BrepGiff: Lightweight Generation of Complex B-rep with 3D GAT Diffusion

Hao Guo[1*] guoh0215@mail.nwpu.edu.cn
Xiaoshui Huang[2*] huangxiaoshui@sjtu.edu.cn
Hao jiacheng[3*] haojiacheng@email.ncu.edu.cn
Yunpeng Bai[4] bai_yunpeng99@u.nus.edu
Hongping Gan[1] ganhongping@nwpu.edu.cn
Yilei Shi[1 †] yilei_shi@nwpu.edu.cn

[1]Northwest Polytechnical University
[2]Shanghai Jiaotong University
[3]Nanchang University
[4]National University of Singapore

Abstract

Despite advancements in Computer-Aided-Design (CAD) generation, direct generation of complex Boundary Representation (B-rep) CAD models remains challenging. The difficulty arises from the parametric nature of B-rep data, complicating the encoding and generation of its geometric and topological information. In this paper, we introduce BrepGiff, a lightweight generation approach for high-quality and complex B-rep based on 3D Graph Diffusion. First, we transfer B-rep models into 3D graphs representation. Specifically, BrepGiff extracts and integrates topological and geometric features to construct a 3D graph where nodes correspond to face centroids in 3D space, preserving adjacency and degree information. Geometric features are derived by sampling points in the UV domain and extracting face and edge features. BrepGiff then applies Graph Attention Network (GAT) to enforce topological constraints from local to global during the degree-guided diffusion process. With the 3D graph representation and diffusion process, BrepGiff significantly reduces the computational cost and improves the quality, thus achieving lightweight generation of complex models. Experiments show that BrepGiff can generate complex B-rep models (>100 faces) using only 2 RTX4090 GPUs, achieving state-of-the-art performance in B-rep generation.

1. Introduction

Computer-Aided Design (CAD) models are indispensable for creating nearly all tangible industrial products, but designing CAD is labor-intensive and time-consuming, resulting extensive research into automating the CAD generation process. Existing methods often rely on sequences of commands [19, 28, 29] or Constructive Solid Geometry (CSG) representations [27, 30, 42]. However, these formats have limitations in fully capturing the detailed geometric and topological information required for complex designs.

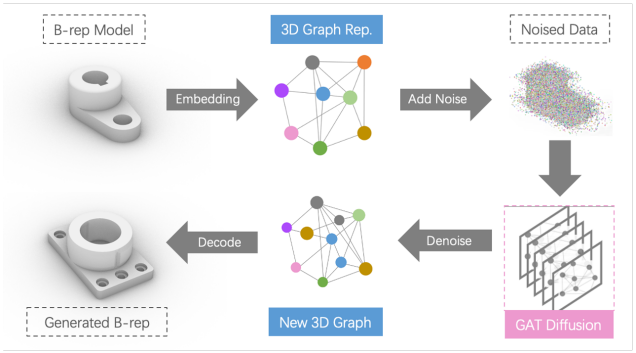


Figure 1. BrepGiff enables lightweight generation of complex B-rep models by embedding B-rep models into 3D graphs and designing the diffusion process using Graph Attention Network.

Boundary Representation (B-rep) emerges as a superior format because it describes 3D models using parameterized faces, edges, and vertices, encompassing both geometric and topological information crucial for understanding the model [31]. Despite its advantages, generating B-rep data directly poses significant challenges due to the difficulties in extracting and integrating topological and geometric features properly, as well as encoding parameterized models into a learnable format. Some researchers explored step-by-step approaches that segment point clouds or distance fields to generate B-rep models [11, 22, 23]. As the first direct generation method, BrepGen [39] creates the entire B-rep model in a single process, maintaining model details with higher flexibility and efficiency. However, while it enhanced the quality of B-rep data generation, this method used 2D tree to embed B-rep data, and primarily focused on

* Equal contribution
†Corresponding author

the generation of CAD data within 50 faces or fewer. There remains a lack of effective solutions for generating more complex models.

In this paper, we introduce **BrepGiff**, a lightweight approach that directly generates high-quality complex CAD models in B-rep format. Given the inherent 3D nature and the explicit *face-edge-node* relationships in 3D CAD models, we represent B-rep as 3D graphs. Each graph node corresponds to a face of the B-rep model, characterized by its centroid in the UV domain projected back to its actual coordinate. Each graph edge represents the edge of B-rep model, depicting the topological feature between faces. We directly treat the vertex of B-rep as feature of edges, thus reducing the complexity of the representation. Besides, to optimize the graph for training, we use the Minimum Spanning Tree (MST) to build the final centered 3D graph. As a result, we reduce computational overhead and allows for the generation of more complex CAD model.

We employ a degree-guided 3D Graph Attention Network (GAT) diffusion model for generation. During the forward noising process, we add noise to 3D graph edges by selectively removing some, guided by node degrees in the graph. For nodes, we add noise to their features. Our denoising process employs a GAT model to predict edges, utilizing a local-to-global generation strategy to maintain topological constraints while restoring complex B-rep data. In addition, we generate edges for predicted active nodes, which reduces the number of nodes to process and improves computational efficiency. Throughout the process, degree guidance and topological constraints ensure that the generated edges match the characteristics of the original B-rep model. Finally, BrepGiff uses a decoder to reconstruct the B-rep model from the 3D graph. Experiments demonstrate that BrepGiff can generate complex B-rep models with over 100 faces, maintaining high quality and validity while being lightweight and trainable with significantly fewer GPU resources. Overall, our contributions include:

- A novel lightweight 3D graph-based representation of B-rep that effectively integrates both topological and geometric features into a unified framework.
- A B-rep degree-guided diffusion process utilizing a 3D Graph Attention Network (GAT) with cross attention, enabling efficient training and high-quality generation of complex B-rep models.
- Evaluations showing that BrepGiff achieves state-of-the-art performance on B-rep generation.

2. Related Works

2.1. CAD Generation

CAD model generation has attracted increasing attention in recent years, particularly generating from point clouds. Several algorithms have been developed to address this task [2, 7, 9, 40]. For example, Point2Cyl [2] segments point clouds into extrusion regions and uses predicted segmentation and surface normals to infer 2D sketches and extrusion parameters. However, it only captures the final CAD model, missing intermediate design steps. To overcome this, Constructive Solid Geometry (CSG) has been used to construct CAD models by combining basic shapes through Boolean operations [27, 30, 41, 42]. Although the CSG method is feasible, it is often overly complex for typical industrial applications. In contrast, most CAD software relies on feature-based modeling, which represents shapes as a sequence of commands (e.g., sketching and extrusions). [34]. Recent deep learning advances have enabled CAD generation using these sketch-extrusion approaches [4, 14, 19, 21, 29, 37, 38], including methods such as ExtrudeNet [28], SECAD-Net [20], and CAD-SIGnet [29].

However, most studies focus on generating CAD command sequences, which offer limited topological and geometric information compared to B-rep. Few efforts have addressed B-rep generation with limited performance.

2.2. B-rep Generation

A B-rep consists of geometric primitives like parametric curves and surfaces, as well as topological elements such as vertices, edges, and faces. Researchers have explored B-rep segmentation and classification, with notable contributions including CADOPs-Net [8], Hierarchical-CADNet [6], and frameworks like BRepNetAT, GraphRO, and UVNet [3, 15, 18]. In terms of B-rep generation, one common approach segments point clouds before constructing the B-rep, as seen in methods like ComplexGen [11], Point2CAD [22], and Split-and-Fit [23]. Recent approaches generate B-rep without intermediate segmentation. For example, Solidgen [16] uses Transformers and pointer networks to generate B-rep without CAD modeling sequence supervision, while CADParser [44] constructs a graph for each input B-rep CAD model for direct generation via Transformers. Closely related to our work is BrepGen [39], which utilizes a latent diffusion module for B-rep generation. However, generating high-quality and complex B-rep data remains challenging. Our work seeks to address these limitations, achieving more efficient and precise B-rep generation.

2.3. 3D Generation with Diffusion Models

Diffusion models are an advanced generation method that involves transforming original data into Gaussian white noise by progressively adding random noise and then reconstructing the data by gradually eliminating this noise [13]. In 3D generation, diffusion models have been applied to diverse tasks, including point cloud-to-esh conversion[12], 3D reconstruction from images [24, 26, 36], and CAD model generation and retrieval. 3DShape2VecSet [43] proposes a latent set diffusion framework for 3D shape diffu-

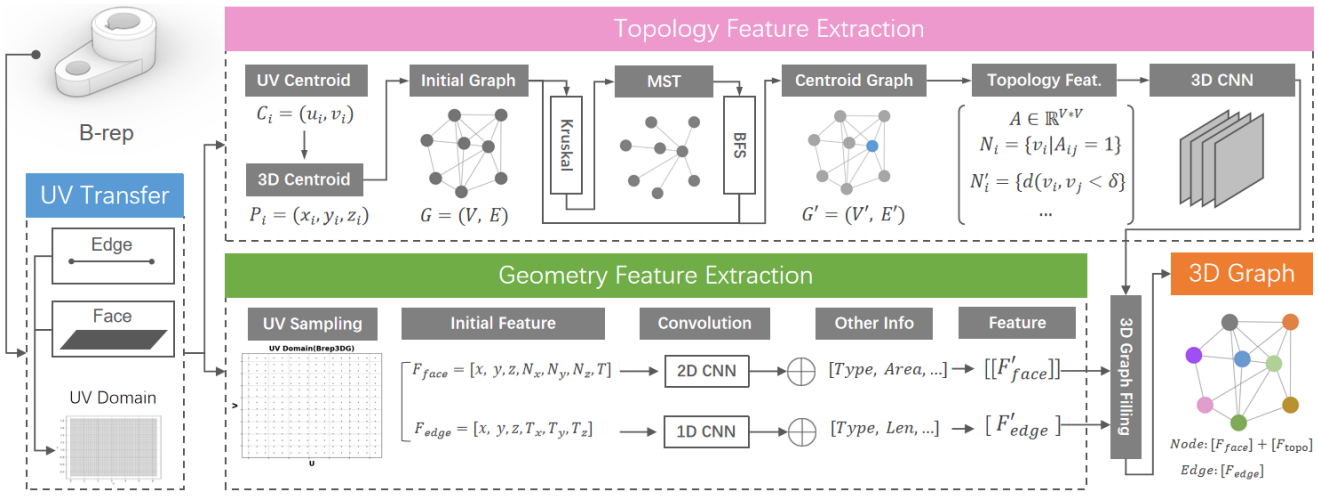


Figure 2. BrepGiff encodes topological and geometric feature of B-rep data into a 3D graph. Topology is used for construct the 3D graph structure. Geometry is used for extract attributes of the graph.

sion generation and point cloud completion. DiffCAD [10] uses diffusion to learn a probabilistic model for for CAD generation and alignment. GetMesh [25] employs a latent set diffusion model to generate meshes from point clouds. Polydiff [1] applies a discrete denoising diffusion model to mesh data, generating 3D triangular networks.

These advancements indicate that diffusion models hold great potential in handling complex 3D structures like B-rep models, promoting more refined and accurate 3D model generation. Our research integrates Graph Attention Networks (GAT) [33] with diffusion models to address the limitations of current B-rep generation methods, with the goal of enhancing efficiency and quality in the creation and editing of B-rep models.

3. 3D Graph Based Encoding

B-rep has two main features: **Topology** and **Geometry**. This poses two challenges when encoding B-rep into a training-friendly representation: 1) how to extract geometry and topology features respectively; 2) how to integrate these two features. To address these issues, BrepGiff embeds B-rep models into 3D graphs with the topology of the B-rep models, and assign the extracted geometry feature as the attributes of the graph, integrating geometric and topological features (Fig. 2). This 3D Graph-based encoding method simplifies complex CAD models and enhances the lightweight generation.

3.1. Topology Encoding

Here, we describe our method to establish a 3D graph structure with the topology information of B-rep model, as well as extracting the topological feature.

3D Graph Construction. To build the 3D graph, we first calculate the centroid of a B-rep's each face in UV domain. For a model with N faces, the centroid in UV domain for each face is denoted as $C_i = (u_i, v_i)$, where $i = 1, 2, \ldots, N$. We then map these centroids back to the CAD coordinate, and obtain the actual centroid coordinates of each face, represented as $P_i = (x_i, y_i, z_i)$. Based on this, an initial 3D graph $G = (V, E)$ is constructed, where V represents the set of vertices with each vertex v_i in V, corresponding to a face centroid $\mathbf{P}_i$. E represents the set of edges, with each edge e_{ij} in E, indicating the connection between two face centroids $\mathbf{P}_i$ and $\mathbf{P}_j$. Then, we use Kruskal's algorithm to construct a Minimum Spanning Tree (MST), ensuring the graph is acyclic while all vertices remain connected. This is to simplify the graph's complexity for training while preserving its fundamental connectivity. Finally, we use Breadth-First Search (BFS) to search from any vertex across the entire MST to identify the face with the highest connectivity as the central face, leading to the construction of an optimized 3D graph $G' = (V', E')$.

Topology Feature Extraction. To extract the topology features of the 3D graph G', following information must be extracted and combined to form a vector $\mathbf{X} = [A, N, C, D]$. $A \in \mathbb{R}^{V \times V}$ is adjacency matrix, where $A_{ij} = 1$ if there exists an edge e_{ij} connecting vertices v_i and v_j, otherwise $A_{ij} = 0$. $N = [\mathbf{n}_1, \mathbf{n}_2, \ldots, \mathbf{n}_V]^T$ denotes the matrix of neighbor information for all vertices. The vector $\mathbf{n}_i \in \mathbb{R}^k$, where k is the dimension of neighbor information, is calculated by concatenating direct neighbor sets $N_i = \{v_j \mid A_{ij} = 1\}$ and potential neighbor sets $N_i' = \{v_j \mid d(v_i, v_j) < \delta\}$. $d(v_i, v_j)$ denotes the distance between vertices v_i and v_j, and δ is a predefined distance threshold. $C = [L, \chi]^T$ is a 2-dimensional vector calculated by concatenating The number of loops L and Euler characteristic

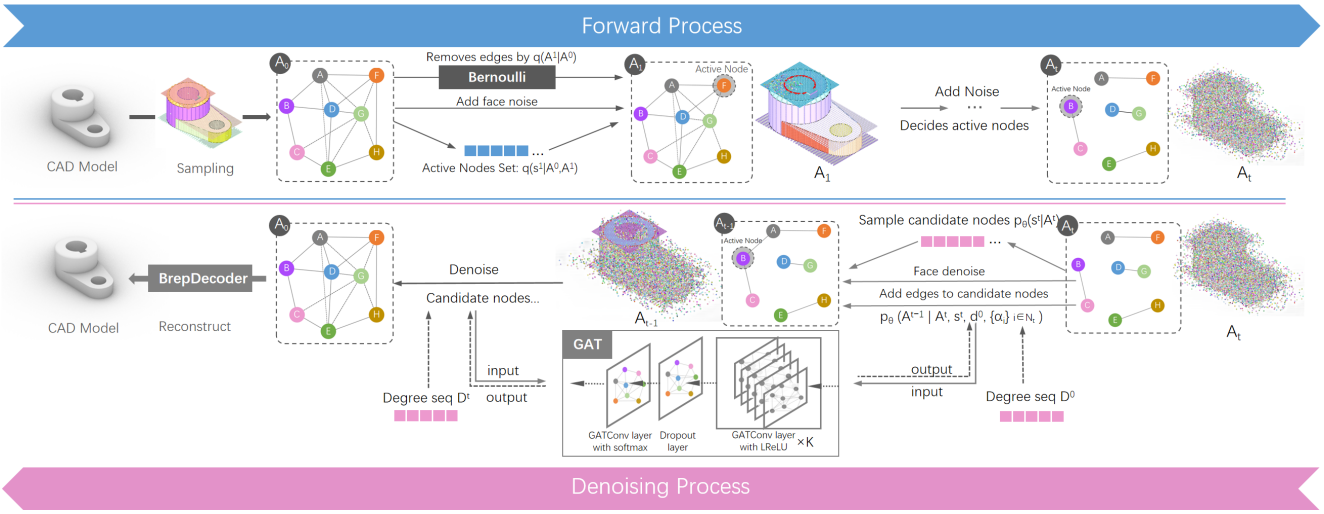


Figure 3. Our model's diffusion process. BrepGiff adds noise to the 3D graph by removing edges and denoise the graph by build edges guided by GAT and degree information.

$\chi = V - E + F$ in the graph. $D = [\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_V]^T$ represents the degree distribution matrix. The degree distribution vector $\mathbf{d} \in \mathbb{R}^V$, where $\mathbf{d}_i$ represents the degree of vertex v_i, is also considered. Finally, the integrated feature vector $\mathbf{X}$ is fed into a 3D CNN [32] for processing. The 3D CNN learns to extract high-level abstract representations from these features. Through a series of convolutional and pooling operations, the 3D CNN outputs a 32-dimensional feature vector $\mathbf{Y} \in \mathbb{R}^{32}$, which comprehensively reflects the topological information of the B-rep model.

3.2. Geometry Encoding

We extract geometric features as attributes of the 3D graph. Basically, we sample the faces and edges of the B-rep model in the UV domain, and using 1D and 2D CNN convolution to reduce the dimensional geometric information and concatenate them with property features.

Face Feature. To extract features of surfaces, we first transfer parametric surfaces into UV domain. Then we uniformly sample the UV-domain surface into 32×32 points. Each sampled point is a 7-dimensional vector, consisting of coordinates $[X, Y, Z]$, surface normal $[N_x, N_y, N_z]$, and the relationship between the point and the trimming surface T, forming a discrete representation $f_{\text{geo}} \in \mathbb{R}^{32\times32\times7}$. We then apply 2D CNN convolutions on the sampled points for dimensionality reduction and get a 128-dimentional vector F_{geo} for the geometric features of each face. In addition, we also calculate other 4 features of faces: type F_{typ}, area F_{area}, degree F_{deg}, index F_{indx}, all as 32-dimentional vector. We then concatnate these 5 features to form the final face feature $F_{face} = \text{Concat}(F_{\text{geo}}, F_{\text{typ}}, F_{\text{area}}, F_{\text{deg}}, F_{\text{indx}}) \in \mathbb{R}^{256}$.

Edge Feature. To capture the geometric features of edges, we uniformly sample points in the UV domain. Each point is a 6-dimensional vector, including 3D coordinates $[X, Y, Z]$ and corresponding tangent vectors $[T_x, T_y, T_z]$. These points create a discrete representation of edges $e_{\text{geo}} \in \mathbb{R}^{m\times6}$, with m being the sample amount along an edge. We then apply 1D CNN to these points for dimensionality reduction, resulting in a 128-dimensional vector E_{geo} for the geometric features of each edge. Additionally, we calculate property features of edges: type E_{typ}, length E_{len}, convexity E_{con}, and the degree sequence of neighboring faces E_{adj}, each as a 32-dimensional vector. We concatenate these features to form the final edge feature $F_{edge} = \text{Concat}(E_{\text{geo}}, E_{\text{typ}}, E_{\text{len}}, E_{\text{con}}, E_{\text{adj}}) \in \mathbb{R}^{256}$.

4. GAT-based Diffusion

4.1. Forward Add Noise Process

Drawn on EDGE [5], BrepGiff designs a customized diffusion process based on Denoising Diffusion Probabilistic Models(DDPM) [13] for B-rep data generation. To add noise process to 3D graph, we remove edges guided by node degrees. To add noise to the node geometry and edge geometry, we follow the noise variance schedule to directly add noise to the nodes' features. In each step, Gaussian noise is added to the edges based on the nodes' degrees, according to Bernoli equation:

$$q(\mathbf{A}^{1:T}, \mathbf{s}^{1:T}|\mathbf{A}^0) = \prod_{t=1}^{T} q(\mathbf{A}^t|\mathbf{A}^{t-1}) q(\mathbf{s}^t|\mathbf{A}^{t-1}, \mathbf{A}^t) \quad (1)$$

where $\mathbf{A}^{1:T}$ and $\mathbf{s}^{1:T}$ are the sequences of all adjacency matrices and all node active states from step $t = 1$ to step $t = T$, respectively. $\mathbf{A}^t$ is the adjacency matrix at step t.

When we add noise to the 3D graph, we also keep a copy of the nodes whose edges have been changed, as well as

their sequence. These node copies and their corresponding sequences are considered as the ***Active Node*** set, which is used in the denoising process for generating B-rep data. Below is a simple formula representation of active nodes:

$$s_t^i = \begin{cases} 1 & \text{if } d_{t-1}^i \neq d_t^i \text{ and } d_{t-1}^i \neq 1 \text{ and } d_t^i \neq 1 \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

where s_t^i represents the active state of node i at step t. If $s_t^i = 1$, node i is active; if $s_t^i = 0$, node i is inactive. d_{t-1}^i and d_t^i represent the degrees of node i at different time steps. It should be noted that different from EDGE, we keep at least one edge for each node, ensuring that nodes with a degree of one will not become isolated during the diffusion process. This is required for applying GAT that requires at least one edge for each node.

4.2. GAT-based Denoising

We employ a Graph Attention Network (GAT) to guide the topological and geometric joint denoising of B-rep models. Unlike conventional GATs that focus solely on node interactions, BrepGiff introduces **cross-attention mechanisms** between nodes (256D) and edges (256D) to co-optimize face and edge geometric features through bidirectional feature interactions. The detailed workflow is as follows:

4.2.1 Cross-Attention Mechanism Design

For each node v_i (corresponding to face features $h_i \in \mathbb{R}^{256}$) and its connected edge e_{ij} (corresponding to edge features $f_{ij} \in \mathbb{R}^{256}$), we compute bidirectional attention weights to fuse geometric information:

Node-to-Edge Attention The update of edge e_{ij} depends on its connected nodes v_i and v_j. The attention weight α_{ij}^e is calculated as:

$$\alpha_{ij}^e = \text{softmax}\left(\text{LReLU}\left(\mathbf{a}_e^T \left[\mathbf{W}_n h_i \| \mathbf{W}_n h_j \| \mathbf{W}_e f_{ij}\right]\right)\right) \tag{3}$$

where $\mathbf{W}_n, \mathbf{W}_e \in \mathbb{R}^{256\times256}$ are learnable weight matrices, $\mathbf{a}_e \in \mathbb{R}^{768}$ is the attention vector, and $\|$ denotes concatenation.

Edge-to-Node Attention The update of node v_i incorporates features from connected edges. The attention weight β_i^n is computed as:

$$\text{softmax}\left(\text{LReLU}\left(\mathbf{a}_n^T \left[\mathbf{W}_n h_i \| \mathbf{W}_e \text{AvgPool}(\{f_{ij}\}_{j\in\mathcal{N}_i})\right]\right)\right) \tag{4}$$

where AvgPool performs average pooling on connected edge features, and $\mathbf{a}_n \in \mathbb{R}^{512}$ is the attention vector.

4.2.2 Joint Feature Update

Edge Feature Denoising Update edge representations based on node features and current edge features:

$$f_{ij}' = \sigma\left(\alpha_{ij}^e \cdot \mathbf{W}_e \left[\mathbf{W}_n h_i \| \mathbf{W}_n h_j \| f_{ij}\right]\right) \tag{5}$$

ensuring consistency between edge geometric attributes (e.g., curvature, convexity) and adjacent face normals/areas.

Node Feature Denoising Update face features by aggregating neighbor nodes and edge information:

$$h_i' = \sigma\left(\beta_i^n \cdot \mathbf{W}_n h_i + \sum_{j\in\mathcal{N}_i} \gamma_{ij} \mathbf{W}_n h_j\right) \tag{6}$$

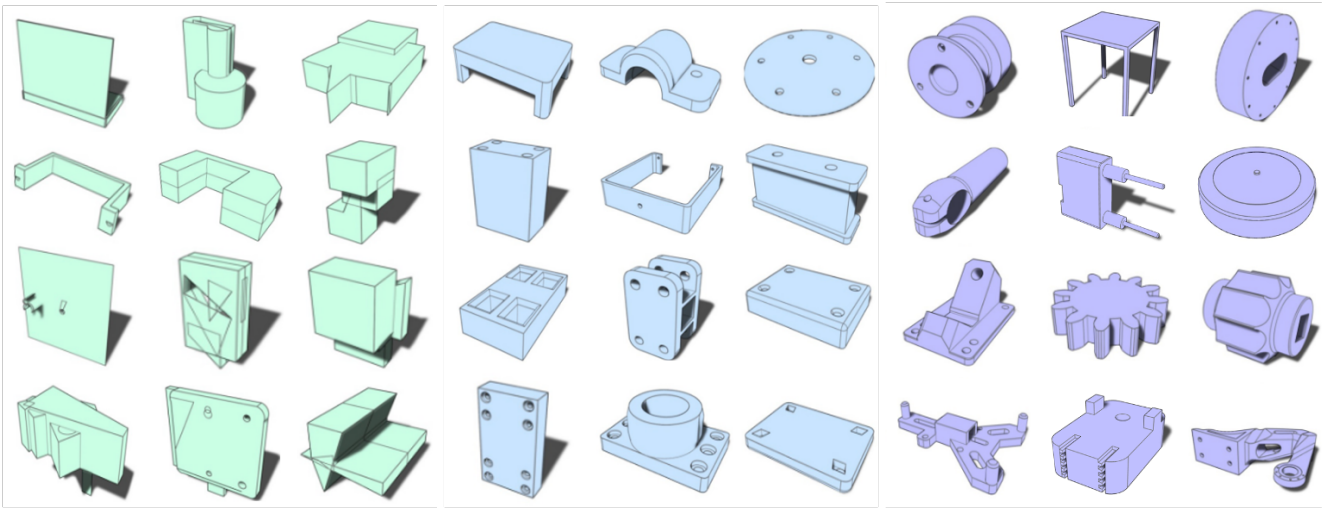
Denoising, which is guided by the degrees sequence of active nodes, contributes significantly to the model's lightweight. At each step, the model identifies the active nodes and predicts the edges for them. The probability of these changes is calculated using the forward and reverse degree change distributions combined with GAT constraints.The process of denoising from local-to-global based on degrees and GAT is key to generating complex CAD models. The denoising process is formulated as such:

$$\begin{aligned} &p_\theta(A_{0:T}, s_{1:T}, d_0 \mid A_T) = p_\theta(d_0) p_\theta(A_T) \\ &\times \prod_{t=1}^{T} \left[p_\theta(s_t \mid A_t, d_0) \prod_{i\in N_t} p_\theta(e_i \mid A_t, s_t, d_0, \alpha_i) \right] \end{aligned} \tag{7}$$

where $p_\theta(d_0)$ and $p_\theta(A_T)$ are the distributions of the initial node degrees and the initial noisy graph. $p_\theta(s_t \mid A_t, d_0)$ is the distribution that predicts which nodes are active, given the current graph A_t and the initial node degrees d_0. $p_\theta(A_{t-1} \mid A_t, s_t, d_0)$ is the distribution that generates the previous graph A_{t-1}, given the current graph A_t, the active node indicators s_t, and the initial node degrees d_0. N_t is the set of active nodes at step t. $p_\theta(e_i \mid A_t, s_t, d_0, \alpha_i)$ is the distribution that predicts the edge existence probability for node i, given the current graph A_t, the active node indicators s_t, the initial node degrees d_0, and α_i is the attention weight computed by GAT. These concepts are operationalized in Algorithm 1, which outlines the GAT-Degree-based graph generation process corresponding to the denoising steps described above.

Algorithm 1 GAT-Degree-based graph generation

Require: Empty graph $\mathbf{A}^T$, graph model $p_\theta(\mathbf{A}^{t-1}|\mathbf{A}^t, \mathbf{s}^t)$, degree sequence model $p_\theta(\mathbf{d}^0)$, and diffusion steps T.
Ensure: Generated graph $\mathbf{A}^0$
  Draw $\mathbf{d}^0 \sim p_\theta(\mathbf{d}^0)$
  for $t = T, \ldots, 1$ **do**
    Determine the active node set $\mathcal{N}_t$ based on the degree changes: $\mathcal{N}_t = \{i \mid s_i^t = 1\}$.
    Compute attention weights α_i for each active node $i \in \mathcal{N}_t$ using GAT.
    Draw $\mathbf{s}^t \sim q(\mathbf{s}^t \mid \deg(\mathbf{A}^t), \mathbf{d}^0)$.
    Draw $\mathbf{A}^{t-1} \sim p_\theta(\mathbf{A}^{t-1} \mid \mathbf{A}^t, \mathbf{s}^t, \mathbf{d}^0, \{\alpha_i\}_{i\in\mathcal{N}_t})$.
  end for



(a) DeepCAD (b) BrepGen (c) BrepGiff(Ours)

Figure 4. This figure demonstrates the generation result of: (a) DeepCAD, (b) BrepGen, and (c) BrepGiff(Ours). We sample CAD model of differen complexity. BrepGiff is capable of generating high quality B-rep CAD models compared to other two techniques.

4.3. B-rep Reconstruction

The denoised 3D graph is converted into a valid B-rep model through a three-stage process:

Parametric Geometry Generation

- *Surfaces*: Face point clouds are fitted to B-spline surfaces via `GeomAPI_PointsToBSplineSurface`.
- *Curves*: Edge points are parameterized as B-spline curves using `GeomAPI_PointsToBSpline`.

Topology-Aware Trimming

- Surface boundaries are trimmed by connected loops, guided by the GNN-predicted adjacency matrix (trained via BCE loss).
- Geometric accuracy is enforced by aligning fitted surfaces/curves with node features through MSE loss.

Watertight Assembly

- Trimmed faces are stitched into a manifold solid using OpenCascade's topological sewing operators.
- Validity constraints (e.g., no self-intersections) are implicitly satisfied by the loss-driven training.

This pipeline ensures that reconstructed B-rep models adhere to both geometric precision (via B-spline fitting) and industrial CAD validity standards.

5. Experiment

5.1. Datasets

To evaluate our model's performance, we conducted experiments using the ABC dataset [17], a large-scale collection of CAD models widely used in geometric deep learning. For unconditional generation metrics comparison, we used the same data as DeepCAD [35]. For other evaluations, we preprocessed the ABC dataset to create the training set by removing models with NURBS and other complex surfaces, controlled the max faces, removed models with multiple shapes and deleted duplicate models.

5.2. Experiment Setup

We implemented BrepGiff in PyTorch and trained the model on two NVIDIA RTX4090 GPUs. To speed up training, we used mixed precision computation. The optimizer we chose is AdamW with a learning rate of 5e-4. We trained the model for 10,000 epochs, generating 64 samples per batch to ensure the model has ample time to learn and converge. The experiment included checks and evaluations every 500 steps to monitor progress and ensure that the model learns as expected.

5.3. Evaluation Metrics

We use 6 metrics to comprehensively evaluate the performance of generative models, including the degree of match between generated and reference data, the novelty and uniqueness of the generated data, and the validity of the generated data. Three of them are same indicators as those employed by Jayaraman et al. [16]. These metrics are described as following:

- *Coverage (COV).* Proportion of reference data having at least one match after each generated data point is assigned to its nearest neighbor in the reference set.
- *Minimum Matching Distance (MMD).* Average Chamfer distance (CD) between the data in the reference set and their nearest neighbors in the generated data.
- *Jensen-Shannon Divergence (JSD).* A symmetric version of the Kullback-Leibler divergence, used to measure the

Table 1. Unconditional generation evaluation results comparison.

Method	COV %↑	MMD %↓	JSD %↓	Novel %↑	Unique %↑	Valid %↑
DeepCAD	65.46	1.29	1.67	87.4	89.3	46.1
SolidGen	71.03	1.08	1.31	99.1	96.2	60.3
BrepGen	73.87	1.04	1.28	**99.8**	**99.7**	62.9
Ours	**75.52**	**0.98**	**1.17**	98.9	98.6	**67.7**

Table 2. Qarameter quantity and training resource consumption.

Model	Parameter	Training Time	GPU	Max Faces
SolidGen	*unknown*	3days	A100×8	50
BrepGen	4.56×10^8	7days	RTX4090×2	50
Ours	7.80×10^7	4days	RTX4090×1	50

similarity between two probability distributions.

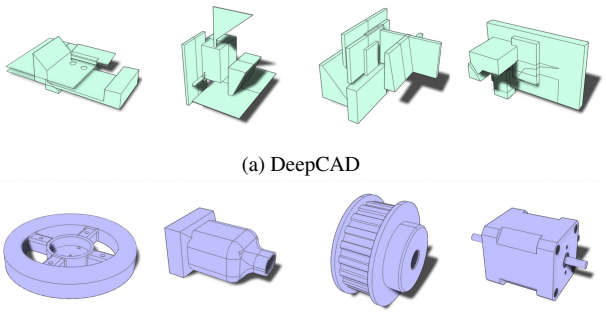
- *Novel*: Percentage of generated models absent in the training set, verified by hashing B-rep structures and parametric features.
- *Unique*: Proportion of distinct models in outputs, ensured through exact hash-based deduplication across the generated set.
- *Valid*: Rate of watertight error-free models, certified via CAD kernel validation of topology and geometry.

5.4. Unconditional Generation

In this experiment, we conducted a comprehensive evaluation of multiple models, including DeepCAD [35], SolidGen [16], BrepGen [39], and our model.

Result Comparison. The experiment results are shown in Table 1. Overall, our model's performance achieves SOTA for most metrics. Specifically, our BrepGiff achieved a better Coverage (COV) of 75.52% compared to BrepGen's 73.87%, indicating better sample diversity and representativeness. Our MMD and JSD values also outperform those of BrepGen, indicating that our generated data distribution closely aligns with the reference data. In terms of Novel and Unique, our BrepGiff is comparable to BrepGen, with most generated data being new and not present in the training set. We achieved a higher validity of 68.2% versus BrepGen's 62.9%, indicating a greater proportion of valid B-rep data and our model's enhanced practicality and reliability.

Lightweightness. We analyze the lightweightness of our model by comparing it with SolidGen and B-rep generation approach , as shown in Table 2. First, our model has only 7.80×10^7 parameters, significantly fewer than both BrepGen (4.56×10^8) and SolidGen. This reduction is achieved through our explicit 3D graph encoding of B-reps, avoiding the multiple diffusion models and VAEs used by BrepGen. In training time, our model completes in 4 days, faster than BrepGen (7 days) while maintaining comparable efficiency to SolidGen (3 days). Notably, our method requires only a single RTX 4090 GPU, whereas BrepGen uses two RTX 4090s and SolidGen requires eight A100 GPUs. This efficiency stems from our graph-based noise addition (via edge removal) and centered 3D graph representation, eliminating the padding and node-copying overheads seen in prior work. Overall, BrepGiff achieves superior lightweightness compared to both baselines.



(a) DeepCAD

(b) BrepGiff

Figure 5. Generated complex CAD models (around 100 faces) comparison for models trained with 2 RTX4090s. Models generated by BrepGiff better represent the shape of real objects.

High Complexity B-rep. We evaluated our model's performance on generating complex B-rep with different maximum face count settings, trained with 2 Nvidia RTX4090 GPUs. The experiment results are shown in Table 3. When the maximum face count was 50, BrepGiff performed best across all metrics. As the maximum face count increased, though the model's performance became worth, it still kept a high generation quality. For example, our model is capable of training with B-rep data of maximum 100 faces and generating B-rep data with valid rate of 62.9%. We did not conduct a performance comparison with BrepGen, as it cannot train on B-rep models containing more than 50 faces with only two RTX 4090 GPUs.

As shown in Figure 5, our generation results are more realistic than other methods. Though DeepCAD [35] is capable of generating complex B-rep data, the visualization shows that most generated models exhibits significant disorganization and fails to accurately represent the actual shape of objects. BrepGen [39] is not capable of training and generating complex B-rep models with 2 RTX4090, thus not

Table 3. Generation results for models with different complexities.

Max Face	COV %↑	MMD %↓	JSD %↓	Novel %↑	Unique %↑	Valid %↑
50	**75.52**	**0.98**	**1.17**	**98.9**	**98.6**	**67.7**
70	72.25	1.67	2.03	98.8	95.9	65.5
100	70.08	3.11	3.80	92.5	90.8	62.9
120	67.31	3.86	4.68	89.5	90.0	48.3

shown in this table. BrepGiff generates models that exhibit a higher degree of accuracy in representing the actual appearance of the objects. Furthermore, it achieves optimal performance in various metrics, including success rate.

5.5. Failure Generation Cases

Figure 6 showcases several failed generation results. The main issues are missing faces and broken faces. It should be noted that the topologies were well recovered, while the recovery of face nodes may be affected by varying levels of noise, leading to insufficient or excessive diffusion steps. When the number of steps is insufficient, the faces cannot be properly recovered. Conversely, when the number of steps is excessive, the faces become over-processed and may break. This issue occurs primarily in complex models.

5.6. Ablation Studies

We conducted three ablation studies to evaluate three key techniques' contribution on our model's performance.

Convolution Effect. We evaluate CNN-based feature extraction on 50K B-rep models (up to 120 faces). As shown in Table 4, using 2D/1D CNNs for dimensionality reduction outperforms raw UV-domain sampling across all metrics: COV (+36%), novelty (+15.7%), uniqueness (+16.1%), and validity (+36.4%).

The performance gain arises from CNNs' ability to learn compact geometric representations by preserving key patterns (e.g., surface normals, edge convexity). In contrast, direct point sampling fails to extract structured features, producing geometically invalid outputs with broken faces (Figure 6)

GAT vs. Non-GAT. We conduct ablation studies on 50K B-rep models ($\leq$ 120 faces). As shown in Table 5, incorporating GAT improves validity by 35.9% and reduces MMD by 16.43%, demonstrating its critical role in enforcing topology-geometry consistency.

For complex models ($>$ 100 faces), the valid rate drops sharply from 67.7% to 31.8% without GAT (Figure 7), as edge-face misalignments disrupt B-rep connectivity.

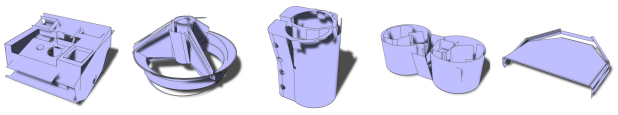


Figure 6. Failure cases generated by BrepGiff.

Table 4. Ablation study on 1D and 2D CNN feature extractors.

Method	COV %↑	MMD %↓	JSD %↓	Novel %↑	Unique %↑	Valid %↑
w/o 1D CNNs	55.22	8.50	12,40	77.5	78.0	50.3
w/o 2D CNNs	32.36	23.96	23.56	62.1	55.9	22.4
Full model	**68.55**	**2.12**	**1.17**	**93.2**	**94.1**	**58.8**

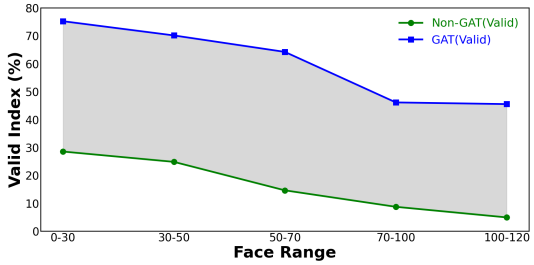


Figure 7. Vaild index of GAT and non GAT generated data in different complexity intervals

Table 5. Ablation study on GNN denoising.

Method	COV %↑	MMD %↓	JSD %↓	Novel %↑	Unique %↑	Valid %↑
w/o GAT	45.52	18.55	20.03	65.6	69.4	22.9
w/ GAT	**68.55**	**2.12**	**1.17**	**93.2**	**94.1**	**58.8**

Table 6. Effect of diffusion steps on B-rep generation quality.

Step	COV %↑	MMD %↓	JSD %↓	Novel %↑	Unique %↑	Valid %↑
100	35.22	45.5	38.90	48.5	30.9	32.1
400	62.15	8.85	11.38	85.2	80.9	52.8
700	68.45	2.89	2.77	92.1	93.0	58.0
1000	**68.55**	**2.12**	**1.17**	**93.2**	**94.1**	**58.8**

Diffusion Step. We evaluate the impact of diffusion steps (100, 400, 700, 1000) on B-rep models with up to 50 faces. As shown in Table 6, increasing steps improves coverage (COV: 35.2% $\rightarrow$ 68.5%, +33.3%) and distribution alignment (JSD: 38.9% $\rightarrow$ 1.17%, -37.7%), demonstrating that more steps yield higher-quality generation.

6. Conclusion

We propose BrepGiff, a novel lightweight method that directly generates complex B-rep model with high quality. The core of BrepGiff lies in the 3D grpah embedding method that encodes topological and geometric feature together. BrepGiff employs diffusion model to add noise in the forward process and uses a degree-guided graph attention network (GAT) for denoising. Experiments show that BrepGiff achieves state-of-the-art performance in B-rep generation tasks, especially for complex B-rep models. BrepGiff provides a solid foundation for future research and applications in direct B-rep generation technology.

7. Acknowledgments

This research was funded by the Fundamental Research Funds for the Central Universities, the Natural Science Basic Research Program of Shaanxi Province under Grant 2024JC-YBQN-0702, and NUS Research Scholarship and NUS ORIA.

References

[1] Antonio Alliegro, Yawar Siddiqui, Tatiana Tommasi, and Matthias Nießner. PolyDiff: Generating 3D Polygonal Meshes with Diffusion Models. *arXiv e-prints*, art. arXiv:2312.11417, 2023. 3

[2] Mikaela Angelina Uy, Yen-yu Chang, Minhyuk Sung, Purvi Goel, Joseph Lambourne, Tolga Birdal, and Leonidas Guibas. Point2Cyl: Reverse Engineering 3D Objects from Point Clouds to Extrusion Cylinders. *arXiv e-prints*, art. arXiv:2112.09329, 2021. 2

[3] Weijuan Cao, Trevor T. Robinson, Yang Hua, Flavien Boussuge, Andrew R. Colligan, and Wanbin Pan. Graph representation of 3d cad models for machining feature recognition with deep learning. *Volume 11A: 46th Design Automation Conference (DAC)*, 2020. 2

[4] Tianrun Chen, Chunan Yu, Yuanqi Hu, Jing Li, Tao Xu, Runlong Cao, Lanyun Zhu, Ying Zang, Yong Zhang, Zejian Li, and Linyun Sun. Img2CAD: Conditioned 3D CAD Model Generation from Single Image with Structured Visual Geometry. *arXiv e-prints*, art. arXiv:2410.03417, 2024. 2

[5] Xiaohui Chen, Jiaxing He, Xu Han, and Li-Ping Liu. Efficient and Degree-Guided Graph Generation via Discrete Diffusion Modeling. *arXiv e-prints*, art. arXiv:2305.04111, 2023. 4

[6] Andrew R. Colligan, Trevor T. Robinson, Declan C. Nolan, Yang Hua, and Weijuan Cao. Hierarchical cadnet: Learning from b-reps for machining feature recognition. *Comput. Aided Des.*, 147:103226, 2022. 2

[7] Qiujie Dong, Rui Xu, Pengfei Wang, Shuangmin Chen, Shiqing Xin, Xiaohong Jia, Wenping Wang, and Changhe Tu. NeurCADRecon: Neural Representation for Reconstructing CAD Surfaces by Enforcing Zero Gaussian Curvature. *arXiv e-prints*, art. arXiv:2404.13420, 2024. 2

[8] Elona Dupont, Kseniya Cherenkova, Anis Kacem, Sk Aziz Ali, Ilya Arzhannikov, Gleb Gusev, and Djamila Aouada. CADOps-Net: Jointly Learning CAD Operation Types and Steps from Boundary-Representations. *arXiv e-prints*, art. arXiv:2208.10555, 2022. 2

[9] Elona Dupont, Kseniya Cherenkova, Dimitrios Mallis, Gleb Gusev, Anis Kacem, and Djamila Aouada. TransCAD: A Hierarchical Transformer for CAD Sequence Inference from Point Clouds. *arXiv e-prints*, art. arXiv:2407.12702, 2024. 2

[10] Daoyi Gao, Dávid Rozenberszki, Stefan Leutenegger, and Angela Dai. DiffCAD: Weakly-Supervised Probabilistic CAD Model Retrieval and Alignment from an RGB Image. *arXiv e-prints*, art. arXiv:2311.18610, 2023. 3

[11] Haoxiang Guo, Shilin Liu, Hao Pan, Yang Liu, Xin Tong, and Baining Guo. Complexgen: Cad reconstruction by b-rep chain complex generation. *ACM Trans. Graph. (SIGGRAPH)*, 41(4), 2022. 1, 2

[12] Anchit Gupta and Anchit Gupta. 3dgen: Triplane latent diffusion for textured mesh generation. *ArXiv*, abs/2303.05371, 2023. 2

[13] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020. 2, 4

[14] Eunji Hong, Minh Hieu Nguyen, Mikaela Angelina Uy, and Minhyuk Sung. MV2Cyl: Reconstructing 3D Extrusion Cylinders from Multi-View Images. *arXiv e-prints*, art. arXiv:2406.10853, 2024. 2

[15] Pradeep Kumar Jayaraman, Aditya Sanghi, J. Lambourne, T. Davies, and Hooman Shayani. Uv-net: Learning from boundary representations. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11698–11707, 2021. 2

[16] Pradeep Kumar Jayaraman, Joseph G. Lambourne, Nishkrit Desai, Karl D. D. Willis, Aditya Sanghi, and Nigel J. W. Morris. SolidGen: An Autoregressive Model for Direct B-rep Synthesis. *arXiv e-prints*, art. arXiv:2203.13944, 2022. 2, 6, 7

[17] Sebastian Koch, Albert Matveev, Zhongshi Jiang, Francis Williams, Alexey Artemov, Evgeny Burnaev, Marc Alexa, Denis Zorin, and Daniele Panozzo. ABC: A Big CAD Model Dataset For Geometric Deep Learning. *arXiv e-prints*, art. arXiv:1812.06216, 2018. 6

[18] J. Lambourne, Karl D. D. Willis, Pradeep Kumar Jayaraman, Aditya Sanghi, Peter Meltzer, and Hooman Shayani. Brepnet: A topological message passing system for solid models. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12768–12777, 2021. 2

[19] Changjian Li, Hao Pan, Adrien Bousseau, and Niloy J. Mitra. Free2cad: Parsing freehand drawings into cad commands. *ACM Trans. Graph. (Proceedings of SIGGRAPH 2022)*, 41 (4):93:1–93:16, 2022. 1, 2

[20] Pu Li, Jianwei Guo, Xiaopeng Zhang, and Dong-ming Yan. SECAD-Net: Self-Supervised CAD Reconstruction by Learning Sketch-Extrude Operations. *arXiv e-prints*, art. arXiv:2303.10613, 2023. 2

[21] Pu Li, Jianwei Guo, Huibin Li, Bedrich Benes, and Dong-Ming Yan. Sfmcad: Unsupervised cad reconstruction by learning sketch-based feature modeling operations. *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4671–4680, 2024. 2

[22] Yujia Liu, Anton Obukhov, Jan Dirk Wegner, and Konrad Schindler. Point2CAD: Reverse Engineering CAD Models from 3D Point Clouds. *arXiv e-prints*, art. arXiv:2312.04962, 2023. 1, 2

[23] Yilin Liu, Jiale Chen, Shanshan Pan, Daniel Cohen-Or, Hao Zhang, and Hui Huang. Split-and-fit: Learning b-reps via structure-aware voronoi partitioning. *ACM Trans. Graph.*, 43:108:1–108:13, 2024. 1, 2

[24] Zhen Liu, Yao Feng, Michael J. Black, Derek Nowrouzezahrai, Liam Paull, and Weiyang Liu. MeshDiffusion: Score-based Generative 3D Mesh Modeling. *arXiv e-prints*, art. arXiv:2303.08133, 2023. 2

[25] Zhaoyang Lyu, Ben Fei, Jinyi Wang, Xudong Xu, Ya Zhang, Weidong Yang, and Bo Dai. GetMesh: A Controllable Model for High-quality Mesh Generation and Manipulation. *arXiv e-prints*, art. arXiv:2403.11990, 2024. 3

[26] Baorui Ma, Yu-Shen Liu, and Zhizhong Han. Learning signed distance functions from noisy 3d point clouds via noise to noise mapping. *ArXiv*, abs/2306.01405, 2023. 2

[27] Daxuan Ren, Jianmin Zheng, Jianfei Cai, Jiatong Li, Haiyong Jiang, Zhongang Cai, Junzhe Zhang, Liang Pan, Mingyuan Zhang, Haiyu Zhao, and Shuai Yi. CSG-Stump: A Learning Friendly CSG-Like Representation for Interpretable Shape Parsing. *arXiv e-prints*, art. arXiv:2108.11305, 2021. 1, 2

[28] Daxuan Ren, Jianmin Zheng, Jianfei Cai, Jiatong Li, and Junzhe Zhang. Extrudenet: Unsupervised inverse sketch-and-extrude for shape parsing. *ArXiv*, abs/2209.15632, 2022. 1, 2

[29] Mohammad Sadil Khan, Elona Dupont, Sk Aziz Ali, Kseniya Cherenkova, Anis Kacem, and Djamila Aouada. CAD-SIGNet: CAD Language Inference from Point Clouds using Layer-wise Sketch Instance Guided Attention. *arXiv e-prints*, art. arXiv:2402.17678, 2024. 1, 2

[30] Gopal Sharma, Rishabh Goyal, Difan Liu, Evangelos Kalogerakis, and Subhransu Maji. CSGNet: Neural Shape Parser for Constructive Solid Geometry. *arXiv e-prints*, art. arXiv:1712.08290, 2017. 1, 2

[31] Ian Stroud. *Boundary representation modelling techniques*. Springer Science & Business Media, 2006. 1

[32] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning Spatiotemporal Features with 3D Convolutional Networks. *arXiv e-prints*, art. arXiv:1412.0767, 2014. 4

[33] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. *arXiv e-prints*, art. arXiv:1710.10903, 2017. 3

[34] 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. *arXiv e-prints*, art. arXiv:2010.02392, 2020. 2

[35] 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 (ICCV)*, pages 6772–6782, 2021. 6, 7

[36] Jiale Xu, Weihao Cheng, Yiming Gao, Xintao Wang, Shenghua Gao, and Ying Shan. InstantMesh: Efficient 3D Mesh Generation from a Single Image with Sparse-view Large Reconstruction Models. *arXiv e-prints*, art. arXiv:2404.07191, 2024. 2

[37] Xianghao Xu, Wenzhe Peng, Chin-Yi Cheng, Karl D. D. Willis, and Daniel Ritchie. Inferring cad modeling sequences using zone graphs. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6058–6066, 2021. 2

[38] Xiang Xu, Karl D. D. Willis, J. Lambourne, Chin-Yi Cheng, Pradeep Kumar Jayaraman, and Yasutaka Furukawa. Skexgen: Autoregressive generation of cad construction sequences with disentangled codebooks. *ArXiv*, abs/2207.04632, 2022. 2

[39] Xiang Xu, Joseph G Lambourne, Pradeep Kumar Jayaraman, Zhengqing Wang, Karl DD Willis, and Yasutaka Furukawa. Brepgen: A b-rep generative diffusion model with structured latent geometry. *arXiv preprint arXiv:2401.15563*, 2024. 1, 2, 7

[40] Bingchen Yang, Haiyong Jiang, Hao Pan, Peter Wonka, Jun Xiao, and Guosheng Lin. PS-CAD: Local Geometry Guidance via Prompting and Selection for CAD Reconstruction. *arXiv e-prints*, art. arXiv:2405.15188, 2024. 2

[41] Fenggen Yu, Zhiqin Chen, Manyi Li, Aditya Sanghi, Hooman Shayani, Ali Mahdavi-Amiri, and Hao Zhang. CAPRI-Net: Learning Compact CAD Shapes with Adaptive Primitive Assembly. *arXiv e-prints*, art. arXiv:2104.05652, 2021. 2

[42] Fenggen Yu, Qimin Chen, Maham Tanveer, Ali Mahdavi Amiri, and Hao Zhang. D^2CSG: Unsupervised Learning of Compact CSG Trees with Dual Complements and Dropouts. *arXiv e-prints*, art. arXiv:2301.11497, 2023. 1, 2

[43] Biao Zhang, Jiapeng Tang, Matthias Niessner, and Peter Wonka. 3DShape2VecSet: A 3D Shape Representation for Neural Fields and Generative Diffusion Models. *arXiv e-prints*, art. arXiv:2301.11445, 2023. 2

[44] Shengdi Zhou, Tianyi Tang, and Bin Zhou. Cadparser: A learning approach of sequence modeling for b-rep cad. In *International Joint Conference on Artificial Intelligence*, 2023. 2