CAD-bench: Benchmarking Language Models on Functional CAD Generation

Anonymous Author(s)
Affiliation
Address
email

Abstract

Language-model agents are increasingly able to operate computer-aided design (CAD) toolchains, but current evaluations for their capabilities in this domain often measure only whether the models build, their rendered appearance, or crude geometric similarity. These proxy and approximate criteria tend to be much easier for the agents to meet, but fail to measure the properties that matter in mechanical design–mating interfaces, exact dimensions, and functional behavior–while overweighting less significant implementation details, such as wall thickness for an outer covering around a gearbox. We introduce CAD-bench, an execution-based benchmark for evaluating language-model CAD agents. CAD-bench contains 17 tasks across four difficulty tiers, ranging from basic solids to threaded mating pairs and functional gear trains. Each submission is executed, exported as geometry, and evaluated with task-specific checks, including dimensional and pose verification, thread-profile analyses, and Blender-based rigid-body simulation for complex machines. The benchmark supports both one-shot CAD-code generation and agent harnesses that produce final STEP artifacts in a Linux environment. Initial results show that CAD-bench is not saturated by current systems. The best standalone model reaches 59.9% overall, and functional tasks remain near zero for most standalone runs. Agent harnesses perform better than one-shot generation, but still fail frequently on the hardest tasks, reaching a maximum score of 68%. No harness, whether one-shot or agent, can achieve a score of 50% on the insane-level tasks. CAD-bench therefore demonstrates the current limited capability of models to complete end-to-end CAD tasks, as well as the gap between measuring their capabilities with functional evaluation and simpler methods.

1 Introduction

Computer-aided design (CAD) is a central representation for physical product development. CAD artifacts encode not only visible shape, but also dimensions, constraints, features, interfaces, and assemblies that downstream engineering tools use for simulation, inspection, and manufacturing. As language-model agents become more capable of writing code, operating tools, and iterating on execution feedback, CAD generation is becoming a natural potential target for agentic automation.

Evaluation, however, remains a bottleneck. A generated CAD artifact can compile, render, and resemble a target object while still failing the engineering task. Common errors include shifted poses, incorrect hole placement, missing mating features, invalid thread profiles, gear collisions, intersecting bodies, and in general, assemblies that fail to transmit motion. These errors impact whether a generated artifact is usable at all.

Prior CAD-generation benchmarks have established important foundations. SketchGraphs represents CAD sketches as geometric constraint graphs [Seff et al., 2020]. CAD-as-Language and

Submitted to 40th Conference on Neural Information Processing Systems (NeurIPS 2026). Do not distribute.

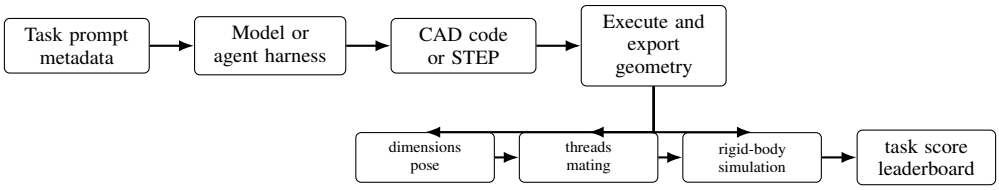

Figure 1: CAD-bench evaluation pipeline. A harness receives a task specification and produces CAD code or a STEP artifact. The runtime executes the submission, exports geometry, and applies task-specific checks for dimensions, pose, thread observables, mating behavior, and rigid-body function before aggregation.

DeepCAD model CAD sketches or construction histories as serialized design programs [Ganin et al., 2021, Wu et al., 2021]. Fusion 360 Gallery provides human-authored design sequences and a programmatic reconstruction environment [Willis et al., 2021]. More recent systems move toward natural-language CAD generation, executable CAD code, or geometric validation. Text2CAD studies text-to-parametric-CAD generation from natural-language prompts [Khan et al., 2024]. CADPrompt pairs natural-language CAD prompts with expert CAD code and uses a visual verification loop [Alrashedy et al., 2024]. CAD-Coder and Text-to-CadQuery target text-to-CAD-code generation with executable feedback or geometric rewards [Guan et al., 2025, Xie and Ju, 2025]. CadEval, CAD-Smith, and CAD Arena provide related evaluation settings based on rendering, geometry, validity, or programmatic measurement [Epoch AI, 2026, Barkley et al., 2026, CAD Arena, 2026]. These benchmarks and systems show that executable geometric feedback is valuable. CAD-bench, however, targets a complementary and more significant question: whether a submitted CAD artifact satisfies mechanically meaningful task requirements.

CAD-bench evaluates complete harnesses rather than a single modeling API or decoding format. A harness receives a task prompt and must produce either CAD code or a STEP artifact. The benchmark then executes the submission, exports geometry where needed, and applies task-specific evaluators. The current release includes 17 tasks across four tiers: basic solids, feature-rich parts, standards-like mechanical components, and functional assemblies. The hardest tasks require not only primitives and simple shapes but also interface compatibility and rigid-body function.

The central claim of CAD-bench is that build success is not a sufficient CAD metric. For example, in the reported agent rows, the right-angle gearbox builds in 96.9% of attempts but averages only 4.0% on the full functional score. CAD-bench is designed to distinguish artifacts that merely execute from artifacts that satisfy the dimensional, interface, and functional constraints of the task.

Contributions. This paper makes four contributions.

- We introduce CAD-bench, a 17-task executable benchmark for language-model CAD agents, spanning primitive solids, feature-rich parts, standards-like components, threaded mating pairs, and functional transmissions.
- We define a harness protocol that supports both one-shot CAD-code generation and agent-produced STEP submissions, allowing model-only and tool-using systems to be evaluated through a shared artifact interface.
- We develop a task-specific scoring stack that combines build success, dimensional and pose checks, reference-geometry gates, thread-profile analysis, and Blender-based rigid-body simulation.
- We report complete baseline sweeps for standalone models and agent harnesses, together with scorer-validation cases, aggregation-sensitivity checks, repeated-run diagnostics, and reproducibility artifacts.

2 Related Work

CAD-bench builds on CAD generation, engineering-document understanding, and executable agent benchmarks.

CAD generation and CAD code benchmarks. Seff et al. [2020] introduce SketchGraphs, a large collection of CAD sketches represented as geometric constraint graphs. Ganin et al. [2021] formulate CAD sketch generation as a language modeling problem over serialized design programs, while Wu et al. [2021] model CAD construction histories as operation sequences. Willis et al. [2021] provide human-authored Fusion 360 design sequences and an environment for programmatic CAD reconstruction. More recently, Khan et al. [2024] introduce Text2CAD for generating sequential CAD models from natural-language prompts, and Alrashedy et al. [2024] introduce CADPrompt, a benchmark of natural-language CAD prompts paired with expert CAD code, along with a visual verification loop. CAD-Coder [Guan et al., 2025] and Text-to-CadQuery [Xie and Ju, 2025] move directly toward text-to-parametric-code generation, using executable CadQuery and geometric rewards such as Chamfer distance. CadEval [Epoch AI, 2026] evaluates text-to-CAD systems with rendering success, Chamfer and Hausdorff distances, and volume or consistency checks. CADSmith [Barkley et al., 2026] is especially close in spirit: it uses execution repair plus programmatic OpenCASCADE measurements and visual judging to refine CadQuery code. CAD Arena [CAD Arena, 2026] provides an emerging public comparison grid for 20 prompts, but its current reported metric is valid executable STL production.

These works establish CAD as a structured generation problem and show that executable geometric feedback matters. CAD-bench differs by scoring submitted artifacts against task-level mechanical requirements, including mating interfaces, thread observables, exact pose, and rigid-body function, rather than construction-sequence, rendering, or point-cloud similarity.

Engineering-document understanding. Doris et al. [2024] introduce DesignQA for evaluating multimodal understanding of engineering documentation. That work is complementary to CAD-bench: understanding drawings and manuals is part of the input side of engineering intelligence, while CAD-bench evaluates output-side artifact generation in a runnable CAD environment.

Executable benchmarks for agents. Recent work has shown the value of realistic, environment-grounded evaluation for agents. AgentBench [Liu et al., 2023], GAIA [Mialon et al., 2023], VisualWebArena [Koh et al., 2024], and OSWorld [Xie et al., 2024] benchmark agents in interactive environments rather than static text tasks. In software engineering, SWE-bench [Jimenez et al., 2023] and MLE-bench [Chan et al., 2024] show that end-to-end execution environments uncover failure modes that synthetic benchmarks miss. RE-Bench [Wijk et al., 2024] and PaperBench [Starace et al., 2025] extend this perspective to open-ended research engineering. CAD-bench adopts the same benchmark philosophy for CAD: models should be evaluated in the artifact-producing environment itself.

Benchmark methodology. BetterBench [Reuel et al., 2024] emphasizes best practices such as transparent reporting, replicability, and careful benchmark lifecycle management, while LiveBench [White et al., 2024] highlights the importance of contamination-resistant evaluation. CAD-bench follows these principles through executable scoring, explicit artifact traces, and a design that makes future task refreshes and extensions straightforward.

3 Motivation: Why CAD Needs an Executable Benchmark

CAD evaluation requires stronger checks than syntactic validity or visual plausibility. A model can generate valid Python, export a watertight mesh, or render an object with the correct silhouette while still producing the wrong dimensions, pose, feature placement, mating geometry, or assembly behavior. In mechanical design, these local errors are often decisive.

Surface-level geometric similarity is also incomplete. Many CAD tasks are defined by constrained interfaces rather than by global shape alone. A screw head with an incorrect socket, a thread that cannot mate, a gear train that collides, or a gearbox that fails to transmit motion may remain visually plausible under coarse rendering or point-cloud metrics. Such artifacts may look close to a reference but fail the engineering role specified by the prompt.

CAD agents also operate in executable tool environments, where they must synthesize parametric programs, reason over APIs and exchange formats, and satisfy exact toolchain semantics. CAD-bench is therefore designed around task-specific observables rather than visual complexity alone: visually

Table 1: CAD-bench task taxonomy. Difficulty is authored for benchmark design; as shown later, model difficulty is not strictly monotone in the same order.

Difficulty	Count	Example tasks	Main capability stress
Easy	3	cube, box, ring spacer	pose, units, simple solids
Medium	4	L-bracket, extrusion segment, slotted plate, hex nut blank	feature placement, boolean modeling, conventions
Hard	6	flange, bolted ring, stepped block, large slotted plate, spur gear, M3 socket-head screw	denser geometry, standards-like structure, gears, threads
Functional	4	gearbox, custom threaded pair, compound gearbox, right-angle compound gearbox	assembly reasoning, interface compatibility, rigid-body functionality

simple tasks may require exact conventions or mating constraints, while visually complex tasks may be straightforward once decomposed into deterministic features.

4 Benchmark Design and Artifact Construction

4.1 Scope and task taxonomy

CAD-bench contains 17 tasks divided into four difficulty levels: easy, medium, hard, and functional. The repository keeps the historical internal label `insane` for the functional bucket, but the paper uses "functional" because that is the capability being measured. Table 1 summarizes the task structure.

The benchmark intentionally mixes single-part and multi-part tasks. The early tasks test whether a model can synthesize basic parametric geometry. Middle tasks add feature density and standards-like details. The final tasks require reasoning about interfaces and physical behavior rather than about isolated solids alone.

4.2 Task packaging

Each task is stored as a prompt, metadata, expected values, and evaluator configuration. In the repository, tasks are specified through files such as `prompt.txt`, `task.toml`, and optional reference assets used for reproducibility. The current public task format includes reference programs in `gold.py`, but those are benchmark-side reference implementations rather than the definition of the task itself. The expected metadata includes task ID, difficulty, evaluator, and task-specific expected values. Public task payloads can be loaded from a local mirror or from a Hugging Face repository.

This packaging supports two important use cases: exact reproduction of published harness runs and evaluation of custom harnesses against the same task set and scoring logic.

The tasks are synthetic but engineering-grounded: they specify concrete dimensions, poses, mating interfaces, thread conventions, and gearbox behavior that a CAD artifact must satisfy. This construction is intended to make the benchmark sound as an evaluation artifact: the prompts define real CAD outcomes, the metadata records the expected observables, and the evaluators test submitted artifacts rather than checking whether a model reproduced a particular text program.

4.3 Runtime and submission modes

The benchmark runtime supports two submission styles. In one-shot mode, current code-oriented harnesses return CAD code inside a structured XML block, with the built-in implementation using Build123D [build123d contributors, 2026]. In step-based agent mode, an interactive harness writes the final CAD artifact to a designated STEP path inside the evaluation environment. In both cases, the runtime executes the submission in a containerized workspace and converts the result into a common scoring pipeline.

This paper reports complete standalone model runs separately from complete agent runs. The agent interface remains part of the benchmark runtime, but agent rows are not mixed into the standalone model table because they use interactive tool feedback and a different wall-clock budget.

4.4 Scoring and verification

Each task returns normalized metrics including `build_success`, `task_score`, `overall_score`, and a benchmark-facing reward:

$$\texttt{reward} = 0.05 \cdot \texttt{build\_success} + 0.95 \cdot \texttt{overall\_score}. \tag{1}$$

At the benchmark level, CAD-bench first averages task `overall_score` within each difficulty bucket. Missing benchmark tasks are inserted as explicit zero-score rows for result and paper aggregation. The final score is the category-weighted mean

$$S = \frac{1S_{\text{easy}} + 2S_{\text{medium}} + 3S_{\text{hard}} + 4S_{\text{functional}}}{1 + 2 + 3 + 4}. \tag{2}$$

This means a zero on a difficulty bucket remains part of the denominator rather than silently dropping that category from the leaderboard. The 1:2:3:4 weighting is a benchmark-design choice rather than a statistical estimate of task importance. It prevents the 13 static tasks from dominating the four functional tasks, while still reporting every bucket separately. We do not use any aggregation rule for statistically significant pairwise model-ranking claims; Section 7 reports a small aggregation-sensitivity check.

The task evaluators are designed to match engineering reality. The benchmark uses:

- mesh- and part-level dimensional checks,
- pose and placement metrics,
- reference-geometry gates that downweight large deviations from the authored solution,
- thread-specific profile and pitch metrics for the custom threaded-pair task, and
- Blender-based rigid-body simulation for gearbox tasks.

Reference-geometry gates are used only as coarse sanity gates on selected tasks. They compare boolean volume differences between the submitted artifact and the reference artifact, then multiply the task score by a smooth penalty when the submitted shape has large missing or extra volume. The task-specific metrics remain responsible for dimensions, poses, threads, and function. This design allows ordinary alternative CAD construction histories, but intentionally reduces credit for artifacts that satisfy a few scalar checks while omitting major geometry.

This combination is central to the benchmark. A large part of CAD-bench's value is that it measures the difference between "the program ran" and "the artifact solved the task."

4.5 Scorer validation and ablations

Because CAD-bench uses task-specific evaluators, the release includes explicit scorer-calibration cases in `metadata/cad-bench-scorer-validation.json`. These cases cover reference submissions, independently authored valid submissions, minor defects, major defects, and nonbuild submissions across representative easy, hard-static, standards-like, threaded-pair, and functional tasks. Table 2 summarizes the main cases.

The validation cases serve two purposes. First, they test invariance to implementation details by giving full credit to independently authored valid artifacts that are not copies of the corresponding `gold.py` programs. Second, they test sensitivity to engineering-relevant defects. Several defective artifacts build successfully but receive low full scores, which demonstrates why build success alone is an inadequate CAD metric. Generic geometry proxies also over-credit local-but-incomplete artifacts, such as a screw without a socket or a flange without bolt holes. For gearbox tasks, the decisive signal is rigid-body behavior: a rigid bridge with plausible axle holes builds and may have the approximately correct bounding box, but receives zero because it cannot transmit the required motion.

Worked M3 example. The M3 socket-head screw evaluator illustrates the scoring pattern. It first checks pose and dimensions: top of head at $z = 0$, full body in $z \leq 0$, head diameter and height, under-head length, and major diameter. It then checks contact geometry: socket depth, across-flats estimate, open radius at multiple depths, and sixfold harmonic signal. Finally it checks task-specific thread observables: helical signal, pitch, handedness, and approximate turns-to-seat. The final score

Table 2: Representative scorer-calibration cases. Full is the CAD-bench `overall_score`. Build is a build-success-only ablation. Proxy is the available generic geometry proxy where one exists; gearbox rows are dominated by functional simulation rather than a generic geometry proxy.

Task	Case	Full	Build	Proxy
Cube	independent valid cube	1.000	1.000	1.000
Cube	correct cube shifted off the required pose	0.000	1.000	0.000
Flange	independently constructed valid flange	1.000	1.000	1.000
Flange	bolt circle 1 mm too small	0.800	1.000	0.800
Flange	missing all four bolt holes	0.176	1.000	0.341
M3 screw	independently authored equivalent	1.000	1.000	1.000
M3 screw	threaded screw with missing socket	0.629	1.000	0.723
M3 screw	plain unthreaded shank	0.273	1.000	0.435
Gearbox	reference functional gear pair	0.962	1.000	–
Gearbox	rigid bridge between axle holes	0.000	1.000	–
Gearbox	reference gear pair shifted off axles	0.000	1.000	–
Threaded pair	reference mating threaded pair	0.974	1.000	–
Threaded pair	plausible unthreaded bolt and nut	0.406	1.000	–
Compound gearbox	rigid three-axle bridge	0.000	1.000	–
Right-angle gearbox	intersecting reference-like assembly	0.000	1.000	–

is a weighted mean over generic pose, generic dimensions, generic contact, task thread, and task insertion modules. This is why the independently authored equivalent fixture receives full credit, a threaded screw without a socket receives partial credit, and an unthreaded cylinder receives only low residual credit despite building.

5 Experimental Protocol

5.1 Model and harness matrix

The benchmark is intended to support a mixed portfolio of closed and open providers. The current reviewer result payload contains 19 complete standalone model sweeps and 32 complete agent rows satisfying the validity criteria in Appendix B.3. Table 3 summarizes these rows in the main paper, while Appendix B records the exact run identifiers and full row-level results. The current standalone model set covers OpenAI, DeepSeek, Gemini, Vercel AI Gateway, and free OpenRouter rows.

The paper focuses on completed full-benchmark runs, meaning sweeps that cover all 17 tasks and reach model output plus benchmark scoring. This choice avoids mixing partially completed campaigns with full evaluations. Provider quota failures, API outages, malformed transport responses, harness setup failures, and upload failures are retained in the provenance artifacts for accounting but excluded from model-quality comparisons and from the reviewer result table because they do not measure the model's CAD capability.

5.2 Reporting protocol

For each completed run, we report benchmark score, per-difficulty aggregates, wall-clock time, and estimated cost when the harness exposes enough usage and pricing data. When multiple complete runs exist for the same provider, model, access mode, and reasoning level, all complete rows in the reviewer result payload are reported, and the exact run identifiers are recorded in Appendix B. The current paper does not claim statistically significant pairwise rankings.

This choice reflects the current state of the experimental log rather than an intended limitation of the benchmark. CAD-bench itself stores the artifacts needed for repeated-run reporting, and larger future campaigns can replace unreplicated full-sweep rows with repeated-run aggregates.

6 Results: What CAD-bench Reveals

6.1 Overall performance

Table 3 summarizes the complete full-benchmark evaluations mirrored for anonymous review. Full leaderboards, run identifiers, wall-clock times, and cost estimates are reported in Appendix B; the

Table 3: Summary of complete CAD-bench baselines. Scores are category-weighted using Eq. (2). Full per-row results and run identifiers are reported in Appendix B.

Setting	Rows	Best overall	Mean easy	Mean hard	Mean functional
Standalone models	19	0.599	0.930	0.585	0.037
Agent harnesses	32	0.677	0.906	0.761	0.141

main text focuses on aggregate conclusions because many configurations currently have only one complete sweep.

Three conclusions emerge from the completed rows.

First, CAD-bench is not saturated. The best current standalone model reaches 0.599 overall, and the best agent harness reaches 0.677. These scores leave substantial headroom despite high performance on many easy and static tasks.

Second, static geometry is substantially easier than functional CAD. Completed standalone runs average 0.585 on hard-static tasks but only 0.037 on functional tasks. Agent harnesses improve both categories, but the same pattern remains: they average 0.761 on hard-static tasks and 0.141 on functional tasks.

Third, the benchmark distinguishes model and harness settings. Agent harnesses generally outperform standalone one-shot generation, consistent with the hypothesis that execution feedback helps repair dimensional, geometric, and interface errors. However, agent success is uneven, and functional assemblies remain difficult even when submissions build successfully.

The difficulty claim is not based only on the names of the tiers. Under the same reported artifacts, replacing CAD-bench's full score with weaker proxies would make the benchmark look easier. Reported agent submissions build the right-angle gearbox in 96.9% of attempts but average only 4.0% full score, and standalone submissions build the gearbox task in 63.2% of attempts but average only 6.7% full score. Thus the functional tasks are not merely harder by our label, they are clearly also harder for the language models.

6.2 Per-difficulty behavior

Across the 19 reported standalone model sweeps, the mean score by difficulty is 0.930 on easy, 0.753 on medium, 0.585 on hard, and 0.037 on functional tasks. Across the 32 reported agent sweeps, the corresponding means are 0.906, 0.849, 0.761, and 0.141. The drop on functional tasks is expected. The remaining drop from medium to hard indicates that standards-like details, dense feature placement, and gear/thread structure still matter after simple part synthesis is solved.

In CAD-bench, difficulty labels are benchmark-design labels, not claims about a strict monotone model ordering. Several hard tasks are still static, deterministic solids with many dimensions but little ambiguity, while some lower-level tasks are sensitive to conventions such as exact orientation, alignment, or hole placement. The current results therefore reinforce the point that CAD evaluation must be task-specific rather than purely complexity-labeled.

7 Discussion and Limitations

CAD-bench is mainly an evaluation artifact rather than a final model leaderboard. Its main claim is that executable CAD scoring exposes failures that build success, rendering validity, and coarse geometry proxies miss. The reported rows are useful diagnostic baselines, but many configurations have only one full sweep. We therefore avoid statistically significant pairwise model-ranking claims. Repeated-run statistics in Appendix B.2 support the static-versus-functional difficulty contrast for seven standalone configurations, and future leaderboard versions should promote rows only after repeated full sweeps under a declared aggregation rule.

The aggregation rule is not significantly driving the main benchmark either. Under the current complete result payload, the top standalone row is unchanged under the paper's weighted score, equal bucket averaging, and task-count weighting: Gemini 3.1 Pro offline scores 0.599, 0.727, and 0.709 respectively. For agent rows, weighted and equal-bucket aggregation both put Codex GPT-5.4 mini

Table 4: Per-task mean `overall_score` and build success across the complete reported rows. Standalone means use 19 model sweeps; agent means use 32 agent sweeps. This table is computed from the 867 task rows in the full reviewer artifact, not from the difficulty aggregates alone.

Difficulty	Task	Standalone score	Standalone build	Agent score	Agent build
Easy	cube	0.895	0.947	0.938	0.969
Easy	box	0.947	1.000	0.844	0.844
Easy	ring spacer	0.947	1.000	0.938	0.938
Medium	L-bracket	0.632	0.842	0.906	0.969
Medium	extrusion	0.868	0.947	0.875	0.875
Medium	slotted plate	0.672	0.947	0.864	0.969
Medium	hex nut blank	0.842	0.895	0.750	0.906
Hard	flange	0.746	0.842	0.906	0.969
Hard	bolted ring	0.789	0.789	0.947	1.000
Hard	stepped block	0.812	0.947	1.000	1.000
Hard	large slotted plate	0.718	0.842	0.853	1.000
Hard	spur gear	0.237	0.526	0.345	0.844
Hard	M3 socket screw	0.207	0.526	0.512	0.875
Functional	gearbox	0.067	0.632	0.316	0.844
Functional	threaded pair	0.012	0.211	0.071	0.938
Functional	compound gearbox	0.069	0.632	0.138	0.719
Functional	right-angle gearbox	0.000	0.158	0.040	0.969

medium web first, at 0.677 and 0.781; task-count weighting instead puts Pi GPT-5.4 high offline first at 0.772. The qualitative conclusion is stable: easy/static CAD is much stronger than functional CAD, while fine-grained row ordering should remain diagnostic until repeated runs.

The benchmark is also small. Seventeen tasks are enough for a high-information initial release that covers primitives, feature placement, standards-like details, threaded interfaces, and simulated mechanisms, but they are not enough to claim broad CAD-agent competence. The public release is therefore intended to be versioned. Appendix A describes the planned lifecycle: immutable public task bundles, private refresh tasks with canary hashes, contamination monitoring, and repeated-run promotion for leaderboard rows.

Task-specific scorers encode engineering assumptions. This is necessary for CAD: a threaded pair, socket-head screw, or gearbox cannot be judged by build success alone. It also creates a risk that a valid alternative solution could miss an authored observable. The scorer-validation cases in Table 2 partially address this by giving full credit to independently authored valid artifacts and low credit to controlled defects, but they are not exhaustive. The intended use of CAD-bench is therefore comparative and diagnostic: inspect per-task traces, scorer components, and validation cases, not only the aggregate score.

8 Conclusion

CAD-bench is an executable benchmark for CAD generation that emphasizes artifact correctness over surface plausibility. The current v0 release shows that models and agent harnesses can often produce runnable CAD, but still fail on standards-like details, mating interfaces, and functional assemblies. The benchmark's main contribution is the scoring interface and reproducible artifact protocol: it separates easy geometry from mechanically meaningful CAD behavior and gives future systems a concrete target for improving assemblies and interfaces that actually work.

References

Kamel Alrashedy, Pradyumna Tambwekar, Zulfiqar Zaidi, Megan Langwasser, Wei Xu, and Matthew Gombolay. Generating CAD Code with Vision-Language Models for 3D Designs. arXiv:2410.05340, 2024. URL: https://arxiv.org/abs/2410.05340.

Jesse Barkley, Rumi Loghmani, and Amir Barati Farimani. CADSmith: Multi-Agent CAD Generation with Programmatic Geometric Validation. arXiv:2603.26512, 2026. URL: https://arxiv.org/abs/2603.26512.

build123d contributors. build123d documentation. 2026. URL: https://build123d.readthedocs.io/en/stable/index.html.

CAD Arena. CAD Arena: Open Benchmark for AI-Generated Parametric CAD. 2026. URL: https://cadarena.dev/.

Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, James Aung, Dane Sherburn, Evan Mays, Giulio Starace, Kevin Liu, Leon Maksin, Tejal Patwardhan, Lilian Weng, and Aleksander Mądry. MLE-bench: Evaluating Machine Learning Agents on Machine Learning Engineering. arXiv:2410.07095, 2024. URL: https://arxiv.org/abs/2410.07095.

Anna C. Doris, Daniele Grandi, Ryan Tomich, Md Ferdous Alam, Mohammadmehdi Ataei, Hyunmin Cheong, and Faez Ahmed. DesignQA: A Multimodal Benchmark for Evaluating Large Language Models' Understanding of Engineering Documentation. arXiv:2404.07917, 2024. URL: https://arxiv.org/abs/2404.07917.

Epoch AI. CadEval. 2026. URL: https://epoch.ai/benchmarks/cad-eval.

Yaroslav Ganin, Sergey Bartunov, Yujia Li, Ethan Keller, and Stefano Saliceti. Computer-Aided Design as Language. arXiv:2105.02769, 2021. URL: https://arxiv.org/abs/2105.02769.

Yandong Guan, Xilin Wang, Xingxi Ming, Jing Zhang, Dong Xu, and Qian Yu. CAD-Coder: Text-to-CAD Generation with Chain-of-Thought and Geometric Reward. arXiv:2505.19713, 2025. URL: https://arxiv.org/abs/2505.19713.

Mohammad Sadil Khan, Sankalp Sinha, Talha Uddin Sheikh, Didier Stricker, Sk Aziz Ali, and Muhammad Zeshan Afzal. Text2CAD: Generating Sequential CAD Models from Beginner-to-Expert Level Text Prompts. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL: https://openreview.net/forum?id=5k9XeHIK3L.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-bench: Can Language Models Resolve Real-World GitHub Issues? arXiv:2310.06770, 2023. URL: https://arxiv.org/abs/2310.06770.

Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Chong Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Ruslan Salakhutdinov, and Daniel Fried. VisualWebArena: Evaluating Multimodal Agents on Realistic Visual Web Tasks. arXiv:2401.13649, 2024. URL: https://arxiv.org/abs/2401.13649.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. AgentBench: Evaluating LLMs as Agents. arXiv:2308.03688, 2023. URL: https://arxiv.org/abs/2308.03688.

Gregoire Mialon, Clementine Fourrier, Craig Swift, Thomas Wolf, Yann LeCun, and Thomas Scialom. GAIA: a benchmark for General AI Assistants. arXiv:2311.12983, 2023. URL: https://arxiv.org/abs/2311.12983.

Anka Reuel, Amelia Hardy, Chandler Smith, Max Lamparth, Malcolm Hardy, and Mykel J. Kochenderfer. BetterBench: Assessing AI Benchmarks, Uncovering Issues, and Establishing Best Practices. arXiv:2411.12990, 2024. URL: https://arxiv.org/abs/2411.12990.

Ari Seff, Yaniv Ovadia, Wenda Zhou, and Ryan P. Adams. SketchGraphs: A Large-Scale Dataset for Modeling Relational Geometry in Computer-Aided Design. arXiv:2007.08506, 2020. URL: https://arxiv.org/abs/2007.08506.

Giulio Starace, Oliver Jaffe, Dane Sherburn, James Aung, Jun Shern Chan, Leon Maksin, Rachel Dias, Evan Mays, Benjamin Kinsella, Wyatt Thompson, Johannes Heidecke, Amelia Glaese, and Tejal Patwardhan. PaperBench: Evaluating AI's Ability to Replicate AI Research. arXiv:2504.01848, 2025. URL: https://arxiv.org/abs/2504.01848.

Colin White, Samuel Dooley, Manley Roberts, Arka Pal, Ben Feuer, Siddhartha Jain, Ravid Shwartz-Ziv, Neel Jain, Khalid Saifullah, Siddartha Naidu, Chinmay Hegde, Yann LeCun, Tom Goldstein, Willie Neiswanger, and Micah Goldblum. LiveBench: A Challenging, Contamination-Free LLM Benchmark. arXiv:2406.19314, 2024. URL: https://arxiv.org/abs/2406.19314.

Haoyang Xie and Feng Ju. Text-to-CadQuery: A New Paradigm for CAD Generation with Scalable Large Model Capabilities. arXiv:2505.06507, 2025. URL: https://arxiv.org/abs/2505.06507.

Karl D. D. Willis, Yewen Pu, Jieliang Luo, Hang Chu, Tao Du, Joseph G. Lambourne, Armando Solar-Lezama, and Wojciech Matusik. Fusion 360 Gallery: A Dataset and Environment for Programmatic CAD Construction from Human Design Sequences. *ACM Transactions on Graphics*, 40(4), 2021. URL: https://arxiv.org/abs/2010.02392.

Hjalmar Wijk, Tao Lin, Joel Becker, Sami Jawhar, Neev Parikh, Thomas Broadley, Lawrence Chan, Michael Chen, Josh Clymer, Jai Dhyani, Elena Ericheva, Katharyn Garcia, Brian Goodrich, Nikola Jurkovic, Megan Kinniment, Aron Lajko, Seraphina Nix, Lucas Sato, William Saunders, Maksym Taran, Ben West, and Elizabeth Barnes. RE-Bench: Evaluating frontier AI R&D capabilities of language model agents against human experts. arXiv:2411.15114, 2024. URL: https://arxiv.org/abs/2411.15114.

Rundi Wu, Chang Xiao, and Changxi Zheng. DeepCAD: A Deep Generative Network for Computer-Aided Design Models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021. URL: https://arxiv.org/abs/2105.09492.

Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. OSWorld: Benchmarking Multimodal Agents for Open-Ended Tasks in Real Computer Environments. arXiv:2404.07972, 2024. URL: https://arxiv.org/abs/2404.07972.

A Benchmark lifecycle, release policy, and broader impacts

The artifact is designed to support a versioned benchmark lifecycle rather than a single frozen leaderboard. The submitted release should be interpreted as CAD-bench v0: a public, reproducible task set with exact run artifacts. Future leaderboard versions should use four controls.

- **Frozen public splits.** Published task bundles are immutable and identified by manifest hashes and dataset revisions.
- **Private refresh tasks.** New functional and standards-like tasks can be staged privately, with salted bundle-hash canaries exported before evaluation to establish that the tasks existed before result collection.
- **Versioned leaderboards.** Rows should be reported against an explicit task-set version; old rows should not be silently mixed with refreshed tasks.
- **Repeated-run promotion.** A configuration should move from diagnostic reporting to a primary leaderboard row only after independent full sweeps are available with a declared aggregation rule.

This policy is already reflected in the release files: task manifests record per-file hashes, the release pins the task dataset revision, and the source artifact contains a canary-export command for private

or unreleased tasks. The current paper still reports v0 diagnostic rows because the full repeated-run campaign is not complete; this is why the claims are framed around benchmark behavior and failure modes rather than statistically significant pairwise model rankings.

CAD agents could provide several benefits: lowering the barrier to prototyping, supporting accessibility-focused device customization, improving engineering education, and reducing repetitive modeling work in low-volume manufacturing workflows.

The same capabilities also create risks. More capable CAD generation could accelerate the design of unsafe hardware, restricted components, or deceptive engineering artifacts. It may also encourage overtrust in automatically generated mechanical designs that have not received appropriate engineering review. Transparent, function-aware benchmarking is one mitigation: it makes it harder to confuse runnable CAD code with mechanically correct design.

B Reproducibility and Reporting Details

B.1 Raw complete result rows

Tables 5–7 record the exact complete rows summarized in Table 3. Scores are reported as percentages for readability. The `run` column is the run identifier corresponding to the stored report artifact. Partial runs and infrastructure-failed rows are excluded from model-quality comparisons.

Table 5: Raw complete standalone model rows.

Rank	Harness	Overall	Easy	Med.	Hard	Func.	Min.	Cost	Run
1	gemini/gemini-3.1-pro-preview-offline-none	59.9	100.0	96.8	70.8	23.2	25.4	5.8625	gemini__gemini-3.1-pro-preview-offline-none_20260428T194054Z_2313253
2	openai/gpt-5.4-pro-web-med	53.9	100.0	96.8	78.5	2.5	325.7	37.0405	openai__gpt-5.4-pro-web-med_20260423T122643Z_1626647
3	vercel/alibaba/qwen-3.6-max-preview-offline-none	52.8	100.0	96.8	76.2	1.5	42.8	3.2293	vercel__alibaba__qwen-3.6-max-preview-offline-none_20260430T003706Z_2473332
4	gemini/gemini-3-flash-preview-offline-none	52.6	100.0	96.8	77.4	0.0	21.1	1.3596	gemini__gemini-3-flash-preview-offline-none_20260428T121645Z_2280737
5	openrouter/tencent/hy3-preview:free-offline-none	51.1	100.0	96.8	60.8	8.7	51.4	–	openrouter__tencent__hy3-preview_free-offline-none_20260428T035110Z_2240039
6	openai/gpt-5.3-codex-spark-offline-med	50.4	100.0	96.8	70.0	0.0	6.1	4.9685	openai__gpt-5.3-codex-spark-offline-med_20260427T023421Z_2115684
7	vercel/alibaba/qwen3.6-27b-offline-none	50.0	100.0	96.8	68.9	0.0	58.6	1.5128	vercel__alibaba__qwen3.6-27b-offline-none_20260430T003707Z_2473345
8	openai/gpt-5.3-codex-spark-offline-xhigh	48.9	100.0	96.8	65.0	0.0	6.0	5.4877	openai__gpt-5.3-codex-spark-offline-xhigh_20260427T024838Z_2122296
9	openai/gpt-5.5-offline-med	48.7	100.0	71.8	65.0	12.0	9.8	8.6697	openai_codex__gpt-5.5-offline-med_20260426T055852Z_2010916
10	openai/gpt-5.5-offline-low	47.5	100.0	71.8	76.3	0.6	10.1	9.7384	openai_codex__gpt-5.5-offline-low_20260426T054800Z_2000292
11	deepseek/deepseek-v4-pro-offline-none	47.5	100.0	59.4	68.0	12.9	55.7	3.6274	deepseek__deepseek-v4-pro-offline-none_20260426T042645Z_1958729
12	openai/gpt-5.5-offline-high	45.6	66.7	71.8	82.1	0.0	8.9	7.8075	openai_codex__gpt-5.5-offline-high_20260426T061252Z_2019625
13	openai/gpt-5.5-offline-xhigh	45.6	100.0	71.8	59.3	8.7	9.9	9.2523	openai_codex__gpt-5.5-offline-xhigh_20260426T062226Z_2023449
14	openai/gpt-5.3-codex-spark-offline-high	40.4	100.0	96.8	36.9	0.0	6.1	5.2469	openai__gpt-5.3-codex-spark-offline-high_20260427T024046Z_2118961
15	deepseek/deepseek-v4-flash-offline-none	39.5	100.0	64.3	55.5	0.0	18.1	0.2778	deepseek__deepseek-v4-flash-offline-none_20260426T052907Z_1985868
16	gemini/gemini-3.1-flash-lite-preview-offline-none	34.2	100.0	75.0	30.8	0.0	19.2	0.1993	gemini__gemini-3.1-flash-lite-preview-offline-none_20260428T050032Z_2249465
17	openai/gpt-5.3-codex-spark-offline-low	33.9	100.0	71.8	31.7	0.0	5.8	4.2688	openai__gpt-5.3-codex-spark-offline-low_20260427T022715Z_2111735
18	openrouter/inclusionai/ling-2.6-1t:free-offline-none	13.9	33.3	2.7	33.3	0.0	12.8	–	openrouter__inclusionai__ling-2.6-1t_free-offline-none_20260428T033043Z_2234914
19	gemini/gemma-3-27b-it-offline_nodocs-none	8.1	66.7	0.0	4.6	0.0	3.7	–	gemini__gemma-3-27b-it-offline_nodocs-none_20260430T022138Z_2493048

B.2 Repeated-run statistical check

The current paper reports complete full 17-task sweeps for every row, but many configurations have only one independent sweep. The paper therefore does not use those rows for pairwise model-ranking significance claims. To test the main qualitative difficulty claim under repeated sampling, the source artifact includes 47 complete local reports, with repeated full-sweep statistics for seven standalone configurations, summarized in `metadata/cad-bench-repeat-stats.json`. Additional attempted repeats across the reported standalone and agent matrix were excluded when

Table 6: Raw complete agent rows.

Rank	Agent	Harness	Overall	Easy	Med.	Hard	Func.	Min.	Cost	Run
1	Codex	codex/gpt-5.4-mini-web-med	67.7	100.0	96.8	78.7	36.8	234.0	2.4446	codex__gpt-5.4-mini-web-med_20260420T000623Z_676664
2	Pi	pi/gpt-5.4-offline-high	67.1	100.0	96.8	82.9	32.1	618.8	4.6416	pi__gpt-5.4-offline-high_20260422T092300Z_1340452
3	Codex	codex/gpt-5.4-offline-high	65.3	66.7	71.8	83.6	47.9	245.2	8.2311	codex__gpt-5.4-offline-high_20260419T190624Z_603644
4	Pi	pi/gpt-5.4-offline-xhigh	64.7	100.0	96.8	88.0	22.3	418.3	4.8298	pi__gpt-5.4-offline-xhigh_20260421T075050Z_1041869
5	Pi	pi/gpt-5.4-web-high	64.6	100.0	96.8	82.9	25.9	281.6	4.5650	pi__gpt-5.4-web-high_20260422T092300Z_1340450
6	Pi	pi/gpt-5.4-mini-web-xhigh	61.0	100.0	100.0	84.5	14.2	240.2	2.4442	pi__gpt-5.4-mini-web-xhigh_20260422T092301Z_1340454
7	Pi	pi/gpt-5.4-mini-offline-high	60.3	100.0	75.0	80.3	28.0	169.0	3.5843	pi__gpt-5.4-mini-offline-high_20260422T151655Z_1441926
8	Pi	pi/gpt-5.4-offline-med	59.9	100.0	96.8	76.2	19.3	304.2	1.9074	pi__gpt-5.4-offline-med_20260421T075052Z_1041872
9	Codex	codex/gpt-5.4-web-low	59.1	100.0	96.8	79.4	14.8	279.8	6.3453	codex__gpt-5.4-web-low_20260419T190502Z_603630
10	Codex	codex/gpt-5.4-web-med	59.1	100.0	71.8	78.0	28.2	260.7	5.7144	codex__gpt-5.4-web-med_20260419T190540Z_603629
11	Pi	pi/gpt-5.4-mini-offline-xhigh	59.0	100.0	96.8	83.5	11.4	282.4	3.4776	pi__gpt-5.4-mini-offline-xhigh_20260422T121627Z_1402824
12	Codex	codex/gpt-5.4-mini-offline-med	57.8	100.0	96.8	52.5	31.7	157.2	3.3316	codex__gpt-5.4-mini-offline-med_20260420T004107Z_683812
13	Codex	codex/gpt-5.4-web-xhigh	56.9	100.0	96.8	89.7	1.7	396.2	12.7137	codex__gpt-5.4-web-xhigh_20260421T075051Z_1041874
14	Codex	codex/gpt-5.4-mini-offline-xhigh	56.9	100.0	96.8	77.8	10.5	120.9	4.0830	codex__gpt-5.4-mini-offline-xhigh_20260422T205507Z_1497871
15	Pi	pi/gpt-5.4-web-xhigh	56.0	100.0	71.8	82.9	16.9	419.3	4.7572	pi__gpt-5.4-web-xhigh_20260421T075052Z_1041870
16	Pi	pi/gpt-5.4-mini-web-med	54.5	100.0	96.8	81.8	1.5	191.2	1.6381	pi__gpt-5.4-mini-web-med_20260422T092300Z_1340457

Table 7: Raw complete agent rows, continued.

Rank	Agent	Harness	Overall	Easy	Med.	Hard	Func.	Min.	Cost	Run
17	Pi	pi/gpt-5.4-mini-web-high	54.0	100.0	96.8	75.7	4.7	130.5	2.5139	pi__gpt-5.4-mini-web-high_20260422T183318Z_1468372
18	Codex	codex/gpt-5.4-mini-web-high	53.8	100.0	96.8	78.3	2.3	389.9	3.6274	codex__gpt-5.4-mini-web-high_20260420T235625Z_845623
19	Codex	codex/gpt-5.4-offline-xhigh	53.5	33.3	71.8	76.2	32.3	289.6	11.4765	codex__gpt-5.4-offline-xhigh_20260419T190535Z_603631
20	Pi	pi/gpt-5.4-offline-low	53.3	100.0	75.0	84.4	7.6	171.2	1.5744	pi__gpt-5.4-offline-low_20260419T061026Z_383966
21	Pi	pi/gpt-5.4-web-med	53.0	100.0	71.8	80.1	11.6	366.2	2.1871	pi__gpt-5.4-web-med_20260421T075053Z_1041868
22	Pi	pi/gpt-5.4-web-low	52.6	100.0	96.8	76.2	0.9	65.1	1.1919	pi__gpt-5.4-web-low_20260422T205357Z_1497569
23	Codex	codex/gpt-5.4-web-high	52.1	66.7	96.8	80.1	5.0	300.5	7.9083	codex__gpt-5.4-web-high_20260419T190546Z_603612
24	Codex	codex/gpt-5.4-mini-offline-high	51.3	100.0	96.8	71.1	1.5	169.1	4.4820	codex__gpt-5.4-mini-offline-high_20260420T004018Z_683339
25	Codex	codex/gpt-5.4-mini-web-xhigh	51.1	100.0	71.8	86.9	1.6	416.0	5.9764	codex__gpt-5.4-mini-web-xhigh_20260421T075051Z_1041875
26	Pi	pi/gpt-5.4-mini-offline-med	49.7	100.0	96.8	64.5	2.4	241.8	2.4102	pi__gpt-5.4-mini-offline-med_20260422T094713Z_1351194
27	Pi	pi/gpt-5.4-mini-web-low	48.1	100.0	75.0	77.1	0.0	189.4	0.5317	pi__gpt-5.4-mini-web-low_20260419T064631Z_413438
28	Codex	codex/gpt-5.4-mini-offline-low	48.1	100.0	100.0	57.5	2.2	111.3	1.6688	codex__gpt-5.4-mini-offline-low_20260420T014421Z_709021
29	Codex	codex/gpt-5.4-offline-low	46.8	33.3	46.8	88.4	18.9	181.1	4.4749	codex__gpt-5.4-offline-low_20260419T190806Z_603646
30	Pi	pi/gpt-5.4-mini-offline-low	37.9	100.0	71.8	38.5	4.9	132.1	0.5499	pi__gpt-5.4-mini-offline-low_20260422T100629Z_1359611
31	Codex	codex/gpt-5.4-offline-med	36.2	33.3	46.8	61.6	12.6	260.4	8.0969	codex__gpt-5.4-offline-med_20260419T190604Z_603645
32	Codex	codex/gpt-5.4-mini-web-low	32.9	66.7	50.0	54.1	0.0	246.5	1.8227	codex__gpt-5.4-mini-web-low_20260420T000933Z_677450

provider quota, authentication, unavailable-model, or timeout failures prevented a complete 17-task report.

Across six `openai/gpt-5.3-codex-spark-offline-med` sweeps, the benchmark score is 0.471 with bootstrap 95% CI $[0.453, 0.490]$. The easy bucket is stable at 1.000, medium is 0.973 with CI $[0.968, 0.984]$, hard is 0.572 with CI $[0.507, 0.643]$, and functional is 0.011 with CI $[0.000, 0.033]$. A paired exact two-sided sign-flip test over full-sweep difficulty aggregates gives $p = 0.03125$ for hard score minus functional score, with mean difference 0.561 and bootstrap 95% CI $[0.486, 0.641]$.

Across the four six-sweep `openai/gpt-5.5-offline-*` repeated checks, benchmark means are 0.477 for low with CI $[0.455, 0.492]$, 0.444 for medium with CI $[0.391, 0.503]$, 0.505 for high with CI $[0.454, 0.563]$, and 0.573 for xhigh with CI $[0.528, 0.612]$. Their hard-minus-functional mean differences are 0.701, 0.483, 0.695, and 0.694, respectively, and each exact sign-flip test gives $p = 0.03125$. The xhigh row has the strongest repeated GPT-5.5 mean here, but these runs still support only a within-configuration static-versus-functional difficulty contrast, not a statistically significant model-ranking claim.

Across six `openrouter/tencent/hy3-preview:free-offline-none` sweeps, the benchmark score is 0.518 with CI $[0.471, 0.557]$; easy is 1.000, medium is 0.823 with CI $[0.729, 0.917]$, hard is 0.664 with CI $[0.585, 0.737]$, and functional is 0.135 with CI $[0.095, 0.176]$. The hard-minus-functional sign-flip test gives $p = 0.03125$, mean difference 0.529, and CI $[0.472, 0.594]$. Across five `openrouter/inclusionai/ling-2.6-1t:free-offline-none` sweeps, the benchmark score is 0.154 with CI $[0.080, 0.236]$; hard-minus-functional remains positive at 0.227 with CI $[0.084, 0.407]$, but the exact sign-flip test is $p = 0.125$ at $n = 5$. These repeated-run checks support the paper's main diagnostic claim that functional CAD remains much harder than static CAD for multiple standalone harnesses, while leaving the model-level ranking tables as diagnostic baselines rather than statistically significant pairwise comparisons.

The source artifact also retains the earlier `metadata/cad-bench-repeat-check.json` two-run sanity record for traceability. Future leaderboard versions should report repeated sweeps in the following format:

- number of independent full sweeps per configuration,
- central estimate (mean or median),
- uncertainty interval (e.g., standard deviation or bootstrap 95% CI),
- policy for selecting primary leaderboard rows when multiple repetitions exist.

The current analysis sections avoid pairwise significance claims and use the complete sweeps as baseline diagnostics.

B.3 Run validity criteria

A run is eligible for a complete-results table if it satisfies all of the following conditions:

- it uses a standalone model harness or an agent harness,
- it attempts the full 17-task benchmark under one fixed provider, model, access mode, and reasoning setting,
- each task reaches model-output capture, STEP export when applicable, and benchmark scoring, and
- failures, if any, are model-output or scored-artifact failures rather than provider quota, API outage, harness setup, or artifact-upload failures.

Provider and infrastructure failures are retained in the experiment logs for auditability, but they are not assigned model-quality scores because they do not evaluate the submitted model on CAD generation.

B.4 Compute and environment reporting

The benchmark runtime uses Dockerized environments, Python 3.11, NumPy, trimesh, and Blender for the functional simulation tasks, together with Build123D in the current code-oriented reference and harness pipeline. The reported local runs used the following environment:

- host hardware: Intel Core i5-7300U CPU, 15 GiB RAM, CPU-only scoring except for Blender's local simulation/rendering stack,
- software versions: Docker 29.4.0, Blender 5.1.1, uv 0.11.7, project Python 3.11 environment from `uv.lock`,
- standalone model wall-clock range in the reported rows: 3.7–325.7 minutes,
- agent wall-clock range in the reported rows: 65.1–618.8 minutes,

- failed-run accounting: provider quota failures are kept in logs but excluded from model-quality comparisons.

B.5 Code, data, and asset availability

The benchmark runtime, scoring code, and harness interface are available in the anonymous source artifact accompanying this paper. Public task payloads are hosted in an anonymous reviewer data mirror, and the reviewer mirror includes a compact result artifact with every complete run reported in this paper.

For reviewer inspection, the artifact has a no-API smoke path: `uv run pytest tests/test_anonymous_artifacts.py tests/test_scorer_validation_artifact.py` checks the anonymous packaging and scorer-calibration artifact, and `uv run python scripts/export_scorer_validation.py` regenerates the calibration JSON. Full model sweeps require provider credentials, but scorer validation and artifact consistency do not.

The anonymous submission release pointers are:

- anonymized source archive: included in the supplemental material and mirrored with the anonymous dataset artifact,
- anonymized public task dataset mirror: included in the supplemental material,
- task dataset revision used for this paper: `20ad0b586b70ac2c5e6fe2b501386cf26a887137`,
- reviewer result artifact: `results/cad-bench-reported-results.json` in the anonymous dataset mirror,
- scorer validation artifact: `results/cad-bench-scorer-validation.json` in the anonymous dataset mirror,
- run artifact bundles: `provenance/<run-id>/report.json` entries in the anonymous reviewer artifact,
- provenance URLs: redacted from submitted metadata when they would reveal non-anonymous infrastructure.

The full reviewer artifact includes Croissant metadata with Responsible AI fields describing intended use, limitations, task authoring, and safety-relevant non-use cases. This matches the 2026 Evaluations & Datasets track expectation that datasets document how they support evaluative claims rather than serving only as raw files.

B.6 Licenses and upstream assets

Table 8 documents the main upstream software and data assets used by the benchmark runtime and reviewer dataset.

Table 8: Primary upstream software assets used by CAD-bench.

Asset	License / terms	Role in benchmark
Build123D	Apache-2.0	current reference implementation and code-harness dependency, not the benchmark definition itself
Blender	GNU GPL	rigid-body simulation and rendering for functional gearbox evaluation
NumPy	BSD-style license	numeric processing in evaluators and utilities
trimesh	MIT	mesh conversion and geometric analysis
Repository code	MIT	benchmark runtime, harnesses, and scoring stack
Public task dataset	MIT	hosted task prompts, metadata, reference programs, and authored benchmark assets
Authored CAD fixtures	MIT	synthetic task-side fixtures, including the M3 socket-head screw equivalent fixture
Closed model APIs	provider terms	external inference services used for benchmarked models

The benchmark tasks and authored fixtures are synthetic benchmark assets rather than redistributed proprietary CAD files.

NeurIPS Paper Checklist

1. **Claims**

Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

Answer: [Yes]

Justification: The abstract and Sections 1 and 4 state the paper's scope as introducing an executable CAD benchmark, documenting its scoring stack, and reporting the currently completed baseline results with explicit limitations around unreplicated full-sweep comparisons.

Guidelines:

- The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [Yes]

Justification: The main results section and Appendix A discuss benchmark size, the current emphasis of the reported baselines on Build123D-based harnesses rather than the benchmark definition itself, incomplete all-provider campaign coverage, current dependence on a small number of completed full sweeps, benchmark exposure risk, and the planned refresh/holdout protocol.

Guidelines:

- The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: The paper is a benchmark and empirical evaluation paper and does not include theoretical results or proofs.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Sections 4 and 5 describe the evaluation protocol and reported metrics, and Appendix A records completed run identifiers, exact dataset revision, code/data availability, and compute details.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The paper describes the benchmark runtime, scoring code, and task packaging as releasable assets, and the appendix describes the anonymous source artifact, reviewer data mirror, pinned task revision, and result artifacts.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Sections 4 and 5 specify the relevant settings for this evaluation paper: provider, model, access mode, reasoning level, task coverage, benchmark score definition, per-difficulty reporting, and build-success aggregation.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: The paper reports bootstrap confidence intervals and exact paired sign-flip tests for repeated full-sweep checks across seven standalone configurations where repeat counts support them. It still explicitly avoids pairwise model-ranking significance claims for configurations without enough repeated independent sweeps, and Appendix A states how repeated-run aggregates should be reported in future leaderboard versions.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.

- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Appendix A reports host hardware, key software versions, full-sweep wall-clock ranges for standalone and agent rows, and how provider quota failures are accounted for.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The work is a benchmark and evaluation study, does not involve deception or human subjects, and includes explicit discussion of limitations, failure modes, and broader impacts in Sections 6 and 7.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: The Limitations section and Appendix A discuss positive impacts for engineering productivity and accessibility, and negative impacts including unsafe hardware design, deceptive artifacts, and overtrust in generated designs.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The paper releases a benchmark runtime and evaluation assets, not a new high-risk generative model or scraped dataset that would require controlled release safeguards of the type described here.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: The bibliography credits the main upstream research assets, and Appendix A includes a dedicated license table for the primary software dependencies, project code, public task dataset, authored fixtures, and provider terms.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: The benchmark task taxonomy, runtime, scoring protocol, run accounting, and release structure are documented in Sections 4–6 and Appendix A.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: The paper does not involve crowdsourcing or research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: The paper does not involve human subjects research.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: LLMs are not involved as any important, original, or non-standard components.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.