0
Total	**358**	20	17	30	44	105	135	2	5

Table 2: LLMs evaluated in MolViBench. All models are accessed via their official APIs. "IR" denotes the Incremental Repair paradigm, while "AC" denotes the Agent Collaboration paradigm.

Model	Provider	Type	Paradigms
Claude Opus 4.6 [17]	Anthropic	Code-specialized	Direct, IR, AC
Claude Opus 4.6 Thinking [17]	Anthropic	Reasoning	Direct, IR, AC
Gemini 3 Pro [18]	Google	Standard	Direct, IR, AC
DeepSeek-V3.2 Reasoner [19]	DeepSeek	Reasoning	Direct, IR, AC
DeepSeek-V3.2 Chat [19]	DeepSeek	Standard	Direct, IR
GPT-5.2 Codex [20]	OpenAI	Code-specialized	Direct, IR
GPT-5.3 Codex [21]	OpenAI	Code-specialized	Direct, IR
Kimi-K2.5 [22]	Moonshot	Standard	Direct
MiniMax-M2.5 [23]	MiniMax	Standard	Direct

firming that higher-order tasks involving multi-step reasoning and workflow orchestration remain a significant challenge. Third, Incremental Repair proves to be the most robust inference paradigm, yielding consistent gains across all models and difficulty levels, whereas Agent Collaboration improves weaker models but degrades performance for stronger ones. Together, these findings highlight the importance of both model capability and inference strategy in tackling complex molecular visualization tasks.

2. Benchmark Statistics

Table 1 summarizes the distribution of tasks across Bloom's taxonomy levels and output types. Several patterns emerge from the distribution. Lower levels (L1 & L2) are dominated by simple output types (`int`, `float`, `bool`, `str`), reflecting single-value computations such as molecular weight or substructure matching. As cognitive complexity increases, structured output types (`list`, `dict`) become overwhelmingly prevalent. Level 4 tasks exclusively return `dict`, encoding multi-field results from complex conditional and iterative workflows, while Level 5 tasks return either `list` or `dict` to capture pipeline outputs such as ranked candidate lists or comprehensive screening reports.

3. Experiments

We evaluate 9 frontier LLMs spanning six providers and two architectural categories: standard code generation models and reasoning-enhanced variants. Table 2 lists all models along with the inference paradigms under which each is tested. The selection covers the current state of the art in code generation, including general-purpose models (Gemini 3 Pro [18], DeepSeek-V3.2 Chat [19]), code-specialized models (Claude Opus 4.6 [17], GPT-5.2/5.3 Codex [20, 21]), and reasoning models (Claude Opus 4.6 Thinking [17], DeepSeek-V3.2 Reasoner [19]).

Table 3: Full results on MolViBench across all inference paradigms. Models ranked by Pass@1 rate. Exec. = executable rate; EM = Exact Match; P@1 = Pass@1; FB = Fallback Pass Rate. L1–L5 report per-level Pass@1. All values in %. Best results are highlighted in red, while second-best results are highlighted in green.

Model	Exec.	EM	P@1	FB	L1	L2	L3	L4	L5
Claude Opus 4.6 Think (IR)	98.9	33.2	39.7	72.6	78.7	50.0	31.9	25.3	7.8
Claude Opus 4.6 Think	94.1	32.7	38.8	69.3	78.7	50.0	30.6	24.0	6.2
Claude Opus 4.6 (IR)	98.6	32.1	38.6	71.8	80.0	45.8	29.2	25.3	7.8
Claude Opus 4.6	94.1	31.0	37.4	68.4	78.7	45.8	27.8	24.0	6.2
Gemini 3 Pro (IR)	98.6	30.4	36.0	58.4	77.3	47.2	22.2	22.7	6.2
Gemini 3 Pro (AC)	95.5	31.0	36.0	57.0	78.7	48.6	15.3	24.0	9.4
Gemini 3 Pro	94.1	29.3	34.9	56.4	76.0	45.8	19.4	21.3	7.8
DeepSeek-V3.2 Reasoner (IR)	86.6	28.8	34.6	52.8	72.0	43.1	31.9	16.0	6.2
Claude Opus 4.6 Think (AC)	95.2	28.5	34.1	65.6	78.7	47.2	27.8	8.0	4.7
Claude Opus 4.6 (AC)	95.0	26.5	34.1	66.2	76.0	47.2	25.0	10.7	7.8
DeepSeek-V3.2 Reasoner (AC)	88.5	28.8	34.1	51.1	73.3	40.3	30.6	13.3	9.4
GPT-5.2 Codex	89.9	27.9	33.2	51.1	76.0	44.4	23.6	9.3	9.4
GPT-5.3 Codex	89.7	27.9	32.7	64.8	76.0	55.6	16.7	5.3	6.2
MiniMax M2.5	84.9	24.6	30.4	50.6	64.0	40.3	26.4	14.7	3.1
DeepSeek-V3.2 Reasoner	74.6	24.9	29.9	45.5	69.3	37.5	19.4	13.3	6.2
DeepSeek-V3.2 Chat	84.6	20.9	28.2	48.9	68.0	40.3	15.3	6.7	7.8
Kimi K2.5	64.2	22.4	27.9	36.9	74.7	43.1	6.9	6.7	4.7

3.1 Prelinminary Results

Table 3 presents the overall performance of different models under various inference paradigms on MolViBench, ranked by Pass@1.

Overall, Claude Opus 4.6 Think with Incremental Repair achieves the strongest performance, attaining a Pass@1 of 39.7% and an executable rate of 98.9%, along with the highest Fallback Pass Rate of 72.6%. The top-four ranked entries are all Claude Opus 4.6 variants, suggesting that both model capability and post-hoc repair contribute meaningfully to performance. DeepSeek-V3.2 Reasoner (IR) and GPT-5.2 Codex occupy the mid-tier, while MiniMax M2.5, DeepSeek-V3.2 Chat, and Kimi K2.5 trail behind. Notably, reasoning-enhanced models like Claude Opus 4.6 Thinking and DeepSeek-V3.2 Reasoner all perform better than their non-reasoning variants.

Meanwhile, Agent Collaboration (AC) achieves comparable or even improved Pass@1 for relatively weaker models (e.g., Gemini 3 Pro and DeepSeek-V3.2 Reasoner), but tends to degrade performance for stronger models such as Claude Opus 4.6. This suggests that the effectiveness of holistic rewriting is highly dependent on the underlying model's capability. For weaker models, AC can provide useful global restructuring and error correction, leading to performance gains. In contrast, for stronger models that already produce largely correct solutions, rewriting often introduces unnecessary modifications that disrupt correct intermediate reasoning, resulting in performance regression. In contrast to AC, Incremental Repair (IR) consistently improves performance across models without introducing regressions, indicating that localized corrections to non-executable or partially incorrect code are more reliable than full-solution regeneration.

References

[1] Jiatong Li, Yunqing Liu, Wenqi Fan, Xiao-Yong Wei, Hui Liu, Jiliang Tang, and Qing Li. Empowering molecule discovery for molecule-caption translation with large language models: A chatgpt perspective. *IEEE Transactions on Knowledge and Data Engineering*, 2024.

[2] Weida Wang, Dongchen Huang, Jiatong Li, Tengchao Yang, Ziyang Zheng, Di Zhang, Dong Han, Benteng Chen, Binzhao Luo, Zhiyu Liu, et al. Cmphysbench: A benchmark for evaluating large language models in condensed matter physics. *arXiv preprint arXiv:2508.18124*, 2025.

[3] Jiatong Li, Junxian Li, Weida Wang, Yunqing Liu, Changmeng Zheng, Dongzhan Zhou, Xiaoyong Wei, and Qing Li. Speak-to-structure: Evaluating llms in open-domain natural language-driven molecule generation. *arXiv preprint arXiv:2412.14642*, 2024.

[4] Gleb V Solovev, Ivan Gurev, Anastasia Vepreva, Ivan Dubrovsky, Alina Zhidkovskaya, Kamil Fatkhiev, Elizaveta Lutsenko, Anastasia Orlova, Nina Gubina, Nikolay O Nikitin, et al. Chemcoscientist: Llm-based multi-agent assistant for automated solving of chemical tasks using data-driven tools. In *2025 IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 2569–2572. IEEE, 2025.

[5] Andres M. Bran, Sam Cox, Oliver Schilter, Carlo Baldassari, Andrew D White, and Philippe Schwaller. Augmenting large language models with chemistry tools. *Nature machine intelligence*, 6(5):525–535, 2024.

[6] Xiangru Tang, Tianyu Hu, Muyang Ye, Yanjun Shao, Xunjian Yin, Siru Ouyang, Wangchunshu Zhou, Pan Lu, Zhuosheng Zhang, Yilun Zhao, et al. Chemagent: Self-updating memories in large language models improves chemical reasoning. In *The Thirteenth International Conference on Learning Representations*, 2025.

[7] Wenyu Zhu, Chengzhu Li, Xiaohe Tian, Yifan Wang, Yinjun Jia, Jianhui Wang, Bowen Gao, Ya-Qin Zhang, Wei-Ying Ma, and Yanyan Lan. Coder as editor: Code-driven interpretable molecular optimization. *arXiv preprint arXiv:2510.14455*, 2025.

[8] Milan Randić. Generalized molecular descriptors. *Journal of Mathematical Chemistry*, 7(1):155–168, 1991.

[9] Violeta Ivanović, Miroslav Rančić, Biljana Arsić, and Aleksandra Pavlović. Lipinski's rule of five, famous extensions and famous exceptions. *Popular Scientific Article*, 3(1):171–177, 2020.

[10] Fatima Noor, Muhammad Junaid, Atiah H Almalki, Mohammed Almaghrabi, Shakira Ghazanfar, and Muhammad Tahir ul Qamar. Deep learning pipeline for accelerating virtual screening in drug discovery. *Scientific Reports*, 14(1):28321, 2024.

[11] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[12] Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

[13] Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R Narasimhan. Swe-bench: Can language models resolve real-world github issues? In *The Twelfth International Conference on Learning Representations*, 2024.

[14] Li Hao, CAO He, Bin Feng, Daniel Shao, Xiangru Tang, Zhiyuan Yan, Yonghong Tian, Li Yuan, and Yu Li. Beyond chemical qa: Evaluating llm's chemical reasoning with modular chemical operations. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2025.

[15] Greg Landrum et al. Rdkit documentation. *Release*, 1(1-79):4, 2013.

[16] David R Krathwohl. A revision of bloom's taxonomy: An overview. *Theory into practice*, 41(4):212–218, 2002.

[17] Anthropic. Introducing claude opus 4.6. `https://www.anthropic.com/news/claude-opus-4-6`, 2026.

[18] Google DeepMind. A new era of intelligence with gemini 3. `https://blog.google/products-and-platforms/products/gemini/gemini-3/#gemini-3`, 2025.

[19] Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, et al. Deepseek-v3. 2: Pushing the frontier of open large language models. *arXiv preprint arXiv:2512.02556*, 2025.

[20] OpenAI. Introducing gpt-5.2-codex. `https://openai.com/index/introducing-gpt-5-2-codex`, 2025.

[21] OpenAI. Introducing gpt-5.3-codex. `https://openai.com/index/introducing-gpt-5-3-codex`, 2026.

[22] Kimi Team, Tongtong Bai, Yifan Bai, Yiping Bao, SH Cai, Yuan Cao, Y Charles, HS Che, Cheng Chen, Guanduo Chen, et al. Kimi k2. 5: Visual agentic intelligence. *arXiv preprint arXiv:2602.02276*, 2026.

[23] MiniMax. Minimax m2.5: Built for real-world productivity. `https://www.minimax.io/news/minimax-m25`, 2026.