Pointer-CAD: Unifying B-Rep and Command Sequences via Pointer-based Edges & Faces Selection

Supplementary Material

In these supplementary materials, we provide the following:

- Additional evaluations and analytical discussions, including comprehensive metrics for text-to-CAD assessment, comparisons on multiple benchmarks, and ablation studies on invalidity ratio;
- Details of the proposed pointer-based representation, including sketch plane selection method, specific vector translation rules, and definitions of geometric special cases in pointer-based referencing;
- Details of the training framework, covering the B-rep encoder, implementation details of the autoregressive decoder, and the training objective;
- Visualization of some dataset cases, the prompts used for annotation, and dataset statistics;
- Implementation Details;

1. Additional Evaluations and Analytical Discussions

1.1. Comprehensive Metrics for Text-to-CAD Evaluation

To provide a more complete assessment of model performance, we further report the Invalidity Ratio (IR), Dangling Edge Length (DangEL), and Self-Intersection Ratio (SIR) following previous works [2, 9]. IR measures generation robustness, while DangEL and SIR quantify complementary aspects of topological soundness. Also, following CADFusion [6] and SkexGen [10], we evaluate our approach using the LVM score, Coverage (COV), Minimum Matching Distance (MMD), and Jensen-Shannon Divergence (JSD).

Clarification on computation of IR:

Text2CAD [3] defines IR as:

$$\text{IR} = \frac{N_{\text{valid}} - N_{\text{post\_build}}}{N_{\text{valid}}}, \tag{1}$$

where N_{valid} is the number of representations that become well-formatted **after post-processing**, and $N_{\text{post\_build}}$ is the subset that can be built into non-zero-volume solids **under the same post-processing pipeline**.

To eliminate the influence of hardcoded post-processing, we adopt another definition of IR across all methods, given by:

$$\text{IR} = \frac{N_{\text{test}} - N_{\text{build}}}{N_{\text{test}}}, \tag{2}$$

where N_{build} denotes the number of generated representations that can be successfully built into non-zero-volume solids **without any post-processing**, and N_{test} is the size of the test set. Under our definition, any output with malformed or incorrect formatting is directly treated as invalid, enforcing a stricter and more realistic failure criterion than the IR used in Text2CAD.

1.2. Additional results on our dataset

Table 1 presents additional evaluation results for various metrics (IR, DangEL, and SIR) on the Recap-DeepCAD dataset. Pointer-CAD consistently outperforms all baselines, achieving the lowest IR and demonstrating superior topological soundness. These improvements show that our pointer mechanism enhances robustness and yields more coherent and structurally consistent CAD constructions.

To further investigate this robustness against error accumulation during sequential generation, Table 2 evaluates single- versus multi-extrusion (≥ 2) models on the Recap-DeepCAD dataset. While baseline methods experience significant performance drops in multi-step scenarios, Pointer-CAD maintains exceptional stability.

Table 3 further evaluates *chamfer* and *fillet* operations. Pointer-CAD is trained on the Recap-OmniCAD$^+$ dataset, whereas baselines that do not support these operations are trained on Recap-OmniCAD for fairness. Within this expanded operation space, Pointer-CAD continues to outperform all baselines, indicating that its modeling accuracy and stability are maintained despite the increased operation diversity.

Finally, Table 4 presents comprehensive generative evaluation metrics, COV, MMD, JSD, and LVM score, across both the Recap-DeepCAD and Recap-OmniCAD$^+$ datasets. Specifically, COV, MMD, and JSD are computed using 1K real and 4K generated samples, averaged over three independent runs. The LVM score is evaluated on the full test set via GPT-4o, utilizing the official evaluation prompt from CADFusion. As demonstrated, our method consistently surpasses all baselines across these metrics on both datasets.

1.3. Results on the Text2CAD Dataset

In addition to the baselines used in the main paper, we also include Text-to-CadQuery [8], which represents CAD models using CadQuery [1] code. We compare our proposed method with existing approaches on the dataset proposed in Text2CAD, whose annotations are parameter-normalized and unit-free, resulting in a simpler annotation format compared with our Recap-DeepCAD and Recap-OmniCAD$^+$

Table 1. **Quantitative comparison on the Recap-DeepCAD dataset across additional metrics.** Pointer-CAD consistently achieves lower invalidity and stronger topological soundness than all baseline methods, demonstrating its superior robustness and reliability in generating coherent CAD constructions.

Model	DeepCAD	Text2CAD	CADmium			Pointer-CAD	
			1.5B	3B	7B	0.5B	1.5B
IR ↓	39.23	30.16	42.84	36.34	31.79	15.02	**8.80**
DangEL ↓	1.80	0.71	3.27	3.88	5.33	0.20	**0.19**
SIR ↓	0.15	0.07	0.10	0.13	0.20	**0.02**	**0.02**

Table 2. **Quantitative error accumulation analysis on Recap-DeepCAD.** We evaluate single- vs. multi-extrusion (≥ 2) models. Our method shows minimal performance drop across steps. Best results are highlighted in red (single) and blue (multi).

Model	DeepCAD		Text2CAD		CADmium				Pointer-CAD			
					1.5B		3B		0.5B		1.5B	
	Single	Multi	Single	Multi	Single	Multi	Single	Multi	Single	Multi	Single	Multi
Line F1 ↑	86.22	63.02	92.54	80.62	92.94	63.51	95.08	56.57	96.54	94.06	**99.34**	**97.65**
Arc F1 ↑	41.74	11.97	56.32	33.21	28.61	9.64	41.73	8.18	70.31	65.91	**96.52**	**93.50**
Circle F1 ↑	86.13	62.70	94.05	78.62	92.75	37.12	93.41	40.48	98.87	96.52	**99.71**	**97.35**
Extrusion F1 ↑	95.34	83.39	99.96	95.90	**100**	68.18	**100**	62.43	99.99	**99.10**	**100**	98.88
CD mean ↓	32.48	46.52	12.82	25.95	7.75	23.53	6.79	24.25	3.80	7.39	**1.70**	**4.35**
CD median ↓	6.92	22.76	1.14	13.24	0.34	12.46	0.26	11.54	0.33	1.76	**0.24**	**0.63**

Table 3. **Quantitative evaluation on the Recap-OmniCAD$^+$ dataset.** Pointer-CAD maintains superior performance over all baselines even under expanded operation diversity.

Model	DeepCAD	Text2CAD	CADmium			Pointer-CAD	
			1.5B	3B	7B	0.5B	1.5B
IR ↓	42.71	33.72	46.65	40.17	34.38	25.37	**19.15**
DangEL ↓	3.34	1.13	3.20	4.32	4.61	0.28	**0.27**
SIR ↓	0.19	0.11	0.12	0.15	0.20	**0.02**	**0.02**

Table 4. **Additional evaluation.** Our method consistently outperforms all baselines across these metrics.

Model	(a) Recap-DeepCAD						(b) Recap-OmniCAD$^+$					
	DeepCAD	Text2CAD	CADmium		Pointer-CAD		DeepCAD	Text2CAD	CADmium		Pointer-CAD	
			1.5B	3B	0.5B	1.5B			1.5B	3B	0.5B	1.5B
COV(%) ↑	71.85	75.90	69.00	70.20	86.97	**89.40**	63.68	72.22	65.70	66.67	**87.57**	87.53
MMD ↓	1.29	0.97	1.20	1.19	0.77	**0.70**	1.48	1.08	1.29	1.16	**0.74**	0.77
JSD (x100) ↓	3.12	3.04	1.45	1.82	0.84	**0.62**	4.03	3.48	1.79	2.21	0.66	**0.65**
LVM Score ↑	5.41	5.82	7.28	7.52	8.46	**8.57**	5.39	5.64	6.98	7.03	**7.99**	7.96

datasets, as shown in Figure 1.

Note that, Text-to-CadQuery constructs its CadQuery dataset sourced from Text2CAD dataset and converts the original Text2CAD models into CadQuery scripts format. However, this conversion procedure introduces non-negligible geometric discrepancies, causing the reconstructed ground truth CadQuery models to deviate from the original Text2CAD geometry. Therefore, for Text-to-CadQuery, we additionally report the Chamfer distance between its predicted models and the converted ground-truth models, as noted in the footnote. As reported in Table 5, Pointer-CAD-1.5B achieves the best IR, sketch operation F1, and CD, while Pointer-CAD-0.5B attains superior extrusion F1, SegE, and DangEL.

Comparison with Text-to-CadQuery. Comparing the results of Pointer-CAD and Text-to-CadQuery, we observe that Text-to-CadQuery achieves lower SIR and FluxEE. This improved performance stems from its use of the mature CadQuery engine and its built-in post-processing, which inherently produces geometrically valid outputs. For instance, the following code directly creates a box without self-intersections while ensuring watertightness.

```
result = Workplane("front").box(
    width, width, thickness
)
```

However, Pointer-CAD achieves smaller SegE and DangEL, even though Text-to-CadQuery relies on a highly engineered drawing pipeline designed to prevent discontinuities and connectivity errors. These results collectively demonstrate that the pointer mechanism in Pointer-CAD produces more coherent and well-structured CAD operations, enabling stronger geometric consistency than both sequence-based and engine-assisted baselines.

1.4. Ablation Study on Invalidity Ratio

To investigate the factors contributing to the observed IR disparity between models trained on the Recap-DeepCAD and Text2CAD datasets, we first compare the annotation pipelines used in each dataset. We note that the main difference lies in the presence of units and whether geometric parameters are normalized. As shown in Figure 1, the annotations in Text2CAD recenter each model at the origin and uniformly scale it to fit within a canonical cube, effectively pre-aligning its position and size to match the final representation, whereas Recap-DeepCAD annotations do not apply any such simplification. We argue that this normalization reduces the difficulty of the task. To validate this, we construct a modified Recap-DeepCAD-Norm dataset by removing all units and normalizing all geometric parameters in the original annotations, following the pipeline used in Text2CAD.

As illustrated in Table 6, all baseline methods trained on these normalized prompts exhibit a substantial reduction in IR, demonstrating that the complex and heterogeneous dimension annotations in Recap-DeepCAD are a primary source of invalid generations. We also note that Text2CAD has a maximum token limit of 512; therefore, for Text2CAD, we specifically filtered out models exceeding this length and then evaluated the remaining IR again. We also observed a reduction in IR after this filtering.

1.5. Quantification of Quantization Error

As discussed in Section 3 of the main paper, command sequences discretize continuous parameters (e.g., coordinates, extrusion distances, angles) into 2^q levels. By default, $q = 8$

Table 5. **Quantitative comparison on Text2CAD datasets.** Pointer-CAD clearly surpasses Text2CAD and CADmium, and remains highly competitive with Text-to-CadQuery despite the latter's reliance on the mature CadQuery engine.

Model	Text2CAD	Text-to-CadQuery-1.5B	CADmium		Pointer-CAD	
			1.5B	3B	0.5B	1.5B
IR ↓	9.58	12.27	9.12	8.51	6.74	**6.59**
Line F1 ↑	83.78	-	79.00	80.30	84.01	**85.77**
Arc F1 ↑	41.54	-	37.72	44.22	37.51	**46.83**
Circle F1 ↑	78.38	-	76.42	78.39	77.93	**78.66**
Extrusion F1 ↑	93.76	-	88.53	88.06	**95.72**	95.50
CD mean ↓	11.78	47.72	11.69	10.83	10.17	**8.57**
CD median ↓	0.33	27.75	0.44	0.37	0.33	**0.30**
SegE ↓	0.41	0.24	0.88	0.91	**0.21**	0.22
DangEL ↓	0.43	0.25	2.10	1.86	**0.16**	0.17
SIR ↓	0.02	**0.01**	0.07	0.06	0.02	0.02
FluxEE ↓	9.40	**0.36**	6.30	4.79	2.56	2.44

* Due to inherent errors in the process by which Text-to-CadQuery constructs its CadQuery dataset, the ground-truth CadQuery models naturally deviate from the original Text2CAD models. When computing the error between the predicted models and these converted ground-truth models, we obtain **a mean Chamfer Distance of 12.79 and a median of 0.32.**

Annotation in Text2CAD Dataset

Create the first part of the CAD model, a cylindrical object with a flat top and bottom, and a slightly tapered middle section. Begin by setting up a new coordinate system with Euler angles of **(0.0, 0.0, 0.0)** and a translation vector of **(0.0, 0.0, 0.0)**. Next, create a sketch on the X-Y plane of the coordinate system. Within the sketch, create a face using a single loop. Within the loop, create a circle with a center at **(0.375, 0.375)** and a **radius of 0.375**. Scale the sketch by a **factor of 0.75** and then extrude the sketch along the Z-axis by a **depth of 0.1046**, creating a new solid body. The resulting part is a cylindrical object with a flat top and bottom, and a slightly tapered middle section. The part has the following dimensions: **length of 0.75, width of 0.75, and height of 0.1046** (including the extrusion depth).

Annotation in Recap-DeepCAD Dataset

To construct the cylindrical part, begin by creating a 2D sketch on the Top plane. In this sketch, draw a circle with its center at the origin **(0, 0)** and a radius of **9.1049cm**. This circle will serve as the base profile for the cylinder.
Next, apply an extrusion operation to the sketch. Extrude the circle along the normal direction by a depth of **2.54cm**. This will create a new solid body in the form of a cylinder with a flat top and bottom.
The resulting cylinder has a diameter of **18.2097cm** and a height of **2.54cm**. The cylinder is symmetrically positioned with its base centered at the origin.

Figure 1. **Prompt comparison.** Recap-DeepCAD dataset includes dimensional values with explicit units, whereas Text2CAD dataset uses normalized, unit-free geometric parameters.

in our experiment. To assess the reconstruction error introduced by different quantization granularities and different representations, we evaluate the median Chamfer Distance between the ground-truth mesh and the mesh reconstructed from its ground-truth command sequence after applying quantization at various bit widths. A lower Chamfer Distance indicates that the representation preserves geometric fidelity more effectively under quantization. As shown in Figure 2, Pointer-CAD consistently exhibits lower quantization error than Text2CAD across all settings. Moreover, as the quantization bit width increases, the gap between the two methods gradually narrows, eventually approaching toward the inherent CD noise floor induced by random point sampling.

Table 6. **Ablation results on the Recap-DeepCAD-Norm dataset.** All baseline methods show a substantial drop in IR, indicating that they depend on memorizing dataset-specific dimensional patterns rather than engaging in genuine geometric reasoning.

Model	Normed	Truncate	IR
Text2CAD	✗	✗	30.16
	✓	✗	15.85
	✗	✓	24.41
Pointer-CAD-0.5B	✗	✗	15.02
	✓	✗	6.13
Pointer-CAD-1.5B	✗	✗	8.80
	✓	✗	5.37

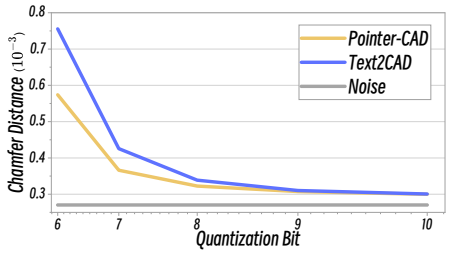


Figure 2. **Quantization Error.** We directly measure quantization error by computing the median Chamfer Distance between each representation before and after quantization, where Pointer-CAD exhibits substantially smaller error than Text2CAD.

1.6. Application of Click Interaction Editing

Since our proposed pointer-based command sequence allows entity selection at each step, we extend the model with token concatenation to incorporate user-interactive selections alongside text instructions, enabling an immersive editing experience. As illustrated in Figure 3, users can interactively select faces or edges on the current B-rep to explicitly specify the operation target, enabling more precise and intuitive editing through direct manipulation in conjunction with text instructions.

2. Details of the Pointer-based Representation

This section elaborates on the implementation logic of the pointer-based representation and the methodology for sketch plane selection.

2.1. Specific Vector Translation Rules

Each token is classified as one of three types: *Label Token*, *Value Token*, or *Pointer*. To simplify the model architecture, we assign non-overlapping integer ranges to label and value tokens, allowing them to be decoded by a single prediction head. However, since a pointer is a reference to a

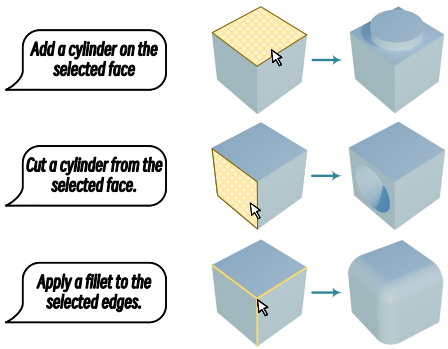


Figure 3. **Illustration of our interactive editing functionality.** Users can directly click on a face or edge of the CAD model and provide a text prompt to specify the desired operation.

geometric entity rather than a simple value, it requires a separate prediction head for decoding. To distinguish pointers from label and value tokens, we reserve two specific integer values within the label/value token space. When the model predicts one of these integers, it signals that the current token is a pointer. These two integers also represent the pointer's state: as shown in Table 7, $<pe>$ indicates an enabled pointer that references an edge or face, whereas $<pd>$ signifies a disabled (inactive) pointer. Specifically, for $<nv>$ and $<ag>$, we first normalize all continuous parameters to the expected range and then quantize into 2^q levels and express them using q-bit integers.

Based on the notation in Table 7, we define the translation rules for each command as shown in Table 8. A CAD model is represented as a sequence of valid sequences, with only the last valid sequence is end with $<em>$.

2.2. Sketch Plane Selection

As defined in the [*Sketch*] notation, a sketch plane is specified by a *Pointer* to a face and a 2D coordinate system [*CS*]. The construction process, illustrated in Figure 4, unfolds in three main steps: First, a base plane is established by selecting a face with the *Pointer*, as shown in Figure 4a. The resulting sketch plane is coplanar with this face. Second, a local coordinate system $U'V'W'$ is constructed on this plane. The normal axis, W', is aligned with the face normal that has a positive dot product with a world axis direction, n, specified by the *Label Token* $<dr>$. The primary in-plane axis, U', is determined by projecting an auxiliary direction, d (listed in Table 9), onto the sketch plane. The second in-plane axis, V', is then derived using the right-hand rule, completing the orthogonal basis $U'V'W'$. As depicted in Figure 4b, the origin of this system is defined by projecting a point P from a world coordinate plane onto the sketch plane along the direction n. Finally, as shown in Figure 4c, the final sketch coordinate system UVW is obtained by applying a counterclockwise in-plane rotation to $U'V'W'$

Table 7. **Special Token Definitions.** This table provides a comprehensive list of all Special Tokens used in our command sequence representation, along with their semantic descriptions.

Notation	ID	Type	Description
<*em*>	1	Label Token	End of model (this step is the final step of the model)
<*es*>	2	Label Token	End of step (additional steps are required after this one)
<*ss*>	3	Label Token	Start of sketch
<*se*>	4	Label Token	Start of extrusion
<*sc*>	5	Label Token	Start of chamfer
<*sf*>	6	Label Token	Start of fillet
<*sp*>	7	Label Token	Start of profile
<*sl*>	8	Label Token	Start of loop
<*sx*>	9	Label Token	Start of curve
<*pe*>	10	Pointer	Pointer to an Edge or Face
<*pd*>	11	Pointer	Empty pointer
<*or*>	$\{12, 13\}$	Label Token	Orientation (Clockwise, Counter-Clockwise)
<*dr*>	$[14, 19]$	Label Token	Direction (X+, X-, Y+, Y-, Z+, Z-)
<*bo*>	$[20, 23]$	Label Token	Boolean (New, Join, Cut, Intersect)
<*nv*>	$[24, 24 + 2^q)$	Value Token	Normalized to $[0, 1]$ and then quantize to 2^q level
<*ag*>	$[24, 24 + 2^q)$	Value Token	Normalized to angle $[0°, 360°)$ and then quantize to 2^q level

Table 8. **Sequence definitions.** Each sequence is defined by a specific combination of commands. The superscript [$^+$] denotes that the element appears one or more times, while the symbol [/] indicates that one of the alternatives can be chosen.

Notation	Sequence	Description
[*P*]	<*nv*> <*nv*> <*pe* / *pd*>	2D point (x, y), snapped to reference or placed freely
[*L*]	<*sx*> [*P*]	Line starting from a 2D point (x, y)
[*C*]	<*sx*> [*P*] <*nv*>	Circle with center at (x, y) and radius r
[*A*]	<*sx*> [*P*] <*ag*> <*or*>	Arc starting from (x, y) with angle α and orientation
[*Loop*]	<*sl*> [*L* / *C* / *A*]$^+$	Closed loop composed of multiple curves
[*Profile*]	<*sp*> [*Loop*]$^+$	2D Region defined by one or more loops
[*CS*]	<*dr*> [*P*] <*ag*> <*nv*>	2D coordinate system in 3D space
[*Sketch*]	<*ss*> <*pe*> [*CS*] [*Profile*]$^+$	Sketch on a plane specified by pointer
[*Extrude*]	<*se*> <*nv*> <*nv*> <*bo*>	Extrude operation with depth and Boolean type
[*EPart*]	[*Sketch*]$^+$ [*Extrude*]	Solid part constructed by extrusion
[*Chamfer*]	<*sc*> <*nv*> <*pe*>$^+$	Chamfer operation on referenced edges
[*Fillet*]	<*sf*> <*nv*> <*pe*>$^+$	Fillet operation on referenced edges
[*VSeq*]	**[*EPart* / *Chamfer* / *Fillet*] <*es* / *em*>**	**Valid sequence**

about the W-axis. An optional scaling factor may also be applied to mitigate quantization errors.

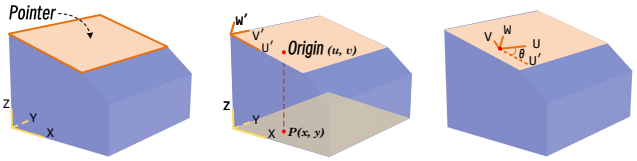
2.3. Geometric Special Cases in Pointer Referencing

While a pointer is generally intended to reference a single, unique geometric entity (i.e., an edge or a face), this one-to-one correspondence breaks down in certain "geometric special cases." These cases occur when multiple entities are geometrically equivalent from a modeling standpoint, such as coplanar faces or collinear edges. In such scenarios, selecting any one of these equivalent entities would result in the same final geometry. Therefore, the ground truth for a pointer is not a single object but rather a set of valid candidates. This section provides precise definitions for these geometric special cases.

Coplanar-Adjacent Faces. A face pointer selects a base face to define a sketch plane. If two or more faces are coplanar (i.e., they lie on the same geometric plane), selecting any of them will result in the same sketch plane definition.

Table 9. **Direction mapping.** In command <*dr*>, each symbol corresponds to a primary direction and its auxiliary direction.

Symbol	Direction	Auxiliary Direction
14	X+	Y+
15	X-	Z+
16	Y+	Z+
17	Y-	X+
18	Z+	X+
19	Z-	Y+



(a) Face selection. (b) Origin definition. (c) Rotation definition.

Figure 4. **Sketch coordinate system construction.** The sketch plane, axes, origin, and rotation are defined step by step to form the local coordinate system UVW.

Therefore, all faces within such a coplanar group are considered valid candidates for the face pointer.

Collinear-Connected Edges. Snapping a sketch point to an existing edge requires an edge pointer. If other edges are collinear with the target edge, pointing to any of them will produce the same snapping result. Therefore, all edges within such a collinear group are considered valid candidates for the edge pointer.

2.4. Pointer Failure Scenarios.

The pointer mechanism successfully resolves geometric references in the vast majority of standard modeling scenarios. Failures are largely confined to extreme cases. Primarily, errors arise from inherent limitations within the B-Rep graph when processing non-manifold topologies—specifically, edges bounded by more than two faces. Although such configurations are typically excluded under standard CAD modeling conventions, their presence introduces significant ambiguity in face selection for operations like chamfer and fillet. As illustrated in Figure 5, encountering a non-manifold edge leads to multiple valid interpretations of a fillet operation, thereby confounding the pointer mechanism.

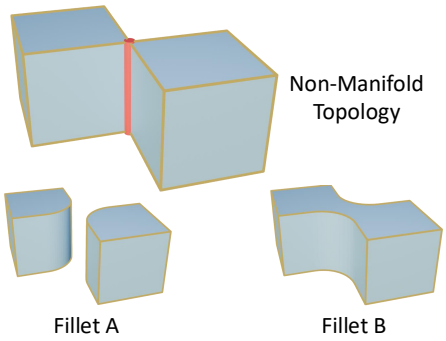


Figure 5. A non-manifold topology leads to multiple valid interpretations of a fillet operation.

3. Details of the training framework.

3.1. B-rep encoder

For each B-rep edge, we uniformly sample 32 points along its parametric curve in 3D space and extract four quantities at each location: point coordinates, tangent and its reverse vector, and first-order derivatives. Each is represented as a 3D vector, and their concatenation yields a 12-dimensional feature per sample. Collecting all samples forms an edge feature tensor of shape 32×12, which serves as input for edge embedding.

For each B-rep face, we uniformly sample its parametric (u, v) domain to construct a regular UV grid of size 32×32. At each grid point, we compute the 3D coordinates, unit surface normal, Gaussian curvature, and a binary visibility mask (set to 1 for interior or boundary samples and 0 otherwise). Concatenating these quantities channel-wise gives an 8-dimensional feature per location, i.e., $3 + 3 + 1 + 1$, producing a face tensor of shape $32 \times 32 \times 8$.

For GNN node embeddings (the B-rep face embeddings), we first apply a 2D convolution to the face tensor to expand it to 256 channels, followed by global adaptive average pooling and a linear projection to a 128-dimensional vector, denoted $h_i^{(0)}$. Similarly, edge embeddings are obtained by applying 1D convolutions to the edge tensor, expanding it to 256 channels, followed by global adaptive average pooling and a linear projection to a 128-dimensional vector, denoted $h_{ij}^{(0)}$. Thus, the graph $\mathcal{G}$ is initialized with node features $h_i^{(0)}$ and edge features $h_{ij}^{(0)}$ for downstream processing.

3.2. Implementation Details of the Autoregressive Decoder

To translate the output of the LLM into our defined command sequence, we process the last hidden state from the model's transformer decoder at each autoregressive decoding step. We employ a dual-head architecture to decode the hidden state into the appropriate token type.

The first head, which we refer to as the **Label/Value Head**, is a linear layer responsible for predicting both *Label Tokens* and *Value Tokens*. Its output comprises three parts, consistent with the tokenization scheme introduced earlier:

- A component corresponding to one of the Label Tokens defined in Table 7.
- A component selecting from two special tokens, <*pe*> and <*pd*>, which indicate the pointer state. When the model predicts the <*pe*>, it indicates that the current token is a pointer, and the output from the second head should be used.
- A component that corresponds to the quantized bins used for continuous Value Tokens, including those for <*nv*> or <*ag*>.

The second head, the **Pointer Head**, is another linear layer specifically designed for decoding pointers. This head's output is a 128-dimensional vector. When the Label/Value Head predicts an active pointer state, this 128-dimensional vector is used to perform a similarity search (via cosine similarity) against the 128-dimensional embeddings of all candidate geometric entities (faces and edges) generated by the B-rep encoder. The entity with the highest similarity score is selected as the pointer's reference. This mechanism allows the model to dynamically ground its generation in the existing B-rep geometry.

3.3. Details of Training Objective

Pointer Prediction. Following CLIP [5], we employ a learnable temperature parameter τ to control the scale of the logits in the loss computation. The parameter is initialized to 0.07, following [7]. To improve training stability, we reparameterize τ as its reciprocal $s = 1/\tau$ and optimize $\log s$ during training, with s clipped to $s \leq 100$ to avoid excessive scaling of the logits. The learning rate for s is set to $lr_s = 0.1 \times lr$, and weight decay is not applied during its optimization.

Overall Objective. The overall training objective $\mathcal{L}$ combines the cross-entropy loss for label/value tokens ($\mathcal{L}_v$) and the contrastive loss for pointer tokens ($\mathcal{L}_p$). The final loss is a weighted sum of these two components, controlled by hyperparameters λ_v and λ_p. In all our experiments, we set $\lambda_v = 0.5$ and $\lambda_p = 0.5$ to give them equal weight.

4. Details of the Dataset

4.1. Dataset Visualization

Figure 6 presents several representative samples from the Recap-OmniCAD$^+$ dataset, showcasing a wide spectrum of model complexity and diversity. As illustrated, our dataset contains a rich variety of models that not only feature complex geometric details such as fillets and chamfers but also exhibit diverse topological structures like holes, pockets, and multi-body components.

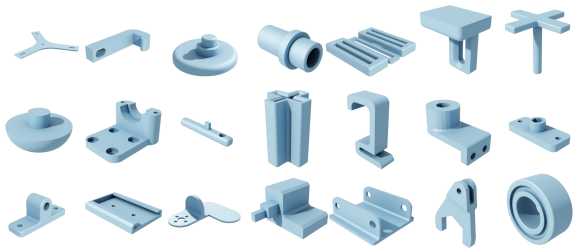


Figure 6. **Representative samples from the Recap-OmniCAD$^+$ dataset.** The figure displays a range of models with varying complexity, from simpler parts with basic features to intricate components incorporating numerous fillets, chamfers, and complex sketches.

4.2. Details of Annotation Prompts

In our dataset construction process, we employ a multi-step approach to generate rich and detailed annotations for each CAD model. First, we utilize the Qwen2.5-vl-72B model to generate a visual description of the model's appearance, using the prompt shown in Figure 9. Next, we use the same model to describe the relative position of the sketch plane within the model, guided by the prompt in Figure 10. To ensure a clear and accurate understanding for the model, we dynamically replace the placeholders in the prompt with the actual sketch plane surface normal vector and facing direction for each CAD model.

The resulting annotations are then combined with modeling parameters extracted from the raw JSON file to create a structured "minimal JSON," as illustrated in Figure 11. This minimal JSON, along with the prompt shown in Figure 12, is then passed to Qwen2.5-72B-Instruct to generate a final natural language description of the modeling process.

4.3. Dataset Statistics

We provide a statistical analysis of our dataset in Figure 7 and Figure 8.

Figure 7 illustrates the distribution of modeling operations. Notably, Recap-OmniCAD$^+$ includes *chamfer* and *fillet* operations, which are absent in the original OmniCAD. The reintegration of these features results in a higher count for all operation types in Recap-OmniCAD$^+$ compared to OmniCAD.

In Pointer-CAD, the command sequence of a complete CAD model is decomposed into three types of operations: sketch–extrude combinations, chamfers, and fillets. Figure 8 presents the statistics of our dataset according to this decomposition. The inclusion of *chamfer* and *fillet* operations increases the overall complexity and the average number of steps required to construct a model. This is reflected in the

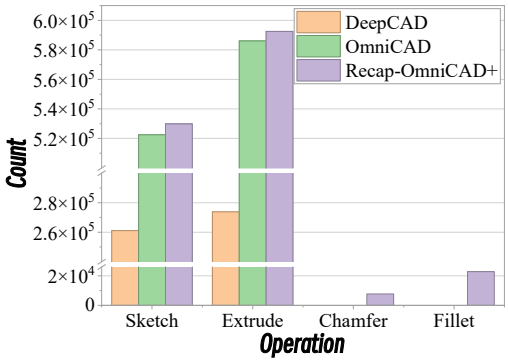


Figure 7. **Distribution of modeling operations across datasets.** The figure illustrates the total count of each modeling operation type for the DeepCAD, OmniCAD, and Recap-OmniCAD$^+$ datasets.

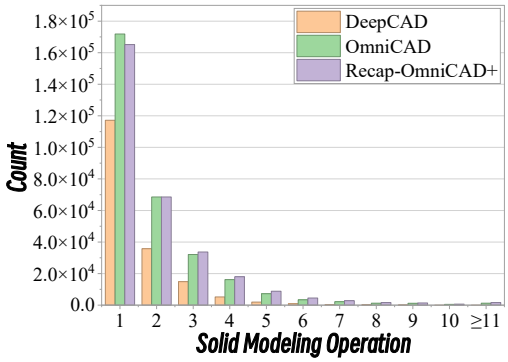


Figure 8. **Distribution of modeling steps per model.** The figure compares the number of solid modeling operations required per model across the datasets.

distribution, where Recap-OmniCAD$^+$ has a slightly lower count of models with a single operation but a consistently higher count for models requiring more than one operation compared to OmniCAD.

Furthermore, both figures highlight that OmniCAD and Recap-OmniCAD$^+$ are significantly more complex than DeepCAD. They feature a greater total number of operations and a higher proportion of models requiring a large number of construction steps. This demonstrates that our datasets are more challenging and better reflect the complexity of real-world CAD modeling tasks.

5. Implementation Details

For the default 0.5B model setting, the entire training process requires approximately 23 hours on 16 NVIDIA H800 GPUs. We use the AdamW optimizer [4] with a learning rate of 1×10^{-4} and a linear decay schedule. For LoRA, the dropout rate is set to 0.1. We use a micro-batch size of 9 with 2 gradient accumulation steps per GPU. The maximum sequence length is 3,072 tokens.

References

[1] CadQuery Contributors. CadQuery, 2025. 1

[2] Prashant Govindarajan, Davide Baldelli, Jay Pathak, Quentin Fournier, and Sarath Chandar. Cadmium: Fine-tuning code language models for text-driven sequential cad design. *arXiv preprint arXiv:2507.09792*, 2025. 1

[3] Mohammad Sadil Khan, Sankalp Sinha, Talha Uddin, Didier Stricker, Sk Aziz Ali, and Muhammad Zeshan Afzal. Text2cad: Generating sequential cad designs from beginner-to-expert level text prompts. *NeurIPS*, 37:7552–7579, 2024. 1

[4] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 8

[5] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, pages 8748–8763. PmLR, 2021. 7

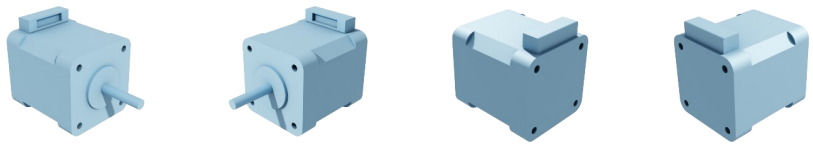
[6] Ruiyu Wang, Yu Yuan, Shizhao Sun, and Jiang Bian. Text-to-cad generation through infusing visual feedback in large language models. In *ICML*, 2025. 1

[7] Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *CVPR*, pages 3733–3742, 2018. 7

[8] Haoyang Xie and Feng Ju. Text-to-cadquery: A new paradigm for cad generation with scalable large model capabilities. *arXiv preprint arXiv:2505.06507*, 2025. 1

[9] Jingwei Xu, Chenyu Wang, Zibo Zhao, Wen Liu, Yi Ma, and Shenghua Gao. Cad-mllm: Unifying multimodality-conditioned cad generation with mllm. *arXiv preprint arXiv:2411.04954*, 2024. 1

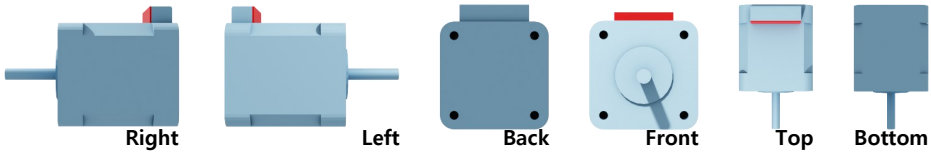
[10] Xiang Xu, Karl DD Willis, Joseph G Lambourne, Chin-Yi Cheng, Pradeep Kumar Jayaraman, and Yasutaka Furukawa. Skexgen: Autoregressive generation of cad construction sequences with disentangled codebooks. *arXiv preprint arXiv:2207.04632*, 2022. 1



You are a senior CAD engineer. I will provide you with **four images** of a 3D model. Your task is to:

1. Generate a **one-word name** for the object, enclosed in <name></name>.
2. Write a **clear and concise one-sentence caption** for the object, enclosed in <caption></caption>, summarizing its overall shape and key structural features. Focus on geometric form, symmetry, major extrusions or cutouts, and distinctive elements. Avoid interpretation or unnecessary detail.

Figure 9. **Prompt for visual description.** This prompt is used with the Qwen2.5-vl-72B model to generate a description of the CAD model's visual appearance.



You are given six orthographic views of the same 3D object in the following fixed order (each image also has a label at the bottom-right corner indicating its view):

1. Right view
2. Left view
3. Back view
4. Front view
5. Top view
6. Bottom view

A **red** planar surface on a light blue object is highlighted.

Red planar surface normal vector (numeric):
(nx, ny, nz) = (REPLACE_NX, REPLACE_NY, REPLACE_NZ)

Authoritative facing-direction hints (precomputed; use exactly as given):
- current X direction: REPLACE_X_DIR (one of: right / left / none)
- current Y direction: REPLACE_Y_DIR (one of: back / front / none)
- current Z direction: REPLACE_Z_DIR (one of: up / down / none)

Coordinate system (right-handed):
- X-axis: left (−) to right (+)
- Y-axis: front (−) to back (+)
- Z-axis: bottom (−) to top (+)

View-to-object 2D mapping rules:
- Right view: left side of image = front, right side of image = back.
- Left view: left side of image = back, right side of image = front.
- Back view: left side of image = right, right side of image = left.
- Front view: left side of image = left, right side of image = right.
- Top view: bottom of image = front, top of image = back.
- Bottom view: top of image = front, bottom of image = back.

.....

Figure 10. **Prompt for sketch plane description.** This prompt guides the model to describe the relative position of the sketch plane, with placeholders for the normal vector and facing direction being dynamically replaced.

```
{
  "parts": {
    "part_1": {
      "sketches": {
        "sketch_1": {
          "name": "Cylinder 1",
          "profile_1": {
            "loop_1": {
              "curve_1": {
                "type": "Circle",
                "center": [0.0, 0.0],
                "radius": 0.5
              }
            }
          },
          "coordinate_system": {
            "reference_plane": "Top",
            "rotation": [0.0, 0.0, 0.0],
            "position": [0.0, 0.0, 0.0]
          }
        }
      },
      "extrusion": {
        "name": "Extrude 1",
        "extrude_operation_type": "NewBodyFeatureOperation",
        "extrude_extent_mode": "OneSideFeatureExtentType",
        "extrude_depth_towards_normal": 0.5,
        "extrude_depth_opposite_normal": 0.0
      },
      "description": {
        "label": "Cylinder",
        "caption": "A cylinder with a height equal to
                    half of its diameter.",
        "length": 1.0,
        "width": 1.0,
        "height": 0.5
      }
    },
    "part_2": {
      "sketches": {
        "sketch_1": {
          "name": "Cylinder 2",
          "profile_1": {
            "loop_1": {
              "curve_1": {
                "type": "Circle",
                "center": [0.0, 0.0],
                "radius": 0.25
              }
            }
          },
          "coordinate_system": {
            "description": "On the top face of the cylinder.",
            "reference_plane": "Top",
            "rotation": [0.0, 0.0, 0.0],
            "position": [0.0, 0.0, 0.5]
          }
        }
      },
      "extrusion": {
        "name": "Extrude 2",
        "extrude_operation_type": "NewBodyFeatureOperation",
        "extrude_extent_mode": "OneSideFeatureExtentType",
        "extrude_depth_towards_normal": 0.1,
        "extrude_depth_opposite_normal": 0.0
      },
      "description": {
        "label": "Cylinder",
        "caption": "A very flat cylinder.",
        "length": 0.5,
        "width": 0.5,
        "height": 0.1
      }
    },
  },
  "dimensions": {
    "x_length": 1.0,
    "y_length": 1.0,
    "z_length": 0.6
  }
}
```


Minimal Json


```
{
  "parts": {
    "part_1": {
      "sketches": {
        "sketch_1": {
          "name": "Cylinder 1",
          "profile_1": {
            "loop_1": {
              "curve_1": {
                "type": "Circle",
                "center": [0.0, 0.0],
                "radius": 0.5
              }
            }
          },
          "coordinate_system": {
            "reference_plane": "Top",
            "rotation": [0.0, 0.0, 0.0],
            "position": [0.0, 0.0, 0.0]
          }
        }
      },
      "extrusion": {
        "name": "Extrude 1",
        "extrude_operation_type": "NewBodyFeatureOperation",
        "extrude_extent_mode": "OneSideFeatureExtentType",
        "extrude_depth_towards_normal": 0.5,
        "extrude_depth_opposite_normal": 0.0
      },
      "description": {
        "label": "Cylinder",
        "caption": "A cylinder with a height equal to
                    half of its diameter.",
        "length": 1.0,
        "width": 1.0,
        "height": 0.5
      }
    },
    "part_2": {
      "fillet": {
        "name": "Top Fillet",
        "fillet_edges": {
          "edge_1": {
            "type": "3D Circle",
            "center": [0.0, 0.0, 0.5],
            "via": [1.0, 0.0, 0.5]
          }
        },
        "fillet_radius": 0.1,
        "fillet_tangent_chain": true
      },
      "description": {
        "label": "Cylinder",
        "caption": "A cylinder with a fillet along its top
                    edge."
      }
    },
  },
  "dimensions": {
    "x_length": 1.0,
    "y_length": 1.0,
    "z_length": 0.5
  }
}
```


Minimal Json

Figure 11. **Examples of the minimal JSON structure.** This figure illustrates two structured 'minimal JSONs' format, which integrates visual annotations and key modeling parameters for the language model.

Annotation Prompt

You are a senior CAD engineer. Your task is to read a CAD model construction process described in JSON format and generate clear, natural language instructions that a junior designer can follow to build the model step by step.

Each part in the JSON is constructed independently and may include:

1. One or more 2D sketches. Each sketch may contain lines, arcs, and circles:
(i) A line is defined by a start point and an end point;
(ii) An arc is defined by a center point, a start point, a sweep angle, and a direction;
(iii) A circle is defined by a center point and a radius.

2. A coordinate system that positions the sketch in 3D space using:
(i) A sketch plane (Top, Right, or Front), which defines the basis for the coordinate system;
(ii) Rotation angles following Z-Y-X order (first rotate around Z, then Y, then X);
(iii) Translation along the x, y, and z directions;
(iv) Optionally, a description field may be present, giving a high-level spatial context of the coordinate system's placement in 3D space.

3. An extrusion operation applied to the sketch or sketches. It includes:
(i) extrude_operation_type, which defines how the extrusion modifies geometry. This may involve adding material, cutting, intersecting, or creating new bodies or components;
(ii) extrude_extent_mode, which defines how far and in which direction the sketch is extruded. Interpret and explain this mode naturally based on the data.

4. A fillet operation, defined by:
(i) fillet_radius, which specifies the radius of the fillet;
(ii) fillet_tangent_chain, which indicates whether the fillet continues smoothly along tangent edges;
(iii) fillet_edges, which specifies the edges to which the fillet is applied.

5. A chamfer operation, defined by:
(i) chamfer_distance, which specifies the distance of the chamfer;
(ii) chamfer_tangent_chain, which indicates whether the chamfer continues smoothly along tangent edges;
(iii) chamfer_edges, which specifies the edges to which the chamfer is applied.

Instructions must follow these rules:

1. Explicitly mention each sketch, even if it has only one shape;
2. Omit unimportant fields like sketch name or body/component name;
3. Wrap all numeric values (coordinates, radii, lengths, distances) in <v></v>;
4. Do not wrap angles, including rotation and sweep_angle;
5. Do not add units; ¡v¿¡/v¿ implies units;
6. Ignore rotations/translations that are all zero and omit zero-valued axes;
7. Use concise, natural engineering language;
8. You may use paragraphs for readability but avoid lists or step numbers;
9. Ignore captions/visualizations that conflict with construction steps;
10. If a coordinate system has a description, integrate it naturally for high-level context.

You will receive only the JSON content in the prompt. Interpret and process it directly.

Figure 12. **Prompt for generating the final natural language description.** This prompt is used with the 'minimal JSON' to generate the final natural language description of the modeling process.