Journal of Computational Design and Engineering, 2024, 11, 110–134
DOI: 10.1093/jcde/qwae005
Advance access publication date: 19 January 2024
Research Article

Brep2Seq: a dataset and hierarchical deep learning network for reconstruction and generation of computer-aided design models

Shuming Zhang[1], Zhidong Guan[1], Hao Jiang[2], Tao Ning[2,*], Xiaodong Wang[1] and Pingan Tan[1]

[1]School of Aeronautic Science and Engineering, Beihang University, Beijing 100191, China
[2]School of Mechanical Engineering and Automation, Beihang University, Beijing 100191, China
*Correspondence: beihang703@gmail.com

Abstract

Three-dimensional (3D) reconstruction is a significant research topic in the field of computer-aided design (CAD), which is used to recover editable CAD models from original shapes, including point clouds, voxels, meshes, and boundary representations (B-rep). Recently, there has been considerable research interest in deep model generation due to the increasing potential of deep learning methods. To address the challenges of 3D reconstruction and generation, we propose Brep2Seq, a novel deep neural network designed to transform the B-rep model into a sequence of editable parametrized feature-based modeling operations comprising principal primitives and detailed features. Brep2Seq employs an encoder-decoder architecture based on the transformer, leveraging geometry and topological information within B-rep models to extract the feature representation of the original 3D shape. Due to its hierarchical network architecture and training strategy, Brep2Seq achieved improved model reconstruction and controllable model generation by distinguishing between the primary shape and detailed features of CAD models. To train Brep2Seq, a large-scale dataset comprising 1 million CAD designs is established through an automatic geometry synthesis method. Extensive experiments on both DeepCAD and Fusion 360 datasets demonstrate the effectiveness of Brep2Seq, and show its applicability to simple mechanical components in real-world scenarios. We further apply Brep2Seq to various downstream applications, including point cloud reconstruction, model interpolation, shape constraint generation, and CAD feature recognition.

Keywords: computer-aided design, boundary representation, 3D deep learning, geometry synthesis, 3D reconstruction, generative model

1. Introduction

Computer-aided design (CAD) software enables the creation of a three-dimensional (3D) geometric model through a series of modeling operations such as 2D sketching, extruding, and Boolean operations. Once the geometric modeling is finished, it is common practice to save the CAD model as a neutral file format, such as IGES or STEP, for data compatibility and versatility between various industrial software systems, including computer-aided engineering and computer-aided manufacturing (CAM). However, unlike native files in CAD software, these neutral geometry files often lack information about the geometric modeling process, including design intent, modeling steps, and editable feature-based modeling parameters, which hinders models reuse and information integration in downstream analysis and manufacturing processes.

To reconstruct the modeling information of raw 3D shapes, Shapiro and Vossler (1991) and Buchele and Crawford (2003) introduced the utilization of half spaces to transform 3D shapes into constructive solid geometry (CSG) models. Since then, extensive research has been conducted in CAD model reconstruction, which can be classified into strategies such as constraint-based construction, feature-based construction, search-based algorithms, tree-based construction, and CSG conversion.

In recent years, machine learning methods, particularly deep learning (DL) methods, have exhibited significant potential in the field of CAD (Buonamici *et al.*, 2018). These methods have proven effective in addressing traditional challenges, including feature recognition (Colligan *et al.*, 2022; Lee *et al.*, 2023; Ning *et al.*, 2023; Yeo *et al.*, 2021), model classification (Meltzer *et al.*, 2021; Takaishi *et al.*, 2020), model retrieval (Sung *et al.*, 2017), model generation (Jayaraman *et al.*, 2022; Kim *et al.*, 2021; Nash *et al.*, 2020; Wu *et al.*, 2021), and design optimization (Zhang *et al.*, 2019). By leveraging the powerful capabilities of DL, it becomes possible to implement new features that traditional methods cannot handle, thereby improving modeling techniques and enhancing human-machine interaction within CAD software (Li *et al.*, 2022).

While DL methods have been applied in CAD, most of the previous research works have not concentrated on the reconstruction of CAD modeling process, and there is a relative scarcity of research focused on boundary representation (B-rep), the industry standard for representing 3D solid models based on parametric curves and surfaces. Consequently, the extension of representation learning research of B-rep data to the domain of model reconstruction remains an urgent and challenging problem. Additionally, there is a deficiency in large-scale CAD model datasets that adequately support DL of feature-based modeling. These datasets

Received: July 18, 2023. **Revised:** January 2, 2024. **Accepted:** January 8, 2024

should encompass B-rep data of 3D shapes and their associated feature-based CAD modeling operations. Despite the establishment of CAD datasets, like Fusion 360 Gallery (Willis *et al.*, 2021) and DeepCAD (Wu *et al.*, 2021), these datasets and corresponding DL methods are limited to two types of CAD modeling operations: 2D sketch and extrusion.

In this work, we leverage DL methods to address the challenge of reconstructing and generating 3D CAD models. We propose Brep2Seq, a deep neural network that directly takes the B-rep data of 3D CAD models as input, inspired by the transformer network (Vaswani *et al.*, 2017) and its adaptation to graph-structured data, known as graphormer (Ying *et al.*, 2021). This Brep2Seq network can encode the B-rep data into a latent space and decode a sequence of CAD feature-based modeling operations. The sequence comprises two types of modeling operations: one for assembling basic geometric primitives (e.g., cuboid, prism, cylinder, cone, and sphere) to form the primary shape of the CAD model, and another for common mechanical features (e.g., hole, groove, fillet, and chamfer). To facilitate DL and training of Brep2Seq, we publish a large-scale dataset consisting of 1 million CAD designs. This dataset encompasses B-rep data of CAD models, as well as parameterized feature-based modeling information.

The novelty and contribution of our work are summarized as follows:

(i) We propose Brep2Seq, a hierarchical graph-to-sequence deep neural network that accurately separates the primary shape and detailed features of CAD models for parameterized feature-based modeling reconstruction.
(ii) We propose a method that combines basic geometric primitives and detailed features for parameterized feature-based modeling, facilitating the synthesis of a large-scale CAD model dataset.
(iii) We achieve the learning of latent space distribution of CAD models and the generation of diverse CAD models by employing adversarial training based on the pre-trained Brep2Seq model.
(iv) The Brep2Seq model can be well applied to various downstream tasks, including point cloud reconstruction, model interpolation, shape constraint generation, and CAD feature recognition.

This paper is organized as follows: Section 2 reviews recent related works. Section 3 describes the representation of CAD construction sequences and the process of automatically synthesizing large-scale dataset. Section 4 introduces the neural network architecture and design details of Brep2Seq. Section 5 presents the results of experiments. Section 6 presents the potential applications in downstream tasks and the discussion is provided in Section 7. Finally, Section 8 concludes this paper. Codes and datasets related to this study could be found at https://github.com/zhangshuming0668/Brep2Seq.

2. Related Work

2.1. 3D shape representation learning

The exploration of 3D shape representation learning began in the fields of computer vision and has evolved into a range of learning-based methods that can handle various geometric data formats, including point clouds, voxels, and polygon meshes. Point clouds, a most common data formats in reverse engineering (RE), represent 3D geometric shapes as a collection of 3D points. Qi *et al.* (2017a) and Qi *et al.* (2017b) subsequently proposed PointNet and PointNet++ for extracting structural relationships and feature representations from point cloud data, enabling their application in downstream tasks such as 3D shape classification, part segmentation, semantic segmentation, and more. 3D voxels is a structured and regular data format that uses a Cartesian grid to describe the 3D shape. Therefore, it is well-suited for encoding with convolutional neural networks (CNNs), as demonstrated by related works (Hilbig *et al.*, 2023; Lee *et al.*, 2022; Qi *et al.*, 2016; Wu *et al.*, 2015). Polygon meshes offer another choice for describing 3D shapes, and Liu *et al.* (2019) and Hanocka *et al.* (2019) have, respectively, introduced non-Euclidean data space convolution and pooling methods suitable for polygon meshes. Furthermore, 3D shapes can be transformed into a 2D multi-view representation for learning. For example, Qin *et al.* (2022) leveraged structured semantic information and geometric topological relationships in sketches and employed a recursive neural network to extract feature representations of CAD models.

The existing methods mentioned above have addressed the problem of 3D shape representation learning from various perspectives. However, they are encumbered by issues such as high network complexity and low shape representation accuracy, limiting their real-world applications. To maximize the preservation of CAD model information, conducting representation learning directly on B-rep data may be a more optimal solution. In recent years, scholars have made constructive progress in the representation learning of B-rep. Cao *et al.* (2020) convert B-rep into graph representation, and then use graph neural networks (GNNs) to encode the geometric shapes and topological relationships. Subsequently, Colligan *et al.* (2022) proposed an improved hierarchical graph convolutional model. Jayaraman *et al.* (2021) utilized CNNs to extract the embedding of surfaces and curves, followed by employing GNNs to get the embedding of B-rep models. Lambourne *et al.* (2021) proposed a novel graph convolution method specifically designed for B-rep data. This convolution method centers around the coedge and describes the relationship between coedge and faces using multiple adjacency matrices, enhancing the capture of the global and local topological relationship in B-rep. Recently, some scholars have introduced attention mechanisms to model relationships between nodes in the B-rep graph. Lee *et al.* (2023) proposed BRepGAT to segment machining feature in B-rep models based on graph attention networks. Hou *et al.* (2023) introduced a fusion self-attention graph convolutional networks (GCNs) framework called FuS-GCN. Their work integrates an attention aggregation function designed to allocate increased attention to key nodes, consequently enhancing the feature representation capability of the B-rep graph.

While the above methods realize the application of traditional CNNs and GNNs on the B-rep data, our work aims to improve the representation learning capability of the complex B-rep models by introducing the transformer architecture and further increasing the parameter scale of the neural network.

2.2. CAD model reconstruction

The reconstruction of digital models from measured data has long been a fundamental objective in the fields of engineering and computer science. This process, commonly referred to as “RE” or “CAD model reconstruction”, aims to generate 3D mathematical surfaces and geometric features that faithfully represent the geometry of a physical part (Alai, 2013; Motavalli, 1998; Varady *et al.*, 1997). This is a key problem with diverse engineering applications, such as quality control, rapid mold manufacturing, design of custom-fit parts, and product analysis and optimization. The primary workflow of RE consists of the following tasks: (i) data capture and pre-processing (Barbero & Ureta, 2011), (ii) segmentation and feature classification (Agathos *et al.*, 2007; Di Angelo & Di Stefano, 2015; Song *et al.*, 2021), and (iii) surface fitting and CAD model generation (Benkő *et al.*, 2001).

One of the most critical tasks in RE is to transform the input raw data (point clouds, meshes, etc.) into B-rep models, involves surface fitting and CAD model generation (Buonamici *et al.*, 2018). While accurately representing 3D objects as B-rep models describes geometric shapes and spatial locations, it falls short in capturing higher-level semantic descriptions. This limitation complicates tasks like making modifications or automatically generating efficient and effective process plans. For instance, while these representations can capture the shape of a hole, they do not explicitly convey that it is a true cylindrical hole. Consequently, even a straightforward task like altering the hole's diameter can pose challenges for a designer. Therefore, it becomes necessary to further infer corresponding parameterized CAD programs based on the reconstructed B-rep model.

One way to address this challenge is to restore the CSG structure of 3D CAD models. In CSG, shapes are formed by combining primitive solids (e.g., spheres, cylinders, and boxes) using Boolean operations such as union, intersection, and difference. CSGNet (Sharma *et al.*, 2018) employed CNN as an encoder to identify the geometric features of 2D images or 3D voxels and utilized recurrent neural network (RNN) as a decoder to predict geometric voxel description parameters, followed by reinforcement learning training methods to analyze the CSG tree structure. Likewise, UCSG-Net (Kania *et al.*, 2020), CSG-Stump (Ren *et al.*, 2021), and CAPRI-Net (Yu *et al.*, 2022) employed supervised learning methods to predict the CSG tree structure.

Another approach is to use a feature-based parametric representation that embeds design intentions through geometric features, parameters, and constraints (Safdar *et al.*, 2020). Zhang *et al.* (2020) utilized virtual topology operations and Boolean segmentation loops search strategy to achieve automatic geometric boundary segmentation of B-rep solid models, thereby facilitating the reconstruction of the feature-based modeling process. Similarly, Xu *et al.* (2021) decomposes B-rep models into zone graphs, using GCN and search ranking algorithms to infer the CAD modeling process for 3D shapes (sketching, extrusion, and Boolean operations). Furthermore, Point2Cyl (Uy *et al.*, 2022), through point cloud segmentation and recognition, predicts command sequences for constructing 3D shapes, specifically cylindrical extrusions. Similarly, Lambourne *et al.* (2022) employed a differentiable loss function to directly compare predicted parameters with the target 3D shape, parsing 3D voxel models into modeling operation sequences involving sketches and extrusions. Willis *et al.* (2021) conceptualized CAD modeling as a Markov process and employing reinforcement learning to derive corresponding operation sequences involving sketches and extrusions.

2.3. CAD model generation

CAD model generation focuses on creating novel 3D shapes that have not been previously observed. Recent research has predominantly concentrated on the generation of 3D shapes using deep neural networks. DeepCAD (Wu *et al.*, 2021) implemented a self-encoding method for the generation of CAD modeling operation sequences based on the transformer networks. Similarly, SkexGen (Xu *et al.*, 2022) is an autoregressive sequence-to-sequence generative model using multiple parallel encoders and decoders, enabling the analysis and generation of the CAD modeling operation sequence encompassing points, edges, faces, sketches, and extrudes. In contrast, SolidGen (Jayaraman *et al.*, 2022) is an autoregressive model that generates B-rep models directly using a three-layer encoding and decoding structure. Some other research works are devoted to generating CSG structures or structured representations, using tree structures (Li *et al.*, 2017; Roberts *et al.*, 2021), hierarchical structures (Gao *et al.*, 2019; Jones *et al.*, 2020; Li *et al.*, 2020), or graph structures (Mo *et al.*, 2019) to depict the interconnections among geometric elements and the relationships between the whole and its constituents. Instead of directly generating parametric descriptions of basic geometric elements (e.g., points, curves, and surfaces) in 3D models, our work aims to explore a new approach to CAD model generation by generating a sequence of feature-based modeling operations.

3. Dataset

In engineering, most mechanical products are designed and manufactured based on specific features that possess a strong structural nature. Commercial CAD software, such as Autodesk's AutoCAD, supports feature-based design by creating 3D shapes from 2D profiles using methods like extrusion or revolution. Open-source CAD packages, such as FreeCAD, offer a variety of feature-based modeling tools including primitive-based modeling relying on geometric primitives (cubes, cylinders, cones, torus, etc.). CAM software like EZFeatureMill supports form features that are closely associated with machining operations, such as stock, holes, slots, pockets, grooves, and bosses. Based on the observation of the feature-based modeling workflow as mentioned earlier, our work seek to leverage parameter-driven primitives and features to represent CAD models, as well as recover the editable modeling operations with help of DL techniques.

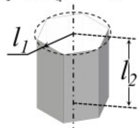
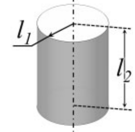
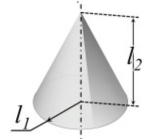
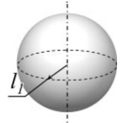
To our knowledge, there are existing CAD model datasets encompassing both manually designed CAD model datasets such as ABC (Koch *et al.*, 2019), Fusion 360 Gallery (Willis *et al.*, 2021), DMU-Net (Dekhtiar *et al.*, 2018), and FabWave-3D (Angrish *et al.*, 2019), and program synthesized CAD model datasets like FeatureNet (Zhang *et al.*, 2018), MFCAD (Cao *et al.*, 2020), SolidLetters (Jayaraman *et al.*, 2021), and MFCAD++ (Colligan *et al.*, 2022). However, the aforementioned CAD model datasets cannot be directly used for parametrized feature-based modeling reconstruction of the B-rep model. Consequently, we establish a large-scale CAD dataset with the CAD construction sequence, suitable for training our network through supervised learning. We employ a random synthesis algorithm to automatically generate CAD models while obtaining the corresponding B-rep data and CAD construction sequences. The CAD model synthesis algorithm is developed based on the open-source geometric kernel OpenCAScade.

The construction sequence of each CAD model in the dataset consists of two types of operations that are common in feature-based modeling. The first are the principal primitives, which can be cuboids, prisms, cylinders, cones, and spheres. The second are detailed features, which encompass 24 common mechanical features described by Zhang *et al.* (2018), such as slot, through-hole, step, fillet, chamfer, and so on. Although these modeling operations are limited, according statistics given by Nourse (1980), approximately 85% of machined parts can be modeled using simple flat and quadric surfaces. Therefore, their reasonable combination sufficiently expresses the geometric shape of the majority of engineering components.

3.1. Principal primitive

Each principal primitive that comprises a CAD model is described by two types of parameters: primitive type and design parameters.

Table 1: Modeling operations and parameters of principal primitives.

Operation	Parameters	Description	Visualization
B(Box)	$l_1, l_2, l_3, t_x, t_y, t_z, q_0, q_1, q_2, q_3$	Cuboid defined by length (l_1), width (l_2), and height (l_3).	
P(Prism)	$l_1, l_2, e, t_x, t_y, t_z, q_0, q_1, q_2, q_3$	e-Sides prism defined by radius (l_1) and height (l_2).	
C(Cylinder)	$l_1, l_2, t_x, t_y, t_z, q_0, q_1, q_2, q_3$	Cylinder defined by radius (l_1) and height (l_2).	
K(Cone)	$l_1, l_2, t_x, t_y, t_z, q_0, q_1, q_2, q_3$	Cone defined by radius (l_1) and height (l_2).	
S(Sphere)	l_1, t_x, t_y, t_z	Sphere defined by radius (l_1).	

The primitive type, denoted by $o_n \in \{B, P, C, K, S\}$, represents the specific type of primitive, including the cuboid (B), prism (P), cylinder (C), cone (K), or sphere (S). The design parameters $\{l_1, l_2, l_3, e, t_x, t_y, t_z, q_0, q_1, q_2, q_3\}$ are used to describe the geometry and spatial position of a primitive, where l_1, l_2, l_3, and e define the geometric shape; t_x, t_y, and t_z define the translation transformation relative to the global coordinate system origin; and the quaternion values q_0, q_1, q_2, and q_3 define the rotation transformation of primitive. A detailed description of the operation types and parameters can be found in Table 1.

3.2. Detailed feature

Building upon the previous research of FeatureNet (Zhang *et al.*, 2018), MFCAD (Cao *et al.*, 2020), and MFCAD++ (Colligan *et al.*, 2022), we integrate the 24 mechanical features in our work, as shown in Fig. 1. The detailed features of CAD models can be effectively characterized by their feature type and design parameters. The design parameters $\{l, w, r, w_1, d, x_1, y_1, z_1, x_2, y_2, z_2\}$ describe the geometric shape and spatial position. Sketch contours and 3D geometric shapes of detailed features can be defined through combinations of parameters like length l, width w, radius r, depth d, and others. In the case of all detailed features, we employ two 3D points, denoted as $p_1(x_1, y_1, z_1)$ and $p_2(x_2, y_2, z_2)$, to specify their spatial positions. For features like through slots, holes, blind slots, and blind holes, p_1 and p_2 serve to establish their axial orientation as well as the starting and ending positions of the feature. In the case of step features, p_1 and p_2 define the feature's outline. For fillet features, p_1 and p_2 are used to determine the edges where the fillet operation should be applied. Detailed definitions of these features can be found in Table A.1 in the Appendix.

3.3. Dataset generation

When using CAD software to model 3D shapes, designers commonly draw a sketch on a plane associated with the existing 3D entity and perform subsequent operations such as extrude and rotation. Our automatic process replicates this modeling process to synthesize CAD models. We position the first primitive at the origin of the global coordinate system and add the following primitives sequentially. We randomly select a face from the current geometric shape G (e.g., random extraction using area as weight) to serve as the base face or reference plane for a 2D sketch. To ensure that the synthesized CAD models possess specific engineering characteristics and conform to standardization, we determine the position and shape of the 2D sketch (rectangle, polygon, and circle) by considering general spatial geometric constraints such as coplanarity, concentricity, tangency, coedge, equal height, etc. (as shown in Fig. 2).

For how to appropriately add detailed features into the primary shape, we follow the approach as described in MFCAD++ (Colligan *et al.*, 2022). By specifying the sketch plane, sketch outline, and extrude depth, we sequentially add detailed features and record the parameters of feature-based modeling operations.

By employing the random synthesis algorithm described above, a B-rep model and a construction sequence consisting of primitives and features are obtained. However, it is crucial to recognize that a B-rep model can be derived from various construction sequences, resulting in a one-to-many relationship between the B-rep model and construction sequences. Therefore, further steps are required to ensure that each sample label (construction sequence) in the synthesized dataset represents the optimal parameterized feature-based modeling process for the B-rep model. Below, we elaborate on the process of determining the unique construction sequence for the B-rep model.

(i) In accordance with the Occam's razor principle and CSG tree optimization methods (Friedrich *et al.*, 2020, 2019), the number of principal primitives is minimized through a generate-check-regenerate process. This involves examining whether there are two or more principal primitives in the construction sequence that can be merged and simplified. For example, this may require replacing two adjacent

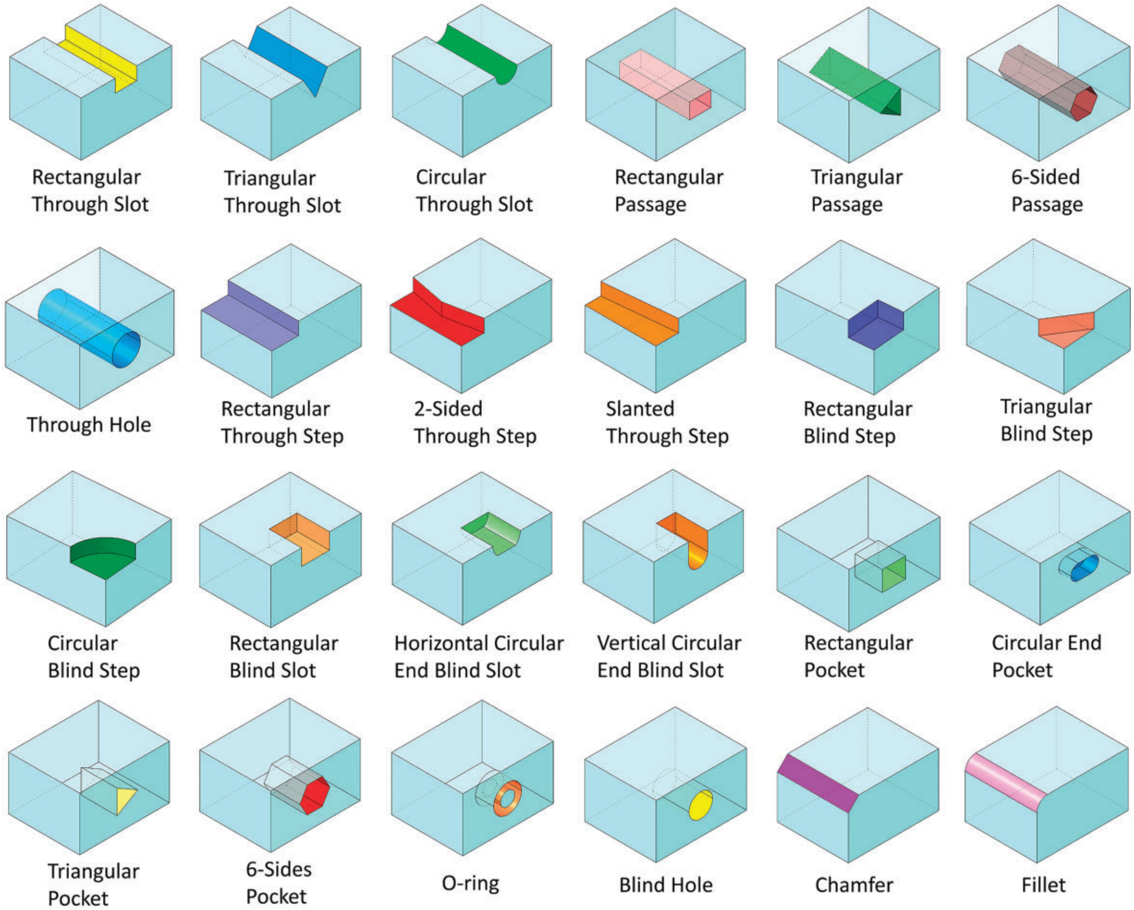


Figure 1: Detailed features.

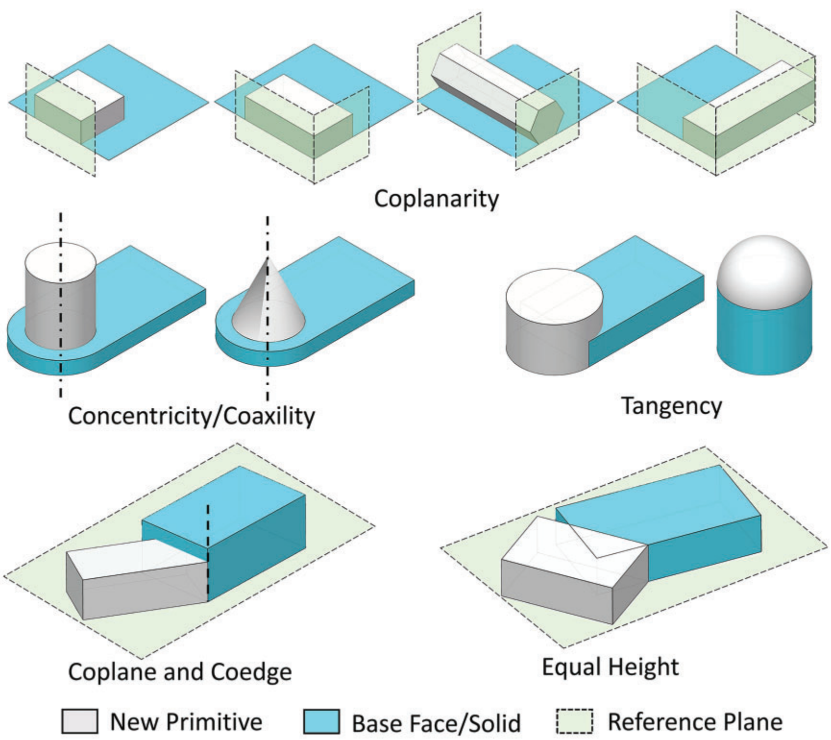


Figure 2: General spatial geometric constraints.

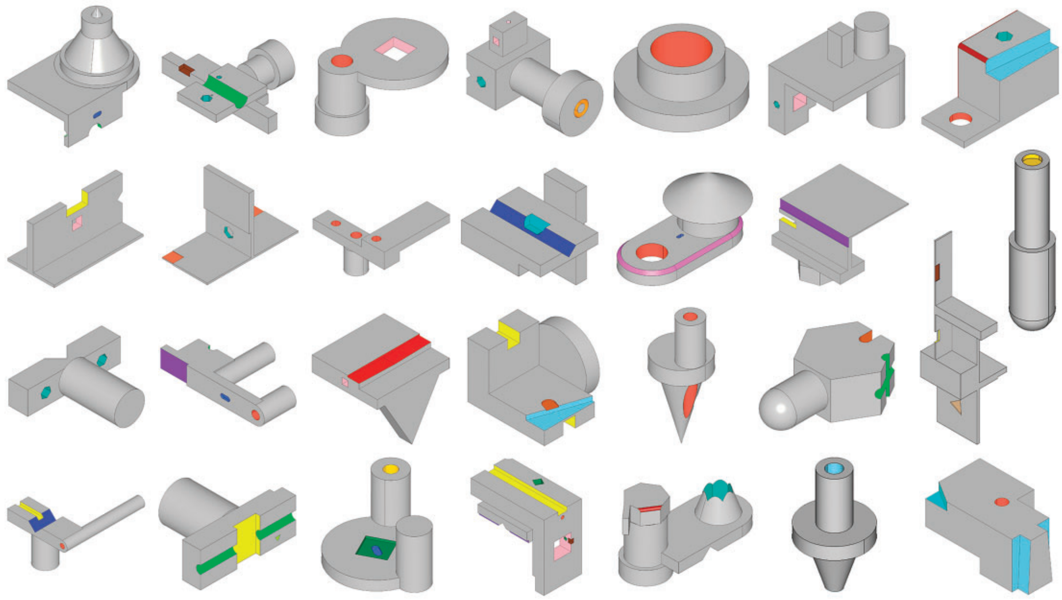


Figure 3: Samples from our synthetic CAD dataset.

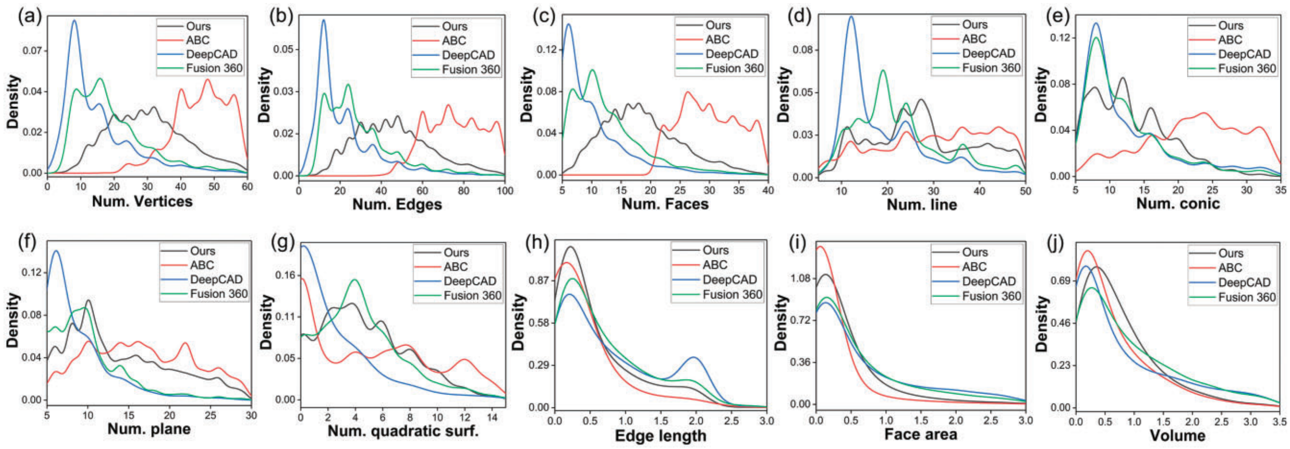


Figure 4: Comparison of distributions computed from our dataset, ABC dataset, DeepCAD dataset, and Fusion 360 dataset.

cubes with a larger cube or merging two coaxial and adjacent cylinders into one cylinder.

(ii) To exclude scenarios where one principal primitive is entirely covered by another, the intersecting volume between any two principal primitives is calculated. This ensures that each principal primitive represents an advantageous half-space of the B-rep model, meeting the minimal redundancy condition of "prime implicants" (Shapiro & Vossler, 1993).

(iii) To determine the order of principal primitives, evaluation metrics such as "proximity" (Shapiro & Vossler, 1993) and "unique representation" (Anderson *et al.*, 1988) from CSG optimization methods are referenced. The principal primitive with the largest volume is placed at the start of the modeling sequence, and subsequent primitives are selected from the remaining set based on adjacency to the current solid and having a larger volume (in cases of equal volume, sorting is based on the ascending lexicographical order of the 3D coordinates of their center points).

(iv) The 24 detailed features are categorized into six groups and added in the following order: step, through slot, hole, blind slot, blind hole, and fillet. For detailed features within the same group, they are arranged in ascending lexicographical order based on the coordinates of their starting feature point p_1.

Our synthesized dataset consists of 1 million CAD designs, and Fig. 3 displays some CAD models sampled from it. To evaluate the realism of the synthesized CAD models, we further perform a comparative analysis between our dataset and several representative real-world CAD datasets (ABC, DeepCAD, and Fusion 360 Gallery). This analysis involves the examination of geometric attributes, following the methodology proposed by Jayaraman *et al.* (2022) and Willis *et al.* (2021). The results are presented in Fig. 4, where the curve is fitted using Gaussian kernel density estimation. The number of geometric elements in each CAD model can give a good indication of the complexity of the dataset. Figures 4(a)–(c) illustrate the distribution of the number of three fundamental topological elements: vertices, edges, and faces. In our dataset, the peak values fall between the ABC dataset and Fusion360 dataset, with approximately 20–30 vertices, 30–50 edges, and 15–20 faces. Regarding the geometric attributes of edges and

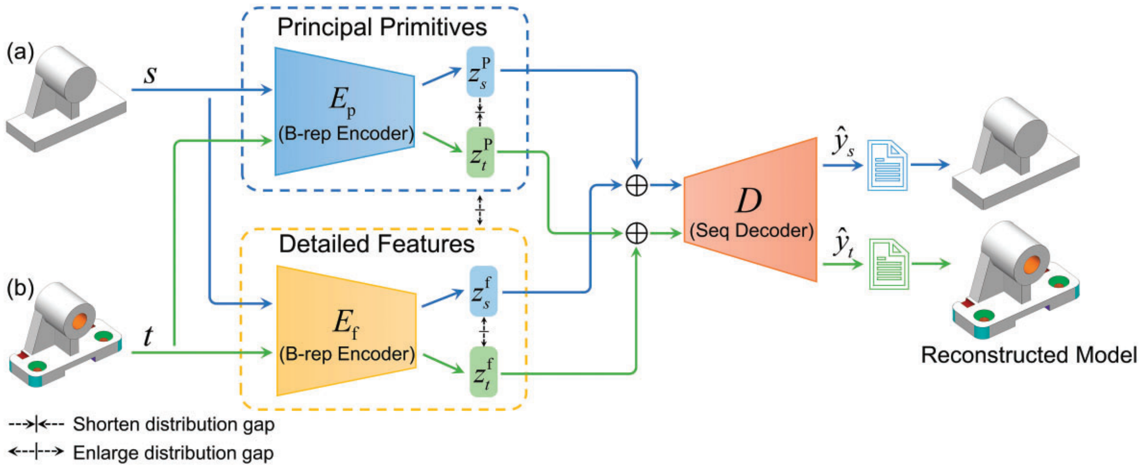


Figure 5: The hierarchical network architecture of Brep2Seq.

faces, the distribution in our dataset is closer to that of the Fusion 360 dataset. Finally, concerning the distribution of curve length, surface area, and solid volume (Figs 4h–j), all four datasets exhibit a close alignment, indicating that our dataset can capture some geometric attributes inherent to real CAD models.

4. Method

4.1. Network architecture

When observing, comprehending, or constructing a new 3D object, people usually focus on the primary shape before considering its detailed features. This pattern has guided our research on B-rep model representation learning. We believe that CAD models consist of two parts, primary shapes and detailed features, whether they are in synthetic datasets or real CAD models. These two categories of features exhibit distinct geometric shapes and local topological characteristics. Furthermore, within our parameterized construction sequences, there exists a differentiation in the label space for these two feature types. Therefore, to achieve a more detailed control over the modeling process, we partition the challenge of B-rep model representation learning into the extraction of feature representations for primary shapes and detailed features.

Inspired by orthogonal spatial decomposition theories and decoupled representation learning methods (Jia *et al.*, 2010; Salzmann *et al.*, 2010; Sanchez *et al.*, 2020), we adopt a hierarchical network architecture to explicitly encode the primary shape and detailed features of B-rep models, as shown in Fig. 5. During training, our synthetic dataset provides N pairwise B-rep models, where model s_i contains only primary shape features (as shown in Fig. 5a), and the other model t_i, based on the former, includes additional detailed features (as shown in Fig. 5b).

$$\mathcal{M} = \{m_i\}_{i=1}^{N}, \quad m_i = (s_i, t_i) \tag{1}$$

Here, both s_i and t_i represent B-rep data of CAD models, sharing the same value range and parameter space.

Within the hierarchical network, two B-rep encoders project B-rep data into two orthogonal latent subspaces, $\mathcal{W}_p$ and $\mathcal{W}_f$, as indicated by equations (2) and (3):

$$z_{s,i}^{p} = E_p(s_i), \; z_{t,i}^{p} = E_p(t_i), \; z_{s,i}^{p}, z_{t,i}^{p} \in \mathcal{W}_p \tag{2}$$

$$z_{s,i}^{f} = E_f(s_i), \; z_{t,i}^{f} = E_f(t_i), \; z_{s,i}^{f}, z_{t,i}^{f} \in \mathcal{W}_f. \tag{3}$$

The principal primitive B-rep encoder E_p is used to identify the commonalities between pairwise CAD models s_i and t_i, that is the primary shape of the models. This results in embeddings $z_{s,i}^{p} \sim P_\phi\left(z_{s,i}^{p}|s_i\right)$ and $z_{t,i}^{p} \sim P_\phi\left(z_{t,i}^{p}|t_i\right)$. The detailed feature B-rep encoder E_f is employed to identify the differences between s_i and t_i, which correspond to the models' detailed features, and produces embeddings $z_{s,i}^{f} \sim P_\psi\left(z_{s,i}^{f}|s_i\right)$ and $z_{t,i}^{f} \sim P_\psi\left(z_{t,i}^{f}|t_i\right)$. This explicit construction of latent spaces for primary shape and detailed features enhances the capability to capture different modal feature information. It also makes the learned feature representations more explicit for downstream applications.

We sum z^p and z^f, and feed them into the sequence decoder D to obtain the parameterized feature-based construction sequence $\hat{y}$ of the original B-rep model.

$$\hat{y}_{s,i} = D\left(z_{s,i}^{p} + z_{s,i}^{f}\right), \quad \hat{y}_{t,i} = D\left(z_{t,i}^{p} + z_{t,i}^{f}\right) \tag{4}$$

In the principal primitive latent subspace $\mathcal{W}_p$, embeddings $z_{s,i}^{p}$ and $z_{t,i}^{p}$ of pairwise B-rep models with the same primary shape should be as similar as possible. Similarity can be described by measuring the distribution distance:

$$\min_{E_p} Dist\left(P_\phi\left(z_s^p|s\right), P_\phi\left(z_t^p|t\right)\right) \tag{5}$$

The distribution distance $Dist(\cdot)$ can be Jensen–Shannon divergence (JSD) or maximum mean discrepancy (Gretton *et al.*, 2006). To minimize the distribution distance, we incorporate a similarity loss function during the training of the neural network, which is specifically defined in Section 4.2.2

In the detailed feature latent subspace $\mathcal{W}_f$, embeddings $z_{s,i}^{f}$ and $z_{t,i}^{f}$ representing detailed features of pairwise B-rep models (one with detailed features, the other without) should be as dissimilar as possible. Furthermore, two latent subspace $\mathcal{W}_p$ and $\mathcal{W}_f$ reflect two opposite aspects of B-rep information, and they should be separated as much as possible. We achieve the separation of these feature subspaces by imposing orthogonal constraints between them. Assuming the batch size for training the neural network is N, the tensor representations of the principal primitive and detailed features are as expressed in equations (6) and (7). We will utilize the difference loss function defined in Section 4.2.3 to ensure the orthogonality of $\{Z_s^f, Z_t^f\}$, $\{Z_s^p, Z_s^f\}$ and $\{Z_t^p, Z_t^f\}$.

$$Z_s^p = \left[z_{s,1}^{p}, \cdots, z_{s,N}^{p}\right]^T, \quad Z_t^p = \left[z_{t,1}^{p}, \cdots, z_{t,N}^{p}\right]^T \in \mathbb{R}^{N\times d_z} \tag{6}$$

$$Z_s^f = \left[z_{s,1}^{f}, \cdots, z_{s,N}^{f}\right]^T, \quad Z_t^f = \left[z_{t,1}^{f}, \cdots, z_{t,N}^{f}\right]^T \in \mathbb{R}^{N\times d_z} \tag{7}$$

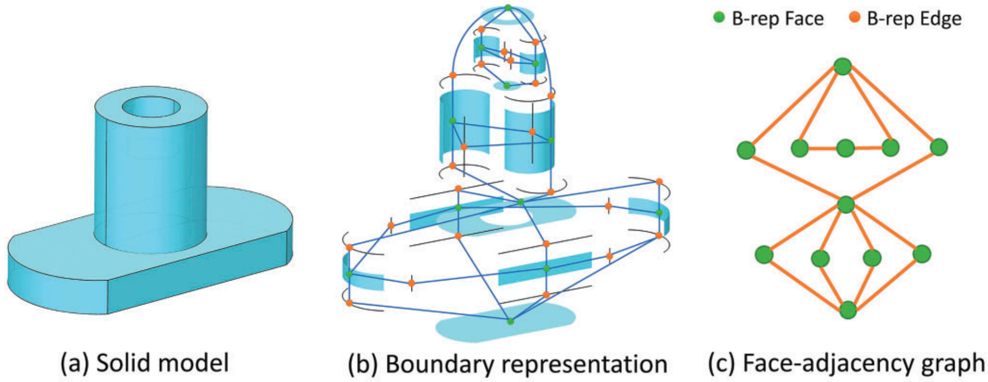


Figure 6: Different representations of a CAD model: (a) solid model; (b) B-rep describes the solid model through a closed boundary comprising vertices, edges, and faces; and (c) B-rep data can be converted into a face-adjacency graph.

4.1.1. *B-rep encoder*

Unlike regular data such as voxels or images, the B-rep data of a CAD model are hierarchical and irregular. Figure 6 illustrates three different representations of a CAD model. The B-rep of a 3D solid model mainly comprises a set of faces $f = \{f_1, f_2, \ldots, f_{|f|}\}$ and a set of edges $e = \{e_1, e_2, \ldots, e_{|e|}\}$. In our work, the structural information of B-rep data can be expressed using an undirected graph $\mathcal{G} = \{\mathrm{f}, \mathrm{e}\}$, where graph nodes represent B-rep faces and graph edges represent B-rep edges. The topological relationship between faces and edges can be described by two relationship matrices: $F \in \{0, 1\}^{|f| \times |f|}$ indicates the adjacent relationship between pairs of faces, while $E \in \{0, 1\}^{|e| \times |f|}$ indicates the adjacent relationship between edges and faces. If face f_i and face f_j are connected by an edge, the value of F_{ij} is 1, otherwise it is 0. If edge e_i is connected to face f_j, the value of E_{ij} is 1, otherwise it is 0.

Inspired by graphormer (Ying *et al.*, 2021), the latest graph convolution method based on the transformer, we adopt its principles and propose an encoding method to achieve the representation learning of CAD model, enable comprehensive extraction of both geometric and topological information from B-rep data. The encoder-decoder architecture of Brep2Seq is shown in Fig. 7.

To conduct representation learning on the B-rep graph, we first need to acquire feature representations for the graph nodes. Each topological face is described as a trimmed surface, composed of a base surface and boundary loops. Consequently, feature representations for the topological face are extracted by considering the following four aspects.

(i) Encoding the 3D geometric shape. For generality, we initially convert the base surface into NURBS parameter surfaces. Following the approach of UV-Net (Jayaraman *et al.*, 2021), we uniformly sample 10 × 10 2D grid points along both u and v directions in the parameter space and then calculate the local features of these grid points, including (i) the 3D point coordinates (x, y, and z), (ii) 3D absolute surface normals consistently pointing outwards, and (iii) a trimming mask with values of 1 and 0, representing samples in the visible region and trimmed region. Subsequently, a CNN processes 2D UV-grids with typically seven channels (three xyz, three normals, and one trimming mask) to encode the geometry shape and outputs the feature vector $x_{\mathrm{geom}(f_i)}$.

(ii) Encoding the surface type. Specifically, we obtain $x_{\mathrm{type}(f_i)}$ through an embedding layer $x_{\mathrm{type}(f_i)} = W_{\mathrm{type}}\delta_i$. Here, $W_{\mathrm{type}} \in \mathbb{R}^{d_x \times 6}$ is a learnable matrix and $\delta_i \in \mathbb{R}^6$ is a one-hot vector indicating the origin surface type among the six surface types (plane, cylinder, cone, sphere, torus, and B-spline surface).

(iii) Encoding the surface area. Since we limit the range of the CAD model's bounding box in the xyz-axis to [−1, 1], the maximum area of a topological face does not exceed 24. Therefore, we can discretize the continuous surface area values into 256 dimensions one-hot vector $\tau_i \in \mathbb{R}^{256}$ within its range [0, 24]. Then, we can obtain $x_{\mathrm{area}(f_i)}$ through an embedding layer $x_{\mathrm{area}(f_i)} = W_{\mathrm{area}}\tau_i$.

(iv) Encoding the number of adjacent faces. We follow graphormer's centrality encoding (Ying *et al.*, 2021) to acquire the learnable embedding vectors $x_{\mathrm{adj}(f_i)}$ based on the number of adjacent faces of each face.

In summary, we formulate the feature representation of graph nodes in the input layer: it is a sum of the above four feature vectors, that is,

$$H^{(0)} = \left[h_1^{(0)\mathrm{T}}, h_2^{(0)\mathrm{T}}, \ldots, h_n^{(0)\mathrm{T}}\right]^{\mathrm{T}} \in \mathbb{R}^{n \times d_h},$$
$$h_i^{(0)} = x_{\mathrm{geom}(f_i)} + x_{\mathrm{type}(f_i)} + x_{\mathrm{area}(f_i)} + x_{\mathrm{adj}(f_i)} \tag{8}$$

where $x_{\mathrm{geom}(f_i)}$ represents the geometric shape, $x_{\mathrm{type}(f_i)}$ represents the base surface type, $x_{\mathrm{area}(f_i)}$ represents the face area, $x_{\mathrm{adj}(f_i)}$ represents the centrality of graph node, and the dimension $d_h = 256$ in our work.

By utilizing the topological relationship matrices F and E mentioned above, we can get the shortest distance between any two B-rep faces on the graph, resulting in the graph distance matrix $D_g \in \mathbb{Z}^{|f| \times |f|}$. Subsequently, the encoding of graph distance can be obtained through an embedding layer:

$$A_1 \in \mathbb{R}^{|f| \times |f| \times d_E}. \tag{9}$$

To represent the positional relationship between any two faces in 3D Euclidean space, we calculate the marginal distributions of D2 distance and A3 distance using the method introduced by Rea *et al.* (2005). 512 pairs of points are randomly sampled between two faces, and the ratio of the Euclidean distance between each point pair to the diagonal length of the CAD model's bounding box is calculated. The interval [0,1] is divided into d_E sub-intervals, and the approximate expression of D2 distance, $d^{\mathrm{D2}}(f_i, f_j) \in \mathbb{R}^{d_E}$, can be obtained by counting the frequency of the ratio value within each sub-interval. Likewise, the approximate expression of the A3 distance, $d^{\mathrm{A3}}(f_i, f_j) \in \mathbb{R}^{d_E}$, can be obtained. Consequently, the positional relationship in Euclidean space can be represented by the

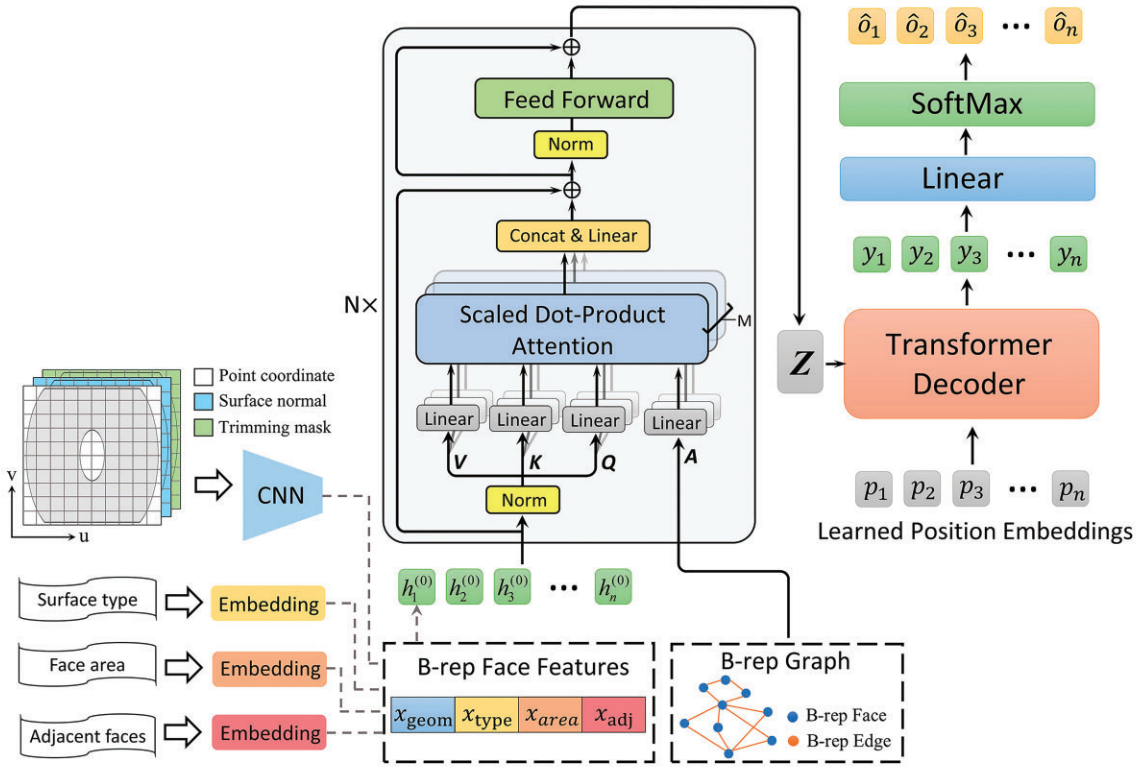


Figure 7: The encoder-decoder architecture of Brep2Seq.

following encoding:

$$A_2 \in \mathbb{R}^{|f|\times|f|\times d_E},\ A_{2(i,j)} = d^{D2}(f_i, f_j) + d^{A3}(f_i, f_j). \tag{10}$$

Whether the common edge between directly adjacent faces or the faces and edges on the shortest graph path between two non-adjacent faces may reveal some important geometric and topological information. For any ordered pair of faces, we can get their shortest path $\{f_0, e_1, f_1, e_2, \ldots f_{N-1}, e_N, f_N\}$ on the graph. The relationship between two adjacent faces f_i and f_{i-1} on the shortest path can be expressed as follow:

$$a_i = \phi\left((1+\varepsilon)x_{e_i} + f_\Theta\left(x_{f_{i-1}} + x_{f_i}\right)\right) \tag{11}$$

where ϕ is the multi-layer perceptron (MLP), ε is the learnable weight coefficient, f_Θ is the linear layer, x_{e_i} is the feature vector of the ith edge e_i, and x_{f_i} is the feature vector of the ith face f_i.

The geometric and topological relationship between the ordered pair of faces (f_i, f_j) can be obtained by summarizing all the face-edge-face relationships along the shortest graph path:

$$A_3 \in \mathbb{R}^{|f|\times|f|\times d_E},\ A_{3(i,j)} = \frac{1}{N}\sum_{i=1}^{N}(a_i \odot w_i) \in \mathbb{R}^{d_E} \tag{12}$$

where $w_i \in \mathbb{R}^{d_E}$ is a learnable weight coefficient.

The B-rep encoder is built upon the transformer. We derive the hidden state $H^{(l)}$ of layer l from the hidden state $H^{(l-1)}$ of layer $l-1$ through a multi-head self-attention module (MultiHead) and a fully connected feedforward network (FFN). Additionally, the pre-layer normalization (LN) operation is employed.

$$H'^{(l)} = \mathrm{MultiHead}\left(\mathrm{LN}\left(H^{(l-1)}\right)\right) + H^{(l-1)},$$
$$H^{(l)} = \mathrm{FFN}\left(\mathrm{LN}\left(H'^{(l)}\right)\right) + H'^{(l)}. \tag{13}$$

There are M self-attention modules in each sub-layer:

$$\mathrm{MultiHead}(H) = \mathrm{Concat}(\mathrm{head}_1, \ldots, \mathrm{head}_M)W^O \tag{14}$$

$$\begin{aligned}\mathrm{head}_m &= \mathrm{self-att}(H, A_1, A_2, A_3)\\ &= \mathrm{softmax}\left(\frac{Q_m K_m^T}{\sqrt{d_k}} + A_1(W_{a1}^m)^T + A_2(W_{a2}^m)^T + A_3(W_{a3}^m)^T\right)V_m,\\ &\forall m \in \{1, \ldots, M\},\ Q_m = HW_q^m,\ K_m = HW_k^m,\ V_m = HW_v^m\end{aligned} \tag{15}$$

where $W^O \in \mathbb{R}^{Md_v\times d_h}$ is the output projection matrix; $W_q^m \in \mathbb{R}^{d_h\times d_k}$, $W_k^m \in \mathbb{R}^{d_h\times d_k}$, and $W_v^m \in \mathbb{R}^{d_h\times d_v}$ are the projection matrices; $W_{a1}^m \in \mathbb{R}^{d_E}$, $W_{a2}^m \in \mathbb{R}^{d_E}$, and $W_{a3}^m \in \mathbb{R}^{d_E}$ are weight embedding matrices, and the dimension $d_E = 64$.

To obtain the graph embedding of B-rep data, our encoder employs the graph pooling function introduced by Gilmer *et al.* (2017). We set up a virtual node connected with all nodes, and use the feature vector of this virtual node in the last layer as the feature representation for the CAD model.

4.1.2. *Sequence decoder*

The modeling process of a CAD model can be described as a sequence consists of modeling operations that need to be executed sequentially. The construction sequence predicted by the Brep2Seq sequence decoder can be represented in the following form:

$$M = [O_1, \cdots, O_{N_p+N_f}]. \tag{16}$$

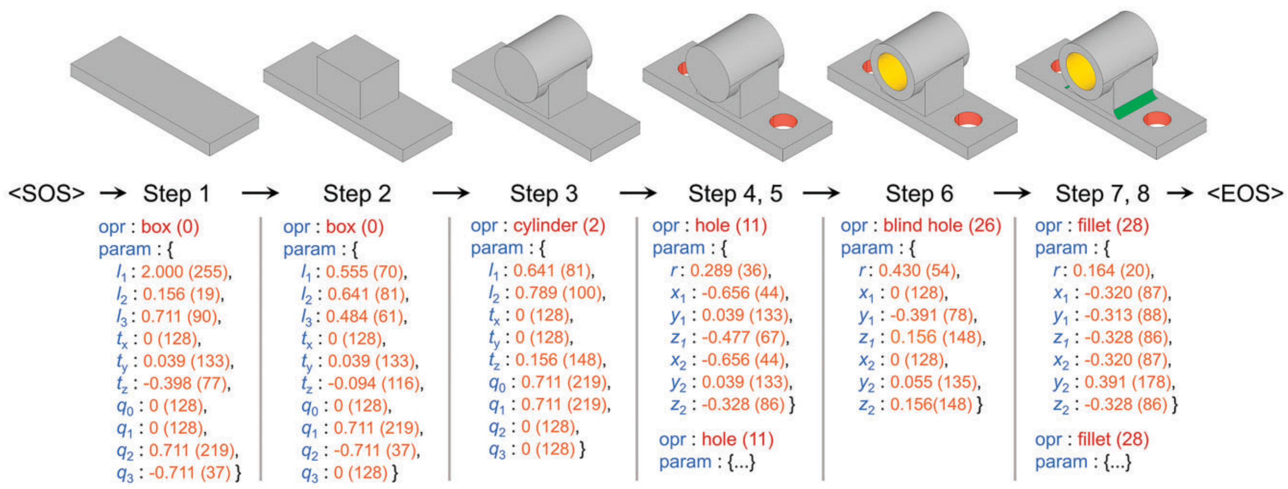


Figure 8: Modeling process of a CAD model and the corresponding CAD construction sequence.

Each modeling operation O_i is described by two types of parameters: operation type o_i and design parameters p_i, i.e.,

$$O_i = (o_i, p_i)\,. \tag{17}$$

To make the construction sequence suitable for neural network training, several problems need to be addressed:

(i) The numbers of modeling operations required for constructing different CAD models vary.
(ii) The numbers of parameters involved in different modeling operations vary.
(iii) Some parameters are continuous, while others are discrete, and their value ranges differ.

Therefore, we have adopted a network-friendly representation (Wu *et al.*, 2021), which regularizes the dimensions of the modeling sequence to ensure compatibility with the transformer network. Firstly, the number of modeling operations within the sequence is fixed, encompassing N_p principal primitives and N_f detailed features. Additionally, we insert a <SOS> token at the start of the sequence and an <EOS> token at the end of the sequence. If the number of modeling operations is less than the maximum sequence length, the sequence is padded with empty <EOS> commands until reaching the designated length. In our work, we set N_p and N_f to 8 and 10, respectively, aligning with the complexity of CAD models in the synthetic dataset, which consists of a maximum of eight principal primitives and 10 detailed features.

Second, we use a fixed-length vector to represent the design parameter of modeling operations. The parameter vector for principal primitives is denoted as $p_i = [l_1, l_2, l_3, e, t_x, t_y, t_z, q_0, q_1, q_2, q_3] \in \mathbb{R}^{11}$, while the parameter vector for detailed features is denoted as $p_i = [w, l, r, w_1, d, x_1, y_1, z_1, x_2, y_2, z_2] \in \mathbb{R}^{11}$. The specific definitions of the elements in the parameter vector can be found in Sections 3.1 and 3.2. Unused parameters are set to -1, and the <SOS> and <EOS> tokens correspond to parameter vectors with values of -1.

Finally, we normalize the data, scaling the range of the bounding box of each CAD model in the XYZ axes to $[-1,1]$. This normalization restricts the ranges of parameters for principal primitives or detailed features, allowing us to quantize their values into 256 levels.

The sequence decoder takes the latent vector $z \in \mathbb{R}^{d_z}$ as input and generates the feature vectors $Y = [y_1^T, y_2^T, \ldots, y_{N_p+N_f}^T]^T \in \mathbb{R}^{(N_p+N_f)\times d_y}$, in which each feature vector $y_i \in \mathbb{R}^{d_y}$ represents a CAD modeling operation. Next, feature vector y_i is passed through two linear layers $FC(d_y, N_{opr})$ and $FC(d_y, N_{param} \times 257)$ to obtain $y_i^{opr} \in \mathbb{R}^{N_{opr}}$ and $y_i^{param} \in \mathbb{R}^{N_{param}\times 257}$, and the softmax function is used to derive the specific operation type o_i and design parameter p_i of the modeling operation. The dimensions d_z and d_y are 256 in our work. $N_{opr} = 5 + 24 + 2 = 31$ represents the number of all modeling operation types and token <SOS> and <EOS>, and $N_{param} = 11$ is the length of the parameter vector. Figure 8 shows the modeling process of a CAD model and the corresponding CAD construction sequence.

4.2. Training

The B-rep encoder and sequence decoder are jointly trained. The loss function consists of three components: the CAD construction sequence reconstruction loss function, the difference loss function, and the similarity loss function, as depicted in equation (18), where β and γ are weight coefficients used to balance all terms ($\beta = \gamma = 0.25$ in our experiments).

$$\mathcal{L} = \mathcal{L}_{recon} + \beta\mathcal{L}_{difference} + \gamma\mathcal{L}_{similarity}. \tag{18}$$

4.2.1. *Reconstruction loss function*

We utilize the standard cross-entropy function to measure the difference between the CAD construction sequence predicted by network and ground truth:

$$\mathcal{L}_{recon} = \mathcal{L}_{primitives} + \lambda\mathcal{L}_{features} \tag{19}$$

$$\mathcal{L}_{primitive} = \sum_{i=1}^{N_p} \mathrm{CrossEntropy}(o_i, \hat{o}_i) + \sum_{i=1}^{N_p}\sum_{j=1}^{N_{param}} \mathrm{CrossEntropy}(p_{i,j}, \hat{p}_{i,j}) \tag{20}$$

$$\mathcal{L}_{feature} = \sum_{i=1}^{N_f} \mathrm{CrossEntropy}(o_i, \hat{o}_i) + \sum_{i=1}^{N_f}\sum_{j=1}^{N_{param}} \mathrm{CrossEntropy}(p_{i,j}, \hat{p}_{i,j}) \tag{21}$$

where λ is weight coefficient used to balance both reconstruction loss functions ($\lambda = 2$ in our experiments).

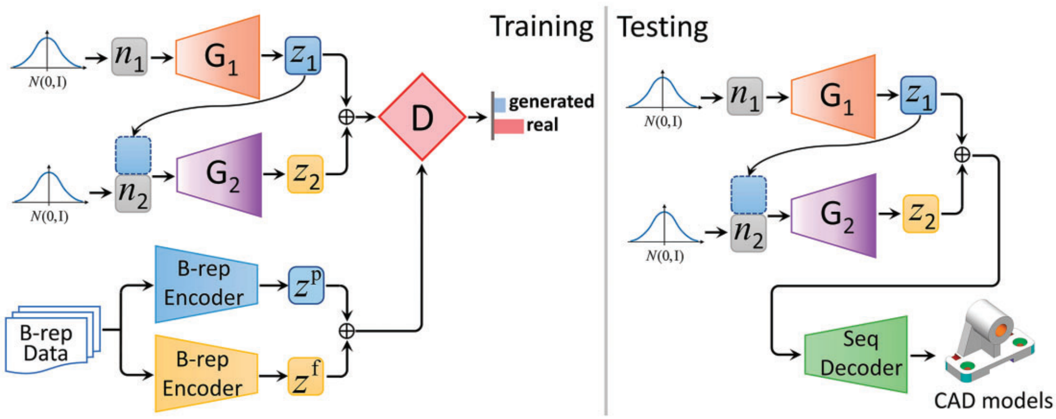


Figure 9: The architecture of GAN for CAD model generation.

4.2.2. *Similarity loss function*

We employ a similarity loss function to minimize the dissimilarity between the latent vectors Z_s^p and Z_t^p. The similarity loss function can be calculated using either the maximum mean difference (MMD) method (Gretton *et al.*, 2006) or the domain adversarial neural networks (DANN) method (Ganin *et al.*, 2016).

In our work, we use kernel functions to calculate the MMD between latent vectors Z_s^p and Z_t^p:

$$\mathcal{L}_{\text{similarity}}^{\text{MMD}} = \frac{1}{(N_b)^2}\sum_{i,j=0}^{N_b}\mathcal{K}\left(z_{s,i}^p, z_{s,j}^p\right) - \frac{2}{(N_b)^2}\sum_{i,j=0}^{N_b}\mathcal{K}\left(z_{s,i}^p, z_{t,j}^p\right) + \frac{1}{(N_b)^2}\sum_{i,j=0}^{N_b}\mathcal{K}\left(z_{t,i}^p, z_{t,j}^p\right) \tag{22}$$

where $\mathcal{K}(x_i, x_j)$ is the kernel function that satisfies the inner product relationship $\mathcal{K}(x_i, x_j) = \langle\phi(x_i), \phi(x_j)\rangle$, and N_b is the batch size. The kernel function that we use is the Gaussian kernel $\mathcal{K}(x_i, x_j) = \sum_n \eta_n \exp\left(-\|x_i - x_j\|^2/2\sigma_n^2\right)$. Detailed definition of MMD method is present in the Appendix 1.

The DANN method is a kind of adversarial training method in the field of transfer learning. It uses the gradient reversal layer (GRL) $\mathcal{Q}$ to make the domain discriminator $\mathcal{D}$ unable to accurately judge whether the latent vector is from the source domain or the target domain. Given a function $f(u)$, the GRL is defined as $\mathcal{Q}(f(u)) = f(u)$, $\frac{d}{du}\mathcal{Q}(f(u)) = -\frac{d}{du}f(u)$. Our work uses this method to make the distribution of the latent vectors $z_{s,i}^p$ and $z_{t,i}^p$ as close as possible and defines the similarity loss function based on the DANN method as

$$\mathcal{L}_{\text{similarity}}^{\text{DANN}} = \sum_{i=0}^{N_b}\log\left[\mathcal{D}\left(\mathcal{Q}\left(z_{s,i}^p\right)\right)\right] + \sum_{i=0}^{N_b}\log\left[1 - \mathcal{D}\left(\mathcal{Q}\left(z_{t,i}^p\right)\right)\right] \tag{23}$$

where the discriminator $\mathcal{D}$ is implemented as a MLP.

4.2.3. *Difference loss function*

We calculate the difference loss function between the latent vectors by utilizing the Frobenius norm of the matrix:

$$\mathcal{L}_{\text{difference}} = \left\|\left(Z_s^f\right)^T Z_t^f\right\|_F^2 + \left\|\left(Z_s^p\right)^T Z_s^f\right\|_F^2 + \left\|\left(Z_t^p\right)^T Z_t^f\right\|_F^2. \tag{24}$$

4.3. Generation

To automate the generation of CAD models, we employ a generation adversarial network (GAN) to learn the distribution $P_r(z)$ of latent vector z representing the real-world 3D CAD model. In our work, the GAN comprises a generator and a discriminator, both implemented as MLPs. The generator $G(n;\theta)$ takes a random vector $n \in \mathbb{R}^{d_n}$ drawn from a normal distribution as input and produces a latent vector $z \in \mathbb{R}^{d_z}$ that represents the CAD model. The discriminator $D(z;\phi)$ assesses whether the latent vector z conforms to the distribution of the ground-truth CAD model in the feature space. Figure 9 illustrates the architecture of our GAN. Consistent with the hierarchical B-rep encoder, the generator is divided into two layers: $G_1(n_1;\theta_1)$ and $G_2(z_1, n_2;\theta_2)$. The summation of generated hidden vectors z_1 and z_2 is used as the input for discriminator D.

$$z_1 = G_1(n_1;\theta_1),\ n_1 \sim N(0, I), z_1 \sim P_{\theta_1}(z_1) \tag{25}$$

$$z_2 = G_2(z_1, n_2;\theta_2),\ n_2 \sim N(0, I), z_2 \sim P_{\theta_2}(z_2|z_1). \tag{26}$$

During training, we employ Wasserstein distance (Arjovsky *et al.*, 2017) as a metric to quantify the dissimilarity between the latent vector generated by the generator and the latent vector representing the real CAD model. In each iteration, the trained B-rep encoder is employed to encode the hidden vectors z^p and z^f, which respectively represent the principal primitives and detailed features of the real CAD model, and the GAN is optimized based on the corresponding objective function as follows.

The optimization objective of discriminator D is donated as equation (27). It aims to assign the highest possible score to the latent vector of the real CAD model, while assigning the lowest possible score to the latent vector generated by the generator G.

$$\max_{\phi}\left\{\mathbb{E}_{z^p,z^f}\left[\log D\left(z^p + z^f\right)\right] + \mathbb{E}_{z_1\sim P_{\theta_1}, z_2\sim P_{\theta_2}(z_2|z_1)}\left[\log\left(1 - D(z_1 + z_2)\right)\right]\right\}. \tag{27}$$

The optimization objective of generator G is donated in equation (28). It aims to maximize the score assigned by discriminator D to the latent vector generated by generator G:

$$\max_{\theta_1,\theta_2}\mathbb{E}_{n_1,n_2\sim N(0,I)}\left\{\log D\left(G_1(n_1) + G_2(G_1(n_1) + n_2)\right)\right\}. \tag{28}$$

5. Experiments and Results

5.1. Evaluation metrics

In order to evaluate the performance of the Brep2Seq network in the tasks of CAD model reconstruction, we adopt the following evaluation metrics:

CAD construction sequence accuracy. We calculate the accuracy of modeling operation types and parameters by comparing the differences between the actual CAD construction sequence M and the predicted CAD construction sequence $\hat{M}$. The accuracy of

Table 2: Reconstruction results on our synthetic dataset and real-world datasets.

Dataset	ACC$_{\text{opr-primitive}}$↑	ACC$_{\text{param-primitive}}$↑	ACC$_{\text{opr-feature}}$↑	ACC$_{\text{param-feature}}$↑	Valid ratio↑	IoU↑	CD × 10^2↓
Synthetic	99.32	95.11	99.48	97.72	97.51	92.87	0.19
DeepCAD					88.13	70.23	3.35
Fusion 360					86.79	78.24	1.65

the CAD construction sequence of principal primitives is defined as follows:

$$ACC_{opr-primitive} = \frac{1}{N_p} \sum_{i=1}^{N_p} \mathbb{I}\left[o_i = \hat{o}_i\right] \tag{29}$$

$$ACC_{param-primitive} = \frac{1}{K} \sum_{i=1}^{N_p} \sum_{j=1}^{|\hat{p}_i|} \mathbb{I}\left[\left|p_{i,j} - \hat{p}_{i,j}\right| < \eta\right] \mathbb{I}\left[o_i = \hat{o}_i\right] \tag{30}$$

where $K = \sum_{i=1}^{N_p} \mathbb{I}\left[o_i = \hat{o}_i\right] |p_i|$ is the total number of parameters belonging to all correctly predicted operations, and $\mathbb{I}[\cdot] \in \{0, 1\}$ is the indicator function.

Similarly, the accuracy of the CAD construction sequence of detailed features is defined as follows:

$$ACC_{opr-feature} = \frac{1}{N_f} \sum_{i=1}^{N_f} \mathbb{I}\left[o_i = \hat{o}_i\right] \tag{31}$$

$$ACC_{param-feature} = \frac{1}{K} \sum_{i=1}^{N_f} \sum_{j=1}^{|\hat{p}_i|} \mathbb{I}\left[\left|p_{i,j} - \hat{p}_{i,j}\right| < \eta\right] \mathbb{I}\left[o_i = \hat{o}_i\right]. \tag{32}$$

Valid ratio. Due to the potential errors in the predicted results, the CAD model synthesized according to the CAD construction sequence may contain geometric and topological inaccuracies. To determine the validity of the CAD construction sequences, we employ the following criteria to calculate the corresponding validity ratio:

(i) The B-rep model should represent a watertight 3D manifold consisting of a single 3D solid.
(ii) The B-rep model should not contain minimal edges or minimal faces.
(iii) The boundary loops of all faces must be composed of coedges in the correct order.
(iv) All faces should not have self-intersecting boundary loops.

IoU measures the intersection over the union of reconstruction model $\hat{M}$ and actual model M:

$$\text{IoU}\left(\hat{M}, M\right) = \frac{\left|\hat{M} \cap M\right|}{\left|\hat{M} \cup M\right|} \tag{33}$$

where $|\cdot|$ represents the volume of intersection or union.

Chamfer Distance (CD, Fan *et al.*, 2017). We uniformly sampled 2000 points on the surface of the reconstructed model $\hat{M}$ and actual model M to calculate the CD value:

$$\text{CD}\left(\hat{M}, M\right) = \sum_{p \sim \hat{M}} \min_{p' \sim M} \left\|p - p'\right\|^2 + \sum_{p' \sim M} \min_{p \sim \hat{M}} \left\|p' - p\right\|^2 \tag{34}$$

In order to evaluate the performance of Brep2Seq in the task of CAD model generation, we employ the metrics introduced by Achlioptas *et al.* (2018): coverage (COV), minimum matching distance (MMD), and JSD. The calculation details of these metrics are described in Appendix 1.

5.2. Implementation details

Network settings. The Brep2Seq network is implemented in PyTorch and trained for 500 epochs with a batch size of 128 on an NVIDIA RTX 3080Ti GPU. The Brep2Seq network consists of six layers of transformer blocks for both the B-rep encoders and the sequence decoders. Each block includes sixteen attention heads, pre-LN, a feed-forward dimension of 512, and a dropout rate of 0.1. Surface CNNs in B-rep encoder is defined as: Conv(7, 64, 3) → Conv(64, 128, 3) → Conv(128, 256, 3) → Pool(1, 1) → FC(256, 64), where Conv(i, o, k) is an image convolutional layer with i input channels, o output channels, and kernel size k, Pool(n, n) is an adaptive average pooling layer which outputs a $n \times n$ feature map, and FC(i, o) is a fully connected layer which takes an input in i-D vector and maps it to o-D vector.

The AdamW optimizer with parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\varepsilon = 10^{-8}$, and weight decay coefficient = 0.01 is employed during training. The initial learning rate is set to 0.001, and a warm-up stage of 10 000 steps is used. We incorporate the ReduceLROnPlateau scheduler to vary the learning rate when the loss function value no longer decreases.

In our GAN model for CAD construction sequence generation, we adopt the same architecture as DeepCAD (Wu *et al.*, 2021), where the generator and discriminator are both MLPs with four hidden layers.

Dataset. We synthesized a dataset consisting of 1 million CAD models with ground-truth labels, following the methodology described in Section 3.3. The dataset is randomly partitioned into training (95%), validation (2.5%), and test (2.5%) sets. To enhance the training efficiency, we normalize the data and scale the bounding box range of each CAD model along the x, y, and z axis to [−1,1]. Furthermore, we discretize all the design parameters of principal primitives and detailed features, converting continuous parameters into 256-dimensional values within their respective ranges.

To evaluate the generalization ability of our Brep2Seq network, we employ two recent CAD datasets: DeepCAD (Wu *et al.*, 2021) and Fusion 360 Gallery (Willis *et al.*, 2021). We remove any duplicates or overly complex CAD models and extract the B-rep data of each model. Subsequently, we preprocess the data by scaling the bounding box range of each CAD model along the x, y, and z axis to [−1,1]. Consequently, we obtain a total of 159 740 models from DeepCAD and 7,015 models from Fusion 360 Gallery.

5.3. CAD model reconstruction

We evaluated the CAD model reconstruction performance of Brep2Seq on our synthetic dataset, presenting the results in Table 2 (top row) and Fig. 10. We observe that Brep2Seq achieved an accuracy of over 99% in predicting the types of CAD modeling operations and an accuracy of over 90% for the modeling operation parameters. This demonstrates Brep2Seq's ability to qualitatively identify the components of CAD models, including principal primitives and detailed features. Furthermore, Brep2Seq successfully reconstructs a significant proportion of valid models (97.51%) with high shape fidelity (IoU of 92.87% and CD of 0.19). Nevertheless, it should be noted that the predicted CAD modeling parameters are discrete values, whereas they are continuous in engineering practice. This difference introduces challenges in achieving complete

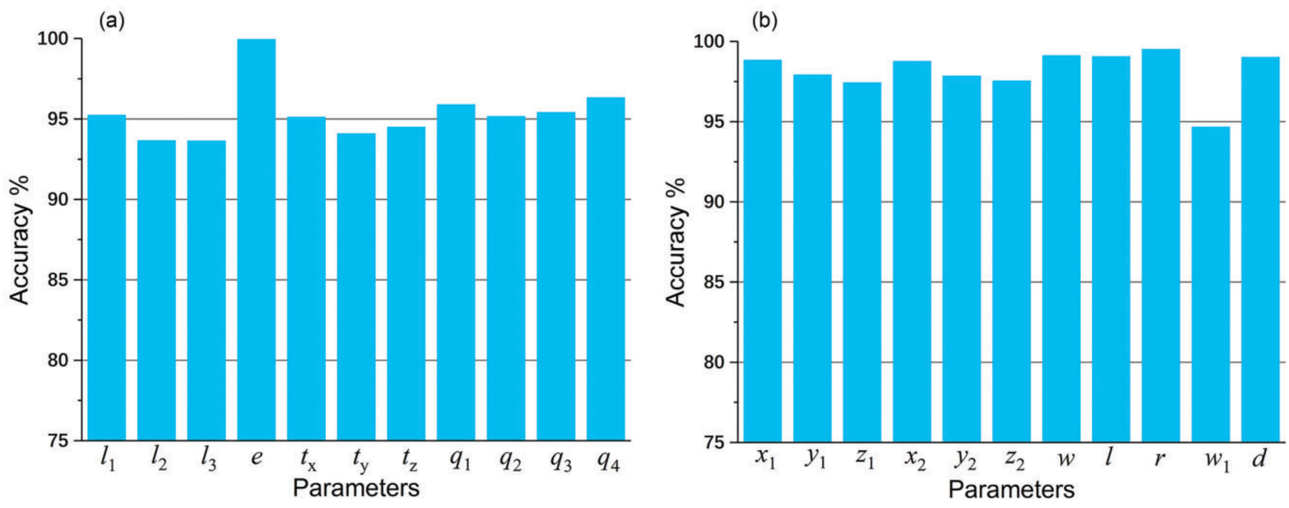


Figure 10: Reconstruction accuracies of CAD modeling parameters on our synthetic dataset: (a) principal primitives; (b) detailed features.

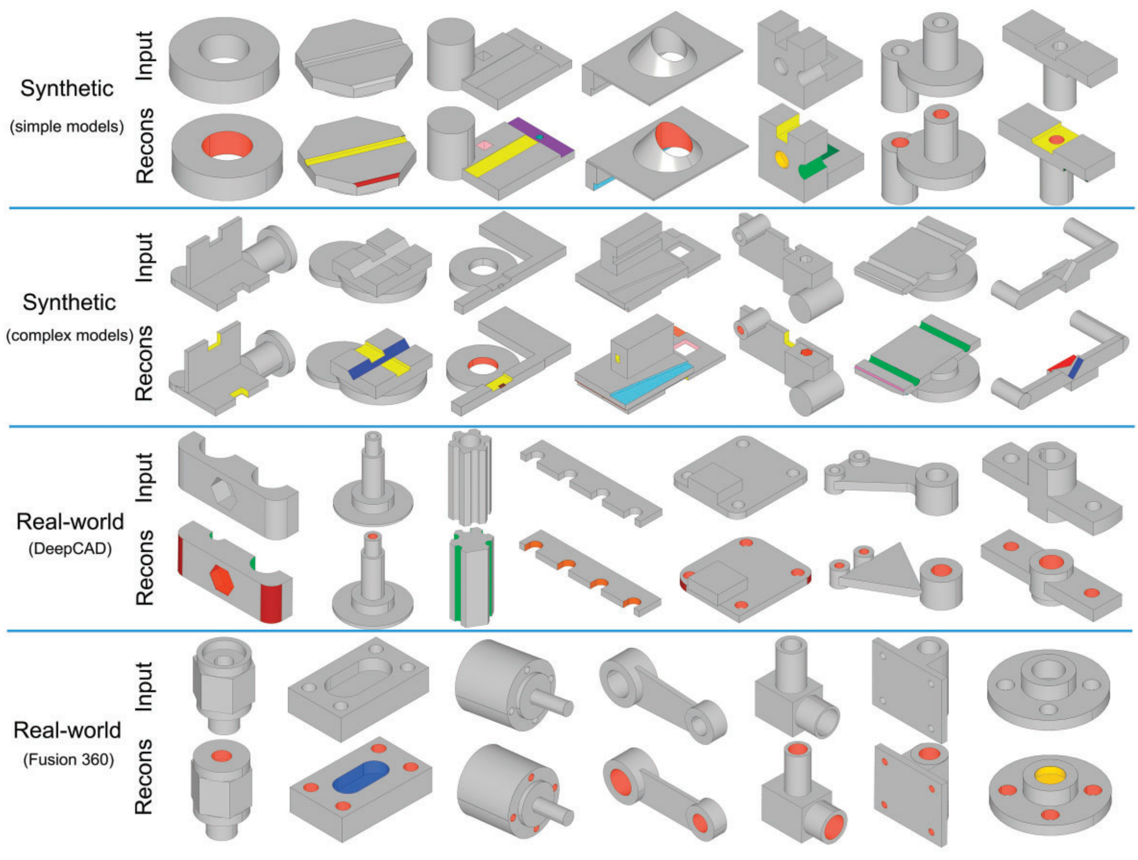


Figure 11: Examples for CAD model reconstruction. We show a selection of reconstructed models from our synthetic dataset and real-world datasets. Top two rows: Examples of CAD models in our synthetic dataset. Bottom two rows: Examples of manual modeling CAD models in DeepCAD and Fusion360 Gallery datasets.

consistency between the reconstructed model and the 3D shape of the target.

We further evaluated the performance of trained Brep2Seq on the DeepCAD and Fusion 360 Gallery datasets, and the corresponding test results are presented in Table 2 (bottom two rows). Due to the absence of parametrized feature-based modeling sequence labels in these datasets, the calculation of CAD Construction Sequence Accuracy is not feasible. The results indicate a decline in the performance of Brep2Seq on real-world CAD datasets, with the valid ratio decreasing by approximately 10%, and the geometric shape fidelity metrics, IoU and CD, also displaying different degrees of degradation.

To present reconstruction results qualitatively, we randomly sampled models from test set and display them in Fig. 11. Furthermore, we selected three CAD models from the Fusion360 Gallery dataset to show the reconstruction of modeling processes

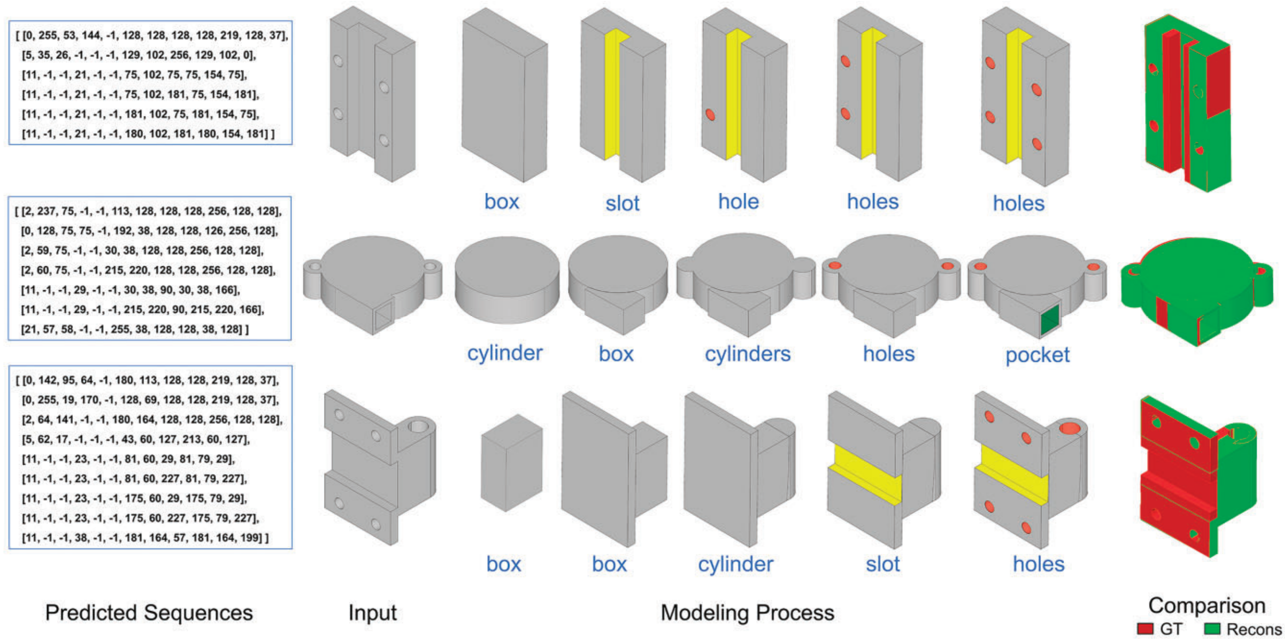


Figure 12: Examples for predicted construction sequences and modeling processes. We selected three CAD models from Fusion360 Gallery dataset to show the predicted construction sequences, the reconstruction process, and a direct comparison with the ground truth model.

in Fig. 12, illustrating the reconstructed models, the predicted construction sequences, the modeling process, and the comparison with the ground truth model.

5.3.1. *Ablation study*

To evaluate the validity of the neural network model design, parameters setting, and training strategy, we have designed the following ablation experiments. We compare the CAD model reconstruction performance on our synthetic dataset.

Comparison of different neural network representations. Brep2Seq utilizes a neural network based on the transformer architecture to realize the representation learning of B-rep, while previous works commonly employed GNNs based on the message-passing mechanism, as demonstrated by Cao *et al.* (2020), Jayaraman *et al.* (2021), and Colligan *et al.* (2022). Here we compare the learning capabilities of these two neural network representations in the task of CAD model reconstruction. To facilitate the comparison, we employ the UV-Net (Jayaraman *et al.*, 2021), which incorporates feature aggregation of faces and edges.

The results are reported in Table 3, where our Brep2Seq network is denoted as $\text{Ours}_{(\text{transformer})}$, and the GNN UV-Net is denoted as $\text{GNN}_{(\text{msg.pass})}$. Compared with the GNN based on message-passing, Brep2Seq achieves about 10% improvement in accuracy for all modeling parameters. Although the valid ratios of the two networks are close, Brep2Seq yields a 13.36% higher IOU value and 0.89 lower CD value. We believe that the larger receptive field and higher model complexity of the transformer network are advantageous in solving the representation learning problem of complex CAD models, while traditional message-passing GNNs can only "aggregate" and "combine" neighboring faces and edges adjacent to each B-rep face.

Comparison of different geometric and topological information. Unlike sequential data, B-rep data of CAD models are complex due to the presence of non-Euclidean geometric entities and discrete topological entities. Our work focuses on utilizing graph distance, Euclidean space distance, and face/edge geometric features to encode the structural information within the graph representation of B-rep. To assess the effectiveness of this neural network design, we explore various encoding methods in this ablation study.

In the first variant, we exclude all geometric information and focus solely on the essential topological information: each graph node is exclusively represented by the centrality embedding $x_{\text{adj}(f_i)}$ and the topological relationships between graph nodes are indicated by the graph distance embedding A_1 described in Section 4.1. We denote this approach as "Topo". In the second variant, we enhance each graph node by incorporating the surface type embedding $x_{\text{type}(f_i)}$ and face area embedding $x_{\text{area}(f_i)}$, and introduce the concave-convex property embedding of the coedge between each two adjacent B-rep faces, similar to the input information used in conventional graph decomposition methods based on face adjacency graphs. We denote this approach as "Topo++".

The results shown in Table 4 indicate that the "Topo" performed the worst among all the variants, with a 13.76% reduction in IoU and significant degradation in CD compared with our proposed method. Due to "Topo++" incorporating the geometric shape information, the accuracy of model reconstruction has been improved. This indicates that not only topology but also geometric shape and spatial position play a crucial role in improving the accuracy of CAD model reconstruction.

Comparison of different hierarchical network architectures. Four different hierarchical structure networks are employed to evaluate their capability in CAD construction sequences prediction. The "non-hierarchical" only utilizes a B-rep encoder and a sequence decoder to simultaneously predict the composition of principal primitives and detailed features in CAD models. The "parallel" employs two independent B-rep encoders and sequence decoders without constraint. The "$\text{Hierarchical}_{(\text{MMD})}$" incorporates a difference loss function and a similarity loss function based on the MMD method, whereas "$\text{Hierarchical}_{(\text{DANN})}$" employs a similarity loss function based on the DANN method.

The results presented in Table 5 demonstrate that the hierarchical network enhances prediction accuracy. Notably, the perfor-

Table 3: CAD model reconstruction results under different neural networks on synthetic dataset.

Method	$ACC_{opr\text{-}primitive}\uparrow$	$ACC_{param\text{-}primitive}\uparrow$	$ACC_{opr\text{-}feature}\uparrow$	$ACC_{param\text{-}feature}\uparrow$	Valid ratio↑	IoU↑	CD × $10^2\downarrow$
$\text{Ours}_{(\text{transformer})}$	**99.32**	**95.11**	**99.48**	**97.72**	**97.51**	**92.87**	**0.19**
$\text{GNN}_{(\text{msg.pass})}$	97.35	84.98	97.62	89.33	91.05	79.51	1.08

Table 4: CAD model reconstruction results with different geometric and topological information on synthetic dataset.

Method	$ACC_{opr\text{-}primitive}\uparrow$	$ACC_{param\text{-}primitive}\uparrow$	$ACC_{opr\text{-}feature}\uparrow$	$ACC_{param\text{-}feature}\uparrow$	Valid ratio↑	IoU↑	CD × $10^2\downarrow$
Topo	96.83	79.29	94.22	80.63	89.59	79.11	2.46
Topo++	97.78	86.24	96.17	89.47	93.61	85.94	0.97
Ours	**99.32**	**95.11**	**99.48**	**97.72**	**97.51**	**92.87**	**0.19**

Table 5: CAD model reconstruction results with different hierarchical network architectures on synthetic dataset.

Method	$ACC_{opr\text{-}primitive}\uparrow$	$ACC_{param\text{-}primitive}\uparrow$	$ACC_{opr\text{-}feature}\uparrow$	$ACC_{param\text{-}feature}\uparrow$	Valid ratio↑	IoU↑	CD × $10^2\downarrow$
Non-hierarchical	96.74	91.52	96.07	92.29	97.23	90.45	0.49
Parallel	96.08	91.83	95.45	91.77	96.85	90.15	0.78
$\text{Hierarchical}_{(\text{MMD})}$	98.64	94.12	98.99	97.13	**97.99**	92.23	0.25
$\text{Hierarchical}_{(\text{DANN})}$	**99.32**	**95.11**	**99.48**	**97.72**	97.51	**92.87**	**0.19**

Table 6: CAD model reconstruction results of DeepCAD network and Brep2Seq network trained on the DeepCAD dataset and Fusion 360 Gallery dataset.

Dataset	Method	Valid ratio↑	IoU↑	CD × $10^2\downarrow$
DeepCAD	DeepCAD	83.47	**76.77**	**1.37**
	Ours	**88.13**	70.23	3.35
Fusion 360	DeepCAD	75.68	36.93	10.1
	Ours	**86.79**	**78.24**	**1.65**

mance of unconstrained parallel networks is inferior to that of the non-hierarchical networks, which indicates that there exists a strong correlation between the detailed features and the principal primitives. In terms of the similarity loss function, the DANN method performs better.

5.3.2. *Comparison*

To the best of our knowledge, there are currently no existing DL models similar to Brep2Seq, which takes the B-rep data of CAD models as input and predicts the CAD construction sequence of principal primitives and detailed features. For comparison, we chose DeepCAD (Wu *et al.*, 2021), the closest model to our work, designed to generate sequences of sketch and extrude operations. We perform this comparison using the DeepCAD dataset and the Fusion 360 Gallery dataset.

Since the DeepCAD network cannot directly handle B-rep data as input, we utilize their pre-trained encoder to generate the latent vectors of CAD sequences, and then train our B-rep encoder to replicate these embeddings. During testing, we first employed the trained B-rep encoder to generate a latent vector of the B-rep data, and then use the pre-trained DeepCAD sequence decoder to generate CAD construction sequences.

The results are presented in Table 6. The DeepCAD network performs better on the DeepCAD dataset, likely due to its pre-training on this dataset. In contrast, Brep2Seq demonstrates better generalization ability on the Fusion 360 Gallery dataset. Notably, Brep2Seq achieved a higher valid ratio across all datasets. We believe that this can be attributed to the representation of CAD modeling process, where Brep2Seq directly predicts parametrized feature-based modeling sequences. In the parametrized feature-based modeling process, each principal primitive and detailed feature possesses a fixed template. As long as they can be logically assembled, a CAD model with correct topology can be obtained, thereby simplifying the model reconstruction problem.

To provide qualitative results, we randomly sample models from the DeepCAD dataset and Fusion 360 Gallery dataset for display in Figs 13 and 14.

5.4. CAD model generation

Based on the pre-trained Brep2Seq model, we trained the GAN using a subset of the DeepCAD dataset and the Fusion 360 Gallery dataset. Some examples of CAD models generated by our approach are presented in Fig. 15.

To evaluate the capacity of Brep2Seq in generating CAD models, we randomly generated 10 000 samples and compared them with two other deep generation models: DeepCAD (Wu *et al.*, 2021) and SkexGen (Xu *et al.*, 2022). Although these models work in different ways and employ different CAD model expressions, they were trained on the same datasets. The quantitative comparison under the metrics as described in Section 5.1 is presented in Table 7. The results indicate that our method performs comparably to DeepCAD and SkexGen.

6. Applications

The experiments above show that our Brep2Seq network can well understand the B-rep data. Leveraging its hierarchical encoder-decoder architecture, the pre-trained Brep2Seq network can serve as both an encoder to extract feature representations from B-rep data and a decoder to generate CAD models by predicting the composition of principal primitives and detailed features. In this section, we discuss the potential applications of the Brep2Seq network.

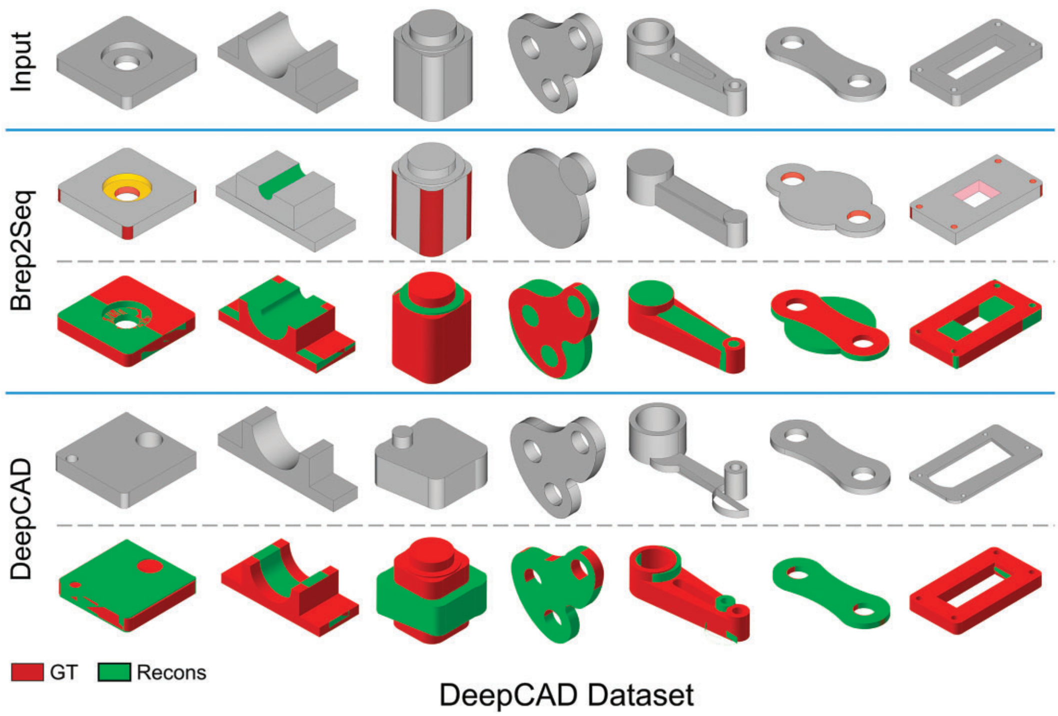


Figure 13: Randomly selected reconstructed models from DeepCAD dataset. Top row: target models; middle row: reconstructed models by executing CAD construction sequences predicted by Brep2Seq decoder; bottom row: reconstructed models by executing CAD construction sequences predicted by DeepCAD decoder.

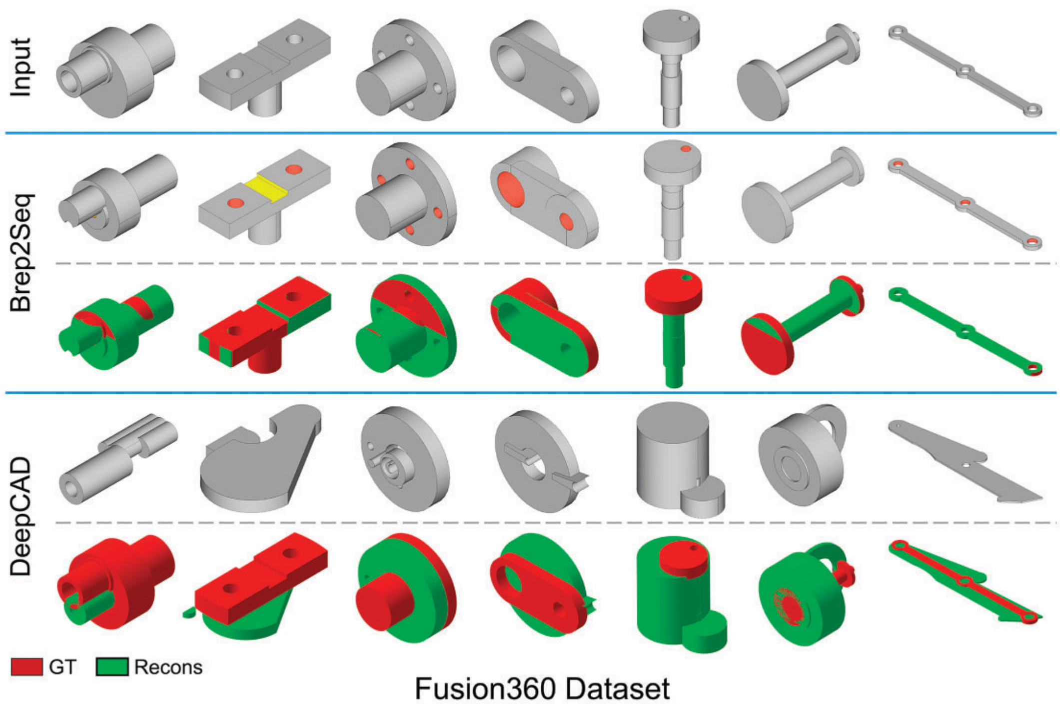


Figure 14: Randomly selected reconstructed models from Fusion 360 Gallery dataset. Top row: target models; middle row: reconstructed models by executing CAD construction sequences predicted by Brep2Seq decoder; bottom row: reconstructed models by executing CAD construction sequences predicted by DeepCAD decoder.

6.1. Reconstruction from point clouds

It is a significant engineering problem to convert irregular and non-Euclidean point clouds into structured representations. Our Brep2Seq network provides a viable feasible solution to this problem: we can leverage an existing point cloud encoder to extract a latent vector, and then employ our pre-trained sequence decoder to reconstruct a CAD model with a 3D shape similar to the input point cloud. Particularly, we obtain point clouds by uniformly

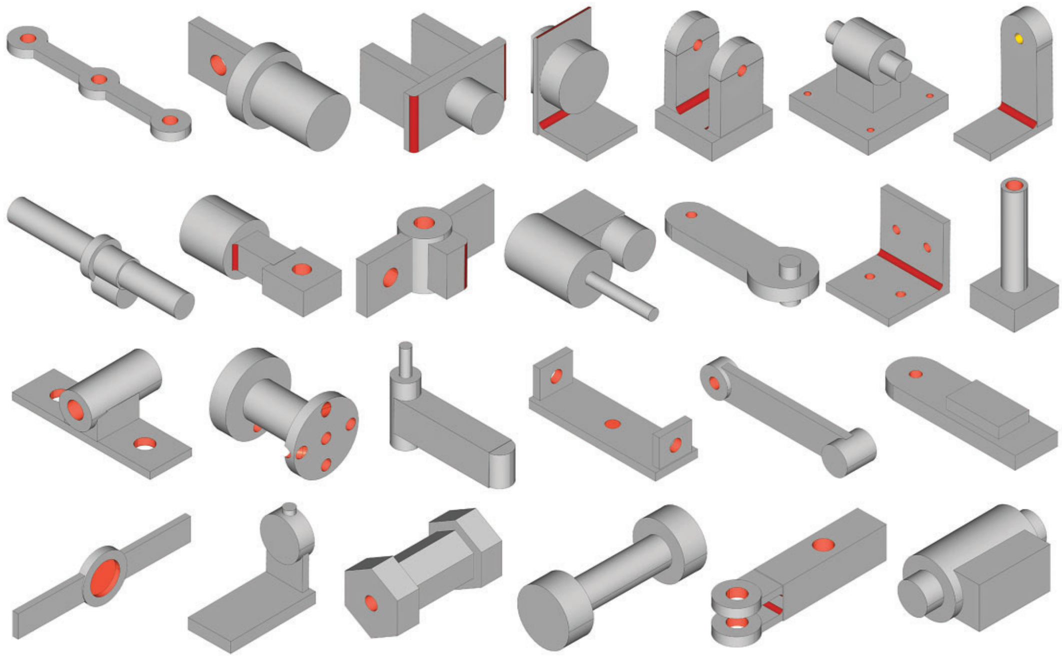


Figure 15: Randomly generated CAD models from our Brep2Seq.

Table 7: Quantitative evaluations on the CAD model generation.

Dataset	Valid ratio↑	COV↑	MMD × 10^2↓	JSD × 10^2↓
Ours	92.0	82.1	**1.10**	1.69
DeepCAD	**98.4**	79.2	1.52	2.56
SkexGen	95.1	**83.5**	1.43	**0.81**

sampling 1024 points from each CAD model in our dataset and train the PointNet++ (Qi *et al.*, 2017b) encoder to map point clouds into the same latent space as our B-rep encoder. During inference, given a point cloud, we apply the trained PointNet++ encoder to map it into a latent vector, which is then decoded into a CAD model using our pre-trained sequence decoder. Figure 16 illustrates the results of point clouds reconstruction.

6.2. CAD model interpolation

Shape interpolation enables visual observation of the evolutionary process from a source model to a target model, providing valuable insights for CAD design. Our pre-trained B-rep encoder allows us to extract latent vectors from a given pair of CAD models and perform linear interpolation on them. Then, the interpolated models are generated by our pre-trained sequence decoder. Figure 17 presents several examples of interpolations, illustrating that the Euclidean distance in the latent space learned by our B-rep encoder can reliably measure CAD model similarity.

6.3. Conditional generation

Given the latent vector of principal primitives, various latent vectors of detailed features can be randomly sampled from the latent space, following the conditional probability distribution learned by GAN. Consequently, CAD models with consistent primary shapes but varying detailed features are produced. Figure 18 illustrates the result of conditional generation.

6.4. CAD feature recognition

We now consider the problem of recognizing feature from CAD models, a well-known problem with various applications, including computer-aided process planning and CAD modeling history reconstruction. Our Brep2Seq network can perform CAD feature recognition by using the predicted parameterized feature-based modeling operations. Figure 19 illustrates the implementation process of B-rep segmentation. Given a B-rep model as input, we reconstruct it based on our pre-trained Brep2Seq network.

While there are geometric and topological differences between the reconstructed CAD model and the input B-rep model, they share the same principal primitive and detailed feature components. We can precisely transfer the labels of principal primitives and detailed features to the original shape, thus obtaining the feature recognition results. Specifically, we first obtain the features (principal primitives or detailed features) corresponding to all faces in the reconstructed model. For any face f_i in the original B-rep model, we uniformly place 512 sample points on it, and establish the mapping between sample points and faces of the reconstructed model by calculating the nearest spatial distance. The features of the matched face are assigned to the sample points. Then, based on the feature labels of these sample points, we can perform a voting process to compute face-level features. Figure 20 shows several examples of CAD feature recognition on our dataset.

7. Discussion

The experimental results presented in Section 5.3 demonstrate that our proposed Brep2Seq model can accurately reconstruct the parameterized feature-based modeling operation sequence of the original B-rep model. Compared with previous GNN methods on B-rep graphs, our approach effectively improve the extraction of geometric and topological information of B-rep models by introducing a transformer's multi-head self-attention mechanism. Additionally, our proposed hierarchical network architecture explic-

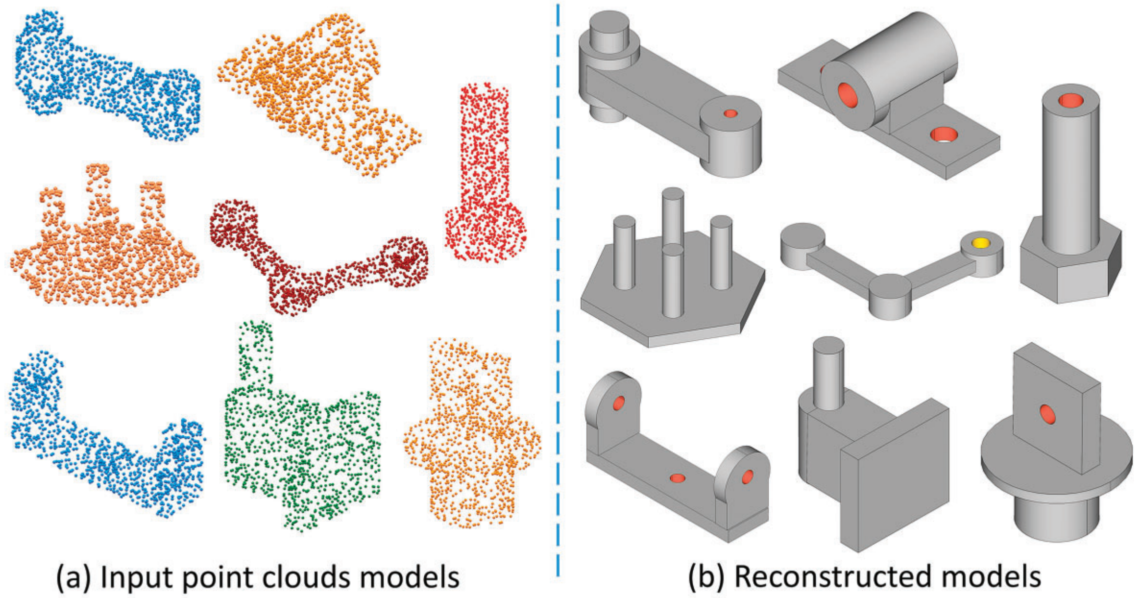


Figure 16: CAD models reconstructed from point clouds. (a) Input point clouds; (b) Reconstructed CAD models.

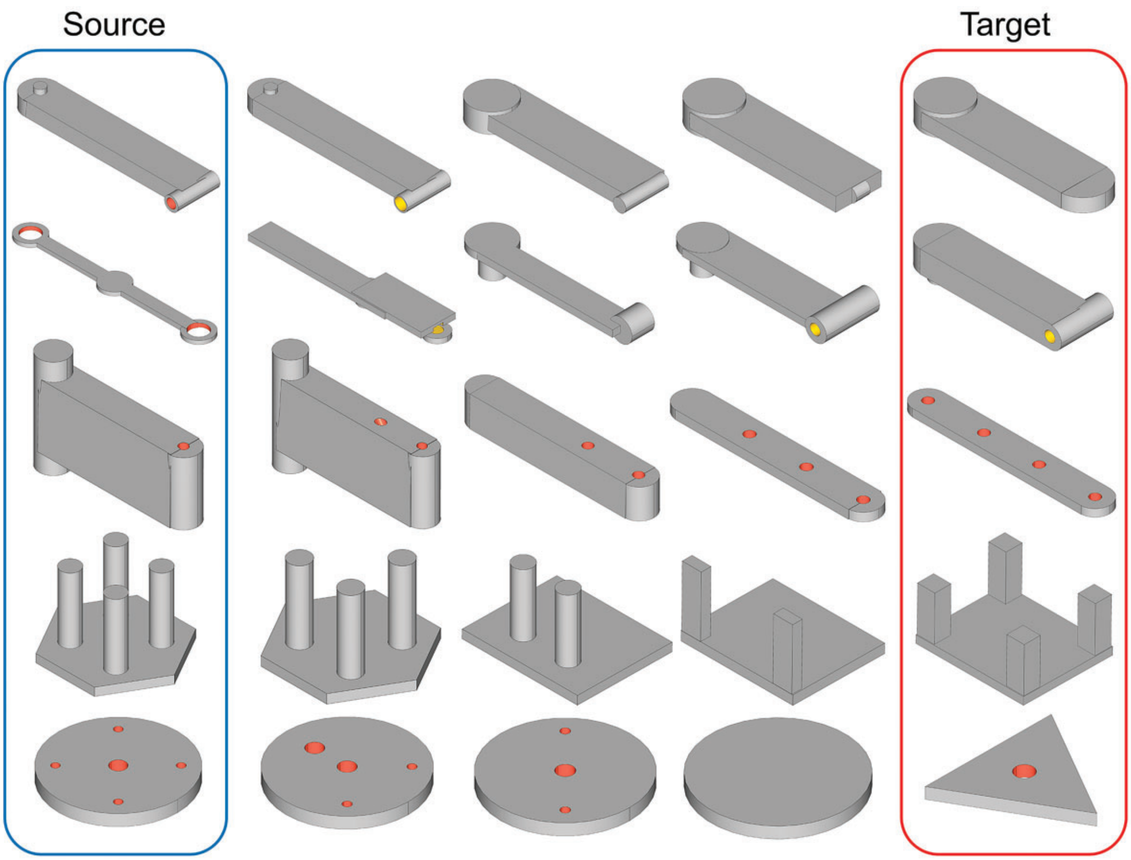


Figure 17: Interpolation of CAD models. The leftmost column are source CAD models and the rightmost column are target CAD models. Between them are CAD models with topological and geometric changes over the interpolation.

itly separates primary shape and detailed features, with the similarity loss function and difference loss function providing the necessary constraints to ensure the neural network's performance. In Fig. 21, we swapped the detailed features embeddings z^f of a pair of models in the left column, resulting in a pair of reconstructed models where the detailed features were swapped while the primary shape remained unchanged. Conversely, we exchanged the principal primitive embeddings z^p of a pair of models in the right column, and the reconstructed models exhibited a corresponding swap in the primary shape. This demonstrates that the two en-

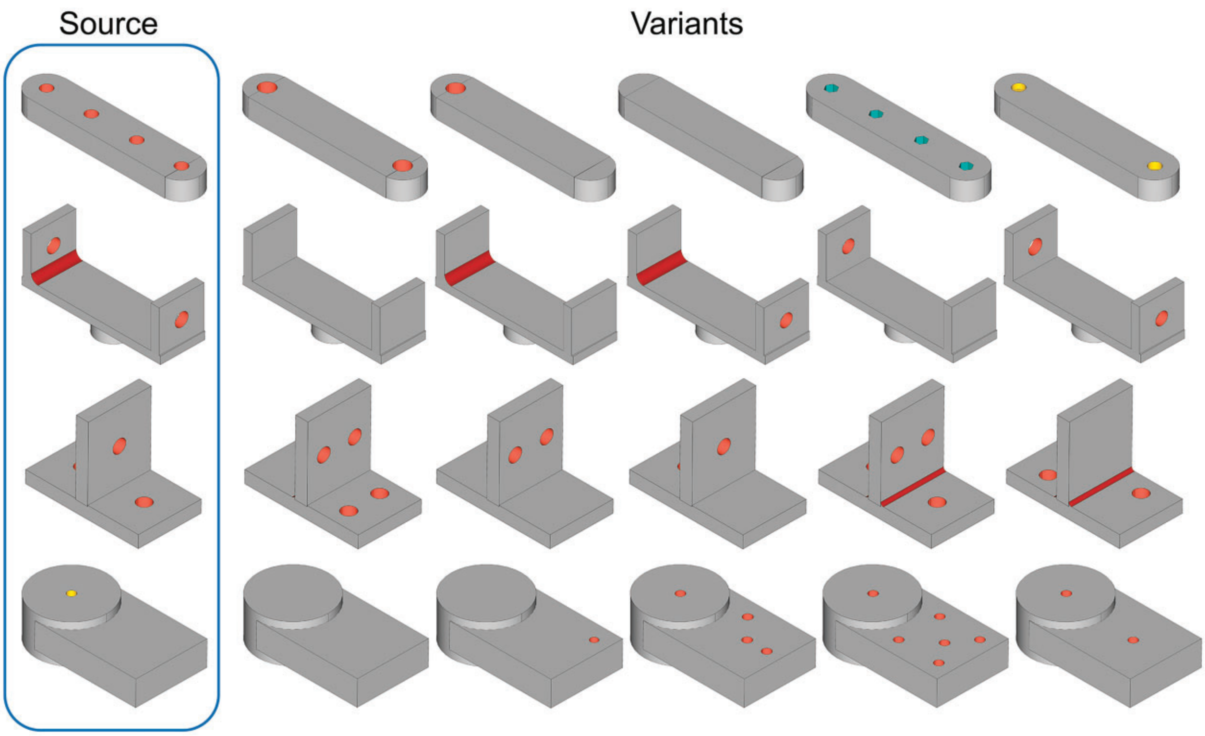


Figure 18: Conditional generation of CAD models. The leftmost of each row is the source model, and the rest are variants of the source model with different detailed features.

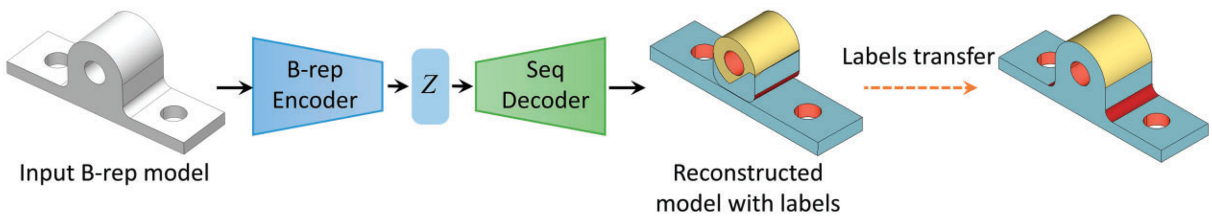


Figure 19: Implementation process of feature recognition based on Brep2Seq.

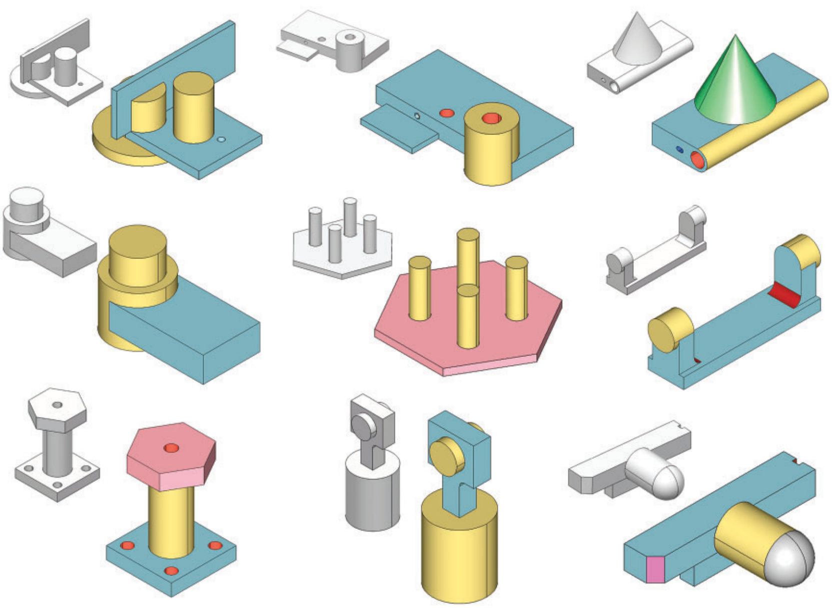


Figure 20: CAD feature recognition results. The results of feature recognition are represented by different colors of faces and the input B-rep model is shown on the top-left.

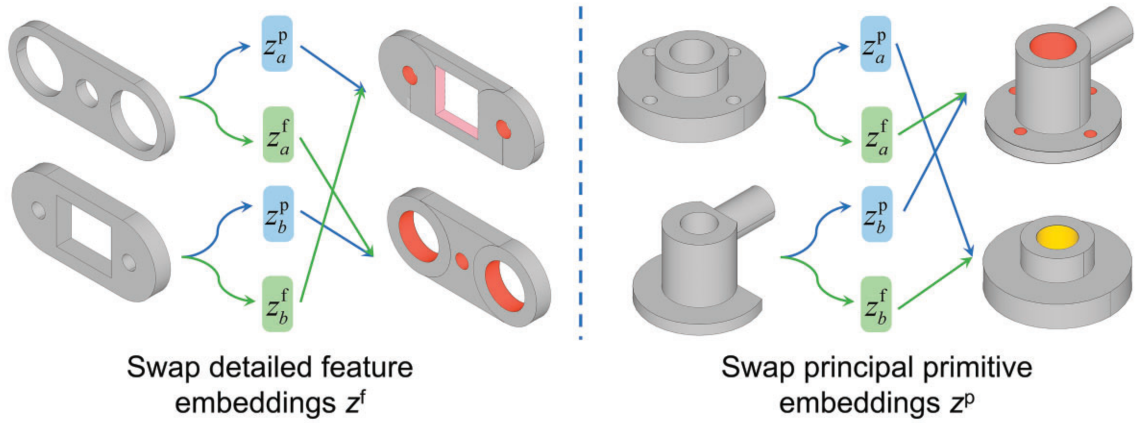


Figure 21: Examples of reconstructed models with swapped principal primitive embeddings or detailed feature embeddings.

coders in the hierarchical network extract decoupled representations: z^p encodes the primary shape, while z^f encodes the detailed features.

We observed that Brep2Seq's performance on real CAD datasets (DeepCAD dataset and Fusion 360 Gallery dataset) was less than satisfactory. We attribute this to two factors. First, while the B-rep models in the synthetic dataset share the same primitives and features as real models, there are differences in data distribution, as indicated in Fig. 4. The second reason is that the high-level semantics of the synthetic B-rep models do not align with real models, thus failing to capture certain design intentions in engineering. These inconsistencies limit the neural network's generalization ability. Nevertheless, the synthetic dataset offers a diverse range of B-rep models for pretraining the neural network, enabling it to learn common geometric features and topological structures. In future work, by leveraging the strengths and mitigating the weaknesses, we can first perform pretraining on our large-scale synthetic dataset and then designing fine-tuning optimization algorithms for specific task based on a small sample of labeled real CAD models.

Not all CAD construction sequences predicted by Brep2Seq can result in 3D CAD models with correct topology. As the complexity of geometry and topology increases, it may lead to failure results, such as the geometry mismatch and topological errors. Figure 22 provides examples of these issues. In this figure, reconstructed models a and b exhibit significant geometric differences from the ground truth models, but they do not contain topological errors and are considered "valid" cases. Reconstructed models c and d are not single continuous 3D solids and are segmented into multiple parts, thus they are invalid. Reconstructed models e to h illustrate cases with topological errors, where models e and f exhibit incorrect boundary loop orders on certain trimming surfaces, while models g and h contain minimal edges or minimal faces, neither of which conforms to the "valid" criteria. Figure 23 illustrates the relationship between sequence length and three CAD reconstruction metrics. The CAD models in our synthetic dataset consist of a maximum of 8 principal primitives and 10 detailed features, where more features imply more complex geometric shapes. Therefore, with an increase in construction sequence length, the difficulty for Brep2Seq to recognize the primitives and features of these complex B-rep models also increases. Specifically, as shown in Fig. 23a, when the sequence length is less than the median value of 9, the valid ratio is nearly 100%. Subsequently, it gradually decreases, indicating an increasing likelihood of generating invalid models. Similarly, in Figs 23b and c, when the sequence length below 9, the CD distance maintains a low level (less than 1.0×10^{-2}), and the IoU value is relatively high (greater than 85%). As the sequence length increases, the CD distance increases and the IoU value decreases, indicating the geometric fidelity tends to decline.

Our work serves as a proof-of-principle demonstration of using deep neural networks to encode B-rep data and predict the corresponding parameterized feature-based modeling operation sequences. While current methods are limited to five types of principal primitives and 24 types of mechanical features, there is a straightforward path to extending and enhancing the existing framework. For instance, by referring to the feature definitions outlined in Section 3, additional mechanical feature types can be introduced. Moreover, modeling operations based on 2D sketches, such as extrude, revolve, sweep, loft, and blend features, can be introduced to further enhance the ability to represent complex 3D shapes. This expansion can enrich the synthetic dataset and enhance the expressive capabilities of neural network models.

8. Conclusions

In summary, we proposed Brep2Seq, a neural network designed to work on B-rep models, addressing the problem of missing modeling process information in CAD neutral files. Brep2Seq can reconstruct the CAD modeling process by decomposing the B-rep models into principal primitives and detailed features. It also can generate new models by sampling from the latent vector space. Specifically, the following conclusions can be drawn:

(i) Brep2Seq leverages the geometric shape and topological relationship information within the CAD models, based on the structural characteristics of B-rep data.
(ii) Brep2Seq employs a hierarchical network architecture to accurately separate the primary shape of a CAD model from its detailed features, facilitating parameterized feature-based modeling reconstruction.
(iii) We evaluated the model reconstruction and generation capabilities of Brep2Seq on DeepCAD and Fusion 360 Gallery datasets. Compared with similar works, Brep2Seq exhibited comparable performance in terms of the valid ratio of modeling, shape fitting accuracy, and diversity of generated models. Additionally, we demonstrated the benefits and versatility of Brep2Seq in downstream tasks.
(iv) We established a large-scale CAD model dataset, the first B-rep model dataset synthesized by a parametrized feature-

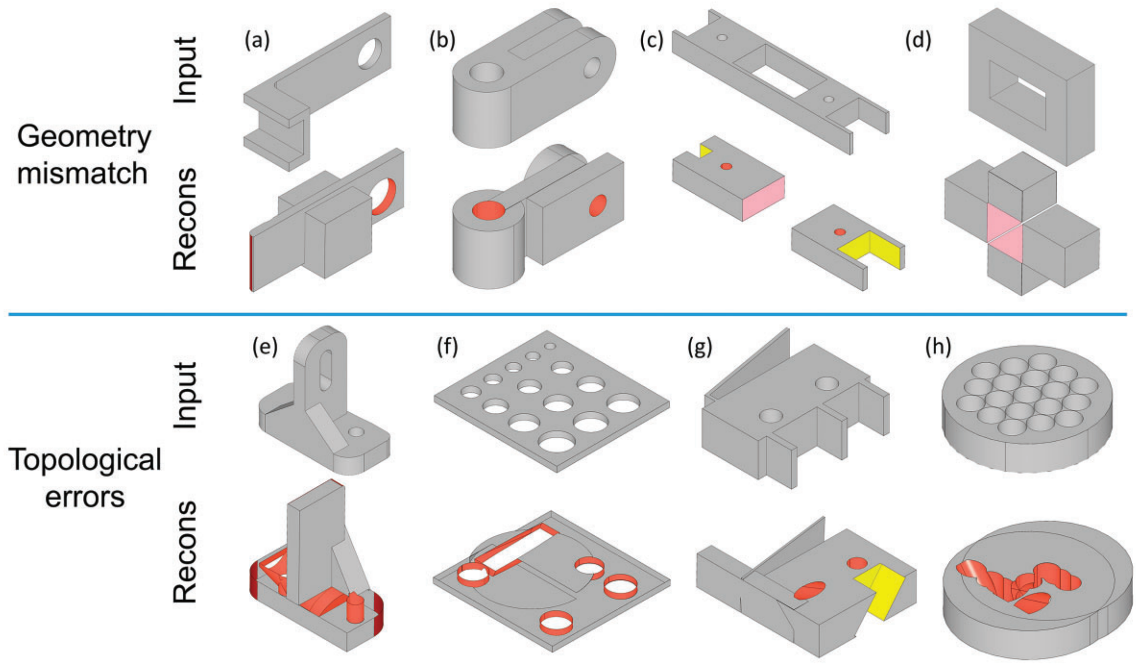


Figure 22: Negative examples for CAD model reconstruction.

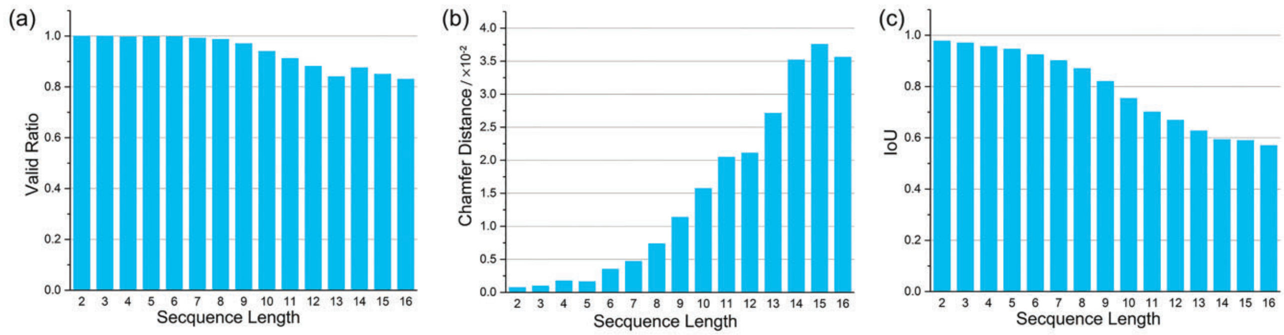


Figure 23: Relationship between the sequence length and geometric fidelity.

based modeling method based on principal primitives and detailed features. This dataset will contribute to the advancement of research in the representation learning of B-rep models.

However, since Brep2Seq was trained on the synthetic dataset, it shows diminished performance and limited generalization ability when applied to real-world CAD models, although we believe these issues can be overcome by introducing fine-tuning optimization algorithms with training examples from real-world. Therefore, it is necessary to improve the 3D shape representation capability of the neural network by further enriching the geometric feature types and incorporating more parametrized feature-based modeling methods. Additionally, exploring a self-supervised learning method based on pre-trained models is also a topic for future research.

Acknowledgments

This work is funded by National Key Project of China (Grant No. GJXM92579).

Conflict of interest statement

None declared.

References

Achlioptas, P., Diamanti, O., Mitliagkas, I., & Guibas, L. (2018). Learning representations and generative models for 3D point clouds. In *Proceedings of the 35th International Conference on Machine Learning*(Vol. **80**, pp. 40–49). PMLR.

Agathos, A., Pratikakis, I., Perantonis, S., Sapidis, N., & Azariadis, P. (2007). 3D mesh segmentation methodologies for CAD applications. *Computer-Aided Design and Applications*, **4**, 827–841. https://doi.org/10.1080/16864360.2007.10738515.

Alai, S. (2013). A review of 3D design parameterization using reverse engineering. *International Journal of Emerging Technology and Advanced Engineering*, **3**, 171–179.

Anderson, J. A. D. W., Sullivan, G. D., & Baker, K. D. (1988). Constrained constructive solid geometry a unique representation of scenes. In *Proceedings of the Alvey Vision Conference*(pp. 14.1–14.6). Alvety Vision Club. https://doi.org/10.5244/C.2.14.

Angrish, A., Craver, B., & Starly, B. (2019). "FabSearch": A 3D CAD model-based search engine for sourcing manufacturing services. *Journal of Computing and Information Science in Engineering*, **19**, 041006. https://doi.org/10.1115/1.4043211.

Arjovsky, M., Chintala, S., & Bottou, L. (2017). Wasserstein generative adversarial networks. In *Proceedings of the 34th International Conference on Machine Learning*(Vol. **70**, pp. 214–223). PMLR.

Barbero, B. R., & Ureta, E. S. (2011). Comparative study of different digitization techniques and their accuracy. *Computer-Aided Design*, **43**, 188–206. https://doi.org/10.1016/j.cad.2010.11.005.

Benkő, P., Martin, R. R., & Várady, T. (2001). Algorithms for reverse engineering boundary representation models. *Computer-Aided Design*, **33**, 839–851. https://doi.org/10.1016/S0010-4485(01)00100-2.

Buchele, S. F., & Crawford, R. H. (2003). Three-dimensional halfspace constructive solid geometry tree construction from implicit boundary representations. In *Proceedings of the Eighth ACM Symposium on Solid Modeling and Applications*(pp. 135–144). Association for Computing Machinery. https://doi.org/10.1145/781606.781629.

Buonamici, F., Carfagni, M., Furferi, R., Governi, L., Lapini, A., & Volpe, Y. (2018). Reverse engineering modeling methods and tools: A survey. *Computer-Aided Design and Applications*, **15**, 443–464. https://doi.org/10.1080/16864360.2017.1397894.

Cao, W., Robinson, T., Hua, Y., Boussuge, F., Colligan, A. R., & Pan, W. (2020). Graph representation of 3D CAD models for machining feature recognition with deep learning. In *Proceedings of the International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*(Vol. **84003**, p. V11AT11A003). American Society of Mechanical Engineers. https://doi.org/10.1115/DETC2020-22355.

Colligan, A. R., Robinson, T. T., Nolan, D. C., Hua, Y., & Cao, W. (2022). Hierarchical CADNet: Learning from B-reps for machining feature recognition. *Computer-Aided Design*, **147**, 103226. https://doi.org/10.1016/j.cad.2022.103226.

Dekhtiar, J., Durupt, A., Bricogne, M., Eynard, B., Rowson, H., & Kiritsis, D. (2018). Deep learning for big data applications in CAD and PLM–research review, opportunities and case study. *Computers and Industry*, **100**, 227–243. https://doi.org/10.1016/j.compind.2018.04.005.

Di Angelo, L., & Di Stefano, P. (2015). Geometric segmentation of 3D scanned surfaces. *Computer-Aided Design*, **62**, 44–56. https://doi.org/10.1016/j.cad.2014.09.006.

Fan, H., Su, H., & Guibas, L. J. (2017). A point set generation network for 3D object reconstruction from a single image. In *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*(pp. 2463–2471). https://doi.org/10.1109/CVPR.2017.264.

Friedrich, M., Fayolle, P.-A., Gabor, T., & Linnhoff-Popien, C. (2019). Optimizing evolutionary CSG tree extraction. In *Proceedings of the Genetic and Evolutionary Computation Conference*(pp. 1183–1191). Association for Computing Machinery. https://doi.org/10.1145/3321707.3321771.

Friedrich, M., Roch, C., Feld, S., Hahn, C., & Fayolle, P.-A. (2020). A flexible pipeline for the optimization of CSG trees. In *Proceedings of the 28th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG) 2020*(pp. 79–88). https://doi.org/10.48550/arXiv.2008.03674.

Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., March, M., & Lempitsky, V. (2016). Domain-adversarial training of neural networks. *Journal of Machine Learning Research*, **17**, 1–35.

Gao, L., Yang, J., Wu, T., Yuan, Y.-J., Fu, H., Lai, Y.-K., & Zhang, H. (2019). SDM-Net: Deep generative network for structured deformable mesh. *ACM Transactions on Graphics (TOG)*, **38**, 1–15. https://doi.org/10.1145/3355089.3356488.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., & Dahl, G. E. (2017). Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning*(pp. 1263–1272). PMLR.

Gretton, A., Borgwardt, K., Rasch, M., Schölkopf, B., & Smola, A. (2006). A kernel method for the two-sample-problem. In *Advances in neural information processing systems*(Vol. **19**). MIT Press. https://doi.org/10.7551/mitpress/7503.003.0069.

Hanocka, R., Hertz, A., Fish, N., Giryes, R., Fleishman, S., & Cohen-Or, D. (2019). MeshCNN: A network with an edge. *ACM Transactions on Graphics (ToG)*, **38**, 1–12. https://doi.org/10.1145/3306346.3322959.

Hilbig, A., Vogt, L., Holtzhausen, S., & Paetzold, K. (2023). Enhancing three-dimensional convolutional neural network-based geometric feature recognition for adaptive additive manufacturing: A signed distance field data approach. *Journal of Computational Design and Engineering*, **10**, 992–1009. https://doi.org/10.1093/jcde/qwad027.

Hou, J., Luo, C., Qin, F., Shao, Y., & Chen, X. (2023). FUS-GCN: Efficient B-rep based graph convolutional networks for 3D-CAD model classification and retrieval. *Advanced Engineering Informatics*, **56**, 102008. https://doi.org/10.1016/j.aei.2023.102008.

Jayaraman, P. K., Lambourne, J. G., Desai, N., Willis, K., Sanghi, A., & Morris, N. J. W. (2023). Solidgen: An autoregressive model for direct B-rep synthesis. *Transactions on Machine Learning Research*. https://doi.org/10.48550/arXiv.2203.13944.

Jayaraman, P. K., Sanghi, A., Lambourne, J. G., Willis, K. D., Davies, T., Shayani, H., & Morris, N. (2021). UV-Net: Learning from boundary representations. In *Proceedings of the 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*(pp. 11698–11707). https://doi.org/10.1109/CVPR46437.2021.01153.

Jia, Y., Salzmann, M., & Darrell, T. (2010). Factorized latent spaces with structured sparsity. In *Advances in neural information processing systems*(Vol. **23**). Curran Associates, Inc.

Jones, R. K., Barton, T., Xu, X., Wang, K., Jiang, E., Guerrero, P., Mitra, N. J., & Ritchie, D. (2020). Shapeassembly: Learning to generate programs for 3D shape structure synthesis. *ACM Transactions on Graphics (TOG)*, **39**, 1–20. https://doi.org/10.1145/3414685.3417812.

Kania, K., Zieba, M., & Kajdanowicz, T. (2020). UCSG-Net- unsupervised discovering of constructive solid geometry tree. In *Advances in neural information processing systems*(Vol. **33**, pp. 8776–8786). Curran Associates, Inc.

Kim, M.-J., Lee, K.-H., Han, Y.-S., Lee, J., & Nam, B. (2021). Generating 3D texture models of vessel pipes using 2D texture transferred by object recognition. *Journal of Computational Design and Engineering*, **8**, 475–487. https://doi.org/10.1093/jcde/qwaa090.

Koch, S., Matveev, A., Jiang, Z., Williams, F., Artemov, A., Burnaev, E., Alexa, M., Zorin, D., & Panozzo, D. (2019). ABC: A big CAD model dataset for geometric deep learning. In *Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*(pp. 9593–9603). https://doi.org/10.1109/CVPR.2019.00983.

Lambourne, J. G., Willis, K. D., Jayaraman, P. K., Sanghi, A., Meltzer, P., & Shayani, H. (2021). BrepNet: A topological message passing system for solid models. In *Proceedings of the 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*(pp. 12768–12777). https://doi.org/10.1109/CVPR46437.2021.01258.

Lambourne, J. G., Willis, K., Jayaraman, P. K., Zhang, L., Sanghi, A., & Malekshan, K. R. (2022). Reconstructing editable prismatic CAD from rounded voxel models. In *Proceedings of the SIGGRAPH Asia 2022 Conference Papers*(pp. 1–9). Association for Computing Machinery. https://doi.org/10.1145/3550469.3555424.

Lee, H., Lee, J., Kim, H., & Mun, D. (2022). Dataset and method for deep learning-based reconstruction of 3D CAD models containing machining features for mechanical parts. *Journal of Computational Design and Engineering*, **9**, 114–127. https://doi.org/10.1093/jcde/qwab072.

Lee, J., Yeo, C., Cheon, S.-U., Park, J. H., & Mun, D. (2023). BrepGAT: Graph neural network to segment machining feature faces in a B-rep model. *Journal of Computational Design and Engineering*, **10**, 2384–2400. https://doi.org/10.1093/jcde/qwad106.

Li, C., Pan, H., Bousseau, A., & Mitra, N. J. (2022). Free2CAD: Parsing freehand drawings into CAD commands. *ACM Transactions on Graphics (TOG)*, **41**, 1–16. https://doi.org/10.1145/3528223.3530133.

Li, J., Niu, C., & Xu, K. (2020). Learning part generation and assembly for structure-aware shape synthesis. In *Proceedings of the AAAI Conference on Artificial Intelligence*(Vol. **34**, pp. 11362–11369). https://doi.org/10.1609/aaai.v34i07.6798.

Li, J., Xu, K., Chaudhuri, S., Yumer, E., Zhang, H., & Guibas, L. (2017). Grass: Generative recursive autoencoders for shape structures. *ACM Transactions on Graphics (TOG)*, **36**, 1–14. https://doi.org/10.1145/3072959.3073637.

Liu, M., Yao, F., Choi, C., Ayan, S., & Ramani, K. (2019). Deep learning 3D shapes using alt-az anisotropic 2-sphere convolution. In *Proceedings of the International Conference on Learning Representations*.

Meltzer, P., Shayani, H., Khasahmadi, A., Jayaraman, P. K., Sanghi, A., & Lambourne, J. (2021). UVStyle-Net: Unsupervised few-shot learning of 3D style similarity measure for B-reps. In *Proceedings of the 2021 IEEE/CVF International Conference on Computer Vision (ICCV)*(pp. 9670–9679). https://doi.org/10.1109/ICCV48922.2021.00955.

Mo, K., Guerrero, P., Yi, L., Su, H., Wonka, P., Mitra, N. J., & Guibas, L. J. (2019). StructureNet: Hierarchical graph networks for 3D shape generation. *ACM Transactions on Graphics*, **38**, 1–19. https://doi.org/10.1145/3355089.3356527.

Motavalli, S. (1998). Review of reverse engineering approaches. *Computers and Industrial Engineering*, **35**, 25–28. https://doi.org/10.1016/S0360-8352(98)00011-4.

Nash, C., Ganin, Y., Eslami, S. M. A., & Battaglia, P. (2020). PolyGen: An autoregressive generative model of 3D meshes. In *Proceedings of the 37th International Conference on Machine Learning*(Vol. **119**, pp. 7220–7229). PMLR.

Ning, F., Shi, Y., Cai, M., & Xu, W. (2023). Part machining feature recognition based on a deep learning method. *Journal of Intelligent Manufacturing*, **34**, 809–821. https://doi.org/10.1007/s10845-021-01827-7.

Nourse, B. (1980). Natural quadrics in mechanical design. In *Proceedings of the Autofact West, 1980*(Vol. **1**, pp. 363–378).

Qi, C. R., Su, H., Kaichun, M., & Guibas, L. J. (2017a). PointNet: Deep learning on point sets for 3D classification and segmentation. In *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*(pp. 77–85). IEEE.

Qi, C. R., Su, H., Nießner, M., Dai, A., Yan, M., & Guibas, L. J. (2016). Volumetric and multi-view CNNs for object classification on 3D data. In *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*(pp. 5648–5656). IEEE. https://doi.org/10.1109/CVPR.2016.609.

Qi, C. R., Yi, L., Su, H., & Guibas, L. J. (2017b). PointNet++: Deep hierarchical feature learning on point sets in a metric space. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*(pp. 5105–5114). Curran Associates, Inc.

Qin, F., Qiu, S., Gao, S., & Bai, J. (2022). 3D CAD model retrieval based on sketch and unsupervised variational autoencoder. *Advanced Engineering Informatics*, **51**, 101427. https://doi.org/10.1016/j.aei.2021.101427.

Rea, H., Sung, R., Corney, J., Clark, D., & Taylor, N. (2005). Interpreting three-dimensional shape distributions. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, **219**, 553–566. https://doi.org/10.1243/095440605x31427.

Ren, D., Zheng, J., Cai, J., Li, J., Jiang, H., Cai, Z., Zhang, J., Pan, L., Zhang, M., Zhao, H., & Yi, S. (2021). CSG-Stump: A learning friendly CSG-like representation for interpretable shape parsing. In *Proceedings of the 2021 IEEE/CVF International Conference on Computer Vision (ICCV)*(pp. 12478–12487). https://doi.org/10.1109/ICCV48922.2021.01225.

Roberts, D., Danielyan, A., Chu, H., Golparvar-Fard, M., & Forsyth, D. (2021). LSD-StructureNet: Modeling levels of structural detail in 3D part hierarchies. In *Proceedings of the 2021 IEEE/CVF International Conference on Computer Vision (ICCV)*(pp. 5836–5845). IEEE. https://doi.org/10.1109/ICCV48922.2021.00578.

Safdar, M., Jauhar, T. A., Kim, Y., Lee, H., Noh, C., Kim, H., Lee, I., Kim, I., Kwon, S., & Han, S. (2020). Feature-based translation of CAD models with macro-parametric approach: Issues of feature mapping, persistent naming, and constraint translation. *Journal of Computational Design and Engineering*, **7**, 603–614. https://doi.org/10.1093/jcde/qwaa043.

Salzmann, M., Ek, C. H., Urtasun, R., & Darrell, T. (2010). Factorized orthogonal latent spaces. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*(Vol. **9**, pp. 701–708). PMLR.

Sanchez, E. H., Serrurier, M., & Ortner, M. (2020). Learning disentangled representations via mutual information estimation. In *Proceedings of the Computer Vision – ECCV 2020*(pp. 205–221). Springer International Publishing. https://doi.org/10.1007/978-3-030-58542-6_13.

Shapiro, V., & Vossler, D. L. (1991). Construction and optimization of CSG representations. *Computer-Aided Design*, **23**, 4–20. https://doi.org/10.1016/0010-4485(91)90077-a.

Shapiro, V., & Vossler, D. L. (1993). Separation for boundary to CSG conversion. *ACM Transactions on Graphics (TOG)*, **12**, 35–55. https://doi.org/10.1145/169728.169723.

Sharma, G., Goyal, R., Liu, D., Kalogerakis, E., & Maji, S. (2018). CSGNet: Neural shape parser for constructive solid geometry. In *Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*(pp. 5515–5523). IEEE. https://doi.org/10.1109/CVPR.2018.00578.

Song, J., Lee, J., Ko, K., Kim, W.-D., Kang, T.-W., Kim, J.-Y., & Nam, J.-H. (2021). Unorganized point classification for robust NURBs surface reconstruction using a point-based neural network. *Journal of Computational Design and Engineering*, **8**, 392–408. https://doi.org/10.1093/jcde/qwaa086.

Sung, M., Su, H., Kim, V. G., Chaudhuri, S., & Guibas, L. (2017). Complementme: Weakly-supervised component suggestions for 3D modeling. *ACM Transactions on Graphics (TOG)*, **36**, 1–12. https://doi.org/10.1145/3130800.3130821.

Takaishi, I., Kanai, S., Date, H., & Takashima, H. (2020). Free-form feature classification for finite element meshing based on shape descriptors and machine learning. *Computer-Aided Design and Applications*, **17**, 1049–1066. https://doi.org/10.14733/cadaps.2020.1049-1066.

Uy, M., Chang, Y., Sung, M., Goel, P., Lambourne, J., Birdal, T., & Guibas, L. (2022). Point2Cyl: Reverse engineering 3D objects from point clouds to extrusion cylinders. In *Proceedings of the 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*(pp. 11840–11850). IEEE. https://doi.org/10.1109/CVPR52688.2022.01155.

Varady, T., Martin, R. R., & Cox, J. (1997). Reverse engineering of geometric models'an introduction. *Computer-Aided Design*, **29**, 255–268. https://doi.org/10.1016/S0010-4485(96)00054-1.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*(Vol. **30**, pp. 6000–6010). Curran Associates, Inc.

Willis, K. D., Pu, Y., Luo, J., Chu, H., Du, T., Lambourne, J. G., Solar-Lezama, A., & Matusik, W. (2021). Fusion 360 gallery: A dataset and environment for programmatic CAD construction from human design sequences. *ACM Transactions on Graphics (TOG)*, **40**, 1–24. https://doi.org/10.1145/3450626.3459818.

Wu, R., Xiao, C., & Zheng, C. (2021). DeepCAD: A deep generative network for computer-aided design models. In *Proceedings of the 2021 IEEE/CVF International Conference on Computer Vision (ICCV)*(pp. 6772–6782). IEEE. https://doi.org/10.1109/ICCV48922.2021.00670.

Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., & Xiao, J. (2015). 3D ShapeNets: A deep representation for volumetric shapes. In *Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*(pp. 1912–1920). IEEE.

Xu, X., Peng, W., Cheng, C., Willis, K. D., & Ritchie, D. (2021). Inferring CAD modeling sequences using zone graphs. In *Proceedings of the 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*(pp. 6058–6066). IEEE. https://doi.org/10.1109/CVPR46437.2021.00600.

Xu, X., Willis, K. D., Lambourne, J. G., Cheng, C.-Y., Jayaraman, P. K., & Furukawa, Y. (2022). SkexGen: Autoregressive generation of CAD construction sequences with disentangled codebooks. In *Proceedings of the 39th International Conference on Machine Learning*(Vol. **162**, pp. 24698–24724). PMLR. https://doi.org/10.48550/arXiv.2207.04632.

Yeo, C., Kim, B. C., Cheon, S., Lee, J., & Mun, D. (2021). Machining feature recognition based on deep neural networks to support tight integration with 3D CAD systems. *Scientific Reports*, **11**, 22147. https://doi.org/10.1038/s41598-021-01313-3.

Ying, C., Cai, T., Luo, S., Zheng, S., Ke, G., He, D., Shen, Y., & Liu, T.-Y. (2021). Do transformers really perform badly for graph representation?. In *Advances in neural information processing systems*(Vol. **34**, pp. 28877–28888). Curran Associates, Inc.

Yu, F., Chen, Z., Li, M., Sanghi, A., Shayani, H., Mahdavi-Amiri, A., & Zhang, H. (2022). Capri-Net: Learning compact cad shapes with adaptive primitive assembly. In *Proceedings of the 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*(pp. 11758–11768). IEEE. https://doi.org/10.1109/CVPR52688.2022.01147.

Zhang, W., Yang, Z., Jiang, H., Nigam, S., Yamakawa, S., Furuhata, T., Shimada, K., & Kara, L. B. (2019). 3D shape synthesis for conceptual design and optimization using variational autoencoders. In *Proceedings of the ASME 2019 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference. Volume 2A: 45th Design Automation Conference*(p. V02AT03A017). ASME. https://doi.org/10.1115/DETC2019-98525.

Zhang, Y., Fu, Y., Jia, J., & Luo, X. (2020). An approach to automatic boundary segmentation of solid models using virtual topology: Toward reconstruction of design features. *Journal of Computational Design and Engineering*, **7**, 367–385. https://doi.org/10.1093/jcde/qwaa030.

Zhang, Z., Jaiswal, P., & Rai, R. (2018). FeatureNet: Machining feature recognition based on 3D convolution neural network. *Computer-Aided Design*, **101**, 12–22. https://doi.org/10.1016/j.cad.2018.03.006.

Appendix 1

A1.1. Definitions of detailed features

The parameterized modeling definitions for the 24 detailed features are shown in Table A.1. The spatial position and sketch contours of each feature are determined by reference points $p_1(x_1, y_1, z_1)$ and $p_2(x_2, y_2, z_2)$. The specific geometric dimensions are defined by length (l), width (w, w_1), radius (r), and depth (d).

A1.2. MMD method

When given two finite sample sets $\mathcal{X} = \{x_1, \ldots, x_n\}$ and $\mathcal{Z} = \{z_1, \ldots, z_m\}$ drawn from distribution P and Q, the MMD method projects $\mathcal{X}$ and $\mathcal{Z}$ into a high-dimensional reproducible Hilbert space $\mathcal{H}$. It measures the difference between distributions P and Q as the expected distance:

$$\mathrm{MMD}\,[P, Q] \triangleq \sup_{\|f\|_{\mathcal{H}\leq 1}} \left(\mathbb{E}_{x\sim P}\,[f\,(x)] - \mathbb{E}_{z\sim Q}\,[f\,(z)]\right) \tag{A.1}$$

$$\mathrm{MMD}\,[\mathcal{X}, \mathcal{Z}] \triangleq \left\| \frac{1}{n}\sum_{i=1}^{n} \phi\,(x_i) - \frac{1}{m}\sum_{j=1}^{m} \phi\,(z_i) \right\|_{\mathcal{H}} \tag{A.2}$$

where $f(\cdot)$ is an arbitrary function in $\mathcal{H}$, and $\phi : x \mapsto \mathcal{H}$ is a nonlinear mapping function that projects the original data into a high-dimensional space.

A1.3. Metrics for CAD model generation

We employ the following metrics to quantify the discrepancy between two sets of CAD models: the set $\mathcal{S}$, comprising real CAD models used for training, and the set $\mathcal{G}$, consisting of generated CAD models.

(i) Coverage (COV): COV is the percentage of CAD models in set $\mathcal{G}$ that closely match real CAD models in set $\mathcal{S}$, determined by the closest CD. A higher metric value indicates a greater diversity within the generated set $\mathcal{G}$.

$$\mathrm{COV}\,(\mathcal{S}, \mathcal{G}) = \frac{\left|\left\{\arg\,\min_{Y\in\mathcal{S}} d^{\mathrm{CD}}\,(X, Y)\,|X \in \mathcal{G}\right\}\right|}{|\mathcal{S}|} \tag{A.3}$$

(ii) MMD: MMD is the average distance between set $\mathcal{G}$ and set $\mathcal{S}$, calculated using the closest CD. This metric quantifies the fidelity of the generated set $\mathcal{G}$, where a higher value indicates a closer resemblance to the real CAD models.

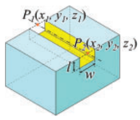
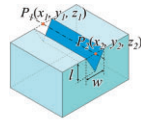
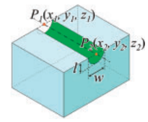
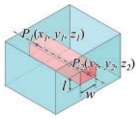
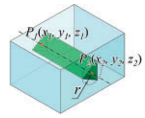
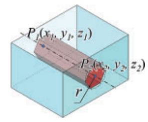
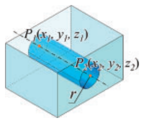
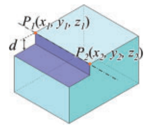
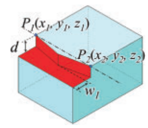
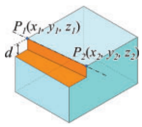
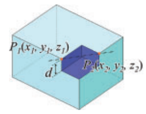
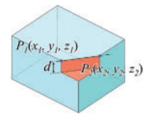
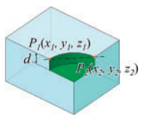
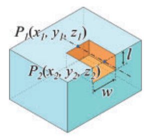
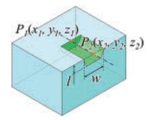
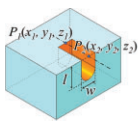
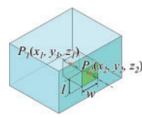
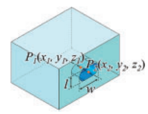
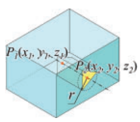
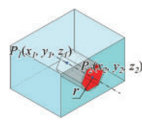
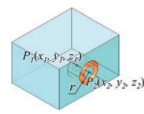
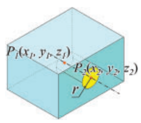
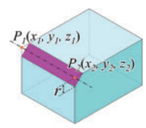
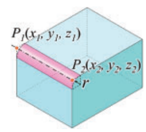
$$\mathrm{MMD}\,(\mathcal{S}, \mathcal{G}) = \frac{1}{|\mathcal{S}|}\sum_{Y\in\mathcal{S}} \min_{X\in\mathcal{G}} d^{\mathrm{CD}}\,(X, Y) \tag{A.4}$$

(iii) JSD: JSD is the similarity between set $\mathcal{G}$ and set $\mathcal{S}$ based on the marginal point distribution.

$$\mathrm{JSD}\,(P_{\mathcal{S}}, P_{\mathcal{G}}) = \frac{1}{2} D_{\mathrm{KL}}\,(P_{\mathcal{S}} \parallel M) + \frac{1}{2} D_{\mathrm{KL}}\,(P_{\mathcal{G}} \parallel M) \tag{A.5}$$

where $M = \frac{1}{2}\,(P_{\mathcal{S}} + P_{\mathcal{G}})$ and D_{KL} is the standard KL-divergence.

Table A.1: Detailed features and their parameters.

Idx	Features	Visualization	Idx	Features	Visualization	Idx	Features	Visualization
0	Rectangular through slot		1	Triangular through slot		2	Circular through slot	
3	Rectangular passage		4	Triangular passage		5	6-Sided passage	
6	Through hole		7	Rectangular through step		8	2-Sided through step	
9	Slanted through step		10	Rectangular blind step		11	Triangular blind step	
12	Circular blind step		13	Rectangular blind slot		14	Horizontal circular end blind slot	
15	Vertical circular end blind slot		16	Rectangular pocket		17	Circular end pocket	
18	Triangular pocket		19	6-Sides pocket		20	O-ring	
21	Blind hole		22	Chamfer		23	Fillet	

Received: July 18, 2023. **Revised:** January 2, 2024. **Accepted:** January 8, 2024