CAD-SIGNet: CAD Language Inference from Point Clouds using Layer-wise Sketch Instance Guided Attention

Supplementary Material

In this document, we provide more details on the used CAD sequence representation (Section (1)), the experimental setup (Section (2)), and the evaluation (Section (3)).

1. CAD Sequence Representation Details

In this section, further details on the CAD sequence representation are provided. Table 1 provides an overview of the tokens and their value ranges within the CAD sequence representation $\mathcal{C}$. Each extrusion sequence is composed of 11 tokens specified in the following order: $\{d^+, d^-, \tau_x, \tau_y, \tau_z, \theta, \phi, \gamma, \sigma, \beta, e_e\}$. On the other hand, a sketch sequence can be defined by a variable number of tokens and follows the hierarchical structure mentioned in [11]. As described in Section 3 of the main paper, a sketch sequence consists of curves represented by a sequence of 2D point coordinates (p_x, p_y). Each curve type is formulated using the following parameters:

- Line: Start and End point.
- Arc: Start, Mid, and End point.
- Circle: Center and top-most point.

Following [10], apart from the non-numerical tokens $\{e_c, e_l, e_f, e_s, \beta, e_e, cls, end, pad\}$, all the other tokens in the CAD sequence $\mathcal{C}$ are quantized to 8 bits. Notably, the first 11 classes are reserved for the non-numerical tokens resulting in a total of $d_t = 266\ (= 2^8 + 11)$ classes. Post quantization, each one dimensional token is augmented into two dimensions with a *pad* token. An example of a CAD sequence representation is depicted in Figure 1.

Sequence Type	Token Type	Token Flags	Token Value	Description
	pad	11	0	Padding Token
	cls	0	1	Start Token
	end	0	1	End Token
	e_s	0	2	End Sketch
	e_f	0	3	End Face
	e_l	0	4	End Loop
	e_c	0	5	End Curve
	(p_x, p_y)	0	$[\![11..266]\!]^2$	Coordinates
Extrusion Sequence	d^+	1	$[\![11..266]\!]$	Extrusion Distance Towards Sketch Plane Normal
	d^-	1	$[\![11..266]\!]$	Extrusion Distance Opposite Sketch Plane Normal
	τ_x	2	$[\![11..266]\!]$	Sketch Plane Origin
	τ_y	3	$[\![11..266]\!]$	
	τ_z	4	$[\![11..266]\!]$	
	θ	5	$[\![11..266]\!]$	Sketch Plane Orientation
	ϕ	6	$[\![11..266]\!]$	
	γ	7	$[\![11..266]\!]$	
	σ	8	$[\![11..266]\!]$	Sketch Scaling Factor
	β	9	$\{7, 8, 9, 10\}$	Boolean (New, Cut, Join, Intersect)
	e_e	10	6	End Extrude

Table 1. Description of different tokens used in our CAD language representation.

2. Additional Details on Experiments

In this section, further details on the experimental procedure are provided.

2.1. Data Preprocessing Details

During preprocessing, each sketch element (faces, loops, and curves) is reordered from the original sequence order. We follow the approach of [10], in which sketch elements are reordered according to their bounding box bottom-left position in ascending order. Furthermore, curves are oriented in a counter-clockwise direction as in [10]. Similar to [10], at most 10 extrusions, are considered for our experiments, resulting in a maximum CAD sequence length of $n_{ts} = 273$. Given the variable number of extrusions within a CAD sequence $\mathcal{C}$, padding tokens (*pad, pad*) are appended at the end of the sequences during training for batch processing. This ensures that every sequence in a batch has a length of 273.

2.2. Training Details

The AdamW [6] optimizer is used during training with a learning rate of 0.001. Additionally, an ExponentialLR scheduler is applied to adjust the learning rate during training, with a decay factor of $\gamma = 0.999$. The dropout rate is set to 0.1. For the LFA [4] k-NN feature aggregation, the number of neighbors is set to 4. In the first two multi-modal transformer blocks, cross-attention is not used between the CAD sequence and point embedding. This design choice is made to prioritize the learning of the intra-modality relationship in early layers. The training time is approximately 6 days for 150 epochs.

2.3. More on Design History Evaluation Metrics

In the main paper, *F1* scores on the extrusions and curve types are reported as a measure of the quality of the predicted CAD sequences. To compute the *F1* scores, the positions of the End Sketch (e_s) and End Extrude (e_e) tokens are initially identified for each ground truth and predicted CAD sequence. This allows us to divide each sequence into a list of sketches and extrusions. The extrusion *F1* score is computed on the numbers of ground truth and predicted extrusion sequences. To compute the *F1* scores for each curve type, the procedure described in Algorithm 1 is used. In this algorithm, the loops of the ground truth

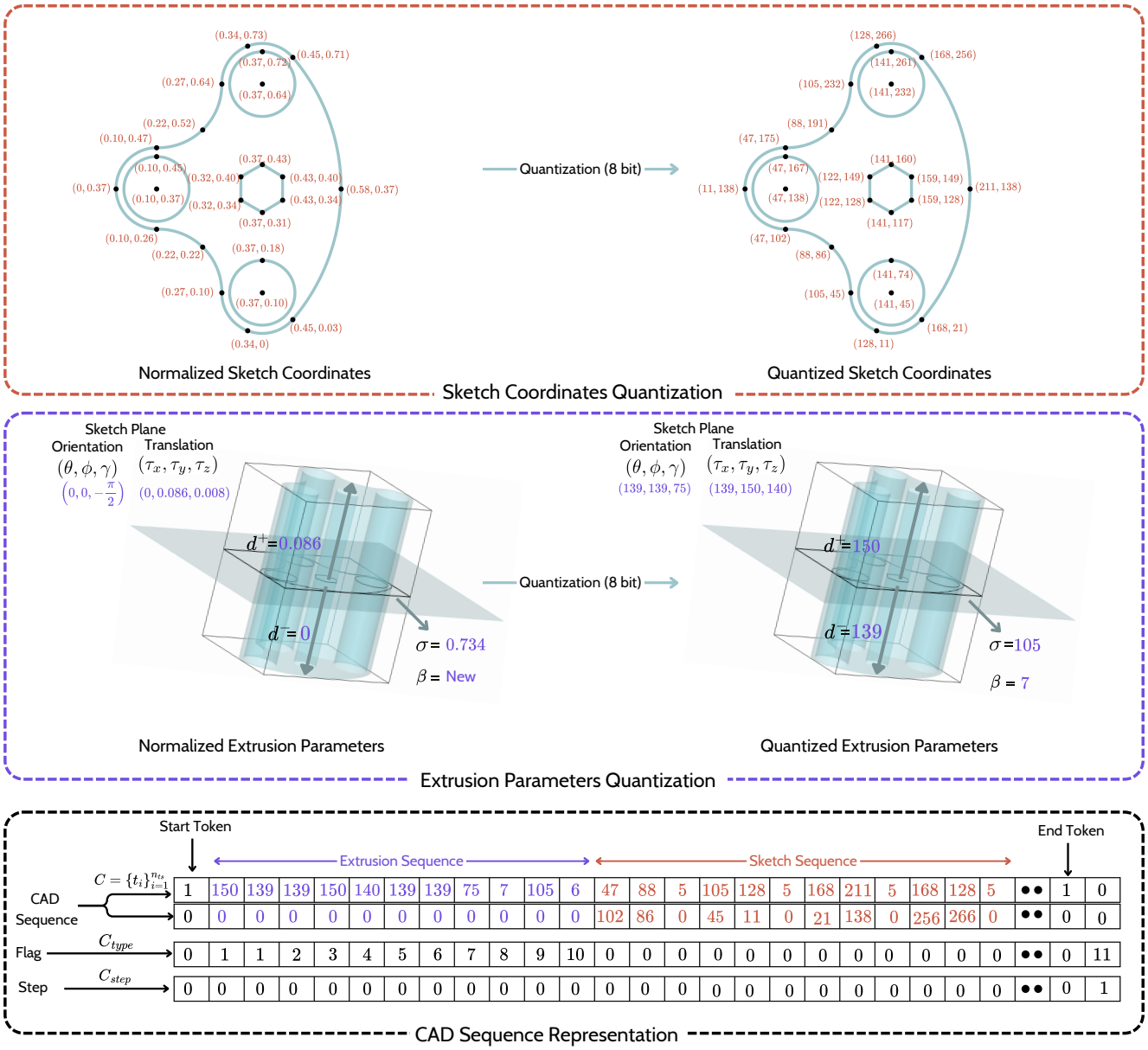

Figure 1. Example of a CAD Sequence Representation. The top and middle panels show the 8-bit quantization process of sketch and extrusion parameters respectively. The bottom panel depicts the construction of the sequence from the different tokens.

and predicted sketches of the same step are matched using the *match_entity_list* function described in Algorithm 2. The *match_entity_list* function employs a Hungarian matching [5] to establish the correspondences between two lists of loops. The cost associated with matching two loops is defined as the sum of the Euclidean distances between their respective bounding box bottom-left and top-right corners. A similar matching strategy is extended for the curves within the matched loops. Finally, the list of matched curve pairs from all the sketches is used to compute the curve type *F1* scores.

Parameter accuracy introduced in [10] is omitted in our work. This is because the CAD sequence representation in DeepCAD [10] differs significantly from ours, particularly in the curve parameterization. Attempting to transform predictions from one representation to another will propagate prediction errors, resulting in an unfair comparison.

For evaluating the CAD reconstruction (obtained by Opencascade [1]), the Chamfer Distance (CD) [3] is computed. This is achieved by uniformly sampling 8192 points from the predicted and ground truth reconstructed CAD models. To ensure scale-invariance in CD computation, the models are normalized within a unit bounding box. Note that the CD can only be computed if the predicted sequence leads to

Algorithm 1: *calculate_metrics*

Data: S_g, S_p
// List of ground-truth and predicted sketches.
Result: Recall, Primitve, F1 for curves.
n_gt ← *length*(S_g)
n_pred ← *length*(S_p)
n_max ← *max*(n_gt, n_pred)
// Over or under-prediction of sketches.
if n_gt ≠ n_pred **then**
  Append None to S_g or S_p until their lengths become n_max.
// List of ground truth and predicted curve types.
y_true ← []
y_pred ← []
for $i \leftarrow 1$ **to** n_max **do**
  // Match loops in the sketch.
  loop_pair ← *match_entity_list*($S_g[i]$.loopList, $S_p[i]$.loopList)
  // Match curves in the matched loop pairs.
  for (l_g, l_p) in loop_pair **do**
    // Get pairs of ground truth and predicted curves from matched loops.
    (c_g, c_p) ← *match_entity_list*(l_g.curveList, l_p.curveList)
    append *curve_type*(c_g) in y_true
    append *curve_type*(c_p) in y_pred
recall ← *multiclass_recall* (y_true, y_pred)
precision ← *multiclass_precision* (y_true, y_pred)
f1 ← *multiclass_f1* (y_true, y_pred)
return recall, precision, f1

a valid CAD model.

3. Additional Evaluation Details

In this section, more results from the different experiments are shown.

3.1. Model Parameters Comparison

Table 2 shows the number of parameters required for each of the networks presented in the main paper. CAD-SIGNet has the lowest number of parameters compared to other baselines.

Model	#Parameters
DeepCAD [10]+PointNet++ [7]	7.4M
SkexGen [11] + PointNet++ [7]	18.7M
HNC [12] + PointNet++ [7]	58.4M
CAD-SIGNet (Ours)	**6.1M**

Table 2. Total number of parameters of CAD-SIGNet compared to different baseline models.

Algorithm 2: *match_entity_list*

Data: e_g, e_p
// List of ground-truth and predicted entities of the same sketch index. Entity can be a loop or a curve.
Result: Matched Curve Pair
n_gt ← *length*(e_g)
n_pred ← *length*(e_p)
n_max ← *max*(n_gt, n_pred)
// Over or under-prediction of entities.
if n_gt ≠ n_pred **then**
  Append None to e_g or e_p until their lengths become n_max.
// Hungarian matching cost matrix.
cost ← []
for $i \leftarrow 1$ **to** n_gt **do**
  for $j \leftarrow 1$ **to** n_pred **do**
    if $e_g[i]$ and $e_p[i]$ is not None **then**
      cost[i][j]=*bbox_distance*($e_g[i], e_p[j]$)
    else
      cost[i][j]=∞
matched_entity_pair=*hungarian_matching*(cost)
return matched_entity_pair

3.2. More Details on Design History Recovery

Table 3 shows supplementary results for the baseline DeepCAD [10] and various versions of CAD-SIGNet as mentioned in the ablation study in Section 5 of the main paper. The precision and recall scores used to compute the *F1* scores are also included.

In the first and second row of Table 3, we provide the results of DeepCAD [10] with two distinct point cloud encoders: the first with PointNet++ [7], and the second using LFA [4] modules used in CAD-SIGNet. The transition to LFA [4] modules from PointNet++ [7] leads to a noticeable decline in recall, precision, and F1 scores for DeepCAD [10]. In CAD-SIGNet without the Layer-wise CA, the point embedding from the last LFA layer are used for cross-attention in the multi-modal transformer blocks. This modification leads to a drastic drop in performance. In contrast, our retained model, which leverages Layer-wise CA, outperforms both the baseline DeepCAD [10] and CAD-SIGNet without Layer-wise CA. Additionally, emphasizing the importance of SGA and Hybrid Sampling, we observe improvements in recall, precision, and F1 scores, especially for arcs and circles.

Figure 4 shows some examples of predicted sketch instances and the corresponding sketches compared to the ground truth ones. Sample 1 from this figure shows that a correct sketch instance prediction has led to the correct

Model	Line			Arc			Circle			Extrusion		
	Recall	Precision	F1	Recall	Precision	F1	Recall	Precision	F1	Recall	Precision	F1
DeepCAD [10] + PointNet++ [7]	69.86	72.40	68.37	12.53	15.21	12.89	59.61	61.95	58.82	81.74	94.87	86.88
DeepCAD [10] + LFA [4]	66.26	71.09	65.04	4.07	6.19	4.41	47.45	50.49	46.76	79.63	92.48	82.90
Ours w/o Hybrid Samp.	76.49	80.12	75.36	26.90	31.50	27.45	71.53	71.78	69.83	93.98	**95.55**	**92.97**
Ours w/o SGA	77.07	**82.13**	76.93	26.12	31.39	26.89	67.1	69.08	66.58	94.17	94.7	92.5
Ours w/o Layer-wise CA	56.63	71.07	56.99	0.73	1.37	0.800	18.34	26.07	19.97	84.81	92.96	84.53
CAD-SIGNet (Ours)	**77.76**	81.35	**77.31**	**27.67**	**33.01**	**28.65**	**72.07**	**72.22**	**70.36**	**94.26**	94.93	92.72

Table 3. Recall, Precision, and F1 scores for the baseline and ablated CAD-SIGNet for lines, arcs, circles, and extrusions. The results are on DeepCAD [10] dataset.

sketch prediction and hence the final predicted CAD model matches the ground truth. In samples 2 and 3, the initial sketch instance is correctly predicted but the corresponding sketches do not match the ground truth ones. However, in these examples, the network can still predict a CAD reconstruction similar in shape to the ground truth. Sample 4 shows that despite an incorrect sketch instance prediction, the final predicted CAD model has the same shape as the ground truth. Sample 5 shows an example in which the correct sketch instance was identified but the network was unable to predict the corresponding sketch sequence. Finally, sample 6 shows that the network sometimes fails to capture some details from the input point cloud.

Figure 5 and 6 showcase more visual results of the reconstructed CAD models from DeepCAD [10] and CAD-SIGNet on the DeepCAD [10] and CC3D [2] datasets, respectively. The top panel of both figures shows that CAD-SIGNet achieves better performance than DeepCAD [10], even for models containing higher curve counts. The bottom panel showcases invalid models of both DeepCAD [10] and CAD-SIGNet. We observe that the invalid outputs generated by CAD-SIGNet are due to syntax errors in the predictions such as a line or an arc with the same start and end point.

Figure 7 displays several qualitative results for the ablated versions of CAD-SIGNet. From those samples, it can be observed that the retained model consistently outperforms its ablated versions (i.e. discarding layer-wise CA, SGA, and hybrid sampling).

3.3. Auto-Completion from User Input Details

As outlined in Section 5.2 of the main paper, Skexgen [11] and HNC [12] are used as baselines for the conditional auto-completion task. As both were not originally designed for conditional auto-completion from point clouds, their training strategy had to be adapted. For SkexGen [11], PointNet++ [7] is used to predict 10 pretrained codebooks from the input point cloud. In the case of HNC [12], the controllable CAD model generation network is retrained with modifications. In addition to the initial sketch and extrusion sequence, we incorporate the full point cloud as input. PointNet++ [7] is used to learn a latent embedding from the input point cloud. The resulting point cloud embedding was then appended to the CAD sequence embedding and passed to the code-tree generator to predict the codes. The generated codes along with the CAD sequence and point cloud embedding are fed to the sketch and extrusion decoder to generate the CAD sequence. Teacher forcing [9] strategy is used during training along with cross-entropy loss. The input CAD sequence is masked from the ground truth CAD sequence during the loss calculation.

During conditional auto-completion inference of CAD-SIGNet, HNC [12], SkexGen [11], the initial ground truth sketch and extrusion sequence along with the input point cloud are provided. The subsequent CAD sequence is autoregressively generated until the end token is predicted.

It is important to highlight that the CAD sequence representation in HNC [12] and SkexGen [11] uses 6-bit quantization while as our representation uses 8-bit quantization. The different quantization schemes lead to different user inputs. Hence, the results reported in Table 4 of the main paper were considering the ratio of CD distances with respect to the user input, rather than CD.

3.4. Performance on Complex Models

The complexity of models that CAD-SIGNet can handle depends on the training dataset. We consider two defining attributes for complexity: the number of extrusions and the total number of curves in the sketches within a CAD model. The DeepCAD dataset [10] which offers the required annotations for CAD-SIGNet includes models with at most 10 extrusions and 44 curves. Figure 2 shows the variation of the median CD w.r.t. the number of extrusions and the total number of curves per CAD model. Unlike DeepCAD [10], the complexity of the input model has minor impact on the performance of CAD-SIGNet.

3.5. Impact of Point cloud Quality

The cross-dataset evaluation on the CC3D dataset [2] in Section 5.3 aims at assessing the ability of CAD-SIGNet to reconstruct CAD models from point clouds that contain realistic scanning artifacts such as noise and self-occlusion. Furthermore, Figure 3 shows some qualitative results from DeepCAD [10] dataset when a part of the input point cloud is missing. The top rows show the input point cloud with the missing points highlighted in red while the middle and bottom rows display the corresponding prediction from CAD-

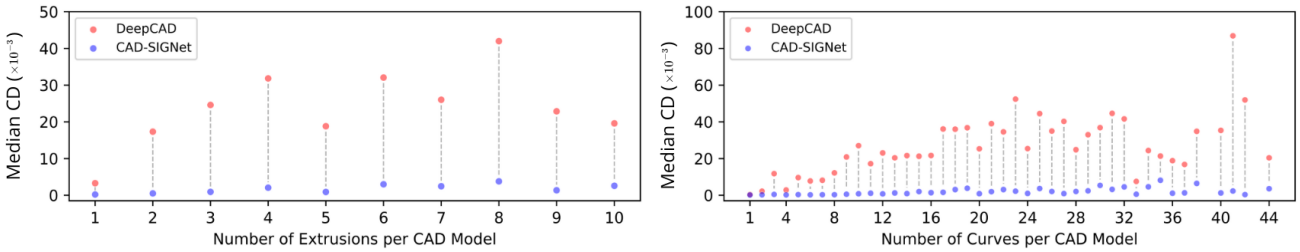

Figure 2. Comparison of median CD for reconstructed CAD models by CAD-SIGNet and DeepCAD [10] w.r.t. to the number of extrusions (**left**) and total number of curves (**right**) per CAD model.

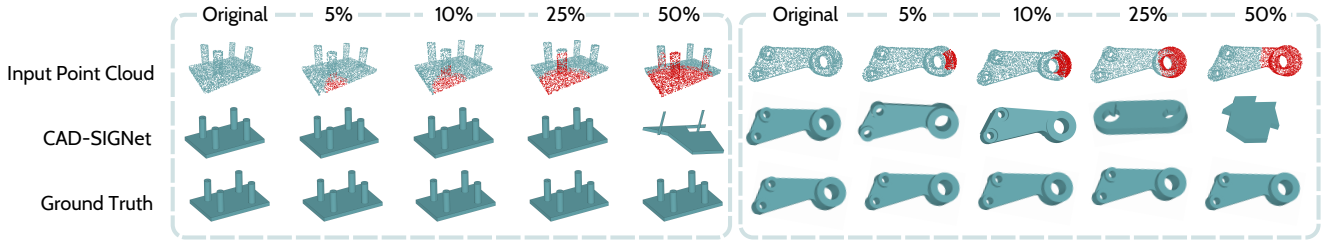

Figure 3. Performance of CAD-SIGNet under varying degrees of input point cloud occlusion. The percentage of missing points is indicated above each result.

SIGNet and ground truth. We can observe that when a small portion of the point cloud is missing, CAD-SIGNet manages to recover a plausible solution. However, if a large portion of the input point cloud is missing, then CAD-SIGNet fails to capture the overall structure of the CAD model.

3.6. Comparison with Point2Cyl

In this section, we compare CAD-SIGNet with Point2Cyl [8], another method addressing the 3D reverse engineering problem from point clouds. Figure 8 shows some qualitative results between Point2Cyl [8] and CAD-SIGNet in terms of the output sketches and 3D reconstructions given an input point cloud. Notably, the shapes predicted by Point2Cyl [8] closely resemble the ground truth shapes. However, there are some major differences between our work and Point2Cyl [8]. Firstly, the sketches in Point2Cyl [8] are predicted as signed distance functions, and a marching square algorithm is applied to deduce the sketch. Such a strategy leads to a non-parametric form of the sketches. Secondly, the final model is a 3D mesh. This implies that the output model cannot be directly edited using CAD software, hence limiting the practical applications of this method. Finally, reconstructing the final model requires manual choice for the boolean operations between the different extrusion cylinders.

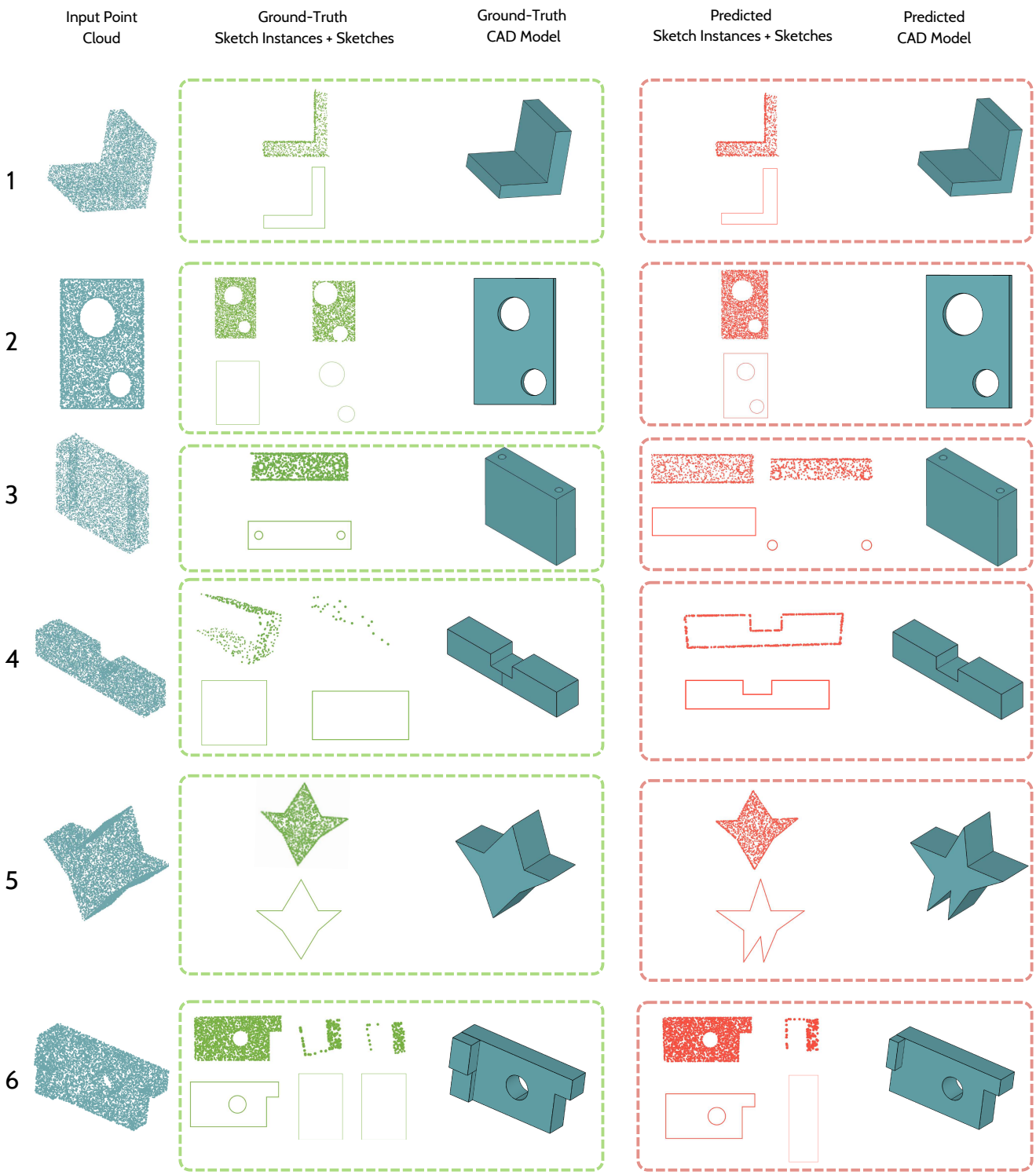

Figure 4. **Sketch Instances**. Ground truth (green) and predicted (red) sketch instances along with their corresponding sketches.

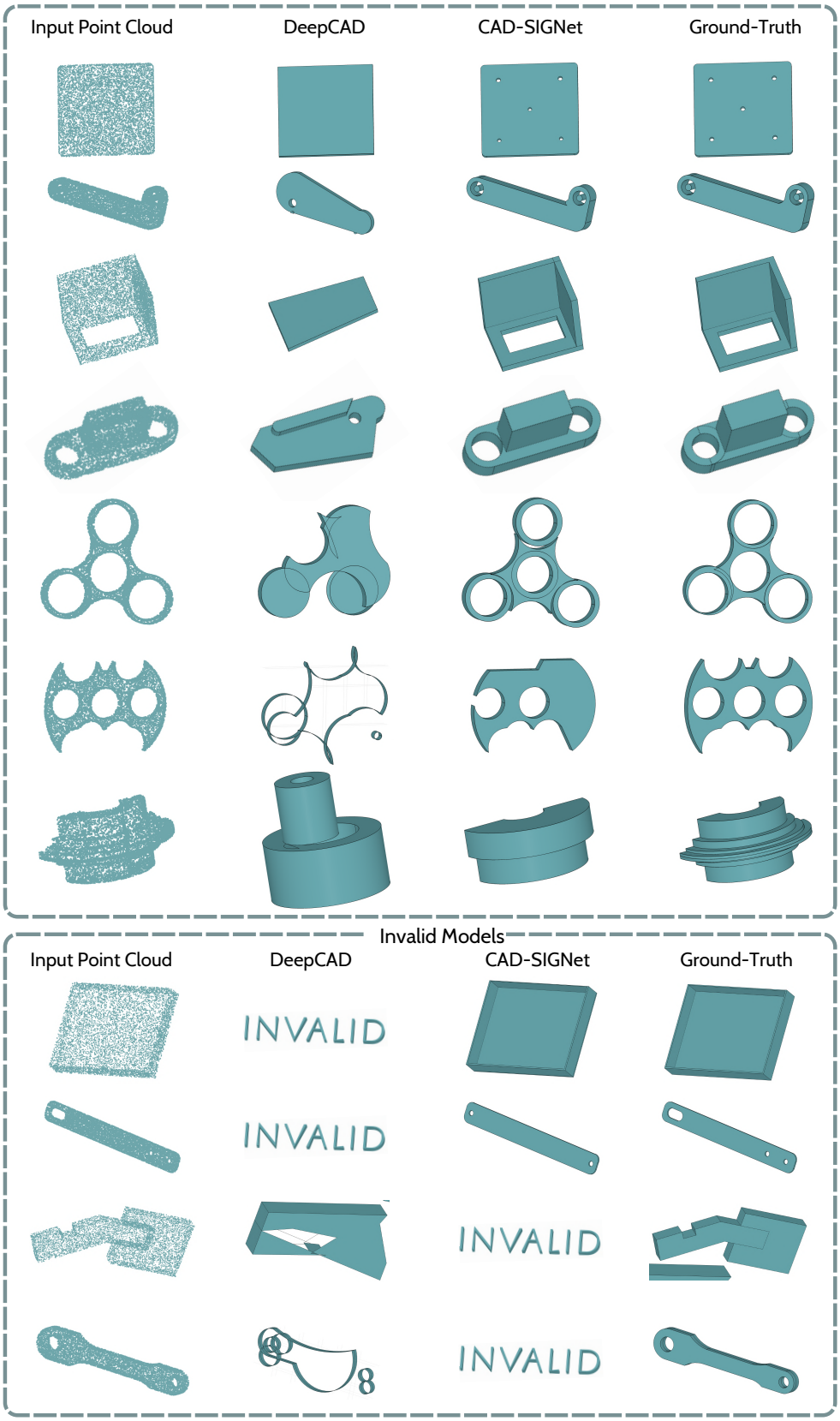

Figure 5. More qualitative results of the reconstructed CAD models from the input point clouds on the DeepCAD [10] dataset. The top panel shows examples of varying complexity. The bottom panel showcases invalid models, observed in both DeepCAD [10] and CAD-SIGNet.

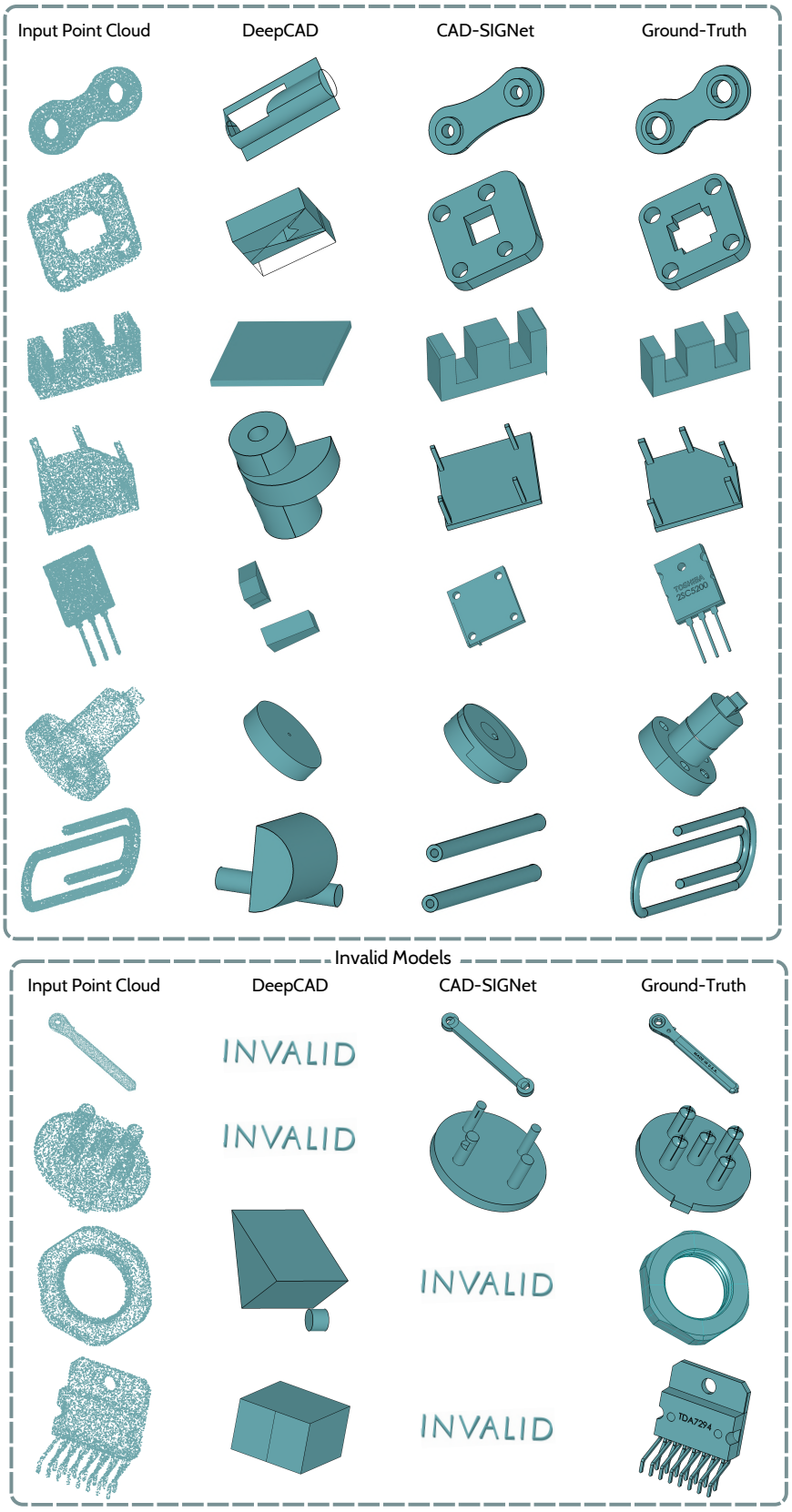

Figure 6. More qualitative results of the reconstructed CAD models from the CAD sequences predicted from input scans on CC3D [2] dataset. Both CAD-SIGNet and DeepCAD [10] are trained on the DeepCAD [10] dataset. The top panel shows examples of varying complexity. The bottom panel showcases invalid models, observed in both DeepCAD [10] and CAD-SIGNet.

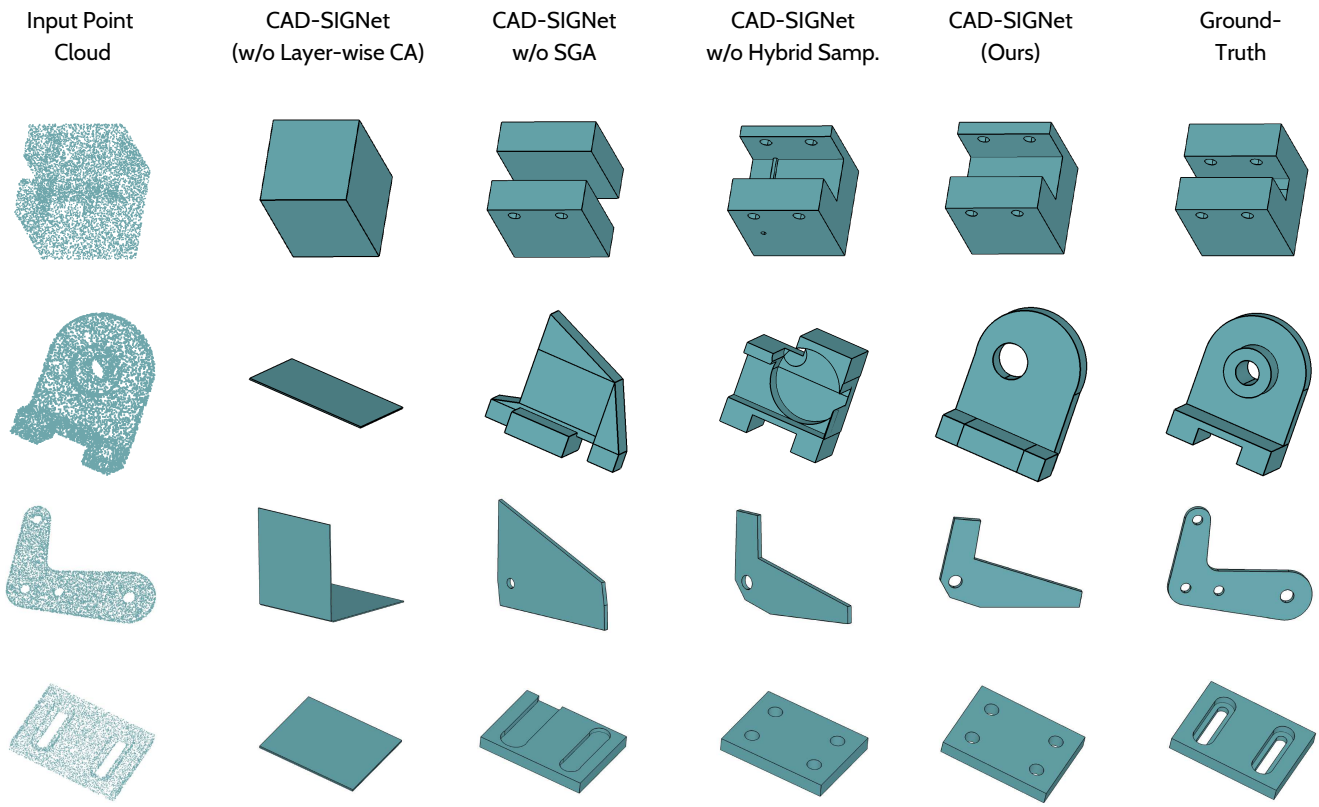

Figure 7. Reconstructed CAD models from predicted CAD sequences on DeepCAD [10] dataset for ablated versions of CAD-SIGNet.

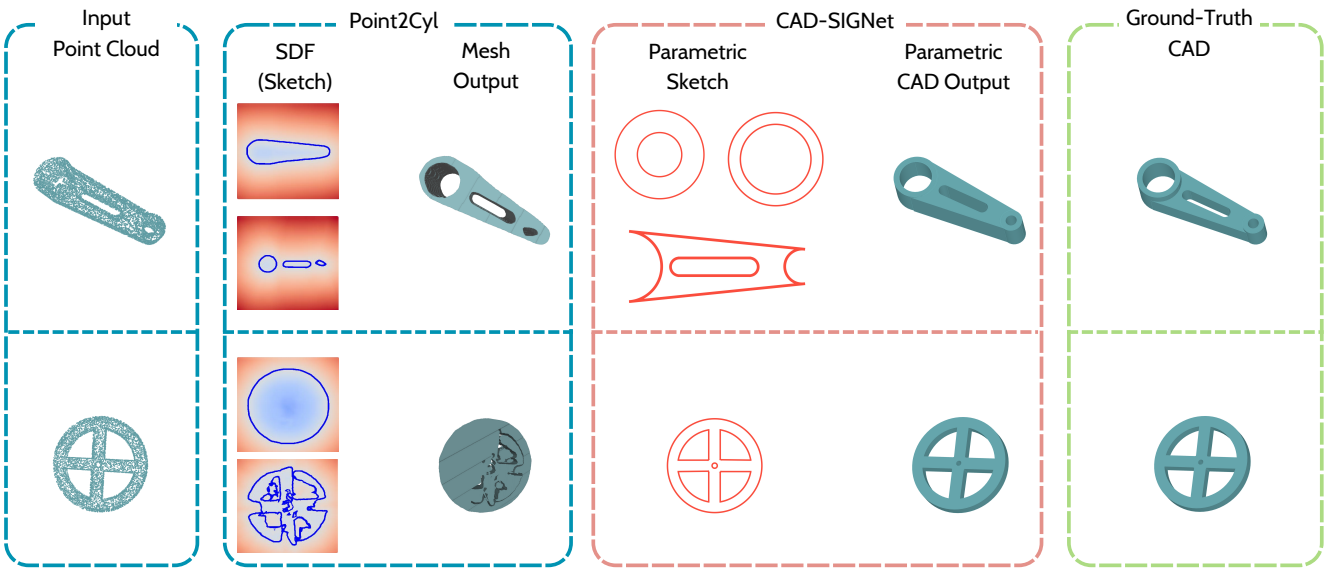

Figure 8. Qualitative results from Point2Cyl [8] and CAD-SIGNet. The output sketches and 3D reconstructions are provided here. The output from Poin2Cyl is not parametric while CAD-SIGNet outputs parametric CAD model.

References

[1] Open cascade. https://dev.opencascade.org/. 2

[2] Kseniya Cherenkova, Djamila Aouada, and Gleb Gusev. Pvdeconv: Point-voxel deconvolution for autoencoding cad construction in 3d. pages 2741–2745, 2020. 4, 8

[3] H. Fan, H. Su, and L. Guibas. A point set generation network for 3d object reconstruction from a single image. In 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 2463–2471, Los Alamitos, CA, USA, 2017. IEEE Computer Society. 2

[4] Qingyong Hu, Bo Yang, Linhai Xie, Stefano Rosa, Yulan Guo, Zhihua Wang, Niki Trigoni, and Andrew Markham. Learning semantic segmentation of large-scale point clouds with random sampling. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2021. 1, 3, 4

[5] Harold W. Kuhn. The Hungarian Method for the Assignment Problem. Naval Research Logistics Quarterly, 2(1–2):83–97, 1955. 2

[6] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019. OpenReview.net, 2019. 1

[7] Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In Proceedings of the 31st International Conference on Neural Information Processing Systems, page 5105–5114, Red Hook, NY, USA, 2017. Curran Associates Inc. 3, 4

[8] Mikaela Angelina Uy, Yen-Yu Chang, Minhyuk Sung, Purvi Goel, Joseph G Lambourne, Tolga Birdal, and Leonidas J Guibas. Point2cyl: Reverse engineering 3d objects from point clouds to extrusion cylinders. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 11850–11860, 2022. 5, 9

[9] Ronald J. Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. pages 270–280, 1989. 4

[10] Rundi Wu, Chang Xiao, and Changxi Zheng. Deepcad: A deep generative network for computer-aided design models. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), pages 6772–6782, 2021. 1, 2, 3, 4, 5, 7, 8, 9

[11] Xiang Xu, Karl DD Willis, Joseph G Lambourne, Chin-Yi Cheng, Pradeep Kumar Jayaraman, and Yasutaka Furukawa. Skexgen: Autoregressive generation of cad construction sequences with disentangled codebooks. In International Conference on Machine Learning (ICML), pages 24698–24724, 2022. 1, 3, 4

[12] Xiang Xu, Pradeep Kumar Jayaraman, Joseph G. Lambourne, Karl D.D. Willis, and Yasutaka Furukawa. Hierarchical neural coding for controllable cad model generation. In Proceedings of the 40th International Conference on Machine Learning. JMLR.org, 2023. 3, 4