OpCAD: From Prompt-Code Pairs to Operation-Aware CAD Data for Parametric Text-to-CAD

Anonymous ACL submission

Abstract

Parametric text-to-CAD programs can export a valid STEP file while attaching a slot to the wrong face or silently dropping a requested hole – mistakes that "code that runs" supervision misses. We argue this is a dataset problem and build a versioned `build123d` corpus that annotates each program with typed operations, legal ordering, topology-aware attachments, realized effects, and validation traces. Fine-tuning the same Qwen3.5-4B base on four cumulative regimes shows distinct effects: cross-library conversion is brittle, operation expansion buys coverage, topology-aware construction buys fidelity, and only quality-gated export reaches deployment coverage. OpCAD, a 10k-step compact 4B specialist on the final regime, drops invalid outputs from $64.3\%/78.1\%$ to $13.8\%/27.0\%$ on the full L1–L3 Text2CAD-Bench `build123d` split (all-GEO/PRO). A matched-success paired bootstrap locates the gain: Qwen3.5-Plus remains more faithful per item; OpCAD's win is valid-output coverage, not per-item geometric superiority.

1 Introduction

Text-to-CAD generates parametric programs grounded in legal modeling operations, not static shapes. A useful generation translates natural-language design intent into code that a geometric kernel will accept, export, edit, and re-execute, which makes the training data the central bottleneck.

Recent text-to-CAD systems include sequential command modeling (Wu et al., 2021; Xu et al., 2022; Khan et al., 2024; Wu et al., 2023; Xu et al., 2024a) and direct CAD-code generation (Xie and Ju, 2025; Yuan et al., 2024), on top of programmatic-CAD datasets and BREP-side generators (Willis et al., 2021; Koch et al., 2019; Seff et al., 2020; Lambourne et al., 2021; Jayaraman et al., 2023; Xu et al., 2024b; Liu et al., 2024). They demonstrate that language models can write CAD code, but they also expose a data-construction gap sharper than in plain code generation (Chen et al., 2021; Austin et al., 2021; Hendrycks et al., 2021; Li et al., 2022; Liu et al., 2023; Zhuo et al., 2025; Jimenez et al., 2024): a prompt–code pair can be valid by accident while failing to teach where features attach, in what order operations are legal, and whether requested holes, shells, grooves, ribs, or mirrored slots actually appear in the exported BREP. Data-centric code-SFT work (Li et al., 2023; Wei et al., 2024; Gunasekar et al., 2023) similarly identifies validity as the dominant lever but does not address topology-aware operation grounding.

We move from prompt–code pairs to *structured operation supervision*: compact units that specify typed operations, legal order, preconditions, topology-aware attachments, realized effects, and validation traces. **(C1)** We introduce operation regimes as a descriptive construction schema and audit vocabulary for parametric text-to-CAD datasets. **(C2)** We construct a versioned `build123d` corpus and analyze how four cumulative construction regimes leave measurable downstream signatures in the same Qwen3.5-4B base. **(C3)** We evaluate OpCAD, a 10k-step compact 4B specialist on the final regime, against Qwen-family models, showing that operation-aware supervision shifts the model toward high valid-output coverage while Qwen3.5-Plus retains higher per-item fidelity on shared successes (Table 4).

2 Structured Operation Supervision

We represent the construction contract for a parametric CAD example as

$$\rho = (O, \prec, P, A, R, V),$$

where O contains typed CAD operations, $\prec$ is legal composition order, P is preconditions, A is topology-aware attachment, R is realized geometric effects, and V is validation traces. The tuple

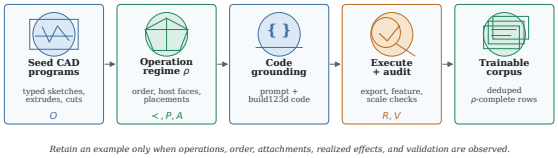

Figure 1: Operation-aware view of parametric CAD supervision. The retained example exposes legal operation order ($\prec$), topology-aware attachment (A), realized feature effects (R), and validation traces (V), rather than only a prompt and final code string.

is not a factorial causal model; it is a construction schema for making implicit CAD assumptions auditable. Operationally, $O, \prec$ mainly expand valid-output coverage, P, A target fidelity after execution succeeds, and R, V push examples toward deployment coverage by requiring promised features to appear in the exported BREP. In `build123d`, for instance, sketches precede extrusion, host solids precede boolean cuts, and feature operations must bind to valid faces, axes, offsets, and dimensions before STEP export. Figure 1 summarizes this view.

3 Versioned Corpus Construction

We build a versioned parametric CAD corpus targeting `build123d`. Each version is a cumulative construction *state*, not a one-factor ablation: D1 converts the inherited CADQuery/Text2CAD-style source into `build123d`; D2 expands parametric idioms with simple templates; D3 adds topology-aware host-face analysis and feature placement; D4 adds execution gates, feature-realization audits, scale filters, prompt-code consistency checks, and global deduplication. Appendix A gives the full state table. Figure 2 connects the corpus footprint to model behavior: D4 retains high-volume compound-operation families, while D1–D4 and OpCAD occupy different coverage/fidelity operating points.

ρ field	D1	D2	D3	D4	Primary target
O: typed operations	○	✓	✓	✓	parametric idioms
$\prec$: legal order	○	✓	✓	✓	valid composition
P: preconditions	–	○	✓	✓	legal host context
A: attachments	–	○	✓	✓	feature placement
R: realized effects	–	–	○	✓	silent-failure removal
V: validation trace	–	○	○	✓	coverage–quality balance

Table 1: Operation-supervision fields exposed by each construction stage and the behavior they are intended to affect. ✓ = explicit and audited, ○ = partial or implicit, – = absent.

4 Evaluation

Benchmark. All evaluation uses the publicly available **Text2CAD-Bench** (Wang et al., 2026) under its `build123d` output protocol: 507 items, each in two prompt styles – GEO (direct geometric description) and PRO (procedural instruction) – for a total of 1,014 Qwen-family jobs. Items are stratified into L1 (primitive constructions: box, cylinder, sphere, simple booleans), L2 (multi-step constructions with patterns, fillets, shells, multi-body assemblies), and L3 (operation-heavy constructions). The construction-state study and Text2CadQuery comparison use the shared L1/L2 scope (816 jobs) because L3 outputs are unavailable for the corresponding D1–D4 runs. All reference programs are part of Text2CAD-Bench and were authored independently of our training pipeline.

Metrics. We report invalid rate (IR), Chamfer distance (CD), and voxel IoU. Generated and ground-truth artifacts are independently bbox-normalized, rigidly aligned, and compared with symmetric bidirectional CD and voxel IoU on a 64^3 grid with ICP refinement on 8,192 sampled points. CD is multiplied by 1,000 in tables. We use median CD/IoU on successful generations to reduce post-alignment outlier sensitivity. Coverage-weighted expected IoU assigns IoU 0 to invalid outputs on the union of jobs; for paired comparisons we use a paired bootstrap on matched-success subsets. All 95% intervals are non-parametric bootstrap with 2,000 resamples, seed 0.

Models. We use two evaluation roles. For the *construction-state* study, Qwen3.5-4B is fine-tuned separately on D1/D2/D3/D4 and selected by the same dev-IR rule (within ± 500 steps, terminating by step 4,517); these rows compare state packages while holding the base, evaluator, decoding, prompt template, and repair budget fixed. For the *deployed compact specialist*, we train the final D4 corpus longer for 10,000 steps and call the resulting model OpCAD. Thus D4 supports the corpus-construction claim, while OpCAD tests how far the final state can push a compact Qwen model at the full L1–L3 evaluation scale. All fine-tuned and proprietary Qwen rows share decoding (greedy, max 2,048 tokens), prompt template, and the three-round `build123d` repair protocol. Qwen3.5-Flash and Qwen3.5-Plus are pinned to the `2026-04-20` stable revision. Text2CadQuery (Xie and Ju, 2025) is executed under a CadQuery $\rightarrow$ STEP adapter;

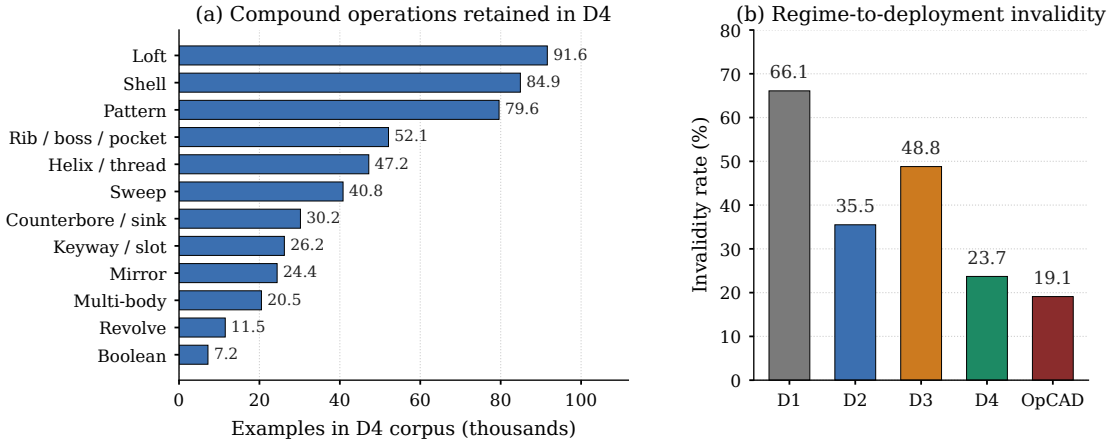

Figure 2: Operation-aware footprint and downstream signature. Left: high-volume compound-operation families retained in D4. Right: invalid rate after fine-tuning the same Qwen3.5-4B base on D1–D4, plus OpCAD, the deployed 10k-step specialist trained on D4. D3 trades coverage for successful-output fidelity; D4/OpCAD shift toward deployment coverage.

failures at code extraction, CadQuery execution, or STEP export count as invalid. Because independent bbox normalization removes absolute scale, the D4 validation gate additionally enforces a $[0.5, 2.0]$ bbox-max-extent ratio against the reference.

5 Results

Construction stages leave downstream signatures. Table 2 shows that data construction changes the operating point, not monotonically "more data" or "longer training" is better. D2 sharply improves valid-output coverage over the converted-source D1. D3 is more selective but gives the strongest L2 successful-output fidelity. D4 changes the objective toward deployable coverage: it has the lowest short-run construction-state IR, while sacrificing some conditional L2 fidelity. OpCAD is therefore not a cell-wise winner over D4; rather, it tests whether the final D4 corpus can train a compact specialist that preserves coverage at the full L1–L3 evaluation scale. Appendix Figure 3 makes this operating-region movement explicit: D2 primarily moves left, D3 moves upward on L2 fidelity but remains selective, and D4/OpCAD move toward the low-IR deployment corner.

Deployed specialist in the Qwen family. Table 3 evaluates OpCAD at full scale. It is not another construction-state checkpoint: it is the deployed 10k-step specialist trained on D4. OpCAD reduces invalid outputs by more than 50 percentage points on both prompt styles relative to the 4B base, with IR intervals disjoint from the base and Qwen3.5-Plus on the ALL scope. This is a deployment result, not a claim that longer D4 training improves every L1/L2 fidelity cell: under this `build123d` generation-and-repair protocol, OpCAD achieves the highest valid-output coverage among the tested Qwen-family systems.

Coverage, not per-item refinement. Table 4 locates the gain. On the matched-success subset where OpCAD and Qwen3.5-Plus both produce a valid STEP, Plus is more faithful by $+0.20$ to $+0.26$ IoU per item across L1 and L2 cells, with paired-bootstrap intervals (Efron and Tibshirani, 1993) excluding zero. The OpCAD gain is therefore coverage and coverage-weighted utility: OpCAD succeeds on a 1.5–$3.3\times$ larger set, while Plus remains stronger at per-item refinement once both systems land valid geometry. The lower block of Table 4 also supports the state story: training on the un-validated D1 corpus damages the base on items the base already solves, while adding parametric-idiom supervision (D2) restores conditional fidelity over D1.

6 Conclusion

Parametric text-to-CAD data should expose typed operations, legal ordering, topology-aware attachments, realized effects, and validation traces, not only final code. In our versioned `build123d` corpus, these construction choices shift a compact Qwen model toward high valid-output coverage; matched-success analysis shows the gain is coverage rather than per-item superiority over proprietary Qwen models. The non-monotonic D3/D4/OpCAD pattern suggests that CAD data

	Model	L1 IR↓	L1 CD↓	L1 IoU↑	L2 IR↓	L2 CD↓	L2 IoU↑
Geo	Qwen3.5-4B	53.1%	53.92	0.594	71.6%	94.57	0.499
	D1	50.2%	68.22	0.323	74.1%	74.94	0.173
	D2	27.1%	67.21	0.454	32.3%	83.98	0.345
	D3	32.9%	**48.00**	0.641	55.7%	**44.25**	**0.681**
	D4	20.3%	54.93	0.589	20.4%	83.42	0.302
	OpCAD	**17.4%**	49.11	**0.670**	**8.5%**	62.05	0.505
Pro	Qwen3.5-4B	69.6%	28.61	0.839	80.6%	96.50	0.438
	D1	62.8%	64.56	0.579	77.6%	104.68	0.181
	D2	30.4%	79.06	0.527	52.7%	99.70	0.191
	D3	39.6%	**39.90**	**0.720**	67.7%	**44.61**	**0.681**
	D4	29.0%	43.46	0.670	**25.4%**	85.42	0.360
	OpCAD	**21.3%**	55.18	0.606	29.4%	62.26	0.492

Table 2: L1–L2 median results under v11 independent normalization. IR is invalid rate; CD is $\times 1000$. OpCAD is trained longer on D4, not a construction-stage checkpoint.

Model	ALL geo	ALL pro	L3 geo	L3 pro
Qwen3.5-4B	64.3 / .188	78.1 / .135	72.7 / .088	90.9 / .050
OpCAD	**13.8 / .441**	**27.0 / .372**	**17.2 / .263**	**34.3 / .208**
Qwen3.5-Flash	56.8 / .304	69.0 / .242	93.9 / .022	98.0 / .016
Qwen3.5-Plus	63.1 / .291	76.3 / .189	99.0 / .000	99.0 / .007
95% bootstrap CIs on IR (%):				
Qwen3.5-4B	[60.2,68.2]	[74.6,81.9]	[63.6,81.8]	[84.8,96.0]
OpCAD	**[10.8,17.0]**	**[23.3,31.2]**	**[10.1,24.2]**	**[24.2,43.4]**
Qwen3.5-Plus	[59.0,67.3]	[72.6,80.1]	[97.0,100]	[97.0,100]

Table 3: Qwen-family results on Text2CAD-Bench `build123d`. Each cell reports IR (%) / E[IoU], with invalid outputs assigned IoU 0. ALL aggregates L1–L3; L3 is the 99-item subset per prompt.

Comparison	N	Δ IoU [95% CI]
Plus is more faithful on shared successes		
OpCAD vs. Plus, L1-geo	105	$+0.200$ $[+0.145, +0.258]$
OpCAD vs. Plus, L1-pro	69	$+0.201$ $[+0.145, +0.260]$
OpCAD vs. Plus, L2-geo	71	$+0.249$ $[+0.179, +0.322]$
OpCAD vs. Plus, L2-pro	35	$+0.256$ $[+0.158, +0.359]$
Data-regime controls		
base vs. D1, ALL	113	-0.161 $[-0.222, -0.099]$
D1 vs. D2, ALL	203	$+0.080$ $[+0.034, +0.126]$

Table 4: Matched-success paired analyses (paired bootstrap, 2,000 resamples, seed 0). Positive deltas mean model B is more faithful on items both systems successfully execute.

construction should be selected for the intended operating point–coverage, conditional fidelity, or deployment robustness–rather than optimized as a single scalar. The broader takeaway is data-centric: CAD corpora should document operation stages and validation predicates, because they determine whether a model learns editable parametric behavior or merely emits plausible code strings.

7 Limitations

The versioned comparison is cumulative rather than a fully controlled causal ablation, and all fine-tunes use one Qwen3.5-4B base. Independent bbox normalization and ICP reduce sensitivity to absolute scale and placement errors, so we report coverage separately and enforce scale gates during corpus construction. L3 state checkpoints, human editability studies, and operation-tree consistency metrics are future work. Text2CadQuery is a cross-library diagnostic rather than a native leaderboard comparison. Our train–benchmark audit finds 0 exact code/prompt overlaps and max prompt-token Jaccard 0.043, but does not rule out structural, parametric, or geometry-level similarity.

References

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

Paul J. Besl and Neil D. McKay. 1992. A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256.

build123d contributors. 2026. build123d: A python-based parametric cad library.

CadQuery contributors. 2026. Cadquery: A python parametric cad scripting framework.

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, and 39 others. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Bradley Efron and Robert J. Tibshirani. 1993. *An Introduction to the Bootstrap*. Chapman and Hall/CRC.

Haoqiang Fan, Hao Su, and Leonidas J. Guibas. 2017. A point set generation network for 3d object reconstruction from a single image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

Suriya Gunasekar, Yi Zhang, Jyoti Aneja, Caio César Teodoro Mendes, Allie Del Giorno, Sivakanth Gopi, Mojan Javaheripi, Piero Kauffmann, Gustavo de Rosa, Olli Saarikivi, Adil Salim, Shital Shah, Harkirat Behl, Xin Wang, Sébastien Bubeck, Ronen Eldan, Adam Tauman Kalai, Yin Tat Lee, and Yuanzhi Li. 2023. Textbooks are all you need. *arXiv preprint arXiv:2306.11644*.

Dan Hendrycks, Steven Basart, Saurav Kadavath, Mantas Mazeika, Akul Arora, Ethan Guo, Collin Burns, Samir Puranik, Horace He, Dawn Song, and Jacob Steinhardt. 2021. Measuring coding challenge competence with APPS. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*.

Pradeep Kumar Jayaraman, Joseph G. Lambourne, Nishkrit Desai, Karl D. D. Willis, Aditya Sanghi, and Nigel J. W. Morris. 2023. Solidgen: An autoregressive model for direct b-rep synthesis. *Transactions on Machine Learning Research*.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. 2024. SWE-bench: Can language models resolve real-world github issues? In *Proceedings of the International Conference on Learning Representations*.

Mohammad Sadil Khan, Sankalp Sinha, Talha Uddin Sheikh, Didier Stricker, Sk Aziz Ali, and Muhammad Zeshan Afzal. 2024. Text2cad: Generating sequential cad designs from beginner-to-expert level text prompts. *Advances in Neural Information Processing Systems*.

Sebastian Koch, Albert Matveev, Zhongshi Jiang, Francis Williams, Alexey Artemov, Evgeny Burnaev, Marc Alexa, Denis Zorin, and Daniele Panozzo. 2019. ABC: A big CAD model dataset for geometric deep learning. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Joseph G. Lambourne, Karl D. D. Willis, Pradeep Kumar Jayaraman, Aditya Sanghi, Peter Meltzer, and Hooman Shayani. 2021. Brepnet: A topological message passing system for solid models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, and 1 others. 2023. Starcoder: May the source be with you! *Transactions on Machine Learning Research*.

Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, Thomas Hubert, Peter Choy, Cyprien de Masson d'Autume, Igor Babuschkin, Xinyun Chen, Po-Sen Huang, Johannes Welbl, Sven Gowal, Alexey Cherepanov, and 7 others. 2022. Competition-level code generation with alphacode. *Science*, 378(6624):1092–1097.

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 *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*.

Yujia Liu, Anton Obukhov, Jan Dirk Wegner, and Konrad Schindler. 2024. Point2cad: Reverse engineering cad models from 3d point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Qwen Team. 2025. Qwen3 technical report. *arXiv preprint*.

Ari Seff, Yaniv Ovadia, Wenda Zhou, and Ryan P. Adams. 2020. Sketchgraphs: A large-scale dataset for modeling relational geometry in computer-aided design. In *Proceedings of the International Conference on Machine Learning*.

Liang Wang and 1 others. 2026. Text2cad-bench: A benchmark for llm-based text-to-parametric cad generation. ArXiv preprint arXiv:2605.18430.

Yuxiang Wei, Zhe Wang, Jiawei Liu, Yifeng Ding, and Lingming Zhang. 2024. Magicoder: Empowering code generation with oss-instruct. *Proceedings of the International Conference on Machine Learning*.

Karl D. D. Willis, Yewen Pu, Jieliang Luo, Hang Chu, Tao Du, Joseph G. Lambourne, Armando Solar-Lezama, and Wojciech Matusik. 2021. Fusion 360 gallery: A dataset and environment for programmatic cad construction from human design sequences. *ACM Transactions on Graphics*, 40(4).

Rundi Wu, Chang Xiao, and Changxi Zheng. 2021. Deepcad: A deep generative network for computer-aided design models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*.

Sifan Wu, Amir Khasahmadi, Mor Katz, Pradeep Kumar Jayaraman, Yewen Pu, Karl D. D. Willis, and Bang Liu. 2023. CAD-LLM: Large language model for cad generation. In *NeurIPS Workshop*.

Haoyang Xie and Feng Ju. 2025. Text-to-cadquery: A new paradigm for cad generation with scalable large model capabilities. *arXiv preprint arXiv:2505.06507*.

Jingwei Xu, Chenyu Wang, Zibo Zhao, Wen Liu, Yi Ma, and Shenghua Gao. 2024a. CAD-MLLM: Unifying multimodality-conditioned cad generation with mllm. *arXiv preprint arXiv:2411.04954*.

Xiang Xu, Pradeep Kumar Jayaraman, Joseph G. Lambourne, Karl D. D. Willis, Zhengqing Wang, and Yasutaka Furukawa. 2024b. Brepgen: A b-rep generative diffusion model with structured latent geometry. *arXiv preprint arXiv:2401.15563*.

Xiang Xu, Karl D. D. Willis, Joseph G. Lambourne, Chin-Yi Cheng, Pradeep Kumar Jayaraman, and Yasutaka Furukawa. 2022. Skexgen: Autoregressive generation of cad construction sequences with disentangled codebooks. *Proceedings of the International Conference on Machine Learning*.

Haocheng Yuan, Jing Xu, Hao Pan, Adrien Bousseau, Niloy J. Mitra, and Changjian Li. 2024. Cadtalk: An algorithm and benchmark for semantic commenting of cad programs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Terry Yue Zhuo, Minh Chien Vu, Jenny Chim, Han Hu, Wenhao Yu, Ratnadira Widyasari, Imam Nur Bani Yusuf, Haolan Zhan, Junda He, Indraneil Paul, and 1 others. 2025. Bigcodebench: Benchmarking code generation with diverse function calls and complex instructions. In *Proceedings of the International Conference on Learning Representations*.

A Construction Stages

Regime	Source / transformation	Construction signal added	Observed signature
D1	Original CADQuery-derived corpus converted to `build123d`.	Early prompt–code supervision with limited standardized operation-regime metadata; used as the first construction stage, not as a matched restricted-operation ablation.	Brittle parametric behavior: 66.1% overall IR.
D2	Template-style feature expansion over `build123d` examples.	Systematic exposure to common `build123d` idioms and simple feature templates; broader valid-code coverage than D1, but limited topology-aware placement and validation.	Coverage improves to 35.5% IR; L2 geo drops 74.1% → 32.3%.
D3	Topology-aware procedural augmentation.	Candidate host-face analysis, part-archetype inference, feature placement constraints, and engineering-drawing-style prompts.	Best L2 successful-output fidelity, but coverage degrades to 48.8% IR.
D4	Quality-gated export layer built on the D3 generation engine.	Mixed-source generation, execution gating, feature-realization audit, prompt–code consistency checks, feature-scale anomaly checks, and global trainable deduplication.	Best coverage–quality balance: 23.7% overall IR.
OpCAD	**Deployed compact specialist trained for 10k steps on D4.**	**No new construction fields; tests how far the final structured-operation corpus can push the same Qwen3.5-4B base under the deployed training budget.**	**Final deployed coverage: 19.1% L1/L2 IR and 13.8%/27.0% ALL geo/pro IR.**

Table 5: Versioned construction regimes used in the dataset-evolution study. D1–D4 are construction regimes; OpCAD is included to connect the final D4 corpus to the deployed 10k-step specialist, not as a fifth data-construction stage. The comparison is intentionally a construction-regime analysis rather than a single-factor causal ablation: each stage changes how parametric CAD supervision is constructed and filtered.

B Reproducibility Details

Base model and optimization. All five fine-tuned checkpoints (D1/D2/D3/D4/OpCAD) start from the open-weight **Qwen3.5-4B** base (Qwen Team, 2025) with identical tokenizer and architecture. We use AdamW (β_1=0.9, β_2=0.95, weight decay 0.1), a warmup-cosine schedule with peak learning rate 1×10^{-4} and 1,000-step linear warmup, packed sequence length 4,096, and global batch size 256. D1/D2/D3/D4 are selected by the dev-IR rule within ±500 steps and terminate by step 4,517; OpCAD uses the same D4 corpus but is trained for 10,000 steps without early stopping. Decoding is greedy (T=0, max 2,048 new tokens) and identical across all Qwen-family rows, including Qwen3.5-Flash and Qwen3.5-Plus.

Corpus sizes. D1 $\approx$ 0.70M records (pre-dedup), D2 $\approx$ 0.62M (pre-dedup), D3 $\approx$ 0.88M (post-dedup), D4 and the corresponding OpCAD corpus $\approx$ 0.96M (post-dedup).

Repair protocol and Text2CadQuery adapter. Each test prompt is generated, executed in a sandboxed `build123d` (build123d contributors, 2026) environment, and post-checked for (a) Python execution success, (b) successful STEP export, (c) non-empty closed solid; on failure, the failure trace is appended and re-generation is attempted up to three rounds. Only the final accepted generation is scored. The Text2CadQuery (Xie and Ju, 2025) adapter extracts the fenced `python` block, executes it under a sandboxed CadQuery 0.2 (CadQuery contributors, 2026) kernel with a 30 s wall-clock budget per item, and exports STEP via OpenCASCADE. Failure modes are recorded as EXTRACT_FAIL, EXEC_FAIL, EXPORT_FAIL, or EMPTY_SOLID; library mismatches (e.g. CadQuery's `Workplane` vs. `build123d`'s `Plane`) are treated as EXEC_FAIL rather than silently absorbed.

Validation predicates (V). The example-retention gate checks: (i) the program executes within 30 s; (ii) STEP export succeeds; (iii) the solid is non-empty and closed (Euler characteristic > 0); (iv) all feature names referenced in the prompt are realized in the exported BREP; (v) the prompt's named operations occur in the program AST; (vi) the bounding-box max-extent ratio between generation and reference lies in $[0.5, 2.0]$. D4 enforces all six predicates; D3 enforces (i)–(iii); D2 enforces (i)–(ii); D1 enforces none.

Feature-realization audit (R). We extract feature mentions from prompt spans using a fixed CAD vocabulary (e.g., hole, slot, shell, rib, pocket, chamfer, fillet, pattern, mirror, thread) plus numeric/count modifiers. On the code side, we check that the corresponding operation family appears in the normalized `build123d` AST or helper-call trace. On the geometry side, exported BREP summaries verify non-empty solids and operation-specific effects where deterministic checks are available: added/removed face or edge counts for cuts and holes, repeated bounding-box-aligned feature instances for patterns, wall-thickness change for shell-like examples, and scale-ratio sanity checks. Ambiguous cases are rejected rather than repaired into the corpus. These checks are heuristic and can miss subtle geometry, but they prevent examples that merely execute while omitting the named feature.

Evaluator. Generated and reference artifacts are independently bbox-max-extent normalized, rigidly aligned via ICP (Besl and McKay, 1992) on $8{,}192$ sampled points, then voxelized at 64^3 for symmetric bidirectional Chamfer distance (Fan et al., 2017) and voxel IoU. The same evaluator is used for all Qwen-family rows and for the Text2CadQuery external baseline.

Proprietary Qwen baselines. Qwen3.5-Flash and Qwen3.5-Plus are pinned to the `2026-04-20` stable revision and use the identical prompt template and three-round repair protocol as the fine-tuned systems.

Leakage audit. Over $962{,}066$ training rows, $1{,}014$ benchmark prompts, and 506 benchmark reference programs we find 0 exact normalized-code overlaps and 0 exact normalized-prompt overlaps. A rare-token candidate search across $217{,}770{,}684$ MinHash candidate pairs yields no prompt with token Jaccard > 0.3 (max observed 0.043). This lexical audit does not rule out geometry-level similarity.

Bootstrap intervals. All 95% intervals reported in the main paper are non-parametric bootstrap (Efron and Tibshirani, 1993) with $2{,}000$ resamples and seed 0; for matched-success conditional comparisons we use a paired bootstrap on the intersection of successful items. The script is included in the release.

C Text2CadQuery Diagnostic

Level	Prompt	OK/Total	IR↓	CD Med↓	IoU Med↑
L1	geo	173/207	16.4%	169.25	0.061
L1	pro	179/207	13.5%	166.86	0.102
L2	geo	151/201	24.9%	165.28	0.078
L2	pro	164/201	18.4%	176.89	0.072

Table 6: Text2CadQuery on the shared L1/L2 scope. Outputs are executed, exported, independently bbox-normalized against GT, aligned, and evaluated with symmetric bidirectional CD and voxel IoU. CD is multiplied by 1000.

D Additional Visual Diagnostics

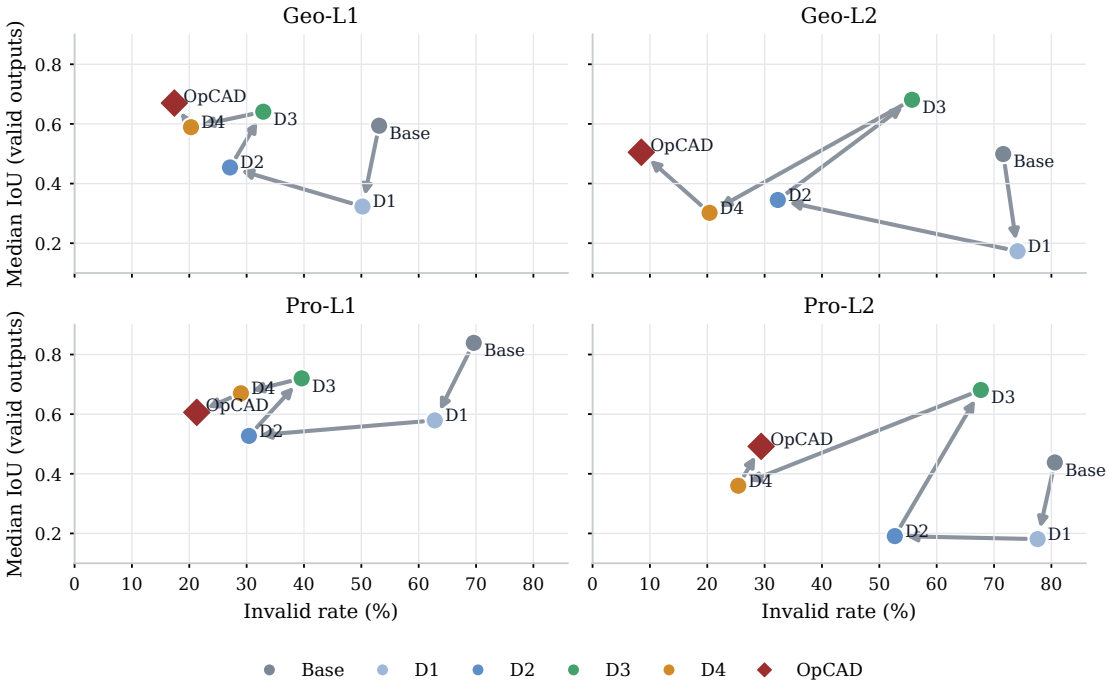

Figure 3: Construction-aware operating regions. Each panel places the same Qwen3.5-4B base and its D1/D2/D3/D4/OpCAD fine-tunes by invalid rate and median IoU on successful outputs. Moving left is higher valid-output coverage; moving up is higher conditional fidelity.

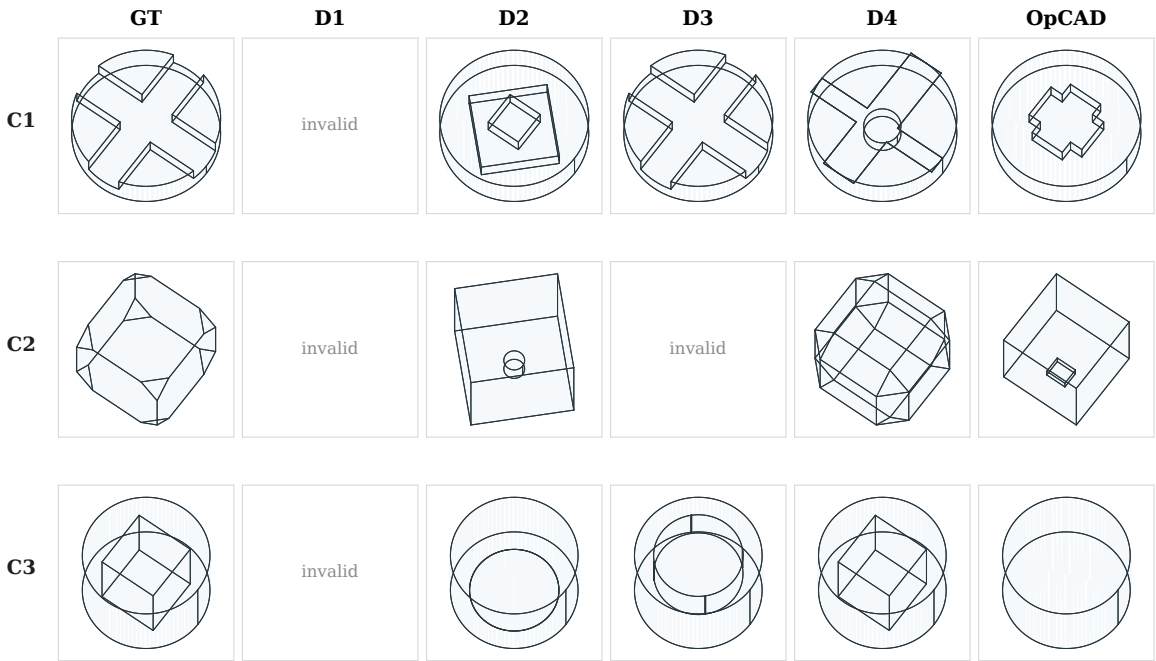

Figure 4: Case-level view of construction stages. C1–C3 show L2 prompts where multiple stages execute but realize different attachments, cutouts, or operation families. The panel is illustrative; quantitative comparisons are reported in Tables 2–4.