CADParser: A Learning Approach of Sequence Modeling for B-Rep CAD

Shengdi Zhou[1] , **Tianyi Tang**[2] and **Bin Zhou**[1*]
[1]State Key Laboratory of Virtual Reality Technology and Systems, Beihang University
[2]University of Waterloo
{zhoushengdi9, tianyitangdhr}@gmail.com, zhoubin@buaa.edu.cn

Abstract

Computer-Aided Design (CAD) plays a crucial role in industrial manufacturing by providing geometry information and the construction workflow for manufactured objects. The construction information enables effective re-editing of parametric CAD models. While boundary representation (B-Rep) is the standard format for representing geometry structures, JSON format is an alternative due to the lack of uniform criteria for storing the construction workflow. Regrettably, most CAD models available on the Internet only offer geometry information, omitting the construction procedure and hampering creation efficiency. This paper proposes a learning approach CADParser to infer the underlying modeling sequences given a B-Rep CAD model. It achieves this by treating the CAD geometry structure as a graph and the construction workflow as a sequence. Since the existing CAD dataset only contains two operations (i.e., Sketch and Extrusion), limiting the diversity of the CAD model creation, we also introduce a large-scale dataset incorporating a more comprehensive range of operations such as Revolution, Fillet, and Chamfer. Each model includes both the geometry structure and the construction sequences. Extensive experiments demonstrate that our method can compete with the existing state-of-the-art methods quantitatively and qualitatively. Data is available at https://drive.google.com/CADParserData

1 Introduction

Parametric Computer-Aided Design (CAD) is the predominant method for creating 3D models of manufactured objects, including automobile parts, electronic devices, and furniture. These CAD files capture the geometry and crucial construction sequence information. This information is vital for preserving design intent, enabling edits, and facilitating downstream tasks such as simulation and manufacturing. Unfortunately, this valuable information is often lost during data

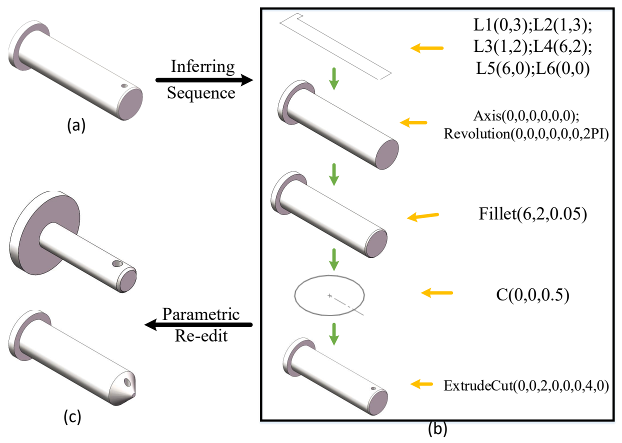

Figure 1: B-Rep CAD models contain the geometry information. Inferring the underlying construction workflow (b) given a B-Rep model (a) can readily re-edit the input model parametrically to generate model CAD models (c).

translation or due to errors, requiring a laborious reverse-engineering process from raw geometry or even 3D scan data.

One common characteristic among these works is their utilization of constructive solid geometry (CSG) as a modeling language. In CSG, shapes are created by combining primitive solids such as spheres, cylinders, and boxes using Boolean operations like union, intersection, and difference. The limited library of parametric primitive shapes in CSG makes it an attractive choice for CAD program inference since it narrows down the search space of potential programs.

Unfortunately, parametric primitive CSG is not the modeling language commonly used in modern CAD workflows. Instead, CAD practitioners employ feature-based modeling to create a solid object by incrementally adding features like holes, slots, or bosses. This iterative process often involves operations on surfaces, as they are more intuitive for users. The resulting object's geometry is stored as a boundary representation (B-Rep). Unlike parametric primitive CSG, feature-based modeling with B-Reps enables Boolean operations between solids, making it more versatile. However, this increased expressiveness poses a challenge for program inference, as the space of feature-based modeling programs is significantly larger.

[*]Bin Zhou is the corresponding author.

While reconstructing CAD operations from raw geometry has been explored in previous research, recent advancements in neural networks for 3D shape generation have sparked renewed interest in CAD reconstruction. However, learning-based approaches to CAD reconstruction have faced a challenge due to lacking a human-designed dataset containing 3D CAD construction sequences. As a result, these approaches have heavily relied on synthetic data for training and testing. The absence of real-world data has constrained the progress in CAD reconstruction, particularly concerning common modeling operations like sketching and extrusion.

To address this challenge, we have introduced a CAD dataset encompassing a wide range of features, aiming to facilitate learning-based approaches in CAD reconstruction. Furthermore, we have developed a deep parser network to infer the CAD sequence for B-Rep models. This paper makes the following contributions:

- We present a new CAD dataset containing ˜40000 models with various features(e.g., Revolve, Fillet, Chamfer) and construction sequence information.
- We propose a novel deep parser network CADParser to infer the CAD construction sequence from a B-Rep model.
- We demonstrate our method outperforms other related works quantitatively and qualitatively.

2 Related Work

2.1 CAD Dataset

Existing large-scale CAD datasets have primarily focused on mesh geometry, point clouds, and voxel grids [Chang *et al.*, 2015; Wu *et al.*, 2015; Kim *et al.*, 2020; Mo *et al.*, 2019], contributing to significant advancements in 3D tasks such as segmentation [Hanocka *et al.*, 2019], classification [Qi *et al.*, 2017], and reconstruction [Mescheder *et al.*, 2019]. However, the standard format for CAD models is Boundary representation (B-Rep), which describes models based on their parametric geometry and topology. B-Rep models enable users to interact directly with faces, edges, and vertices, allowing for seamless alignment and modification of 3D shapes. B-Rep models can be shared across different software platforms as a widely accepted protocol. Nevertheless, B-Rep models lack information about their creation and design processes, which hampers higher-level control over 3D shapes. When a CAD model is accompanied by its construction procedure, designers can easily modify the entire shape through parametric operations rather than focusing solely on local adjustments to vertices and faces as in mesh models. Only recently, a few datasets have provided CAD models and their construction history. Based on their respective focuses, these datasets can be broadly categorized into Sketch and 3D CAD.

In the context of CAD modeling, a sketch refers to a 2D skeleton as the basis for creating 3D shapes. It includes both the 2D geometry primitives and the constraints among them. The SketchGraphs dataset [Seff *et al.*, 2020], the first publicly available benchmark for CAD sketches, comprises 15 million engineering sketches extracted from the Onshape CAD software [Ons,]. This dataset has been instrumental in advancing sketch generation models [Para *et al.*, 2021; Seff *et al.*, 2020; Willis *et al.*, 2021a]. However, the SketchGraphs dataset suffers from significant data duplication. To address this limitation, Ganin et al. [Ganin *et al.*, 2021] have collected a large-scale dataset from the Onshape platform and implemented filtering procedures to mitigate the drawbacks of the SketchGraphs dataset. To the best of our knowledge, these two datasets are currently the only CAD sketch datasets that focus on showcasing how sketches were designed, including the associated constraints.

Despite sketch datasets, there are also 3D CAD datasets that offer the 3D geometry models in B-Rep format and the corresponding construction process. One such dataset is the ABC dataset [Koch *et al.*, 2019], which collects approximately 1 million CAD designs from the Onshape platform. Unlike sketch datasets, the ABC dataset focuses on 3D shapes rather than 2D sketches. However, the initial version of the ABC dataset only includes parametric representations of 3D CAD models, lacking sufficient information to reveal how the models were created. In response to this limitation, Wu et al. [Wu *et al.*, 2021] has created their dataset based on the ABC dataset, consisting of ˜180,000 models. Leveraging the link to the original CAD design provided by the ABC dataset, DeepCAD utilizes Onshape's domain-specific language (DSL) to parse the construction operations and parameters, converting them into JSON format [Pezoa *et al.*, 2016]. Another dataset, the Fusion360 Gallery dataset [Willis *et al.*, 2021b], offers approximately 8,000 CAD designs created by human designers using sketching and extrusion operations. However, like DeepCAD, the Fusion360 Gallery dataset also primarily focuses on sketch profiles and extrusion operations, limiting the diversity of CAD model creation. Furthermore, many CAD models in these two datasets are typically designed by learners rather than professional designers, resulting in greater randomness in reasoning about concurrent 3D shapes compared to basic engineering components. To this end, we have developed our own CAD dataset, specifically targeting authentic engineering designs. Our dataset encompasses a wide range of modeling features to support further CAD reconstruction and analysis research.

2.2 CAD Reconstruction

The primary task of CAD reconstruction is to recover the underlying CAD programs that describe how a model is constructed from scratch, given its representation in B-Rep, mesh, or point cloud format. These sequences are crucial for preserving the editability of CAD models, allowing for subsequent modifications such as model simplification for simulation or adjustments of manufacturing tolerances. Previous works on CAD reconstruction can be broadly categorized into Constructive Solid Geometry (CSG)-based methods and sequence-based methods.

CSG-based methods aim to parse the input model into a binary tree structure, where each leaf represents a primitive object (such as a cube, cylinder, or sphere), and each internal node represents a boolean operation (such as intersection, union, or difference) [Laidlaw *et al.*, 1986]. CSGNet [Sharma *et al.*, 2018] and UCSGNet [Kania *et al.*, 2020] employ neural networks to parse 3D voxel input and generate a CSG tree.

However, CSGNet relies on synthetic data for training, which may not be well-suited for real-world 3D CAD datasets. On the other hand, UCSGNet presents a self-supervised approach to address the challenge of limited CAD data availability. Alternatively, Wu et al. [Wu *et al.*, 2018] and InverseCSG [Du *et al.*, 2018] take a different approach by extracting primitives and constructing a CSG tree bottom-to-up through energy minimization based on voxel or point labels. These methods do not rely on neural networks for the construction process. It's worth noting that all the mentioned CSG-based methods operate on voxel or mesh representations. However, compared to B-Rep, both voxel and mesh representations have lower levels of detail (LOD). In our approach, we feed the B-Rep model directly into the network to extract high-level features of the CAD model. It should be noted that CSG methods face challenges in building complex surfaces, such as Non-uniform rational B-splines (NURBS), which are commonly used in manufacturing CAD models.

Another aspect of CAD reconstruction involves recovering a parametric sequence that describes how 2D skeletons are designed and transformed into 3D solids. This approach, referred to as sequence-based methods, has been addressed by several existing works, namely Sketch2CAD [Li *et al.*, 2020], Fusion360 [Willis *et al.*, 2021b], and ZoneGraph [Xu *et al.*, 2021]. Sketch2CAD focuses on inferring the sequence program by recognizing and segmenting CAD operations from sketches. However, it does not take B-Rep input directly, and during testing, it requires a sketch image to guide the prediction, which is not readily available in real-world scenarios. Fusion360 aims to parse B-Rep input using the Autodesk Fusion 360 desktop CAD application. It predicts the next operation at each step by comparing the target shape with the current state. ZoneGraph employs a graph of zones to represent the B-Rep input, reducing the search space of CAD programs by converting the infinite space with continuous parameters into a finite space with a set of operation sequences. Fusion360 and ZoneGraph take B-Rep input and predict the underlying sequence to construct the input model. However, to simplify the prediction task, these methods start with face extrusion, assuming the faces are already prepared. This approach overlooks the surface creation process and makes it challenging to explicitly import the prediction results into CAD software for downstream edits. Furthermore, these two methods focus only on sketch and extrusion operations for simplicity, which limits the diversity of features within actual CAD models. This work presents a learning approach inspired by DeepCAD [Wu *et al.*, 2021] to predict direct parametric sequences suitable for CAD software. Additionally, we aim to incorporate more rich features of CAD models into our methodology.

2.3 Transformer Based Models

Technically, our work builds upon the Transformer networkk [Vaswani *et al.*, 2017], demonstrating significant success in various 3D shape generation tasks. For example, SceneFormer [Wang *et al.*, 2021] utilizes multiple Transformers for generating indoor scenes, while PolyGen [Nash *et al.*, 2020] employs Transformers to create vertices and faces of 3D meshes. DeepCAD utilizes Transformers to develop standard B-Rep CAD models. Among these works, DeepCAD is particularly relevant to our research. However, unlike these models, our main objective is to parse CAD models in the context of engineering B-Rep input, which enables downstream edits and lightens designers' workload rather than generating entirely new shapes. Essentially, our focus lies in reverse engineering shapes rather than developing them.

3 Method Overview

Our approach takes as input a manufactured CAD model in the format of boundary representation and predicts its construction workflow. To support our learning approach, we also introduce a manufactured CAD dataset.

3.1 CAD Dataset

To address the limited diversity of CAD models in existing datasets, we have created a comprehensive collection of CAD manufacturing models comprising approximately 40,000 models. The construction of our dataset involved the following steps, and we plan to make it publicly available shortly.

Collection of CAD Dataset

We first collect a gallery of manufacturing CAD models online, and all these models are designed via the CAD software SolidWorks. Differing from the Fusion360 Gallery [Willis *et al.*, 2021b] and DeepCAD dataset [Wu *et al.*, 2021], the models of which are usually designed for human learning or just for fun, our gallery data are all manufacturing components, including both the National standard parts and the standard enterprise parts. Each model contains a geometry structure and the corresponding design history. Similar to the Fusion360 Gallery [Willis *et al.*, 2021b], we parse the design history into a representation that is friendly to our learning network. To accomplish this parse process, We build a script employing native SolidWorks APIs. To parse the native model, we analyze the model feature statistically and filter the operations which is less than 90% in the count. So, in our dataset, we parse five modeling operations: sketch, extrusion, revolution, fillet, and chamfer. In total, we collect ˜10000 cad models as the template of our dataset. The specific operations and parameters are listed in Table 3.

CAD Dataset Augmentation

Since our CAD model templates only contain ˜10000 objects, which is insufficient for our learning approach, we need to increase the number of models as much as possible. However, because the CAD model is strictly constrained, random generation of the CAD model is unavailable. To address this challenge, we provide a novel data augmentation method to enrich our dataset. Specifically, we transform the design history to the model sequence so that the modeling process can be considered a series of steps. In particular, sketch and extrusion or sketch and revolution are supposed to be one step. We modify the last step iteratively from back to front, considering that subsequent steps depend on previous ones. The modifications mainly involve cutting the last step and adjusting the parameters of the last modeling operation. Additionally, we validate the shape of each model using the PythonOCC API

Op	S	RP	E	Ec	R	Rc	F	Cf
Ratio	40%	11%	11%	10%	8%	5%	5%	2%

Table 1: CAD operation analysis on our Dataset

Sequence length	0-5	5-10	10-15	15-20	20+
Ratio	64%	23%	5%	3%	5%

Table 2: CAD sequence length analysis on our Dataset

to ensure that only valid CAD models are included. Through this augmentation method, we have expanded our dataset to approximately 40,000 models.

We also performed a statistical analysis on our dataset to examine the distribution of CAD operation types and the lengths of CAD modeling sequences. Table 1 presents the ratios of CAD operations in our collected data, where 'S' means the 2d sketch, 'RP' indicates the reference plane on which the drawing locates, and other operations can refer to Table 3.

Table 2 displays the results of our statistical analysis on the lengths of CAD modeling sequences in our dataset. It is important to note that when calculating the sequence length, we treat the sketch feature as a single unit, although it comprises multiple line and segment operations.

3.2 CAD Representation

In order to learn deep neural networks capable of encoding and predicting CAD models, we first need a well-defined data structure for the boundary representation and construction workflow. Since boundary representation describes the CAD model as a solid volume enclosed by trimmed faces, where a parametric face is also a region connected by model edges, this strict adjacency relationship encourages us to consider it a graph. And construction workflow can be seen as an ordered set of various feature-based operations in the current CAD design software. So, we model it as the feature command sequences reasonably.

Graph Construction

Inspired by the B-RepNet [Lambourne *et al.*, 2021] and Uv-net [Jayaraman *et al.*, 2021], a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is constructed for one input B-Rep CAD model, where vertex $v_i \in \mathcal{V}$ includes the faces, edges and coedges and $\mathcal{E}$ describes the adjacency relationship of them. For face features, we stack the surface geometry feature and uv grids feature introduced from [Lambourne *et al.*, 2021] and [Jayaraman *et al.*, 2021]. Concretely, for edge feature and coedges features, we only use that geometry information referring to [Lambourne *et al.*, 2021]. In this way, the vertex features are generated. According to the relationship that faces $\rightarrow$ edge $\rightarrow$ half edge, we construct the adjacency matrix.

Sequence Construction

We model the construction workflow of a CAD model as the feature-based command sequences because a solid object is created by iteratively adding features such as sketch, extrusion, or revolution. This analogy motivates us to leverage the natural language processing approaches like Transformer networks[Vaswani *et al.*, 2017] to achieve our goal. But different from the natural language vocabulary consisting of individual words, each CAD feature command has a different number of parameters, and a different parameter type contains continuous or discrete values. To make the CAD commands suitable for the NLP approaches, we define the construction workflow $S = (C_1, C_2, ..., C_{N_C})$ where C_i is an individual feature command in the creation step. The command $C_i = (t_i, \boldsymbol{p}_i)$ is defined itself by its command type t_i and corresponding parameters $\boldsymbol{p}_i$. The specification of the CAD command is listed in Table 3.

Commands	Parameters	Description
<SOS>	$\emptyset$	
L(Line)	x, y	line end point
A(Arc)	x, y	arc end point
	α	sweep angle
	f	clockwise flag
C(Circle)	x, y	circle center point
	r	circle radius
E(Extrusion)	t_x, t_y, t_z	sketch plane transformation
	θ, γ, δ	sketch plane rotation
	s	scale of the sketch
	$e1, e2$	extrude distance for both direction
Ec(Extrusion Cut)	= E	
Ax(Revolution Axis)	t_x, t_y, t_z	sketch plane transformation
	θ, γ, δ	sketch plane rotation
R(Revolution)	t_x, t_y, t_z	sketch plane transformation
	θ, γ, δ	sketch plane rotation
	s	scale of the sketch
	α	revolution angle
Rc(Revolution Cut)	= R	
F(Fillet)	p_x, p_y, p_z	3D point in the filleted edge
Cf(Chamfer)	= F	3D point in the chamfered edge
<PAD>	$\emptyset$	
<EOS>	$\emptyset$	

Table 3: CAD commands with their type and parameters.

To tackle the trouble of different numbers of parameters in various CAD commands, we fix the length of a command by stacking the whole of the parameters from all the CAD commands into one vector. In this way, we formulate the command $C_i \in \mathbb{R}^{19\times 1}$, and unused parameters for each command are set to -1. To ensure efficient parallel processing, we use fixed-length N_C commands in each workflow sequence. In this paper, N_C is set to 32 according to the statistic of our dataset. To address the problem of the mixture of both continuous and discrete parameter values in a command, we refer to DeepCAD[Wu *et al.*, 2021]. After normalizing a solid input into a $2 \times 2 \times 2$ cube, each continuous parameter value is quantized into 256 levels. So far, we have transferred the raw CAD commands into a network-friendly representation.

3.3 Network Architecture

We present our deep architecture comprising a graph encoder for boundary representation and a sequence decoder for CAD commands as shown in Figure 2. Our objective is to forecast

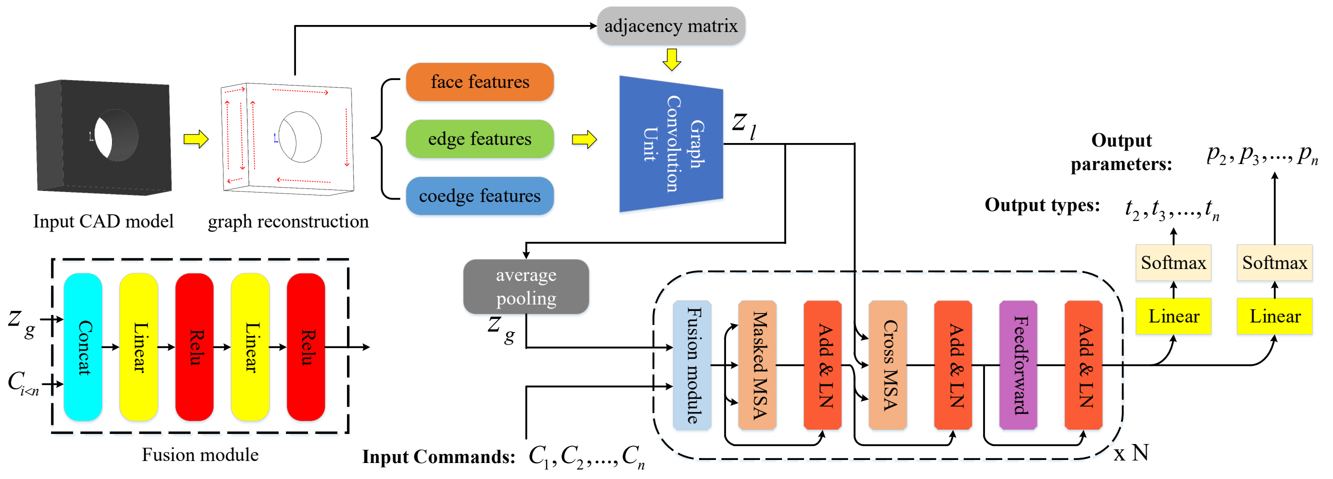

Figure 2: Overview of our network architecture. The input CAD model represented in the B-Rep format is first used to construct the CAD graph. Then, the face features, edge features, co-edge features, and the adjacent matrix are extracted from the graph. These features are passed through the convolution unit to result in the latent vector z_l. z_g is calculated as the average pooling of z_l. Then, z_g and input tokens C_i are combined and fed into the Decoder layer composed of N Transformer Blocks stacks with the cross attention z_l to predict the command type t_i along with the command parameters $\boldsymbol{p}_i$.

the command sequences revealing the construction process of the given B-Rep model.

Embedding

To enable the Transformer[Vaswani *et al.*, 2017] based decoder, we first need to project the discrete CAD commands into highly continuous embedding space of dimension d_E. With the nature of each CAD command C_i having individual command type t_i and corresponding parameters $\boldsymbol{p}_i$, we split the command into two parts and project them into different dimensions respectively. In particular, we formulate the CAD command C_i as $e(C_i) = e^i_{\text{cmd}} + e^i_{\text{param}} + e^i_{\text{ind}} \in \mathbb{R}^{d_E}$.

For command type embedding, we define $e^i_{\text{cmd}} = W^i_{\text{cmd}}\delta^i_c$, where $W^i_{\text{cmd}} \in \mathbb{R}^{d_E \times 12}$ is a learnable matrix, and δ^i_c is a 12-dimensional one-hot vector indicating the command type among the 12 CAD command types.

For command parameters, since we have discretized continuous values and introduced an unused value -1 for the current command, we lead to an embedding dimension $2^8 + 1 = 257$. Thus each command parameter is first projected into the embedding space separately with a matrix $W_X \in \mathbb{R}^{d_E \times 257}$, then we combine the separate parameter embedding and project it into a d_E-dimensional vector with a linear layer $W_{\text{param}} \in \mathbb{R}^{d_E \times 19}$, namely,

$$e^i_{\text{param}} = W^i_{\text{param}} \text{f}(W_X \delta^i_{\boldsymbol{p}}) \tag{1}$$

where f$(\cdot)$ denotes that flatten the combined parameter embedding into one vector.

Finally, we add an index embedding to indict the index of the CAD command given a sequence with a learnable matrix $W_{\text{ind}} \in \mathbb{R}^{d_E \times N_c}$, so $e^i_{\text{ind}} = W^i_{\text{ind}}\delta^i$.

Encoder

To encode the B-Rep model input, we use the backbone proposed by [Lambourne *et al.*, 2021] as our graph encoder. Through the special adjacency relationship named Topological walks, we can perform the convolution operations for the graph and obtain the local latent vector z_{lf}, z_{le}, z_{lc} of faces, edges, and co-edges, respectively. Then, the global feature vector z_{gf}, z_{ge}, z_{gc} of those is computed, followed by an average pooling operation. In this way, we can describe each CAD model with global and local features.

Feature Fusion

To leverage the global and local features of the input model, we design a fusion module that passes through the local feature average pooling to get a global feature and concatenates the input embeddings and global features. The stacked vector is then passed through two blocks composed of a linear layer and a relu layer. The output is fed into the decoder to predict the CAD commands.

Decoder

Considering the effectiveness and excellent performance of Transformer architecture[Vaswani *et al.*, 2017], we leverage the Transformer blocks as our decoder backbone. Specifically, we use 4 Transformer layers as our decoder, with 6 attention blocks in each layer. Our decoder takes as input the unmasked sequence embedding while also performing the cross-attention operation within the Transformer layer with the latent vector from the encoder. And we predict the command type t_i and command parameters $\boldsymbol{p}_i$ separately by feeding the output vector into two different linear layers. With the discrete nature of both t_i and $\boldsymbol{p}_i$, we decompose the joint distribution over the sequence of commands as the product of a series of conditional commands distributions:

$$p(S;\theta) = \prod_{i=1}^{N_c} p(t_i, \boldsymbol{p}_i | t_{<i}, \boldsymbol{p}_{<i}, z; \theta) \tag{2}$$

We model this distribution using an autoregressive fashion that outputs at each step the command and parameters of a predictive distribution for the next sequence command. The model is trained to maximize the log probability of the observed data with respect to the model parameters θ.

3.4 Training Details

We use the AdamW[Loshchilov and Hutter, 2017] optimizer with an initial learning rate 10^{-3}, reduced by a factor of 0.9 every 30 epochs and a linear warmup period of 10 initial epochs. We use a dropout rate of 0.1 in all transformer layers and a gradient clipping of 1.0. We train our networks for 100 epochs with a total batch size of 96 on one 1080Ti GPU, which takes about one day.

4 Experiments

In this section, we evaluate our CADParser on the manufactured dataset. We split our collected models into training and test set, where the test set counts 1000. And we use our training set to train our CADParser approach.

4.1 Evaluate Metric and Comparison Methods

Although the output of our method is command sequences, the B-Rep model can be computed based on the CAD engine. So We use intersection over union (IoU) to evaluate network performance since there is more than one answer for the same input model in some cases. To calculate the IOU between two B-Rep models, we accomplish it using OpenCascade. Since the sequence-based prediction is under strict constraints, it sometimes fails when working. Thus, we introduce the Recall metric to express the failure ratio in the parsing process; clearly, we also consider it a failure case when the IOU is less than 0.5. We also use the parse time as our evaluation metric to validate the efficiency of comparison methods.

There are only a few works to operate directly on the B-Rep models, so we compare with two existing state-of-the-art methods: Fusion360 Gym[Willis *et al.*, 2021b]and ZoneGraph[Xu *et al.*, 2021]. For Fusion360 Gym, since it is based on the Fusion360 software, the algorithm is integrated as a service. We here apply the agent strategy in prediction and utilize the model weights released by the Fusion360 Gym. For ZoneGraph, we train it from scratch as there is no model published. Because training ZoneGraph requires the extrusion tools we lack, we train the ZoneGraph using the Fusin360 reconstruction subset for 10 epochs. When inferring the sequences, we set the option of ZoneGraph with max time to 300 and max step to 15.

Table 4 shows the results of three evaluation metrics. We can see that both fusion360Gym and Zonegraph can achieve more than 90% on the IOU. But the difference between them is Fusion360Gym can work well on the Recall with 82% performance, but Zonegraph can only obtain about 40%, far less than the Fusion360Gym. Our model CADParser achieves 81% on the IOU performance, but for Recall, our model works better than Zonegraph, which approaches 64%, but still less than the Fusion360Gym. However, for the efficiency of the methods, our method is superior to other methods, which is an order of magnitude lower than others.

Methods	IOU	Recall	Runtime(s)
Fusion360 Gym	0.91	**0.82**	20
ZoneGraph	**0.95**	0.42	38.5
Ours	0.81	0.64	**3.9**

Table 4: The quantitative results among the comparison methods. The best result are bold.

The reason for the gap on IOU, we think that our model quantizes the command while Fusion360Gym[Willis *et al.*, 2021b] and ZoneGraph[Xu *et al.*, 2021] deal the problem with that called face extrusion, which doesn't predict the command sequence but instead selects a group of parallel faces to predict the underlying extrude operation. Based on its intrinsic property, face extrusion is very suitable for identifying extrusion operations. But it also has no expansibility, which can't recognize other features. Another reason is that the IoU in three-dimensional space can introduce weight imbalance among CAD features as the IoU metric primarily focuses on the global geometric structure. However, this metric may overlook the importance of other instructions, leading to the neglect of local features.

Figure 3 shows the qualitative comparison, we can see that our CADParser can achieve state-of-the-art performance. Especially for some input models with other types of faces(not plain), Fusion360Gym and ZoneGraph will fail for this part, but our model can reconstruct the corresponding geometry part like the result of the first row.

4.2 Ablation Study

We conduct several ablation studies to qualify the influences of different modules in our model. We mainly perform the ablation study from three perspectives: the encoder features, the decoder backbone, and data augmentation. Here we use IOU and Recall to discover how well the command sequences inferred from the input B-Rep model.

We first evaluate the influence of the encoder feature on the performance of CADParser. To identify which feature is essential for our sequence decoder, we remove the fusion module to predict with only local or global features. As shown in Table 5. We see that removing either global features or local features will cause the reduction of mean IOU and Recall. Without local features, the mean IOU reduces by about 1% while it decreases by about 2% without global features. However, for Recall, removing local features is more degraded than removing global features.

Methods	IOU	Recall
Encoder + LSTM	0.73	0.40
CADParser + Local	0.79	0.62
CADParser + Global	0.80	0.61
CADParser + Local + Global w/o Aug	0.78	0.45
CADParser + Local + Global + Aug	**0.81**	**0.64**

Table 5: Ablation study results on mean IOU and Recall. The best results are bold.

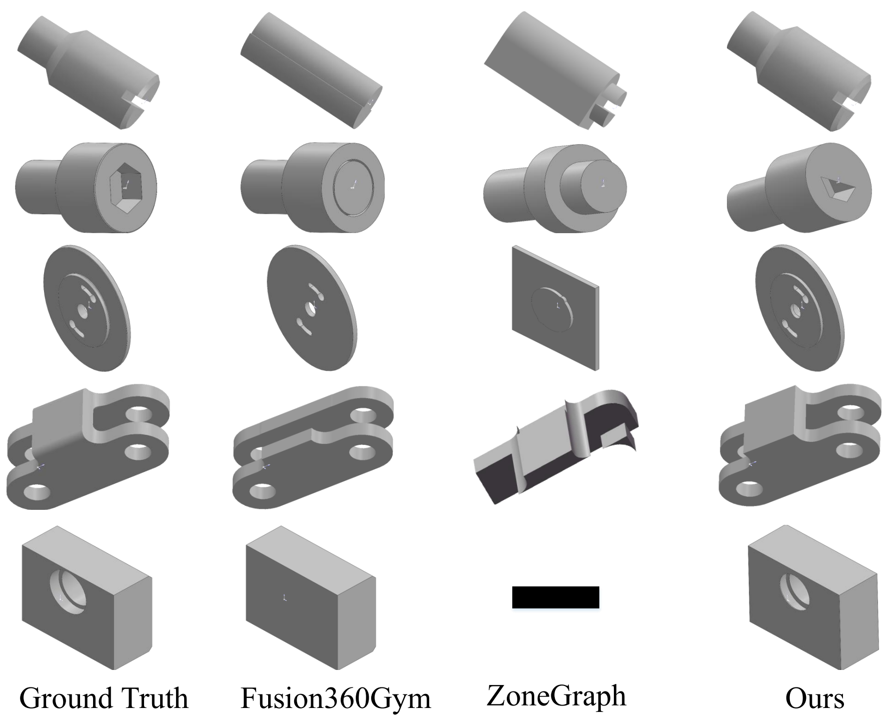

Figure 3: Results comparison of reconstruction, the empty means the failure of the corresponding method.

Since the success of the Transformer networks is highly dependent on a large amount of data, but our training set only contains less than 40000 models, we replace the transformer decoder with an LSTM decoder to identify if the transformer decoder still works well in this case. As Table 5 shows, the Transformer decoder surpasses the LSTM decoder 11% on mean IOU and 60% on Recall. So, even if the training set is not so large, the transformer decoder can still work well and present excellent performance.

We also check the influence of data augmentation on final results. Table 5 shows that removing the data augmentation process causes a 4% decrease in mean IOU. But we can discover a 30% decrease in Recall without data augmentation, which indicates that the necessary augmentation.

5 Discussion and Limitation

Our CADParser aims to directly parse the B-Rep model into the construction sequence compatible with the current CAD softwares. Compared to face extrusion, our method can deal with more features, and it is extensible. In contrast, our models lose the accuracy in the geometry information.

Another disadvantage of the CAD command sequence is that the CAD design is under such strict constraints that our results sometimes fail to reconstruct the geometry model. So, we think in future work, the CAD design constraints can be considered in our model to output more correct and reasonable results, avoiding the enormous error in parsing.

Meanwhile, relying solely on the IoU metric may not comprehensively assess sequence-based reconstruction. So a more reliable metric is expected in future work, considering global and local features.

6 Conclusion

We have presented a learning approach CADParser, a neural network architecture that can directly parse the B-Rep models into command sequences. Unlike previous methods, CADParser aims to operate directly on the manufactured object to explore practical applications. To support the CADParser, we also introduce a CAD dataset containing ˜40000 manufactured objects. Compared to the previous CAD datasets, our dataset extends the diversity of CAD commands. For the performance of our CADParser, we focus on our specific CAD dataset, and our method can approach the state-of-the-art result. Another highlight of our method is can directly output the explicit command sequence, which is compatible with current CAD design software. In future work, we can integrate the algorithm into the software and parse more kinds of models.

Acknowledgments

We would like to thank the anonymous reviewers for their constructive comments. This work was supported in part by National Natural Science Foundation of China (61932003), and Beijing Science and Technology Plan Project (Z221100006322003).

References

[Chang *et al.*, 2015] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

[Du *et al.*, 2018] Tao Du, Jeevana Priya Inala, Yewen Pu, Andrew Spielberg, Adriana Schulz, Daniela Rus, Armando Solar-Lezama, and Wojciech Matusik. Inversecsg: Automatic conversion of 3d models to csg trees. *ACM Transactions on Graphics (TOG)*, 37(6):1–16, 2018.

[Ganin *et al.*, 2021] Yaroslav Ganin, Sergey Bartunov, Yujia Li, Ethan Keller, and Stefano Saliceti. Computer-aided design as language. *Advances in Neural Information Processing Systems*, 34, 2021.

[Hanocka *et al.*, 2019] Rana Hanocka, Amir Hertz, Noa Fish, Raja Giryes, Shachar Fleishman, and Daniel Cohen-Or. Meshcnn: a network with an edge. *ACM Transactions on Graphics (TOG)*, 38(4):1–12, 2019.

[Jayaraman *et al.*, 2021] Pradeep Kumar Jayaraman, Aditya Sanghi, Joseph G Lambourne, Karl DD Willis, Thomas Davies, Hooman Shayani, and Nigel Morris. Uv-net: Learning from boundary representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11703–11712, 2021.

[Kania *et al.*, 2020] Kacper Kania, Maciej Zieba, and Tomasz Kajdanowicz. Ucsg-net-unsupervised discovering of constructive solid geometry tree. *Advances in Neural Information Processing Systems*, 33:8776–8786, 2020.

[Kim *et al.*, 2020] Sangpil Kim, Hyung-gun Chi, Xiao Hu, Qixing Huang, and Karthik Ramani. A large-scale annotated mechanical components benchmark for classification and retrieval tasks with deep neural networks. In *European Conference on Computer Vision*, pages 175–191. Springer, 2020.

[Koch *et al.*, 2019] Sebastian Koch, Albert Matveev, Zhongshi Jiang, Francis Williams, Alexey Artemov, Evgeny Burnaev, Marc Alexa, Denis Zorin, and Daniele Panozzo. Abc: A big cad model dataset for geometric deep learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9601–9611, 2019.

[Laidlaw *et al.*, 1986] David H. Laidlaw, W. Benjamin Trumbore, and John F. Hughes. Constructive solid geometry for polyhedral objects. In *Computer Graphics (Proceedings of SIGGRAPH 86)*, volume 20, pages 161–170, August 1986.

[Lambourne *et al.*, 2021] Joseph G Lambourne, Karl DD Willis, Pradeep Kumar Jayaraman, Aditya Sanghi, Peter Meltzer, and Hooman Shayani. Brepnet: A topological message passing system for solid models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12773–12782, 2021.

[Li *et al.*, 2020] Changjian Li, Hao Pan, Adrien Bousseau, and Niloy J Mitra. Sketch2cad: Sequential cad modeling by sketching in context. *ACM Transactions on Graphics (TOG)*, 39(6):1–14, 2020.

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

[Mescheder *et al.*, 2019] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4460–4470, 2019.

[Mo *et al.*, 2019] Kaichun Mo, Shilin Zhu, Angel X Chang, Li Yi, Subarna Tripathi, Leonidas J Guibas, and Hao Su. Partnet: A large-scale benchmark for fine-grained and hierarchical part-level 3d object understanding. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 909–918, 2019.

[Nash *et al.*, 2020] Charlie Nash, Yaroslav Ganin, SM Ali Eslami, and Peter Battaglia. Polygen: An autoregressive generative model of 3d meshes. In *International Conference on Machine Learning*, pages 7220–7229. PMLR, 2020.

[Ons,] Onshape. https://www.onshape.com/, Accessed: 2019-08-01.

[Para *et al.*, 2021] Wamiq Para, Shariq Bhat, Paul Guerrero, Tom Kelly, Niloy Mitra, Leonidas J Guibas, and Peter Wonka. Sketchgen: Generating constrained cad sketches. *Advances in Neural Information Processing Systems*, 34, 2021.

[Pezoa *et al.*, 2016] Felipe Pezoa, Juan L Reutter, Fernando Suarez, Martín Ugarte, and Domagoj Vrgoč. Foundations of json schema. In *Proceedings of the 25th International Conference on World Wide Web*, pages 263–273, 2016.

[Qi *et al.*, 2017] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[Seff *et al.*, 2020] Ari Seff, Yaniv Ovadia, Wenda Zhou, and Ryan P. Adams. SketchGraphs: A large-scale dataset for modeling relational geometry in computer-aided design. In *ICML 2020 Workshop on Object-Oriented Learning*, 2020.

[Sharma *et al.*, 2018] Gopal Sharma, Rishabh Goyal, Difan Liu, Evangelos Kalogerakis, and Subhransu Maji. Csgnet: Neural shape parser for constructive solid geometry. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5515–5523, 2018.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[Wang *et al.*, 2021] Xinpeng Wang, Chandan Yeshwanth, and Matthias Nießner. Sceneformer: Indoor scene generation with transformers. In *2021 International Conference on 3D Vision (3DV)*, pages 106–115. IEEE, 2021.

[Willis *et al.*, 2021a] Karl DD Willis, Pradeep Kumar Jayaraman, Joseph G Lambourne, Hang Chu, and Yewen Pu. Engineering sketch generation for computer-aided design. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2105–2114, 2021.

[Willis *et al.*, 2021b] Karl DD Willis, Yewen Pu, Jieliang Luo, Hang Chu, Tao Du, Joseph G Lambourne, Armando Solar-Lezama, and Wojciech Matusik. Fusion 360 gallery: A dataset and environment for programmatic cad construction from human design sequences. *ACM Transactions on Graphics (TOG)*, 40(4):1–24, 2021.

[Wu *et al.*, 2015] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015.

[Wu *et al.*, 2018] Qiaoyun Wu, Kai Xu, and Jun Wang. Constructing 3d csg models from 3d raw point clouds. In *Computer Graphics Forum*, volume 37, pages 221–232. Wiley Online Library, 2018.

[Wu *et al.*, 2021] Rundi Wu, Chang Xiao, and Changxi Zheng. Deepcad: A deep generative network for computer-aided design models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6772–6782, 2021.

[Xu *et al.*, 2021] Xianghao Xu, Wenzhe Peng, Chin-Yi Cheng, Karl DD Willis, and Daniel Ritchie. Inferring cad modeling sequences using zone graphs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6062–6070, 2021.