Supplementary Document
DeepCAD: A Deep Generative Network for Computer-Aided Design Models

Rundi Wu Chang Xiao Changxi Zheng
Columbia University
{rundi, chang, cxz}@cs.columbia.edu

A. CAD dataset

We create our dataset by parsing the command sequences of CAD models in Onshape's online repository. Some example models from our dataset are shown in Fig. 9. Unlike other 3D shape datasets which have specific categories (such as chairs and cars), this dataset are mostly user-created mechanical parts, and they have diverse shapes.

Our dataset is derived from the ABC dataset [23], which contains several duplicate shapes. So for each shape in the test set, we find its nearest neighbor in the training set based on chamfer distance; we discard this test shape if the nearest distance is below a threshold.

We further examine the data distribution in terms of CAD command sequence length and the number of extrusions (see Fig. 8). Most CAD command sequences are no longer than 40 or use less than 8 extrusions, as these CAD models are all manually created by the user. A similar command length distribution is also reported in Fusion 360 Gallery [48].

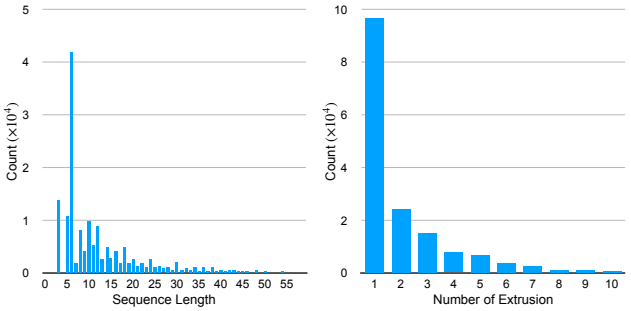

Figure 8. Statistics of CAD training data in terms of command sequence length (left) and number of extrusions (right).

B. Command Parameter Representation

Recall that our list of the full command parameters is $\boldsymbol{p}_i = [x, y, \alpha, f, r, \theta, \phi, \gamma, p_x, p_y, p_z, s, e_1, e_2, b, u]$ in Table 1. As described in Sec. 3.1.2, we normalize and quantize these parameters.

First, we scale every CAD model within a $2 \times 2 \times 2$ cube (without translation) such that all parameters stay bounded: the sketch plane origin (p_x, p_y, p_z) and the two-side extrusion distances (e_1, e_2) range in $[-1, 1]$; the scale of associated sketch profile s is within $[0, 2]$; and the sketch orientation (θ, ϕ, γ) have the range $[-\pi, \pi]$.

Next, we normalize every sketch profile into a unit square such that its starting point (i.e., the bottom-left point) locates at the center $(0.5, 0.5)$. As a result, the curve's ending position (x, y) and the radius r of a circle stay within $[0, 1]$. The arc's sweeping angle α by definition is in $[0, 2\pi]$.

Afterwards, we quantize all continuous parameters into 256 levels and express them using 8-bit integers.

For discrete parameters, we directly use their values. The arc's counter-clockwise flag f is a binary sign: 0 indicates clockwise arc and 1 indicates counter-clockwise arc. The CSG operation type $b \in \{0, 1, 2, 3\}$ indicates *new body*, *join*, *cut* and *intersect*, respectively. Lastly, the extrusion type $u \in \{0, 1, 2\}$ indicates *one-sided*, *symmetric* and *two-sided*, respectively.

C. Network Architecture and Training Details

Autoencoder. Our Transformer-based encoder and decoder are both composed of four layers of Transformer blocks, each with eight attention heads and a feed-forward dimension of 512. We adopt standard layer normalization and a dropout rate of 0.1 for each Transformer block.

The last Transformer block in the decoder is followed by two separate linear layers, one for predicting command type (with weights $W_1 \in \mathbb{R}^{256\times 6}$), and another for predicting command parameters (with weights $W_2 \in \mathbb{R}^{256\times 4096}$). The output, a 4096-dimensional vector, from the second linear layer is further reshaped into a matrix of shape 16×256, which indicates each of the total 16 parameters.

Latent-GAN. Section 3.4 describes the use of latent-GAN technique on our learned latent space for CAD generation. In our GAN model, the generator and discriminator are both MLP networks, each with four hidden layers. Every hidden layer has a dimension of 512. The input dimension (or the noise dimension) is 64, and the output dimension is 256.

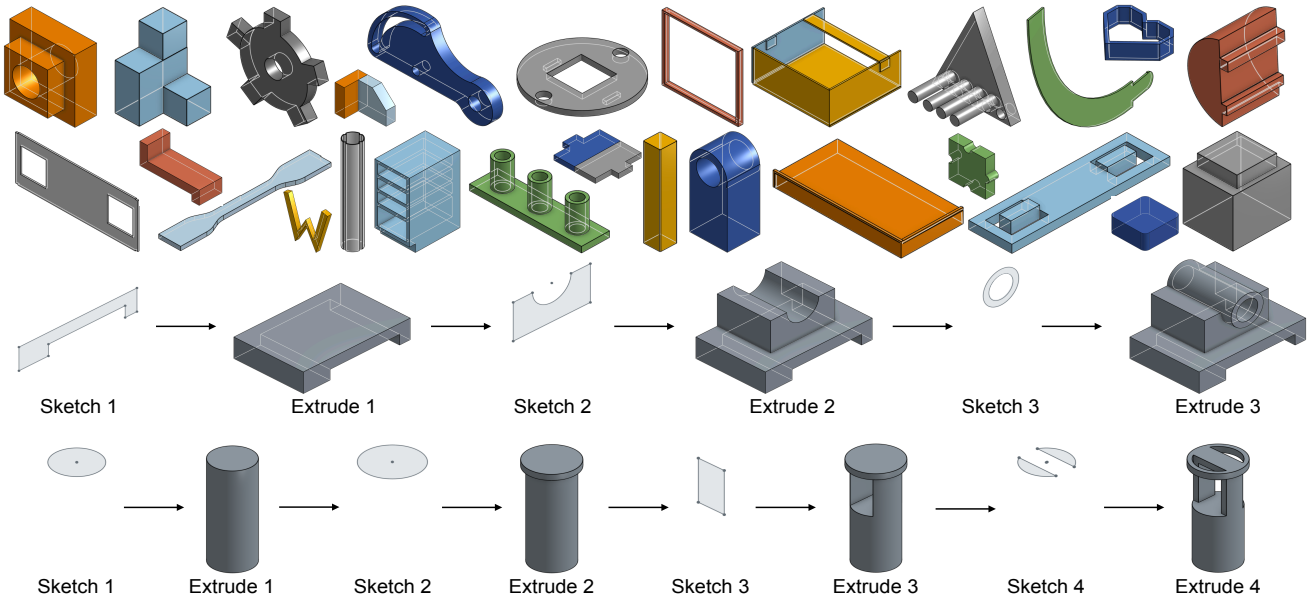

Figure 9. **Our CAD dataset**. Top two rows: examples of CAD models in our dataset. Bottom two rows: examples of CAD construction sequences.

We use WGAN-gp strategy [7, 18] to train the network: the number of critic iterations is set to 5 and the weight factor for gradient penalty is set to 10. The training lasts for $200,000$ iterations with a batch size of 256. In this process, Adam optimizer is used with a learning rate of 2×10^{-4} and $\beta_1 = 0.5$.

D. Autoencoding CAD models

Comparison methods. Here we describe in details the variants of our method used in Sec. 4.1 for comparison.

`Alt-Rel` represents curve positions relative to the position of its previous curve in the loop. As a result, the ending positions of a line and a arc and the center of a circle differ from those in our method, but the representation of other curve parameters (i.e., α, f, r in Table 1) stays the same.

`Alt-Trans` includes in the extrusion command the starting position (s_x, s_y) of the loop, in addition to the origin of the sketch plane. The origin (p_x, p_y, p_z) is in the world frame of reference; the loop's starting position (s_x, s_y) is described in the local frame of the sketch plane. In our proposed approach, however, we translate the sketch plane's origin to the loop's starting position. Thereby, there is no need to specify the parameters (s_x, s_y) explicitly.

`Alt-ArcMid` specifies an arc using its ending and middle positions. As a result, the representation of an arc becomes into (x, y, m_x, m_y), where (x, y) indicates the ending position (as in our method), but (m_x, m_y) is used to indicate the arc's middle point.

`Alt-Regr` regresses all parameters of the CAD commands using the standard mean-squared error in the loss function. The Cross-Entropy loss for discrete parameters (such as command types) stays the same as our proposed approach. But in this variant, continuous parameters are not quantized, although they are still normalized into the range $[-1, 1]$ in order to balance the mean-squared errors introduced by different parameters.

`Ours+Aug` includes randomly composed CAD command sequences in its training process. This is a way of data augmentation. When we randomly choose a CAD model from the dataset during training, there is 50% chance that the sampled CAD sequence will be mixed with another randomly sampled CAD sequence. The mixture of the two CAD command sequences is done by randomly switching one or more pairs of sketch and extrusion (in their commands). CAD sequences that contain only one pair of sketch and extrusion are not involved in this process.

Full statistics for CD scores. In Table 4, we report the mean, trimmed mean, and median chamfer distance (CD) scores for our CAD autoencoding study. "Trimmed mean"

Method	mean CD	trimmed mean CD	median CD
`Ours+Aug`	**6.14**	**0.974**	**0.752**
`Ours`	7.16	1.08	0.787
`Alt-ArcMid`	6.90	1.09	0.790
`Alt-Trans`	7.14	1.09	0.792
`Alt-Rel`	9.24	1.38	0.863
`Alt-Regr`	12.61	3.87	2.14

Table 4. Mean, trimmed mean and median chamfer distances for shape autoencoding. Numerical values are multiplied by 10^3.

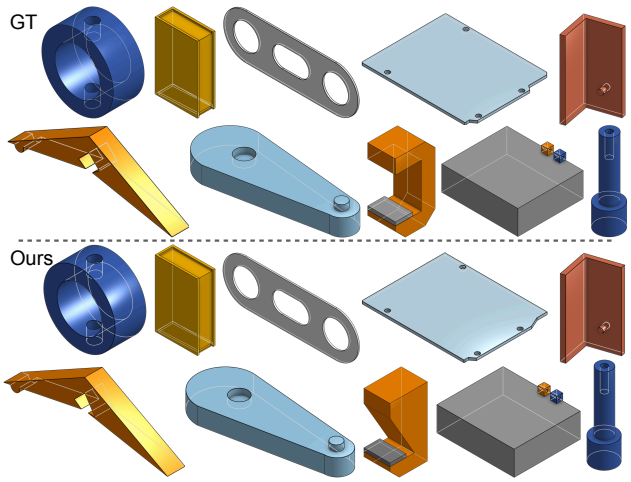

Figure 10. Shape autoencoding results on Fusion 360 Gallery [48] test data. The autoencoder is trained using our own dataset.

ACC_{cmd} ↑	ACC_{param} ↑	median CD↓	Invalid Ratio↓
97.90	96.45	0.796	1.62

Table 5. Quantitative evaluation for shape autoencoding on Fusion 360 Gallery [48] test data. The model is only trained on our proposed dataset. ↑: the higher the better, ↓: the lower the better.

CD is computed by removing 10% largest and 10% smallest scores. The mean CD scores are significantly higher than the trimmed mean and median CD scores. This is because the prediction of CAD sequence in some cases may be sensitive to small perturbations: a small change in command sequence may lead to a large change of shape topology and may even invalidate the topology (e.g., the gray shape in Fig. 11). Those cases happen rarely, but when they happen, the CD scores become significantly large. It is those outliers that make the mean CD scores much higher.

Accuracies of individual parameter types. We also examine the accuracies for individual types of parameters. The accuracy is defined in Sec. 4.1, and the results are shown in Fig. 13. While all the parameter are treated equally in the loss function, their accuracies have some differences. Most notably, the recovery of arc's sweeping angle α has lower accuracy than other parameters. By examining the dataset, we find that the values of sweeping angle α span over its value range (i.e., $[0, 2\pi]$) more evenly than other parameters, but the arc command is much less frequently used than other commands. Thus, in comparison to other parameters, it is harder to learn the recovery of the arc sweeping angle.

E. Generalization on Fusion 360 Gallery [48]

To validate the generalization ability of our autoencoder, we perform a cross-dataset test. In particular, for shape autoencoding tasks, we take the model trained on our proposed dataset and evaluate it using a different dataset provided by Fusion 360 Gallery [48]. These two datasets are constructed from different sources: ours is based on models from Onshape repository, whereas theirs is created from designs in Autodesk Fusion 360. The qualitative and quantitative results are shown in Fig. 10 and Table 5, respectively, showing that our trained model performs well on shape distributions

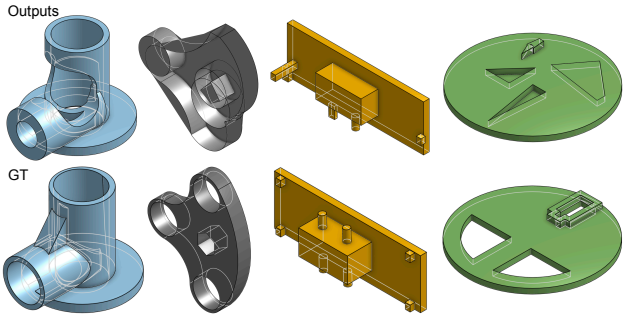

Figure 11. Failure examples in shape autoencoding. Top: our reconstructed CAD outputs. Bottom: ground-truth CAD models.

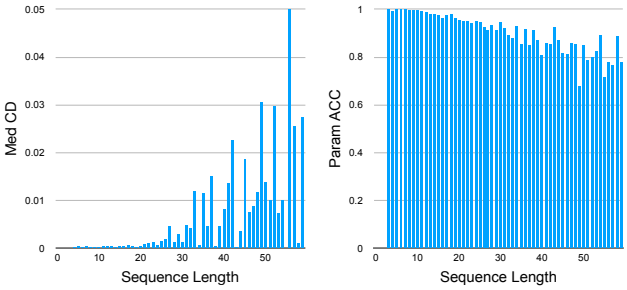

Figure 12. Quantitative metrics for shape autoencoding w.r.t. CAD sequence length. Left: median chamfer distance (the lower the better). Right: parameter accuracy (the higher the better).

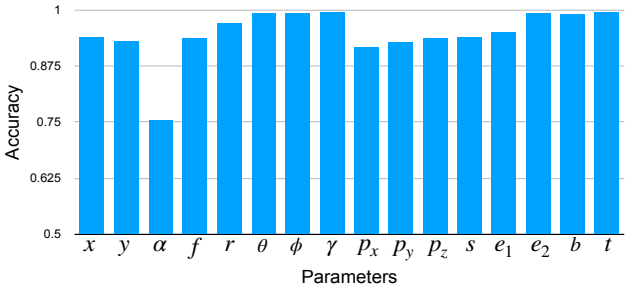

Figure 13. Accuracies for individual parameter types.

Method	ACC_{cmd} ↑	ACC_{param} ↑	median CD↓	Invalid Ratio↓
`Ours`	85.95	74.22	10.30	12.08
`Ours-noise`	84.65	74.23	10.44	13.82

Table 6. Quantitative results for CAD reconstruction from point clouds. `Ours-noise` corresponds to noisy inputs (uniform noise in $[-0.02, 0.02]$ along normal direction). We use the same metrics as in autoencoding task; ACC_{cmd} and ACC_{param} are both multiplied by 100%, and CD is multiplied by 10^3.

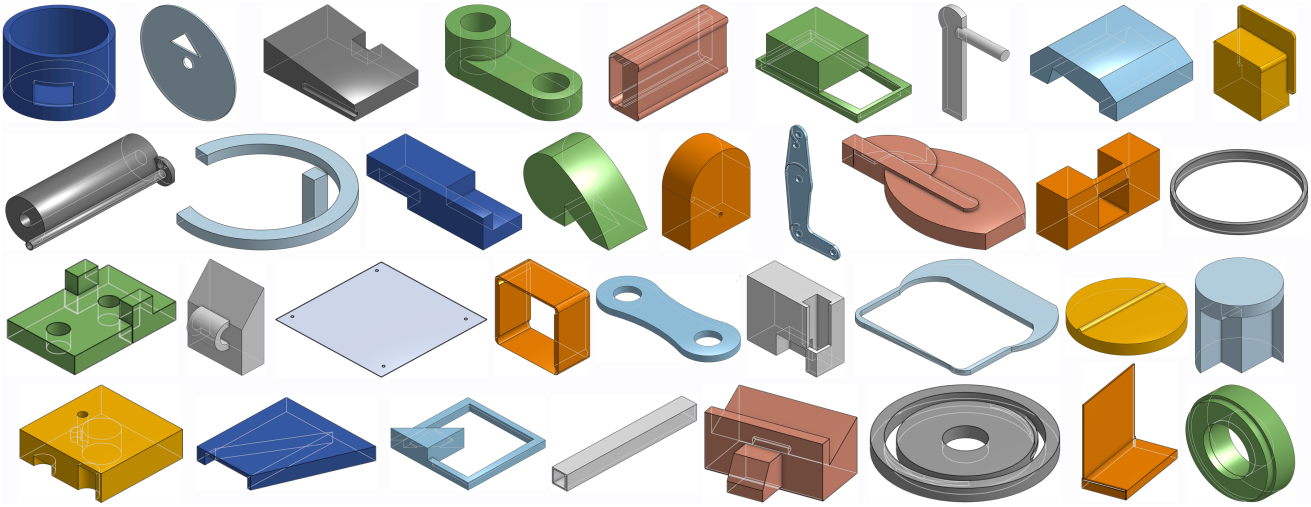

Figure 14. A gallery of our generated CAD models.

that are different from the training dataset.

F. Failure Cases

Not every CAD command sequence is valid. Our method is more likely to produce invalid CAD commands when the command length becomes long. Figure 11 shows a few failed results. The produced gray shape has invalid topology, and the yellow shape suffers from misplacement of small sketches. Figure 12 plots the median CD scores and the parameter accuracies with respect to CAD command sequence length. The difficulties for generating long-sequence CAD models are twofold. As the CAD sequence becomes longer, it is harder to ensure valid topology. Meanwhile, as shown in Fig. 8, the data distribution in terms of the sequence length has a long tail; the dataset provides much more short sequences than long sequences. This data imbalance may cause the network model to bias toward short sequences.

G. Metrics for Shape Generation

We follow the three metrics used in [6] to evaluate the quality of our shape generation. In [6], these metrics are motivated for evaluating the point-cloud generation. Therefore, for computing these metrics for CAD models, we first convert them into point clouds. Then, these metrics are defined by comparing a set of reference shapes $\mathcal{S}$ with a set of generated shapes $\mathcal{G}$.

Coverage (COV) measures the diversity of generated shapes by computing the fraction of shapes in the reference set $\mathcal{S}$ that are matched by at least one shape in the generated set $\mathcal{G}$. Formally, COV is defined as

$$\text{COV}(\mathcal{S},\mathcal{G}) = \frac{|\{\arg\min_{Y\in\mathcal{S}} d^{\text{CD}}(X,Y) | X \in \mathcal{G}\}|}{|\mathcal{S}|}, \quad (6)$$

where $d^{\text{CD}}(X,Y)$ denote the chamfer distance between two point clouds X and Y.

Minimum matching distance (MMD) measures the fidelity of generated shapes. For each shape in the reference set $\mathcal{S}$, the chamfer distance to its nearest neighbor in the generated set $\mathcal{G}$ is computed. MMD is defined as the average over all the nearest distances:

$$\text{MMD}(\mathcal{S},\mathcal{G}) = \frac{1}{|\mathcal{S}|}\sum_{Y\in\mathcal{S}} \min_{X\in\mathcal{G}} d^{\text{CD}}(X,Y). \quad (7)$$

Jensen-Shannon Divergence (JSD) is a statistical distance metric between two data distributions. Here, it measures the similarity between the reference set $\mathcal{S}$ and the generated set $\mathcal{G}$ by computing the marginal point distributions:

$$\text{JSD}(P_{\mathcal{S}}, P_{\mathcal{G}}) = \frac{1}{2} D_{\text{KL}}(P_{\mathcal{S}}||M) + \frac{1}{2} D_{\text{KL}}(P_{\mathcal{G}}||M), \quad (8)$$

where $M = \frac{1}{2}(P_{\mathcal{S}} + P_{\mathcal{G}})$ and D_{KL} is the standard KL-divergence. $P_{\mathcal{S}}$ and $P_{\mathcal{G}}$ are marginal distributions of points in the reference and generated sets, approximated by discretizing the space into 28^3 voxel grids and assigning each point from the point cloud to one of them.

Since our full test set is relatively large, we randomly sample a reference set of 1000 shapes and generate 3000 shapes using our method to compute the metric scores. To reduce the sampling bias, we repeat this evaluation process for three times and report the average scores.