Text2CAD: Generating Sequential CAD Designs from Beginner-to-Expert Level Text Prompts

Mohammad Sadil Khan[*†1,2,3] **Sankalp Sinha**[*1,2,3] **Talha Uddin Sheikh**[1,2,3]

Didier Stricker[1,2,3] **Sk Aziz Ali**[1,4] **Muhammad Zeshan Afzal**[1,2,3]

[1]DFKI [2]RPTU Kaiserslautern-Landau [3]MindGarage [4]BITS Pilani, Hyderabad

Abstract

Prototyping complex computer-aided design (CAD) models in modern softwares can be very time-consuming. This is due to the lack of intelligent systems that can quickly generate simpler intermediate parts. We propose Text2CAD, the first AI framework for generating text-to-parametric CAD models using designer-friendly instructions for all skill levels. Furthermore, we introduce a data annotation pipeline for generating text prompts based on natural language instructions for the DeepCAD dataset using Mistral and LLaVA-NeXT. The dataset contains $\sim$ 170K models and $\sim$ 660K text annotations, from abstract CAD descriptions (e.g., *generate two concentric cylinders*) to detailed specifications (e.g., *draw two circles with center* (x, y) *and radius* r_1, r_2, *and extrude along the normal by* d...). Within the Text2CAD framework, we propose an end-to-end transformer-based auto-regressive network to generate parametric CAD models from input texts. We evaluate the performance of our model through a mixture of metrics, including visual quality, parametric precision, and geometrical accuracy. Our proposed framework shows great potential in AI-aided design applications. Project page is available at https://sadilkhan.github.io/text2cad-project/.

1 Introduction

Computer-Aided Design (CAD) plays a crucial role in industrial design and additive manufacturing (AM), revolutionizing the way products are prototyped [7]. This type of prototyping requires feature-based part modeling [7], precision measurements [40], and creative part editing [60] at different design stages [40, 60]. While CAD software saves the final model as a boundary representation (B-Rep) [22], the design process often involves a chain of 2D sketches (e.g., circles, lines, splines) and 3D operations (e.g., extrusion, loft, fillet) [56, 58, 59, 19]. This representation allows the designers to control the design history and iteratively refine the final models.

Despite their capabilities, modern CAD tools lack the AI-assisted design integration [37]. In Figure 1, we illustrate how an intelligent system capable of generating parametric CAD models from textual descriptions can be utilized to assemble a complex 3D model. Although tools like FreeCAD [1], SolidWorks [46], and Para-Solid [45] offer 3D CAD models from catalogs like McMaster-Carr [2] for the reuse

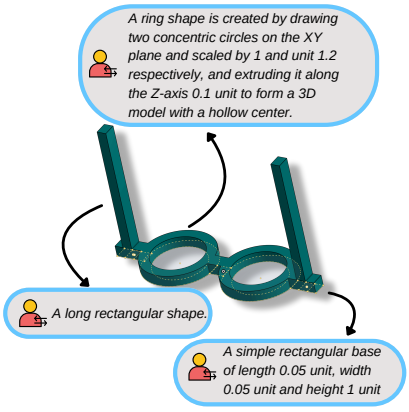


Figure 1: *Designers can efficiently generate parametric CAD models from text prompts. The prompts can vary from abstract shape descriptions to detailed parametric instructions.*

[*]Equal Contributions.

[†]Corresponding author (mohammad.khan@dfki.de)

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

https://doi.org/10.52202/079017-0242

of existing CAD models, no such system currently exists that can generate parametric CAD models from textual design descriptions. One primary challenge for developing such a system is defining suitable textual descriptions for parametric CAD generation, making it difficult to create deep learning methods that accurately convert these descriptions into precise CAD models.

To address this gap, in this paper we propose Text2CAD as the first AI framework for generating parametric CAD models represented by construction sequences (*i.e.,* parameters for 2D sketches and extrusions) from design-related text prompts. We faced two primary challenges in fulfilling this goal: (1) the unavailability of the dataset and (2) a network to map the texts into CAD construction sequences. Towards this end, we introduce a data annotation pipeline to generate a dataset containing textual descriptions of the CAD models in DeepCAD [56] dataset. We leverage the open-source Large Language Models (LLMs) [16] and Vision Language Models [27, 26] for this task. Our annotated text prompts are multi-level in nature ranging from highly abstract (*e.g., a long rectangular shape*, *a thin S-shaped object*) to more specific with detailed parametric descriptions (*e.g., first draw a rectangle from* (x_1, y_1) *and* (x_2, y_2)*.... then extrude the sketch along z-axis..*). These prompts are designed for users of all skill levels and can contain arithmetic logic and numerical expressions as part of the design details. Within this framework, we introduce Text2CAD Transformer [49], a conditional deep-generative network for generating CAD construction language [†] from text prompt in an auto-regressive fashion.

Currently, there are works on text-to-3D generation [25, 34, 51, 32, 12] that have shown significant advancements in creating 3D scenes and shapes from textual descriptions. But existing text-to-3D methods are not applicable for generating CAD models from text descriptions as the final output of these models is neither parametric nor human-editable in nature. Very recently web API from *zoo developers* [4] has introduced CAD generation app using text prompt from users and programmable scripting language (as KittiCADLanguage[†]) for designers to edit and modify. However, the generated CAD models are obtained in the form of solid-body, and not decomposed to its intermediate *sketch-and-extrusion* steps as proposed in our Text2CAD. On the other hand, Given raw numerical data of any parametric CAD model, current state-of-the-art large language models (LLMs), such as pre-trained Mistral-50b [16] or GPT-4 [35] and open source Llama [47] may only derive procedural scripting codes for other APIs, such as FreeCAD [38] or OpenSCAD [30], to generate a model. However, in contrast to our Text2CAD, such LLM augmented CAD generation approach will not be designer-friendly, not suitable for beginner-level designers, will not automate the development process in easy ways, and will restrict the re-usability of the scripts in case of complex shapes. Alternatively, Using state-of-the-art vision language models, such as LLaVa [27, 26], GPT-4V [61], as an alternative for deducing CAD construction sequences performs poorly because of two main reasons – (1) no training datasets are available that provide natural language-based design instructions as annotations for raw CAD construction sequences and (2) most VLMs are trained on categorical description/caption datasets of 3D objects (*e.g.,* LLaVA-NeXT [26] predicts 'two concentric hollow cylinders 'as *toilet paper*). We remove the above limitations in our Text2CAD by creating new large-scale annotations for DeepCAD [56] dataset using responses from LLMs and VLMs to train our multi-modal model. Our contributions can be summarized as follows:

- We propose Text2CAD as the first AI framework for generating parametric 3D CAD models using textual descriptions.
- We introduce a data annotation pipeline that leverages both LLMs and VLMs to generate a dataset that contains text prompts with varying level of complexities and parametric details.
- We propose an end-to-end transformer-based autoregressive architecture for generating CAD design history from input text prompts.
- Our experimental analysis demonstrates superior performance over the two-stage baseline method adapted for the task at hand.

The rest of the sections are organized as follows: Section 2 reviews the related work in CAD domains. Section 3 outlines our data annotation pipeline. Section 4 details our proposed Text2CAD transformer architecture. Section 5 presents our experimental results. Section 6 discusses the limitations of our current framework, and Section 7 concludes the paper.

[†]In this paper, the phrases 'CAD construction language ', 'CAD design history 'and 'CAD construction sequence 'are used interchangeably.

[†]https://github.com/KittyCAD/modeling-app/tree/main?tab=readme-ov-file

2 Related Work

Datasets and Generative models for CAD: Current datasets and generative models for CAD are limited and often not suited for developing knowledge-based CAD applications. Some datasets focus solely on 2D sketch design [43, 10, 44], and other popular datasets like ABC [20], Fusion360 Gallery [55], Thingi10K [62], and CC3D [6, 9] provide 3D meshes, BRep (boundary representation), and other geometry or topology related annotations that are suitable for 3D modeling. DeepCAD [56] dataset, a subset of ABC, and Fusion360 [55] provide CAD construction sequences in the form of *sketch and extrusion* to deduce design history. However, CAD models may consist of numerous other types of operations beside *extrusion*, and such construction sequences with other CAD operations are not available in the current datasets. Finally, there is no dataset available that provides textual design descriptions as annotations to create a conversational AI system for CAD modeling.

Current supervised learning methods that fall under *sequence-to-sequence Sketch/CAD language modeling* [56, 58, 19, 10] filters out unnecessary metadata from lengthy raw design files and represent them as desired sequence of input/output tokens. For instance, Ganin et al. [10] represents design files as messages in Protocol Buffer [48] format. Hierarchical Neural Coding (HNC) method [58] represents the desired design sequence in tree structure of 2D sketch loops, 2D bounding boxes over all loops as profile, and 3D bounding boxes over all profiles as solid. CAD-SIGNet [19] represents CAD construction language as a sequence composed of 2D sketch and extrusion parameters. In Text2CAD method, we map the raw design history obtained from DeepCAD metadata into textual descriptions.

CAD Construction Language using Transformers: Transformer-based [49] network architecture is the preferred choice for many deep learning-based applications related to CAD modeling [56], 3D scan-to-CAD reverse engineering [19, 23], representation learning [18] and others [39]. CAD as a language [10] describe how 2D sketches can be transformed into *design language* by sequencing tokens of 2D parametric curves as message passing units. Mixture of Transformer [49] and Pointer Networks [50] decode the sketch parameters in auto-regressive fashion.

Formalizing constrained 2D sketches, *i.e.,* collection of curves (*e.g., line, arc, circle and splines*) with dimensional and geometric constraints (*e.g., co-incidence, perpendicular, co-linearity*), as a language for CAD modeling has been studied over last few years [36, 10, 33, 54, 24]. However, the first proposal of developing a CAD language interface was suggested decades ago in [41]. Among the recent works in this direction, SketchGen [36] represents 2D sketches as a sequence of the tokens for curves and constraints. The decoder-only transformer model in [36] predicts optimal sketches through nucleus sampling [13] of token embedding vectors, focusing on replicating drawing processes of CAD designers. Polygen [33] method also employs Transformer model [49] to generate detailed 3D polygonal meshes by learning joint distribution on vertices and faces of a CAD. As an extension of [33], TurtleGen [54] also propose decoder-only transformer model to learn joint distribution of vertices and edges together that form sketches and represented as graphs in CAD models.

3D CAD modeling steps as a language is not directly formulated by any state-of-the-art multi-modal CAD learning methods [56, 29, 19, 59, 9, 31, 58, 24]. Khan et al. [19] propose a novel auto-regressive generation of *sketch-and-extrusion* parameters directly from 3D point clouds as input whereas DeepCAD [56], SkexGen [59], HNC [58] and MultiCAD [29] adopts a two-stage strategy to generate the output. MultiCAD [29] adopt multi-modal contrastive learning to associate geometry features with features of CAD construction sequences whereas CAD-SIGNet [19] requires an extra step as user feedback to vote for one of the many generated sketches at current step to predict the next one. Unlike previous approaches, our proposed Text2CAD transformer is the first auto-regressive network that generates CAD construction sequences directly from textual descriptions.

3 Text2CAD Data Annotation

The diagram in Figure 2 outlines the process of generating textual annotations for DeepCAD dataset [56] using Large Language Models (LLMs) [17, 35, 47] and Vision Language Models (VLMs) [27, 26]. These annotations describe the corresponding CAD construction workflow in human interpretable format. To enrich the DeepCAD [56] dataset with textual annotations, we implement a two-stage description generation pipeline using the capabilities of both LLMs and VLMs. The two stages are - (1) generating abstract shape descriptions using VLM, and (2) extracting multi-level textual instructions from LLM based on the shape descriptions and design details provided in the dataset. An example text prompt for the CAD model shown in *top-left* of the Figure 2: '*The CAD*

https://doi.org/10.52202/079017-0242

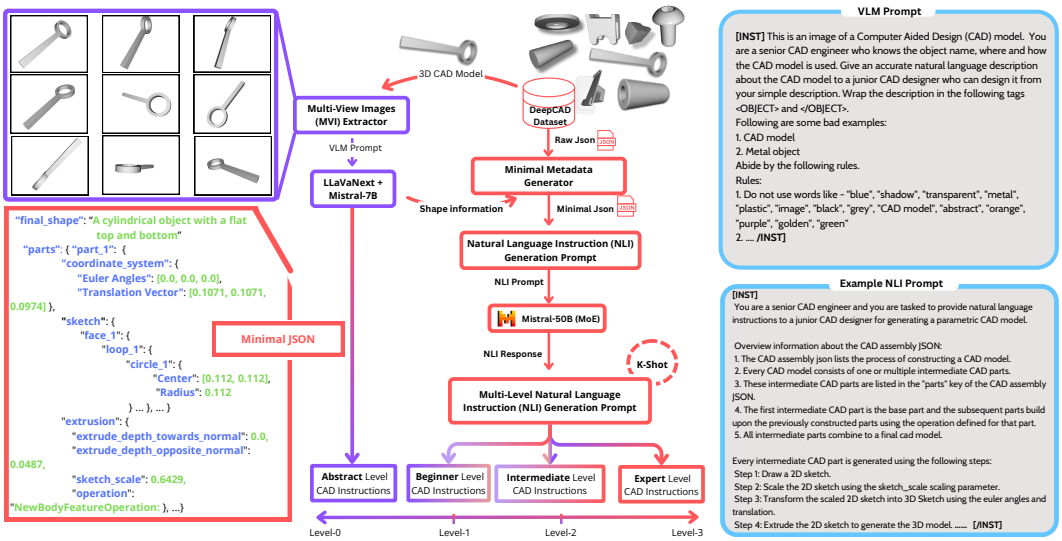


Figure 2: **Text2CAD Data Annotation Pipeline:** Our data annotation pipeline generates multi-level text prompts describing the construction workflow of a CAD model with varying complexities. We use a two-stage method - (Stage 1) Shape description generation using VLM (Stage 2) Multi-Level textual annotation generation using LLM.

model consists of a cylindrical object with a flat top and bottom connected by a curved surface and slightly tapered towards the bottom. This object is created by first setting up a coordinate system, then sketching two concentric circles and drawing a closed loop with lines and an arc on a shared plane. The sketch is then extruded along the normal direction to form a solid body. The resulting part has a height of approximately 0.0487 units '. In this example, the phrase in the violet color is generated by a VLM. An LLM uses this description along with the CAD construction information to generate the prompt.

Shape Descriptions using VLM: The initial step of our annotation generation pipeline involves generating abstract object-level descriptions of the CAD models using LLaVA-NeXT [26] model. The objective in this step is to accurately capture the structural descriptions of the 3D shape, such as *"a ring-like structure"*, *"a cylinder"*, or *"a hexagon with a cylinder on top"*. We generate shape descriptions for both the final CAD model and its intermediate parts. We first produce multi-view images from predetermined camera angles for each individual parts and the final CAD model. These images are then utilized in a predefined prompt (refer to the *top-right* of Figure 2) for the LLaVA-NeXT [26] model to generate simplified shape descriptions of all individual parts as well as the complete final shape.

Multi-level Design Instructions using LLM: In this stage, multiple textual annotations corresponding to different design details of a CAD model are generated using Mixtral-50B [17] through a series of steps (refer to the *middle-column* in Figure 2). The DeepCAD [56] dataset contains CAD construction sequences in JSON format. We first preprocess the raw CAD construction sequences using a 'Minimal Metadata Generator 'which replaces random, meaningless keys with more meaningful terms (*e.g.,*"part_1", "loop_1"). This step aims to reduce the hallucinations [14] by Mixtral-50B [17]. The minimal metadata is further augmented with the shape descriptions for each parts and the final model generated by the VLM. The output of this process is a condensed representation of the shapes and their relational attributes within the CAD design (see *bottom-left* in Figure 2). With the minimal metadata at hand, we then craft a prompt (refer to the *bottom-right* in Figure 2) to generate detailed natural language instructions (NLI) ensuring a minimal loss of information from the minimal metadata. Afterward, the NLI responses are refined by LLM using a k-shot [5] "Multi-Level Natural Language Instruction Generation Prompt" to generate multi-level instructions of different specificity and details. We categorize these levels as:

- **Abstract level (L0):** Abstract Shape Descriptions of the final CAD model extracted using VLM in the first stage.

- **Beginner level (L1):** Simplified Description - Aimed at laypersons or preliminary design stages, this level provides a simplified account of the design steps, eschewing complex measurements and jargon.
- **Intermediate level (L2):** Generalized Geometric Description - This level abstracts some of the details, providing a generalized description that balances comprehensibility with technical accuracy.
- **Expert level (L3):** Detailed Geometric Description with Relative Values - Here, the instructions include precise geometric descriptions and relative measurements, catering to users who require an in-depth understanding or are performing the CAD modeling task.

Our annotations consist of the generated multi-level instructions at the final stage. We generate these annotations over the course of 10 days. It's worth noting that one can directly generate the multi-level instructions from the minimal metadata without creating the detailed natural language instructions in the second stage. We observe that this strategy increases the LLM's tendency for hallucinations [14] and it generates more inaccurate multi-level instructions. Instead our method follows *chain-of-thought* prompting strategy as outlined in [52] which greatly reduces such bottleneck. More details on our annotation pipeline are provided in Section 10 and 11 of the supplementary material.

4 Text2CAD Transformer

The Text2CAD transformer architecture, as shown in Figure 3, is designed to transform natural language descriptions into 3D CAD models by deducing all its intermediate design steps autoregressively. Given an input text prompt $T \in \mathbb{R}^{N_p}$, where N_p is the number of words in the text, our model learns the probability distribution, $P(\mathbf{C}|T)$ defined as

$$P(\mathbf{C}|T) = \prod_{t=1}^{N_c} P(c_t|c_{1:t-1}, T; \theta) \tag{1}$$

where $\mathbf{C}$ is the output CAD sequence, N_c is the number of tokens in $\mathbf{C}$ and θ is the learnable model parameter. We represent $\mathbf{C}$ as a sequence of sketch and extrusion tokens as proposed in [19]. Each token $c_t \in \mathbf{C}$ is a 2D token that either denotes a (1) 2D-coordinate of the primitives in sketch, (2) one of the extrusion parameters (euler angles/translation vector/extrusion distances/boolean operation/sketch scale) or (3) one of the end tokens (curve/loop/face/sketch/extrusion/start sequence/end sequence). Following [56, 19]. We quantize the 2D coordinates as well as the continuous extrusion parameters in 8 bits resulting in 256 class labels for each token. An example CAD sequence representation is provided in Figure 3 (in blue table). For more details, please refer to the supplementary section 9.

Now we elaborate on the various components of the architecture, detailing the processes involved in converting text to CAD representations. Let the input text prompt at timestep $t-1$ be $T \in \mathbb{R}^{N_p}$ and the input CAD subsequence $\mathbf{C}_{1:t-1} \in \mathbb{R}^{N_{t-1} \times 2}$.

Pretrained Bert Encoder: The initial step in the Text2CAD network involves encoding the textual description provided by the user. This description can range from highly abstract, beginner-friendly instructions to detailed, expert-level commands. To handle this diversity, we used a pre-trained BERT (Bidirectional Encoder Representations from Transformers) [8] model, denoted $\text{BERT}_{\text{pre-trained}}$. The input text $T \in \mathbb{R}^{N_t}$ is tokenized and passed through the BERT [8] model to generate contextual embedding:

$$\mathbf{T} = \text{BERT}_{\text{pre-trained}}(T) \tag{2}$$

Here, $\mathbf{T} \in \mathbb{R}^{N_p \times d_p}$ represents the sequence of token embedding vectors that capture the semantic meaning of the input text, where N_p is the number of tokens and d_p is the dimension of the embedding.

Adaptive Layer. An adaptive layer consisting of 1 transformer encoder layer, refines the output $\mathbf{T}$ of the BERT [8] encoder to better suit the CAD domain aligning with the specific vocabulary and structural requirements of CAD instructions. The adaptive layer outputs the embedding $\mathbf{T}_{\text{adapt}} \in \mathbb{R}^{N_p \times d_p}$ using

$$\mathbf{T}_{\text{adapt}} = \text{AdaptiveLayer}(\mathbf{T}) \tag{3}$$

CAD Sequence Embedder: Each token in the input CAD subsequence $\mathbf{C}_{1:t-1}$ is initially represented as a one-hot vector with a dimension of 256, resulting in a one-hot representation, $\mathbf{C}^o_{1:t-1} \in \mathbb{R}^{N_{t-1} \times 2 \times 256}$. For the sake of simplicity, we represent $\mathbf{C}^o_{1:t-1} = [\mathbf{C}^{ox}_{1:t-1}; \mathbf{C}^{oy}_{1:t-1}]$,

https://doi.org/10.52202/079017-0242

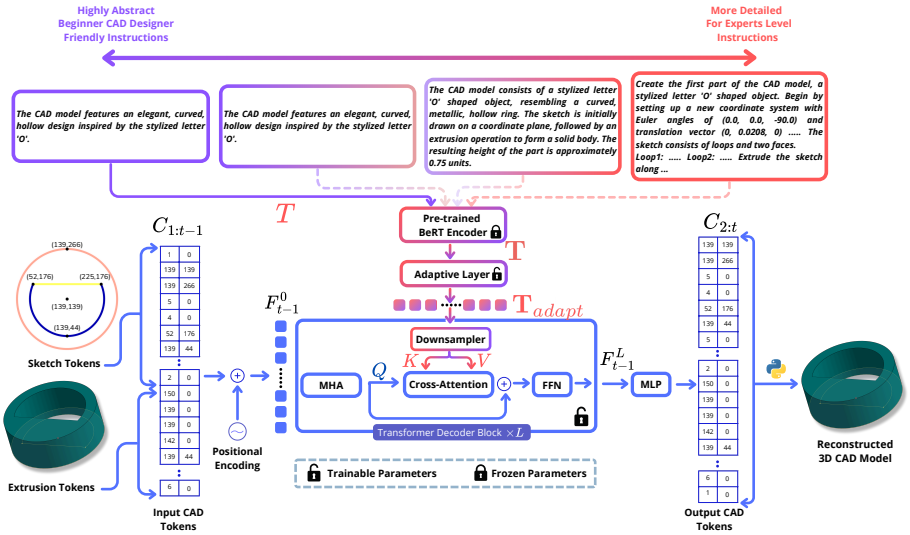


Figure 3: **Network architecture**: Text2CAD Transformer takes as input a text prompt T and a CAD subsequence $\mathbf{C}_{1:t-1}$ of length $t-1$. The text embedding $\mathbf{T}_{adapt}$ is extracted from T using a pretrained BeRT Encoder ([8]) followed by a trainable Adaptive layer. The resulting embedding $\mathbf{T}_{adapt}$ and the CAD sequence embedding $\mathbf{F}^0_{t-1}$ is passed through $\mathbf{L}$ decoder blocks to generate the full CAD sequence in auto-regressive way.

where $\mathbf{C}^{ox}_{1:t-1}, \mathbf{C}^{oy}_{1:t-1} \in \mathbb{R}^{N_{t-1} \times 256}$. The initial CAD sequence embedding $\mathbf{F}^0_{t-1} \in \mathbb{R}^{N_{t-1} \times d}$ is obtained using Eq. 4

$$\mathbf{F}^0_{t-1} = \mathbf{C}^{ox}_{1:t-1}\mathbf{W}^x_{t-1} + \mathbf{C}^{oy}_{1:t-1}\mathbf{W}^y_{t-1} + \mathbf{P} \tag{4}$$

, where $\mathbf{W}^x_{t-1}, \mathbf{W}^y_{t-1} \in \mathbb{R}^{N_{t-1} \times d}$ are learnable weights and $\mathbf{P} \in \mathbb{R}^{N_{t-1} \times d}$ is the positional encoding.

Layer-wise Cross Attention. We use a standard transformer decoder [49] with layer-wise cross-attention mechanism between the CAD and the text embedding within the decoder blocks. The layerwise cross-attention mechanism facilitates the integration of contextual text features with the CAD embedding, allowing the model to focus on relevant parts of the text during CAD construction. Each decoder block l takes as input CAD embedding $\mathbf{F}^{l-1}_{t-1}$ and text embedding $\mathbf{T}_{\text{adapt}}$, where $\mathbf{F}^{l-1}_{t-1}$ is the output of the previous decoder block (for the first decoder block, the input CAD embedding is $\mathbf{F}^0_{t-1}$). At first, the CAD embedding $\mathbf{F}^l_{t-1} \in \mathbb{R}^{N_{t-1} \times d}$ is generated from $\mathbf{F}^{l-1}_{t-1}$ using

$$\mathbf{F}^l_{t-1} = \text{MHA}(\mathbf{F}^{l-1}_{t-1}) \tag{5}$$

, where MHA is the multi-head self-attention [49] operation. Afterwards, We downsample $\mathbf{T}_{\text{adapt}}$ to generate $\mathbf{T}^l_{\text{adapt}} \in \mathbb{R}^{N_p \times d}$ using

$$\mathbf{T}^l_{\text{adapt}} = \mathbf{T}_{\text{adapt}}\mathbf{W}^l_{\text{adapt}} \tag{6}$$

, where $\mathbf{W}^l_{\text{adapt}} \in \mathbb{R}^{d_p \times d}$ is the learnable projection matrix. The cross-attention mechanism involves query ($\mathbf{Q}$), key ($\mathbf{K}$), and value ($\mathbf{V}$) generation using

$$\mathbf{Q} = \mathbf{F}^l_{t-1}\mathbf{W}_Q, \quad \mathbf{K} = \mathbf{T}^l_{\text{adapt}}\mathbf{W}_K, \quad \mathbf{V} = \mathbf{T}^l_{\text{adapt}}\mathbf{W}_V \tag{7}$$

Here, $\mathbf{W}_Q \in \mathbb{R}^{d \times d_q}$, $\mathbf{W}_K \in \mathbb{R}^{d \times d_k}$, and $\mathbf{W}_V \in \mathbb{R}^{d \times d_v}$ are learned projection matrices. The cross-attention output $\mathbf{A} \in \mathbb{R}^{N_{t-1} \times d_v}$ is computed as:

$$\mathbf{A} = \text{Softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right)\mathbf{V} \tag{8}$$

where d_k is the dimensionality of the key vectors. The cross attention mechanism enables the model to dynamically adjust the importance of different parts of the text relative to the CAD sequence. Afterwards, the output embedding of the decoder block l is generated using

$$\mathbf{F}^l_{t-1} \leftarrow \text{LayerNorm}(\text{FFN}(\text{LayerNorm}(\mathbf{F}^l_{t-1} + \text{Dropout}(\text{A})))) \tag{9}$$

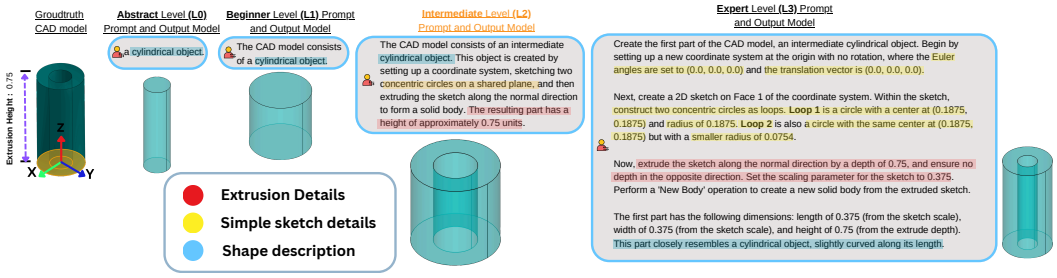


Figure 4: Parametric CAD model generation by Text2CAD transformer using different text prompts. Our text prompts follow a certain structure highlighting the different design aspects of CAD construction workflow (shown in different colors). Abstract (L0) and Beginner (L1) level prompts contain shape descriptions (teal color) whereas Intermediate (L2) and Expert (L3) level prompts are more parametric and contain design details for sketch and extrusion (yellow and red).

,where FFN is the feed foward network [49] and LayerNorm is the Layer Normalization [49]. The complete Transformer decoder block is repeated L times, allowing for deeper integration and refinement of the text and CAD tokens. The final CAD embedding $F_{t-1}^{L} \in \mathbb{R}^{N_{t-1} \times d}$ is passed to an MLP to generate the output CAD sequence. We use Cross-Entropy loss during training.

5 Experiment

Dataset. We use the DeepCAD [56] dataset which contains approximately $\sim$ 150k training CAD sequences and $\sim$ 8k test and validation sequences in sketch-and-extrude format. Following, [56, 19], the sketches and the final CAD models are normalized within a unit bounding box. For each sample in the dataset, four design prompts ranging from abstract to expert levels (**L0, L1, L2, L3**) are generated using our data annotation pipeline resulting in $\sim$ 600k training samples, and $\sim$ 32k test and validation samples.

Implementation Details. Text2CAD transformer consists of $L = 8$ decoder blocks with 8 self-attention heads. The learning rate is 0.001 with AdamW [28] optimizer. Dropout is 0.1. Maximum number of word tokens, N_p is fixed as 512 and CAD tokens N_c as 272. The dimension d_p for the pre-trained Bert encoder [8] embedding $\mathbf{T}$ as well as $\mathbf{T}_{adapt}$ is 1024. The CAD sequence embedding d is 256. Following [19], the first two decoder blocks do not use any cross-attention operation between the text embedding and the CAD sequence embedding. The Text2CAD transformer has been trained with teacher-forcing [53] strategy for 160 epochs using 1 Nvidia A100-80GB GPU for 2 days. During inference, top-1 sampling has been used to autoregressively generate the CAD sequences from an input text.

Baseline. Since there are no existing methods for generating parametric CAD sequences from text prompts, we use DeepCAD [56] and our Text2CAD w/o AL (*i.e.,* without Adaptive Layer) variant as our baselines. To adjust DeepCAD [56] for performing CAD generation from text inputs, the Adaptive Layer [56] (see Section 4) is trained to map the pre-trained BERT [8] embedding $\mathbf{T}$ into the ground truth latent vector z. During inference, the predicted latent vector z is then passed to the pre-trained DeepCAD [56] decoder to generate the CAD sequences. For Text2CAD w/o AL, the pre-trained BERT [8] embedding $\mathbf{T}$ is directly passed to the transformer decoder.

Experimental Setup. For setting up our evaluation protocol, the selection of the input text prompts and desired outcomes of the CAD models are depicted in Figure 4. Our textual annotations follow a certain structure. In the abstract (L0) and beginner (L1) level prompts, the shape descriptions are more prevalent (in teal color in Figure 4). The intermediate level prompts augment simple sketch (in yellow color) and extrusion (in red color) details with the shape descriptions. Expert-level prompts include more precise details for each of the design aspects previously highlighted. In all our experiments discussed below, we use these four levels of prompts following the aforementioned formats. However, we have conducted another experiment, where we interpolate between the abstract and expert prompts to generate multiple new prompts and observe the performance of our model on these prompts. Due to the space restriction, we provide the results of this experiment in the supplementary Section 12.

5.1 Evaluation

Our goal is to evaluate how well the generated CAD sequences align with their respective input text prompts. We concentrate on two primary aspects: (1) examining the *parametric correspondence* between the predicted CAD sequences and the input text prompts, and (2) conducting *visual inspections*

https://doi.org/10.52202/079017-0242

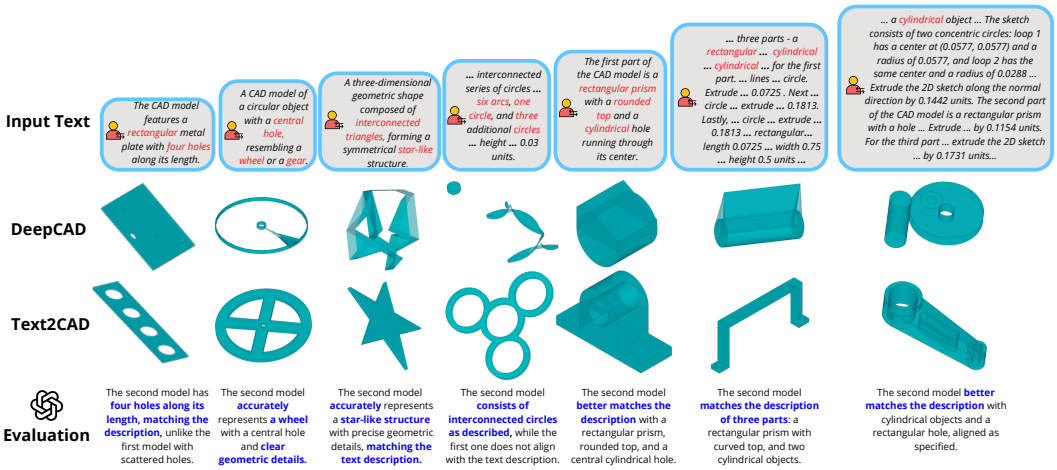


Figure 5: Qualitative results of the reconstructed CAD models of DeepCAD [56] and Text2CAD on DeepCAD [56] dataset. From top to bottom - Input Texts, Reconstructed CAD models using DeepCAD [56] and Text2CAD respectively and GPT-4V [35] Evaluation.

of the reconstructed 3D CAD models. Currently, there is no standardized benchmark for text-to-3D models that can be directly applied to this task. Existing methods for text-to-3D [51, 15] utilize pre-trained CLIP [42] models to measure the alignment between reconstructed 3D models and their text prompts. However, CLIP [42] scores may not adequately evaluate the geometric [57] and the parametric aspects of CAD designs. Therefore, inspired by [57, 11], we employ three evaluation metrics to thoroughly measure the parametric and geometric alignment between the generated CAD designs and the text prompts.

A. CAD Sequence Evaluation: In this evaluation strategy, we comprehensively assess the *parametric correspondence* between the generated CAD sequences with the input text. We use the groud truth CAD sequence for this purpose. We only use this evaluation strategy for expert-level (L3) prompts. Since, expert-level prompts, being highly detailed with parametric details, exhibit a higher one-to-one correspondence with the ground truth CAD construction sequences compared to other levels.

To measure the correspondence between the ground truth and the predicted sequence, the strategy outlined in [19] is followed. We evaluate the F1 scores of the primitives and extrusions by aligning the predicted loops with the ground truth loops within the same sketch using the Hungarian matching algorithm [21] (An example is provided in supplementary Figure 10). The geometric alignment between the ground truth and reconstructed CAD models is measured using the *chamfer distance* (CD). The *Invalidity Ratio* (IR) is calculated to measure the proportion of invalid CAD sequences (*i.e.,*sequences that can not generate any CAD model).

Table 1: Quantitative evaluation between DeepCAD [56] and our method (AL is Adaptive Layer). The scores are evaluated only for **Expert Level** (L3) prompts. The results include F1 scores for primitives and extrusions as well as mean and median CD and IR. CD is multiplied by 10^3.

Model	F1↑				Median CD↓	Mean CD↓	IR↓
	Line	Arc	Circle	Extrusion			
DeepCAD	76.78	20.04	65.14	88.72	32.82	97.93	10.00
Text2CAD w/o AL	78.88	27.18	71.44	93.28	0.82	35.91	2.69
Text2CAD	**81.13**	**36.03**	**74.25**	**93.31**	**0.37**	**26.41**	**0.93**

Table 1 summarizes the quantitative results between the baseline methods and our final Text2CAD transformer. Compared to the baselines (rows 1-2), our model (row 3) achieves higher F1 scores for all the primitives and extrusion, with the most notable being an 80% improvement in the F1 score for arcs. The results indicate a better correspondence between the expert-level prompts and the predicted CAD sequences. Notably, our model significantly outperforms the baseline DeepCAD [56] in terms of median CD and invalidity ratio by a factor of ~ 88.7 and ~ 10.75 respectively. The higher CD despite having relatively high F1 scores for DeepCAD [56] indicates that even though DeepCAD [56] can recognize the primitives and the number of extrusions from the input texts, it

https://doi.org/10.52202/079017-0242

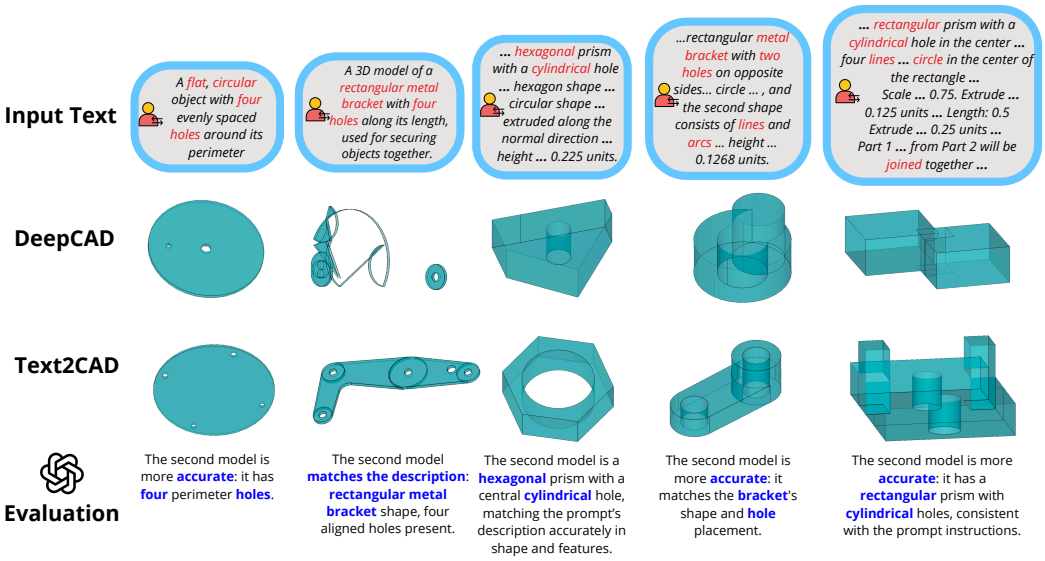


Figure 6: Additional Qualitative results of the reconstructed CAD models of DeepCAD [56] and Text2CAD on DeepCAD [56] dataset. From top to bottom - Input Texts, Reconstructed CAD models using DeepCAD [56] and Text2CAD respectively and GPT-4V [35] Evaluation.

fails to correctly parameterize those primitives and the extrusions. Compared to DeepCAD [56], the Text2CAD transformer predicts more accurate sketches and extrusion parameters thanks to its layerwise cross-attention mechanism. In Figure 5 and 6, some qualitative results are shown.

Table 1 (rows 2 and 3) shows the results between Text2CAD w/o AL and our final model. The incorporation of the Adaptive layer in the Text2CAD transformer improves the F1 scores for primitives such as lines, arcs, and circles. Notably, there is a 2.85% increase of F1 scores for lines, 32.56% for arcs, and 3.93% for circles. Moreover, the improvement is particularly striking in terms of the IR, which sees a remarkable reduction by a factor of ~ 2.9.

Table 2: GPT-4 evaluation of the CAD models generated from 1000 prompts per level and User studies of 100 samples per level. In both evaluations, overall Text2CAD is a favored choice over DeepCAD [56].

Model	GPT-4 Evaluation (%)				User Study-based Evluation (%)			
	Abstract Level (L0)	Beginner Level (L1)	Intermediate Level (L2)	Expert Level (L3)	Abstract Level (L0)	Beginner Level (L1)	Intermediate Level (L2)	Expert Level (L3)
Undecided	0.80	0.5	1	0.70	-	-	-	-
DeepCAD	47.40	**51.15**	40.20	36.06	**50.95**	48.73	44.94	41.14
Text2CAD	**51.80**	48.35	**58.80**	**63.24**	49.05	**51.27**	**55.06**	**58.86**

B. GPT-4V Evaluation: To perform the *visual inspections* of the 3D CAD models generated from abstract (L0), beginner (L1), intermediate (L2), expert (L3) level prompts, GPT-4V [35] has been used. We follow the protocol outlined in [57] and generate a meta prompt that consists of multi-view images of the reconstructed CAD models from both DeepCAD [56] and our model as well as the input text prompts. Following this, GPT-4V [35] provides a verdict on which model predictions accurately reflect the text descriptions in terms of shape and geometry. But if the two models are very similar in shape and match the input text accurately, then it outputs 'Undecided'.

We randomly select 1,000 samples for each level from the test dataset and generate a verdict per sample using GPT-4V [35]. Table 2 (*left*) presents the final results. These results indicate that overall Text2CAD outperforms baseline DeepCAD [56]. Additionally, we observe that the performance gap between Text2CAD and DeepCAD [56] is minimal at the abstract (L0) and beginner (L1) levels, despite Text2CAD losing by 2.8% at the beginner level. However, as the complexity and parametric details of the prompts increase, the gap widens significantly. Text2CAD outperforms DeepCAD [56] at the intermediate (L2) and expert (L3) levels, leading by as much as 18.6% and 27.18% respectively. This indicates that Text2CAD is more inclusive of the highly detailed design prompts. In supplementary Figure 11, we have provided some examples.

https://doi.org/10.52202/079017-0242

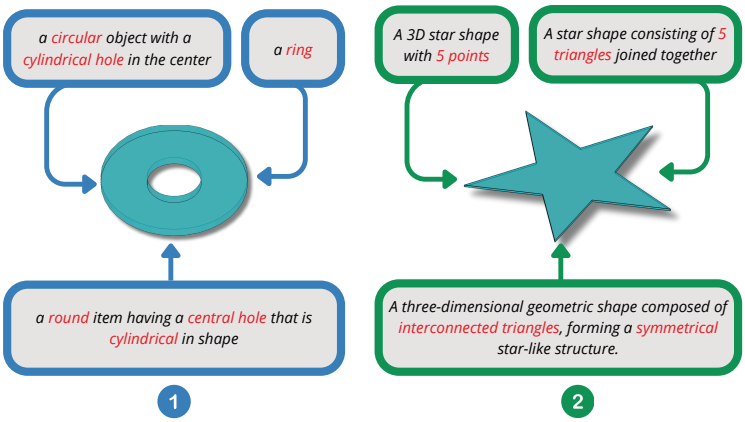


Figure 7: Visual examples of 3D CAD model generation using varied prompts. **(1)** Three different prompts yielding the *same ring-like model*, some *without explicitly mentioning 'ring'*. **(2)** Three diverse prompts resulting in *same star-shaped model*, each emphasizing *different star characteristics*.

C. User Study-based Evaluation: We conduct a user study with 100 randomly selected examples per level to evaluate the preference between our method and DeepCAD [56]. Five CAD designers with varying experience levels participate in the evaluation. Each participant is shown multi-view images of the reconstructed 3D CAD models from both methods side by side, along with their corresponding input texts. They are then asked to determine which CAD model is more geometrically accurate and easier to edit to achieve the desired outcome. Each participant evaluates 20 samples per level. The final result is provided in Table 2 (*right*). To our surprise, the result follows a similar pattern as GPT-4V [35] evaluation with a minimal performance gap in the abstract (L0) and beginner (L1) levels and a wider gap for more detailed intermediate (L2) and expert (L3) prompts.

5.2 Prompt Diversity

The diversity of the generated textual prompts in the Text2CAD dataset depends on both the variety of CAD models available in the DeepCAD [56] dataset, as well as the performance of Mistral [16] and LLaVA-Next [26]. To enhance prompt variety, we have focused more on generating shape descriptions from LLaVA-Next [26] rather than only object names. For example, in our annotation *"a ring"* can be sometimes described as *"a circular object with a cylindrical hole in the center"*. This approach enables our transformer model to learn to generate the same CAD models using different styles of textual prompts. In Figure 7, we show two examples where the network generated same CAD models from various types of prompts.

6 Limitation

Despite the promising results of Text2CAD, several limitations exist. Firstly, LLaVA-NeXT [26] is sensitive to the perspective distortions in multi-view images, which affects the generated shape descriptions and final LLM-generated prompts. Secondly, effective tokenization of numerical parameters in texts is still an open research problem in NLP. In Text2CAD transformer architecture, we used BERT [8] tokenizer which might tokenize some numerical parameters as an *UNK* token. Thirdly, the lack of standardized benchmarks for evaluating text-to-CAD generation poses challenges in assessing model performance comprehensively. Furthermore, the DeepCAD [56] dataset is imbalanced, predominantly featuring rectangular and cylindrical shapes, which limits the model's robustness towards more complex shapes. Some failure cases are described in supplementary Section 13.

7 Conclusion

In this paper, we introduce Text2CAD, the first AI framework designed to generate parametric CAD models from text prompts suitable for users of all skill levels. Our contributions include a two-stage data annotation pipeline using Mistral-50B [16] and LLaVA-NeXT [26] and a novel end-to-end trainable Text2CAD Transformer architecture that effectively transforms natural language instructions into sequential CAD models. Through a comprehensive evaluation, including GPT-4V [35] assessments and user studies by CAD designers, we demonstrate that Text2CAD outperforms the adapted two-stage baseline, especially as the complexity and detail of the prompts increase. Future work will focus on addressing the current limitations, such as reducing annotation inaccuracies and improving dataset diversity, to further enhance the robustness and applicability of Text2CAD.

https://doi.org/10.52202/079017-0242

8 Acknowledgement

This work was in parts supported by the EU Horizon Europe Framework under grant agreement 101135724 (LUMINOUS).

References

[1] Freecad. https://www.freecadweb.org/. Accessed: <Date>. 1

[2] Mcmaster. https://www.mcmaster.com/,. 1

[3] Open3d. http://www.open3d.org/. 19

[4] Zoo.dev. https://zoo.dev/. 2

[5] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020. 4

[6] Kseniya Cherenkova, Djamila Aouada, and Gleb Gusev. Pvdeconv: Point-voxel deconvolution for autoencoding cad construction in 3d. pages 2741–2745, 10 2020. 3

[7] John G Cherng, Xin-Yu Shao, Yubao Chen, and Peter R Sferro. Feature-based part modeling and process planning for rapid response manufacturing. *Computers & industrial engineering*, 34(2):515–530, 1998. 1

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics*, 2019. 5, 6, 7, 10

[9] Elona Dupont, Kseniya Cherenkova, Anis Kacem, Sk Aziz Ali, Ilya Aryhannikov, Gleb Gusev, and Djamila Aouada. Cadops-net: Jointly learning cad operation types and steps from boundary-representations. In *2022 International Conference on 3D Vision (3DV)*, 2022. 3

[10] Y. Ganin, S. Bartunov, Y. Li, E. Keller, and S. Saliceti. Computer-aided design as language. *Advances in Neural Information Processing Systems*, 34, 2021. 3

[11] Yuze He, Yushi Bai, Matthieu Lin, Wang Zhao, Yubin Hu, Jenny Sheng, Ran Yi, Juanzi Li, and Yong-Jin Liu. T^3bench: Benchmarking current progress in text-to-3d generation, 2023. 8

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

[13] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. In *International Conference on Learning Representations*, 2020. 3

[14] Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ArXiv*, abs/2311.05232, 2023. 4, 5, 16

[15] Ajay Jain, Ben Mildenhall, Jonathan T. Barron, P. Abbeel, and Ben Poole. Zero-shot text-guided object generation with dream fields. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 857–866, 2021. 8

[16] Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023. 2, 10, 15, 16

 https://doi.org/10.52202/079017-0242

[17] Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024. 3, 4

[18] Minseop Jung, Minseong Kim, and Jibum Kim. Contrastcad: Contrastive learning-based representation learning for computer-aided design models. 2024. 3

[19] Mohammad Sadil Khan, Elona Dupont, Sk Aziz Ali, Kseniya Cherenkova, Anis Kacem, and Djamila Aouada. Cad-signet: Cad language inference from point clouds using layer-wise sketch instance guided attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 1, 3, 5, 7, 8, 15, 18

[20] 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. 3

[21] Harold W. Kuhn. The Hungarian Method for the Assignment Problem. *Naval Research Logistics Quarterly*, 2(1–2):83–97, March 1955. 8

[22] 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. 1

[23] Joseph George Lambourne, Karl Willis, Pradeep Kumar Jayaraman, Longfei Zhang, Aditya Sanghi, and Kamal Rahimi Malekshan. Reconstructing editable prismatic cad from rounded voxel models. In *SIGGRAPH Asia 2022 Conference Papers*, pages 1–9, 2022. 3

[24] Changjian Li, Hao Pan, Adrien Bousseau, and Niloy J Mitra. Free2cad: Parsing freehand drawings into cad commands. *ACM Transactions on Graphics (TOG)*, 41(4):1–16, 2022. 3

[25] Chen-Hsuan Lin, Jun Gao, Luming Tang, Towaki Takikawa, Xiaohui Zeng, Xun Huang, Karsten Kreis, Sanja Fidler, Ming-Yu Liu, and Tsung-Yi Lin. Magic3d: High-resolution text-to-3d content creation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 2

[26] Haotian Liu, Chunyuan Li, Yuheng Li, Bo Li, Yuanhan Zhang, Sheng Shen, and Yong Jae Lee. Llava-next: Improved reasoning, ocr, and world knowledge, 2024. 2, 3, 4, 10, 15

[27] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In *NeurIPS*, 2023. 2, 3

[28] 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. 7

[29] Weijian Ma, Minyang Xu, Xueyang Li, and Xiangdong Zhou. Multicad: Contrastive representation learning for multi-modal 3d computer-aided design models. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, CIKM '23, page 1766–1776, New York, NY, USA, 2023. Association for Computing Machinery. 3

[30] Felipe Machado, Norberto Malpica, and Susana Borromeo. Parametric cad modeling for open source scientific hardware: Comparing openscad and freecad python scripts. *Plos one*, 14(12):e0225795, 2019. 2

[31] Dimitrios Mallis, Ali Sk Aziz, Elona Dupont, Kseniya Cherenkova, Ahmet Serdar Karadeniz, Mohammad Sadil Khan, Anis Kacem, Gleb Gusev, and Djamila Aouada. Sharp challenge 2023: Solving cad history and parameters recovery from point clouds and 3d scans. overview, datasets, metrics, and baselines. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1786–1795, 2023. 3

[32] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021. 2

[33] 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. 3

[34] Alex Nichol, Heewoo Jun, Prafulla Dhariwal, Pamela Mishkin, and Mark Chen. Point-e: A system for generating 3d point clouds from complex prompts, 2022. 2

[35] Josh OpenAI, Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023. 2, 3, 8, 9, 10, 16, 19, 20, 21

[36] Wamiq Para, Shariq Bhat, Paul Guerrero, Tom Kelly, Niloy Mitra, Leonidas J Guibas, and Peter Wonka. Sketchgen: Generating constrained cad sketches. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 5077–5088. Curran Associates, Inc., 2021. 3

[37] Bonsa Regassa Hunde and Abraham Debebe Woldeyohannes. Future prospects of computer-aided design (cad) – a review from the perspective of artificial intelligence (ai), extended reality, and 3d printing. *Results in Engineering*, 14:100478, 2022. 1

[38] Juergen Riegel, Werner Mayer, and Yorik van Havre. Freecad. *Freecadspec2002. pdf*, 2016. 2

[39] Daniel Ritchie, Paul Guerrero, R Kenny Jones, Niloy J Mitra, Adriana Schulz, Karl DD Willis, and Jiajun Wu. Neurosymbolic models for computer graphics. In *Computer Graphics Forum*, volume 42, pages 545–568. Wiley Online Library, 2023. 3

[40] David Robertson and Thomas J Allen. Cad system use and engineering performance. *IEEE Transactions on Engineering Management*, 40(3):274–282, 1993. 1

[41] Tariq Samad. *A natural language interface for computer-aided design*. Springer Science & Business Media, 1986. 3

[42] Aditya Sanghi, Rao Fu, Vivian Liu, Karl DD Willis, Hooman Shayani, Amir H Khasahmadi, Srinath Sridhar, and Daniel Ritchie. Clip-sculptor: Zero-shot generation of high-fidelity and diverse shapes from natural language. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18339–18348, 2023. 8

[43] Ari Seff, Yaniv Ovadia, Wenda Zhou, and Ryan P Adams. Sketchgraphs: A large-scale dataset for modeling relational geometry in computer-aided design. *arXiv preprint arXiv:2007.08506*, 2020. 3

[44] Ari Seff, Wenda Zhou, Nick Richardson, and Ryan P Adams. Vitruvion: A generative model of parametric cad sketches. In *International Conference on Learning Representations*, 2021. 3

[45] Siemens. ParaSolid. https://www.autodesk.com/products/autocad/. 1

[46] Solidworks. 3D CAD Design Software. https://www.solidworks.com/. Online: accessed 02-June-2022. 1

[47] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023. 2, 3

[48] Kenton Varda. Google protocol buffers: Google's data interchange format. *Technical report*, 2008. 3

[49] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010, Red Hook, NY, USA, 2017. Curran Associates Inc. 2, 3, 6, 7

 https://doi.org/10.52202/079017-0242

[50] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. *Advances in neural information processing systems*, 28, 2015. 3

[51] Zhengyi Wang, Cheng Lu, Yikai Wang, Fan Bao, Chongxuan Li, Hang Su, and Jun Zhu. Prolificdreamer: High-fidelity and diverse text-to-3d generation with variational score distillation. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. 2, 8

[52] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. 5

[53] Ronald J. Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1(2):270–280, 1989. 7

[54] 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. 3

[55] 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. 3

[56] 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, October 2021. 1, 2, 3, 4, 5, 7, 8, 9, 10, 15, 16, 19

[57] Tong Wu, Guandao Yang, Zhibing Li, Kai Zhang, Ziwei Liu, Leonidas Guibas, Dahua Lin, and Gordon Wetzstein. Gpt-4v(ision) is a human-aligned evaluator for text-to-3d generation. In *CVPR*, 2024. 8, 9

[58] Xiang Xu, Pradeep Kumar Jayaraman, Joseph G Lambourne, Karl DD Willis, and Yasutaka Furukawa. Hierarchical neural coding for controllable cad model generation. *arXiv preprint arXiv:2307.00149*, 2023. 1, 3

[59] 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

[60] Yasuhiro Yamamoto and Kumiyo Nakakoji. Interaction design of tools for fostering creativity in the early stages of information design. *International Journal of Human-Computer Studies*, 63(4):513–535, 2005. Computer support for creativity. 1

[61] Xinlu Zhang, Yujie Lu, Weizhi Wang, An Yan, Jun Yan, Lianke Qin, Heng Wang, Xifeng Yan, William Yang Wang, and Linda Ruth Petzold. Gpt-4v(ision) as a generalist evaluator for vision-language tasks. *ArXiv*, abs/2311.01361, 2023. 2

[62] Qingnan Zhou and Alec Jacobson. Thingi10k: A dataset of 10, 000 3d-printing models. *ArXiv*, abs/1605.04797, 2016. 3

9 CAD Sequence Representation

Table 3 shows all the tokens used in our CAD sequence representation. We use the same representation as proposed by Khan et al [19] which uses a *sketch-and-extrude* format. Each 2D sketch consists of multiple faces and each face consists of multiple loops and every loop either contains a line and a arc or a circle. Loops are always closed (*i.e.,*same start and end coordinate). We parameterize the curves in the following way

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

Finally, we represent a sketch using a sequence of 2D coordinates only with specific end tokens for the end of curve, loop, face and sketch. Each extrusion sequence consists of the 10 parameters followed by an end of extrusion token. These are

- **Euler Angles:** 3 parameters (θ, ϕ, γ) determining the orientation of the sketch plane.
- **Translation Vector:** 3 parameters (τ_x, τ_y, τ_z) that describe the translation of the sketch plane.
- **Sketch Scale:** 1 parameter (σ) for scaling the 2D sketches.
- **Extrude distances:** 2 parameters (d^+, d^-) containing the extrusion distances towards and opposite of the normal of the sketch plane.
- **Boolean Operation:** 1 parameter (β) determining the extrusion operation. There are 4 extrusion operation in DeepCAD [56] dataset namely - *solid body, cut, join and intersection*.

Except the boolean operation and all the end tokens, all the 2D sketch parameters as well as the extrusion parameters are quantized in 8 bits.

Table 3: CAD sequence representation used in our experiment.

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

10 Implementation Details for Data Annotation

As mentioned in Section 3, we use LLaVA-NeXT [26] for the VLM oriented task and Mistral-50B [16] for LLM tasks. We use 1 Nvidia A100-40GB GPU to run LLaVA-NeXT [26] and 4 Nvidia A100-80GB GPUs to run Mistral-50B [16].

https://doi.org/10.52202/079017-0242

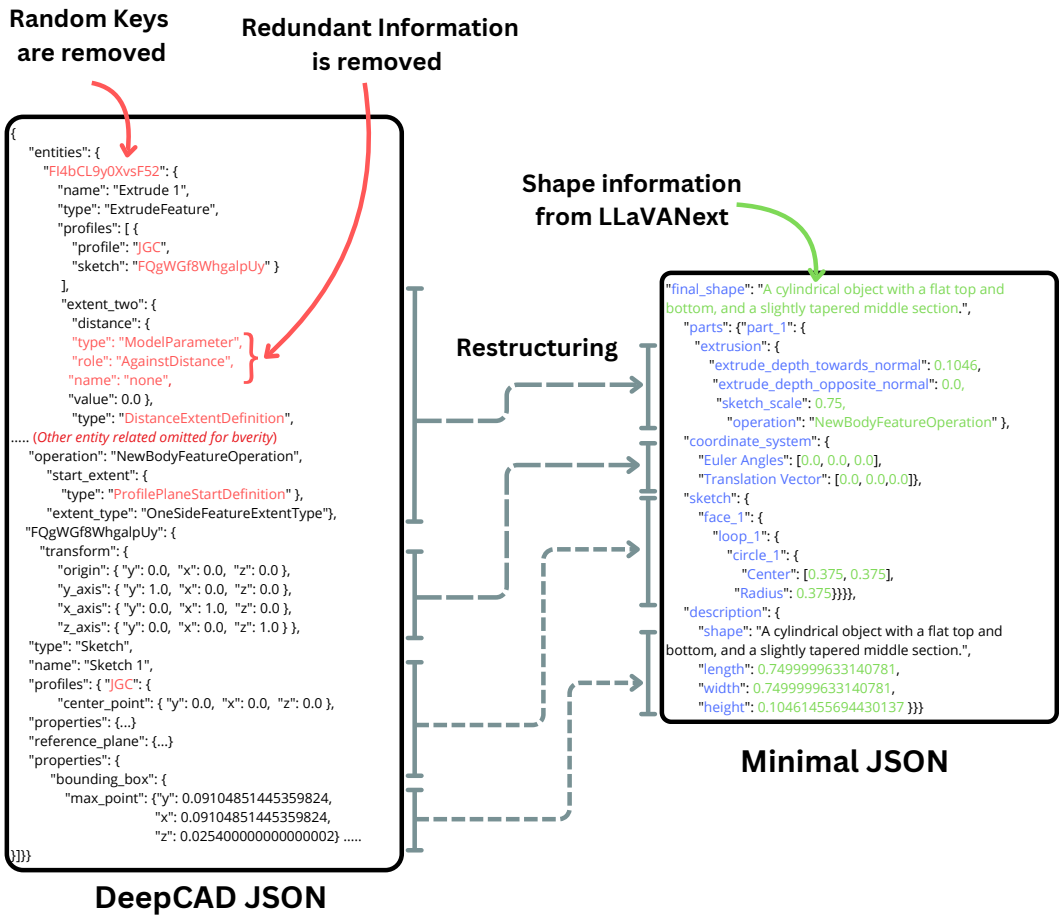


Figure 8: An example of Minimal metadata JSON (*right*) generated from DeepCAD [56] JSON (*left*). During the minimal metadata generation, random keys (*e.g.,"FI4bCL9y0XvsF52"*) or redundant design information (*e.g.,{"type": "ModelParameter", "role": "AgainstDistance"}*) is removed.

11 Additional Details on Data Annotation Pipeline

Since LLMs are prone to hallucinations [14], we employ several strategies to mitigate this issue. Firstly, we observe that when we directly pass the raw DeepCAD [56] jsons to Mistral [16] instead of the minimal metadata to generate the detailed natural language instructions, the model often uses the random keys provided in the dictionary to refer to the corresponding curves or the sketches. Additionally, the json contains redundant information that is not necessary in the final CAD construction. To overcome this, we generate minimal metadata by restructuring the dictionary into a more human-readable format. In Figure 8, we provide an example DeepCAD [56] Json (*left*) and a minimal data(*right*)

12 Additional Experimental Details on Interpolated prompts

In this section, we provide details on our model's performance on text prompts that contain different structure than the training samples. To generate these prompts, we pass all four text prompts (*i.e.,*abstract, beginner, intermediate and expert) of a CAD model to GPT-4V [35] and ask to generate 20 more samples by interpolating between the all the levels. Figure 12 and Figure 13 shows visual results of two examples. The results indicate that Text2CAD can effectively handle varying level of prompt structures, generating accurate and similar CAD models in shape compared to the ground truth. It retains the overall geometry for prompts with highly abstract shape descriptions

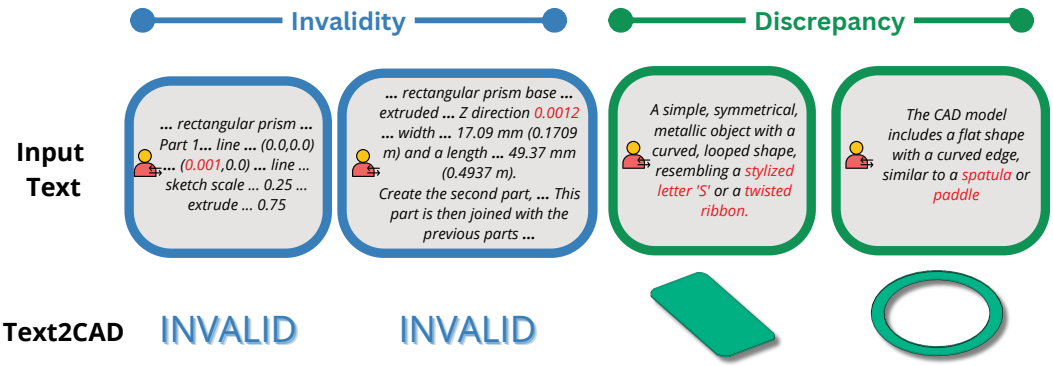


Figure 9: Failure cases for Text2CAD Transformer. **Invalid Samples** (left): The network fails to generate any valid CAD model. **Discrepancy Cases** (right): The generated CAD model does not match the input text prompts.

(*first* and *second* row in Figure 12 and Figure 13) . As the parametric details increase in the text prompts, it generates the precise CAD model as the ground truth (*third* and *fourth* row in Figure 12 and Figure 13)

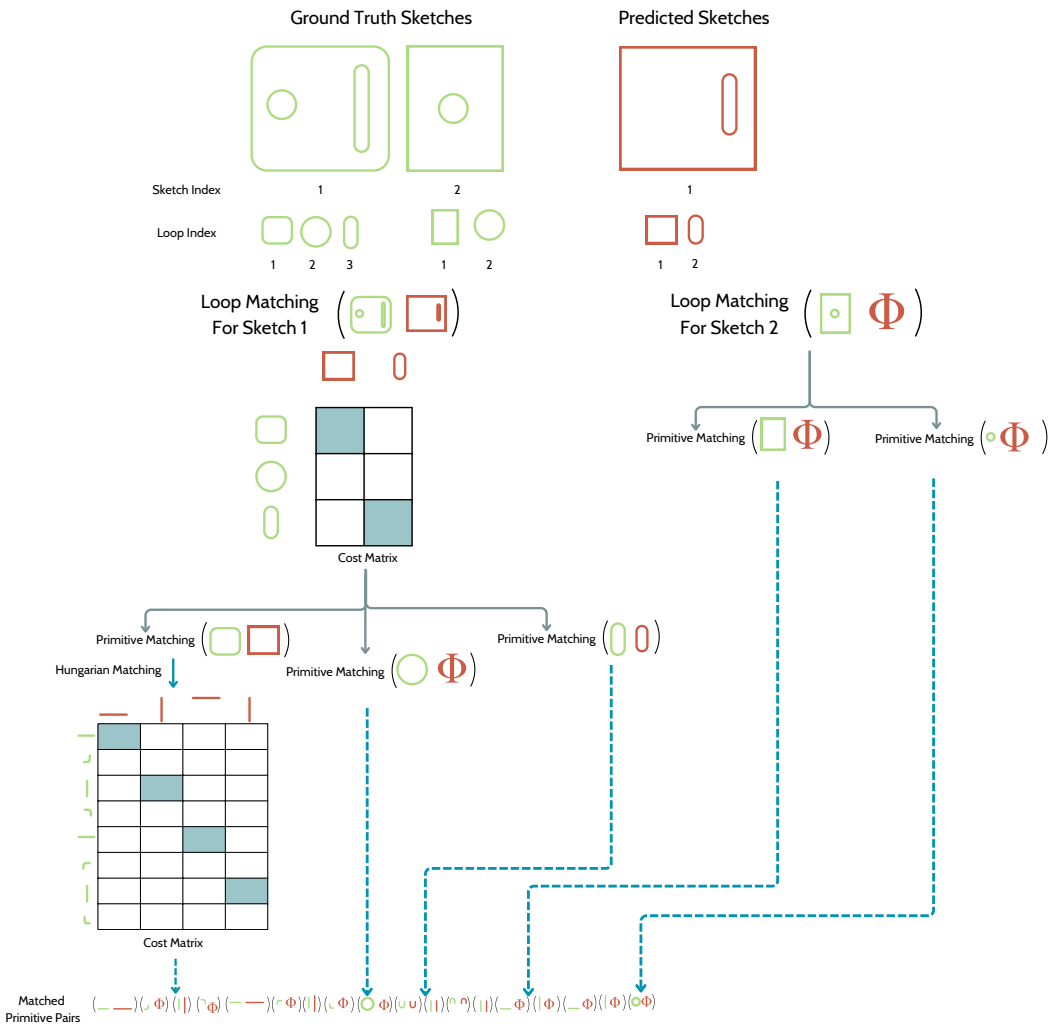
13 Discussion on Failure Cases

In this section, we describe two types of failure cases for Text2CAD Transformer. In Figure 9, we have shown examples of both type of cases.

1. Invalidity: In this scenario, the model fails to generate any CAD model from the text prompts. As reported in Table 1, this occurs in approximately 1% of the test samples. In these cases, the model predicts invalid sketch or extrusion parameters, such as the same start and end points for lines or arcs, or zero values for extrusion depth on both sides.

2. Discrepancy: Discrepancy refers to situations where the generated model does not precisely match the shape described in the text prompts. This is more prevalent in our model than invalidity and is harder to quantify. We notice that this occurs when prompts are more focused on object name (*e.g.*,spatula, paddle) rather than parametric descriptions. We argue that this issue comes from noisy VLM annotations. As mentioned in the Section 6, the perspective distortions in multi-view images can affect the accuracy of shape or object name recognition. These noisy descriptions propagate into other prompt levels using the annotation pipeline.

https://doi.org/10.52202/079017-0242



	Line	Arc	Circle
Recall	$\frac{6}{10} = 0.6$	$\frac{2}{6} = 0.33$	0
Precision	1	1	0
F1	0.46	0.50	0

Figure 10: F1 score calculation for CAD sequence evaluation as proposed in [19].

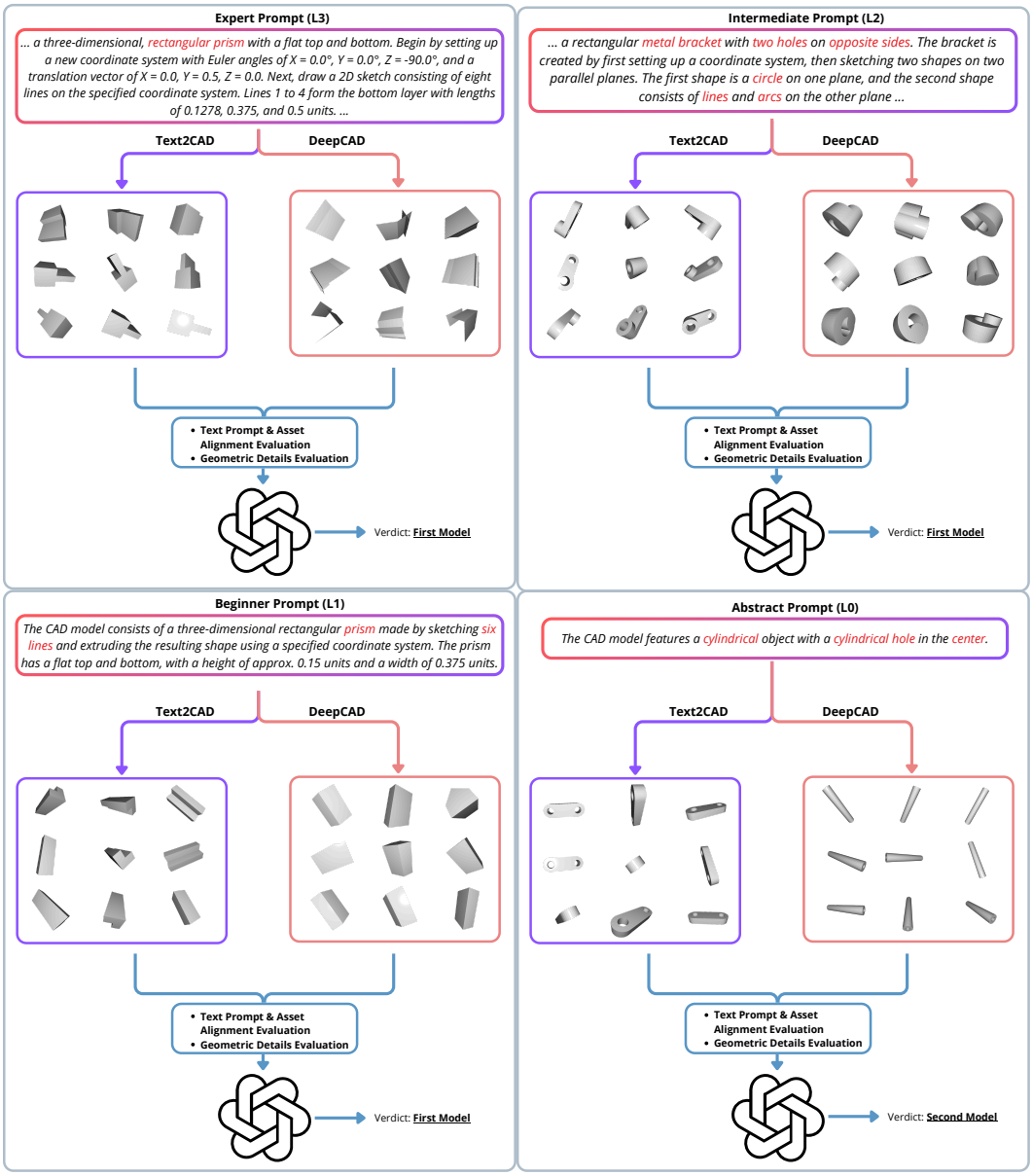


Figure 11: **GPT-4V Evaluation Strategy**: Four prompts (one per level) are randomly sampled from the test set. These prompts are used to reconstruct CAD models from the predicted CAD sequences using both DeepCAD [56] and the proposed Text2CAD. Nine multi-view images of these models are rendered using Open3D [3] and stacked in 3×3 grid, which are used for the GPT-4V [35] evaluation. GPT-4V [35] analyzes their alignment with the initial text prompt and their geometric details and provides a final verdict for this comparison. As shown in the evaluation, our model performs better when input text prompts contain more parametric details.

https://doi.org/10.52202/079017-0242

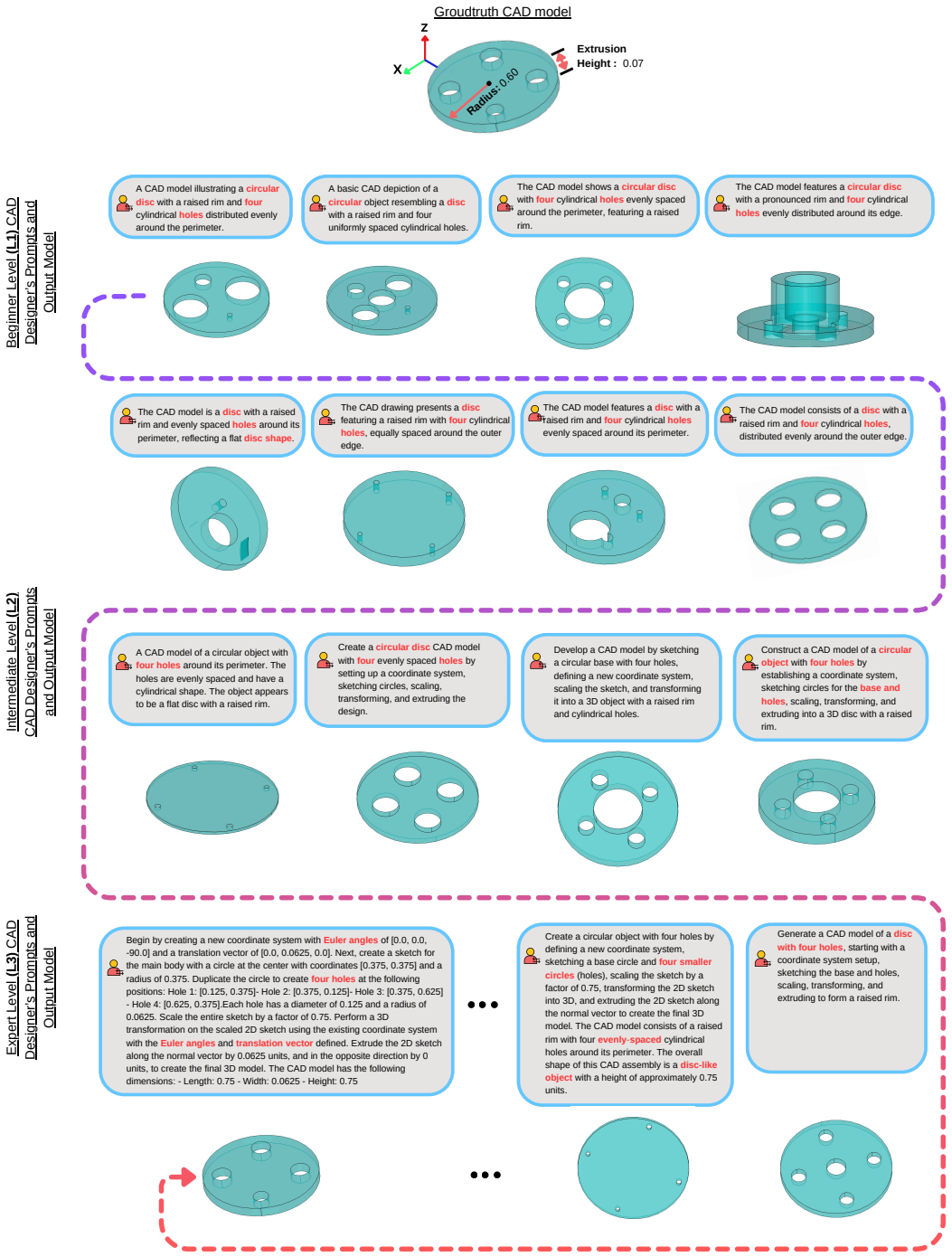


Figure 12: Visual results of Text2CAD on interpolated text prompts generated by GPT-4V [35]. From top to bottom, the geometric details in the text prompts increase.

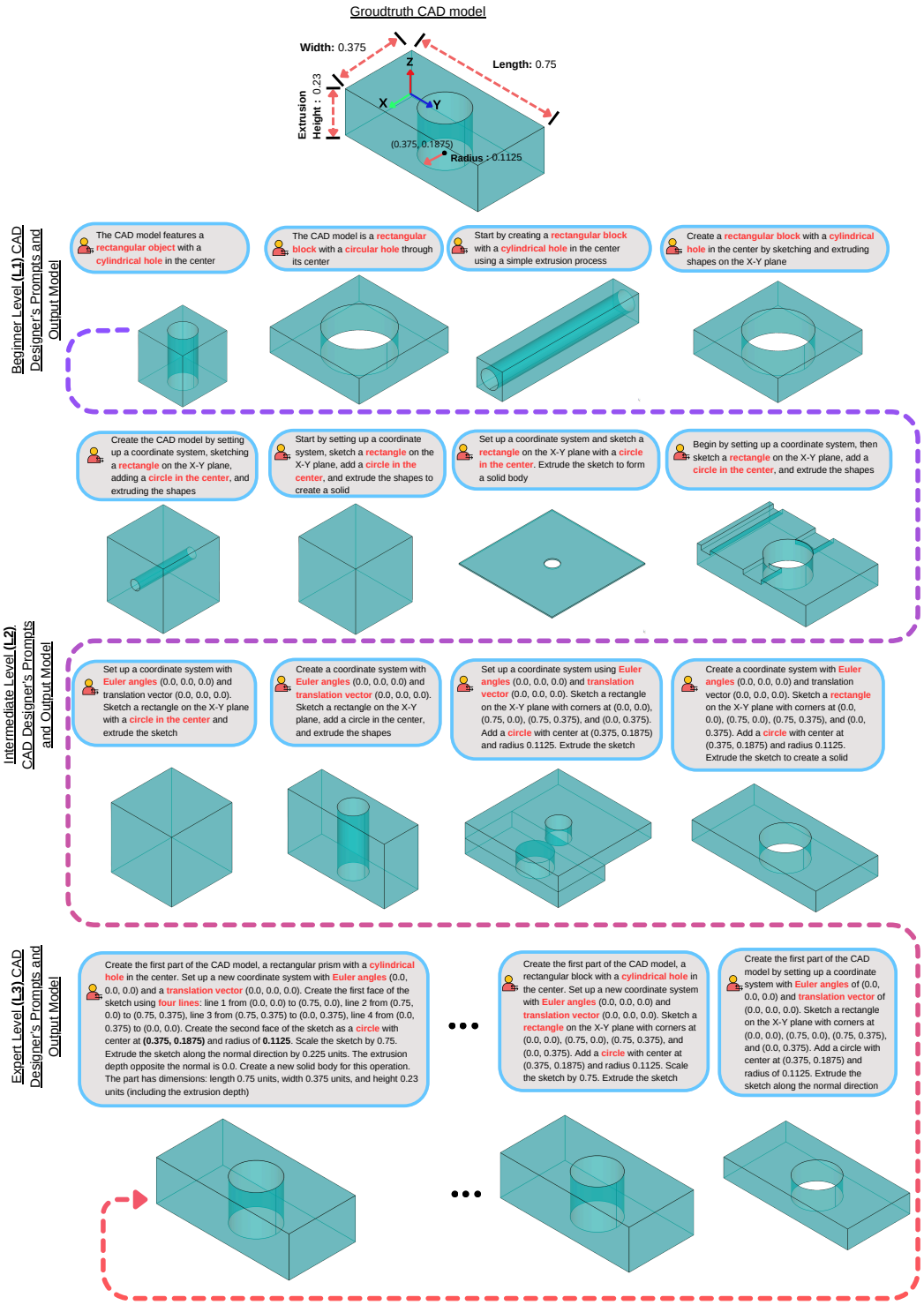


Figure 13: Visual results of Text2CAD on interpolated text prompts generated by GPT-4V [35]. From top to bottom, the geometric details in the text prompts increase.

https://doi.org/10.52202/079017-0242

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: In Section 5 (Experiment), we evaluated our method using various evaluation strategies to justify our contribution.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We have provided a limitation section of our method in the main paper. Please refer to Section 6 (Limitation).

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification: We do not have any theoretical result. Our contribution is focused on novel application in CAD domains.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We have provided the experimental setup in Section 5 (Experiment). We have also provided the LLM and VLM prompts that we used in the Figure 2 and 11.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

https://doi.org/10.52202/079017-0242

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: Currently we have not published our code and dataset. As mentioned in the abstract, we will publish both of them soon.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: We have provided the data-splits, hyperparameter details, training and inference setup in Section 5 (Experiment).

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [NA]

Justification: Given that there is no standardized benchmark for our task (text-to-CAD) at hand, it's not applicable for our method.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Please refer to Section 5 (Experiment) and Section 10 (Implementation Details on Data Annotation Pipeline).

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: Yes we have followed the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We have provided the positive impact of our framework in Section 1 (Introduction). We are not yet aware of any negative societal impacts as of now.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.

https://doi.org/10.52202/079017-0242

- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [Yes]

    Justification: We will implement thorough verification processes to identify and mitigate any potential misuse of our data before releasing the annotations.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We have cited every dataset and other supporting architecture/framework to the best of our knowledge.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.

- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The new assets are generated via our data annotation pipeline. (Please refer to Section 3 (Data Annotation Pipeline).

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: NA

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: NA

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.

https://doi.org/10.52202/079017-0242

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.