GenCAD: Image-Conditioned Computer-Aided Design Generation with Transformer-Based Contrastive Representation and Diffusion Priors

Anonymous authors
Paper under double-blind review

Abstract

The creation of manufacturable and editable 3D shapes through Computer-Aided Design (CAD) remains a highly manual and time-consuming task, hampered by the complex topology of boundary representations of 3D solids and unintuitive design tools. While most work in the 3D shape generation literature focuses on representations like meshes, voxels, or point clouds, practical engineering applications demand the modifiability and manufacturability of CAD models and the ability for multi-modal conditional CAD model generation. This paper introduces GenCAD, a generative model that employs autoregressive transformers with contrastive learning framework and latent diffusion models to transform image inputs into parametric CAD command sequences, resulting in editable 3D shape representations. Extensive evaluations demonstrate that GenCAD significantly outperforms existing state-of-the-art methods in terms of the unconditional and conditional generations of CAD models. Additionally, the contrastive learning framework of GenCAD facilitates the retrieval of CAD models using image queries from large CAD databases—a critical challenge within the CAD community. Our results provide a significant step forward in highlighting the potential of generative models to expedite the entire design-to-production pipeline and seamlessly integrate different design modalities.

1 INTRODUCTION

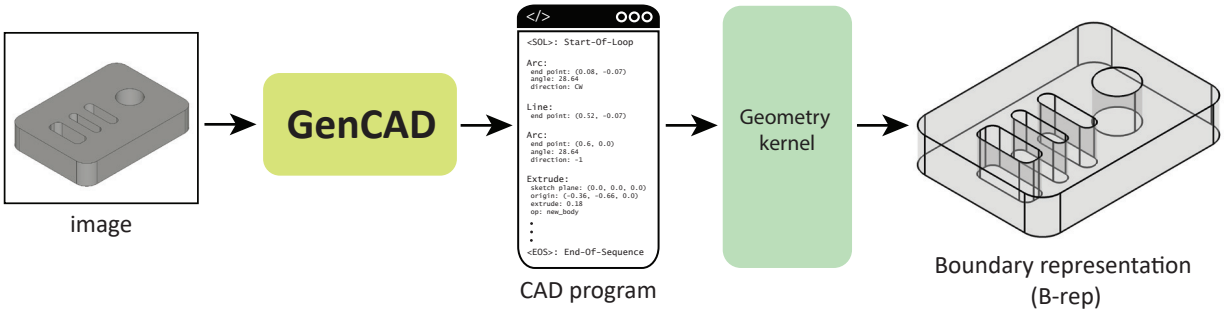

Figure 1: GenCAD demonstrates AI-based generative CAD models conditioned on images

Generating 3D shapes is a fundamental part of the engineering design process that morphs the design from creative intuition to digital 3D shapes. Professional engineers use modern Computer-Aided Design (CAD) models to digitally represent 3D shapes for applications in automotive, aerospace, manufacturing, medical devices, consumer products and architectural designs. Engineering design of such 3D solid modeling is a complex sequential process that requires human expertise and intuition. Since the invention of CAD, most CAD modeling tasks have been manual and typically go through a long iterative process to finalize a design with desired requirements. Additionally, modern CAD software is highly non-intuitive for humans and

comes with a steep learning curve for most practitioners (Piegl, 2005). Based on the motivations of recent literature, we argue that some portion of the 3D CAD modeling process can be automated using learning-based approaches which can offer tremendous acceleration in the design and manufacturing pipeline. An important first step in this direction is to build generative models that can provide valid and real-world 3D CAD models based on user intuitions. As most of the generative model architectures are proposed for image or text generation, a carefully designed framework is needed for 3D CAD modeling in engineering computational design tasks.

Generative models, built upon large neural network architectures, have recently shown impressive performance in image and text generation tasks (Ramesh et al., 2021; Razavi et al., 2019; Saharia et al., 2022; Karras et al., 2019; Rombach et al., 2022). To align the outputs of these generative models with user intentions, they are often conditioned on additional image or text input. Inspired by the success of text and image modalities, many recent studies have focused on building generative models for 3D representations such as meshes (Geuzaine & Remacle, 2009; Feng et al., 2019), voxels (Wang et al., 2017), point clouds (Qi et al., 2017; Qin et al., 2019) and implicit representations (Jun & Nichol, 2023). These applications do not include CAD modeling tasks that require *editable and manufacturable representation* of 3D solids. One of the fundamental challenges in building generative models for CAD is the underlying data structure of the 3D solid model. Since the 1980s, the industry standard in Computer-Aided Design (CAD) has been the use of boundary representations (B-rep) (Weiler, 1986) that encodes geometry information in the parametric representation of surfaces, edges, and vertices. Although B-rep is more complex than other types of 3D shape representations, it offers several benefits. For example, the boundary representation of the 3D solid model is not resolution-dependent and memory-intensive such as mesh, voxels or point-clouds while also being easier to render, unlike their counterparts such as implicit representations. Additionally, the existing off-the-shelf geometry kernels can seamlessly convert a parametric CAD command sequence or a CAD program[1] into a B-rep which makes this representation the de facto industry standard. However, due to the complex topological relationship between geometric entities, B-rep data structure is not directly suitable as input for neural network architectures and an intermediate network-friendly representation must be utilized. Few recent studies have developed CAD datasets that are neural network friendly which makes it feasible to build powerful generative models for direct generating B-reps or CAD models (Koch et al., 2019; Wu et al., 2021; Willis et al., 2021; 2022). Most of these approaches utilize one of the following: graph-based representation of 3D solid models (Jayaraman et al., 2021; Lambourne et al., 2021), a sequential representation of CAD modeling operations (Xu et al., 2022; Wu et al., 2021), or a specialized variant of the B-rep such as indexed list data structure (Jayaraman et al., 2022).

Learning to generate the CAD modeling sequence can provide more usefulness than directly learning to generate a B-rep. We argue that direct B-rep generation is less attractive as it does not encode the underlying design history. The sequence of solid modeling operations, or CAD program, is critical to modern CAD software and offers a more flexible and interpretable representation than the direct B-rep. More specifically, a B-rep model can be thought of as a sequence of parametric CAD commands that can create a 3D solid shape using an off-the-shelf geometry kernel. Although this representation of 3D CAD is straightforward and scalable, most available literature focuses on the unconditional generation of CAD programs which is not aligned with any user input. We argue that these unconditional generative models lack the true potential of generative CAD models for useful 3D solid modeling tasks. To align the generative CAD models with user intention, we propose a new type of neural network-based architecture that is conditioned on CAD images. The idea of image-to-CAD conversion or image-to-technical drawing conversion is of particular importance to the design community (Dori & Tombre, 1995; Dori & Wenyin, 1999; Nagasamy & Langrana, 1990; Gümeli et al., 2022; Majumdar & Seethalakshmy, 1997). Although several attempts have been made in the past, most of the approaches use heuristics and are application specific which makes them difficult to generalize. Our approach is largely motivated by the recent success of text-to-image and image-to-mesh literature (Alliegro et al., 2023; Tyszkiewicz et al., 2023; Xu et al., 2024a).

Here we focus on the problem of learning to generate CAD programs in terms of parametric command sequences based on CAD images. We propose a new generative model architecture, generative CAD (GenCAD), that can sequentially generate an entire CAD program aligned with an input CAD-image as shown in

[1]A sequence of parameterized CAD modeling operations is denoted as a CAD program

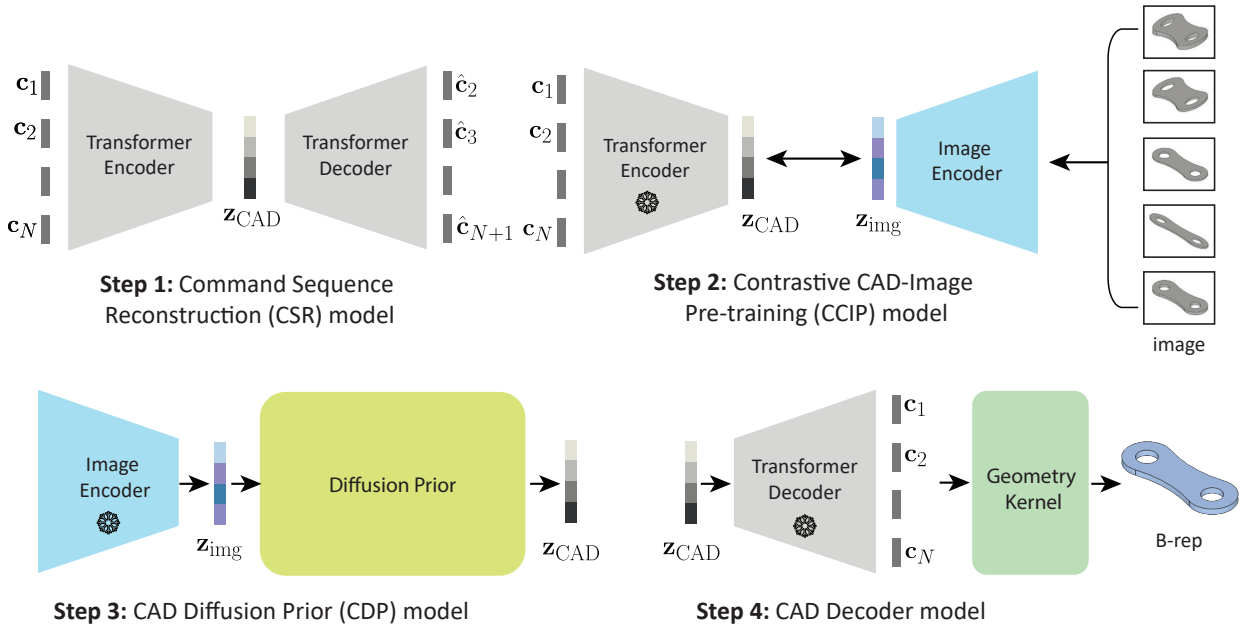

Figure 2: **GenCAD:** The proposed framework, GenCAD, consists of four steps: 1) a transformer-based encoder-decoder architecture is trained autoregressively to learn the latent representation of the vectorized CAD commands, 2) a contrastive-learning based model is used to learn the joint representation of the latent space of CAD command sequence and CAD image, 3) image conditional CAD generation can be achieved by sampling CAD latents from the diffusion model conditioned on image latents, and 4) using the trained transformer decoder to predict the CAD commands autoregressively. Note that ❄ denotes a frozen model that is not updated during training.

Figure 2. Our main idea is to learn the joint distribution of the latent representation of the CAD command sequences and the CAD images using a contrastive learning-based approach. Next, we develop a conditional latent diffusion model as a prior network that can generate CAD latent conditioned on input image latent. Finally, we present a transformer-based decoder model that can create the CAD program from the input CAD latent. Note that the final output of GenCAD is not merely a 3D solid model but a CAD program which is an entire sequence of parameterized CAD commands. This is critical because the CAD command sequence can be converted to B-rep models or other convenient representations such as mesh, point clouds, or voxels using any off-the-shelf geometry kernel. This trivial conversion allows GenCAD to be directly useful in generating conceptual 3D engineering design which has the potential to automate many complex 3D solid modeling tasks. Another major benefit of GenCAD is its ability to successfully perform image-based CAD program retrieval tasks, a major challenge in the engineering design community. We show that our framework can utilize the learned joint representation of CAD images and CAD command sequences to retrieve CAD programs using image input. Our contributions can be summarized as the following:

- We develop a transformer-based autoregressive model for representation learning of CAD sequences and show that our autoregressive model is more accurate in reconstructing CAD sequences than state-of-the-art models

- We propose and develop GenCAD–an image conditional generative model for CAD and show that conditional generation of CAD outperforms unconditional models in terms of diversity, fidelity, and statistical distance

- We show that contrastive learning enables image-based CAD retrieval which is more than 15× accurate in retrieval when compared to image-to-image search

2 RELATED WORK

2.1 CAD as a language modeling problem

The recent success of deep neural networks for language modeling tasks (Brown et al., 2020; Vaswani et al., 2017) has allowed many domains to formulate language-like problems for specific applications. Although learning design intent and developing intelligent CAD systems have been topics of interest since the 1990s (Ault, 1999; Ohsuga, 1989), treating CAD as a language modeling task is a somewhat recent approach.Ganin et al. (2021) showed that CAD can be converted to a language modeling task by considering protocol buffers (PB) for describing CAD sketch structures. The authors used PB to describe various sketch entities and constraints. Extending this idea to 3D is non-trivial due to the requirement of specific protocol buffers for 3D CAD commands. Similarly, another study by Para et al. (2021) focused on developing language-like problem formulation for CAD sketches that can handle various primitives and constraints. This approach is also limited to 2D sketches and scalability of the approach to 3D CAD is also non-trivial. Wu et al. (2021) also focused on converting CAD as a language modeling problem. This approach utilizes the transformer architecture for sequential 3D CAD generation and develops a simple vocabulary for CAD commands. This approach is capable of both sketch and 3D operation but a limited number of sketch and 3D operations. Note that the generative model proposed by Wu et al. (2021) is entirely unconditional and does not allow any user input during generation.

2.2 Datasets for CAD

Unlike other representations of 3D shapes such as meshes, point clouds, voxels, or implicits, there is a lack of large-scale datasets for CAD. There are only a few publicly available CAD datasets, among which the ABC dataset is the largest containing, 1 million CAD models obtained from online public repositories (Koch et al., 2019). This dataset contains B-reps but does not provide any design histories or labels. Later the DeepCAD dataset was developed by cleaning the ABC dataset (Wu et al., 2021). DeepCAD dataset provides design history by parsing each CAD design from the onshape repository. In addition to this, the Fusion 360 dataset provides both design history, assembly, and face segmentation although the dataset is quite small for learning generalizable models (Willis et al., 2021; 2022; Lambourne et al., 2021). Additionally, the MFCAD and MFCAD++ datasets provide machining feature recognition data in terms of B-reps (Cao et al., 2020; Colligan et al., 2022).

2.3 Generative models for CAD

In addition to learning specific tasks directly from B-rep data, recent studies have also developed generative models by directly synthesizing from B-rep data structure (Xu et al., 2022; Jayaraman et al., 2022; Xu et al., 2024b). Recent generative approaches mostly focus on utilizing transformer-based architecture for generating unconditional sketches (Xu et al., 2022; Para et al., 2021) or 3D CAD (Xu et al., 2022; Wu et al., 2021; Xu et al., 2024b; Jayaraman et al., 2022). In contrast to directly synthesizing the B-rep, the CAD generation process (CAD program) is of crucial importance due to its potential in the automation of many design tasks. Most importantly, the parameterized CAD operations or commands contain valuable design history. Wu et al. (2021) developed DeepCAD, a transformer-based latent generative adversarial network (l-GAN) model, that can generate unconditional CAD. Most of the generative CAD approaches only consider unconditional generation or specific data structures such as graph representation from boundary representation of CAD models. To align CAD model generation with user intention, a conditional input, such as an image, is needed–unlike the models developed in previous studies. To the best of our knowledge, there has been no generative model reported in the literature that can provide an entire CAD program from an image input.

3 METHOD

In contrast to the studies discussed in section 2, our main motivation for this work is to develop a learning-based approach that is scalable and can generate conditional 3D CAD. More specifically, we are interested in

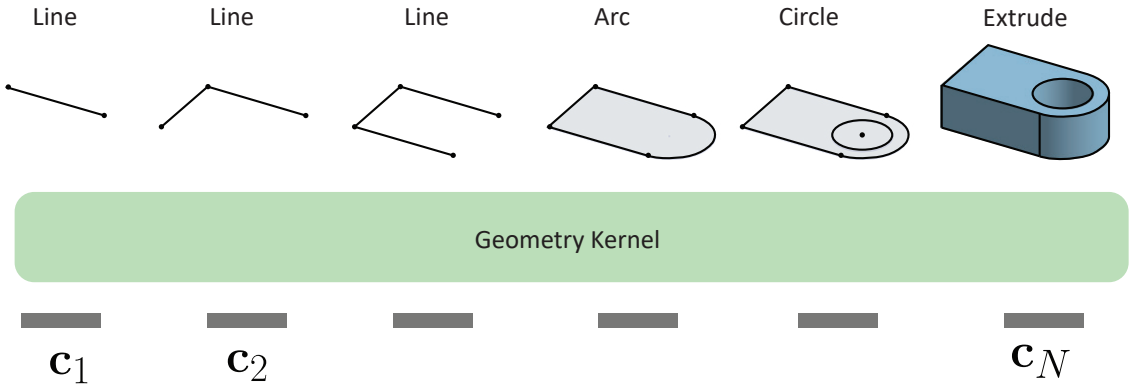

Figure 3: **CAD as a language modeling problem using geometry kernel:** Our main idea is based on the fact that real-world CAD design is sequential and a learning-based approach should capture the correlation in the sequential design from the large-scale dataset. This sequential approach converts the CAD problem into a language modeling problem where vector representation of CAD commands, $\mathbf{c}_i \in \mathbb{R}^d$, are fed into an off-the-shelf geometry modeling kernel to sequentially create the 3D solid geometry such as Boundary-representation (B-rep). In this toy example, the sequence is as follows: line $\rightarrow$ line $\rightarrow$ line $\rightarrow$ arc $\rightarrow$ circle $\rightarrow$ extrusion.

creating CAD programs that are conditioned on images. In the following, we describe our main contribution, a three-step framework, for generating image-conditional CAD models.

3.1 From natural language to CAD language

To represent the CAD program in a neural network-friendly representation, we opt for a natural language-like representation of a CAD program. The key idea is to represent a CAD model as a sequence of parameterized CAD commands that can be fed into a standard geometry kernel to create the 3D solid model. The goal of the parameterization is to allow the CAD command to have a fixed-dimensional vector representation. Each of the CAD commands is analogous to tokenization in natural language processing (NLP) tasks. Similarly, the fixed dimensional vector representation of the CAD command can be thought of as the embedding of a token in NLP tasks. A major difference between the tokenization of words and the tokenization of CAD commands is that the CAD commands are parameterized where the parameters ensure geometric consistency. Each CAD command represents the type of CAD operation that is being used and the associated parameters required to perform this CAD operation. An example of this mechanism is shown in Figure 3 where the vectorized CAD commands are fed into a geometry kernel to create the 3D solid geometry sequentially. Note that the type of the CAD operation is discrete and is similar to NLP tokens, but the parameter values are continuous.

Here we closely follow the CAD command representation criteria proposed by Wu et al. (2021). Specifically, each CAD command, $\mathbf{c}_i \in \mathbf{R}^{17}$, can be represented as $\mathbf{c}_i = (t_i, \mathbf{p}_i)$ where $\mathbf{t}_i \in \mathbb{R}$ represents the command type and $\mathbf{p}_i \in \mathbb{R}^{16}$ represents the parameters of each command. Note that each CAD command has a different number of parameters but they can be converted to a fixed dimensional vector by concatenating additional parameters and masking the values for unused parameters. Additionally, the parameters are carefully chosen so that they are sufficient for the geometry kernel to create the geometric entity. The types of CAD tokens used in this study are described in the following.

Special tokens: We define two special tokens $\langle$`SOL`$\rangle$ and $\langle$`EOS`$\rangle$ to represent the start of a loop of a sketch and end of sequence respectively. We draw motivation from the NLP literature which also defines special tokens to represent the start of a sentence or end of a sentence. These special tokens do not require any parameters and only indicate the state of the CAD geometry.

Sketch token: We define three tokens to represent a sketch; `Line` token, `Circle` token, and `Arc` token. In a geometry kernel, a line can be generated using only the end point coordinate, (x, y), if the start point is

known. As we are generating a CAD sketch sequentially, each sketch command provides us with the starting position for the next command. A circle can be generated based on the following three parameters; the center of the circle (x, y) and the radius of the circle, r. Similarly, an arc can be represented by the four parameters: end point of the arc, (x, y), sweep angle, α and directional flag of the arc, f which represents either a clockwise or counter-clockwise arc.

Extrusion token: One of the most complex yet widely used operations in CAD is the extrusion command. Representing extrusion requires keeping track of a sketch plane, extrude direction, and distance. To represent the current sketch plane we use these six parameters: three for the orientation of the current sketch plane (θ, ϕ, γ) and three for the origin of the current sketch plane (p_x, p_y, p_z). Additionally, a scale parameter is used to represent the scale of the sketch profile s. Extrude distance can be represented by two parameters (e_1, e_2) for each side of the sketch. Finally, to sequentially generate the solid, we need to either create a new solid body or join, cut, and intersect the extruded body from the existing solid body. So, we use a boolean parameter to indicate the type of operation of the body based on the extrude command. Finally, we use another parameter to represent whether it is a one-sided or two-sided extrude. As a result, the extrude command is represented using a total of 10 parameters.

3.2 Dataset

Here we use the DeepCAD dataset (Wu et al., 2021) to train and evaluate all of our models. This dataset is created based on the ABC dataset which is obtained from the publicly available human CAD designs from Onshape Inc. (Onshape, 2007). Note that the ABC dataset only contains the B-rep data and does not provide any design history. The DeepCAD dataset is created by parsing the design history of each CAD using the Onshape API. As CAD designs are complex and can be difficult to learn for sophisticated 3D shapes, this dataset is limited to sketch and extrude operations to make it more approachable for neural network-based models. Thus, this dataset does not contain other CAD operations, such as edge operations (fillets/chamfers), revolve, and mirror. The sketch operation is also limited to lines, circles, and arcs. After filtering out the ABC dataset based on these operations, the DeepCAD dataset contains $178,238$ CAD designs and is the largest CAD dataset with design history. Not all of these designs can render a 3D solid which makes it difficult to extract CAD images for our purpose. We filter this dataset based on 3D solid creation using an off-the-shelf geometry kernel, Open Cascade, and end up with a total dataset of $168,674$ CAD models with command sequences. Among these, we use $152,530$ for training, 8515 for validation, and 7629 for testing for all the models reported in this study. For CAD images, we create five different versions of each CAD model by changing the scale of the solid model in the x, y, and z axes. Each image is grayscale and has dimensions of $1 \times 448 \times 448$. Note that some of the scaling operations do not create a valid CAD and we omit those images. We end up with $840,947$ images obtained from the CAD dataset.

3.3 GenCAD: conditional generation of CAD programs

Using the CAD language modeling formulation in section 3.1, we propose **Gen**erative **CAD** (GenCAD), a four-step framework for conditional CAD program generation. First, we develop an autoregressive transformer encoder-decoder model, the Command Sequence Reconstruction (CSR) model, using CAD command sequences as input to learn the latent representation of the final 3D solid geometry model. Second, we develop a Contrastive CAD-Image Pretraining (CCIP) model with ResNet-based image encoder and contrastive loss to jointly learn the latent representation of the CAD image with the latent representation of the CAD command sequences from step 1. Third, we develop a CAD diffusion prior (CDP) model that can generate CAD latents, and optionally take image latents, from step 2, as input. Finally, the CAD latent samples generated from the CDP model are fed into a transformer-based decoder model that can generate the entire CAD command sequence autoregressively. To make our approach scalable for large-scale datasets and models, we utilize the pre-trained CAD encoder from step 1 during the training of the CCIP model instead of training from scratch. We also use the pre-trained CAD decoder model from step 1 for decoding the CAD latent from the CDP model. Once, we have the sequence of CAD commands, we can use any off-the-shelf geometry kernel to obtain the final 3D solid model. By combining these four steps, we show that our approach can generate valid CADs conditioned on image inputs which is shown in Figure 2.

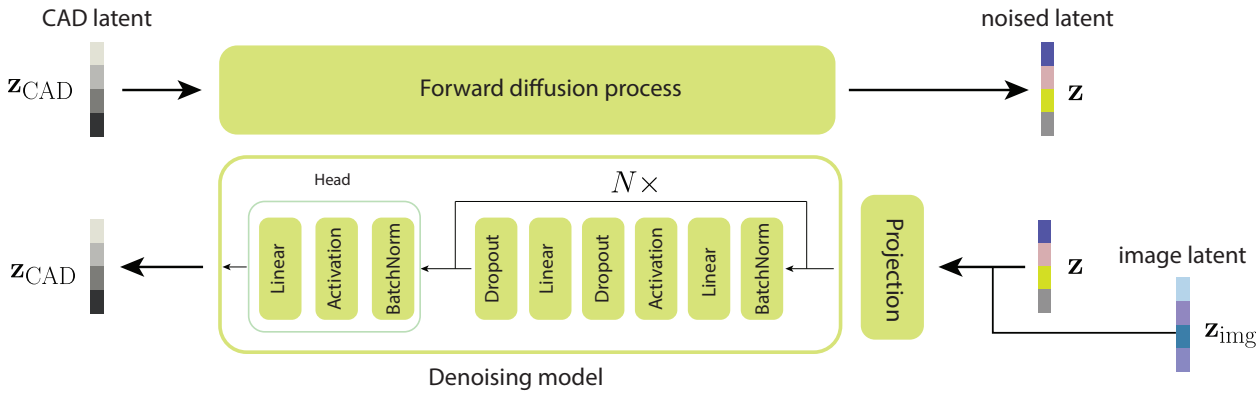

Figure 4: **CAD Diffusion Prior (CDP):** The diffusion prior takes CAD latent as well as optional image latent as input. The denoising model of the diffusion prior is based on MLP-ResNet architecture proposed by Gorishniy et al. (2021).

3.3.1 Command Sequence Reconstruction (CSR): Latent representation learning for CAD programs

The first step of our method is to learn an underlying latent representation of the CAD program or the command sequences. This is analogous to the pre-training of modern large language models (LLM) for downstream applications. We draw motivation from the recent success of large-scale generative pre-trained models (GPT) (Brown et al., 2020; Radford et al., 2019) and propose a transformer-based autoregressive model for learning the CAD commands. The main idea is to predict the next CAD command at each timestep of the sequence. This autoregressive training is done by observing the previous CAD commands within the sequence and masking all the future commands at each timestep. We use a standard transformer encoder and decoder network (Vaswani et al., 2017) with causal masking for both the CAD sequence encoder and the decoder. As the command parameters contain discrete and continuous values, we opt for learning the distribution in the latent space instead of the parameter space.

As mentioned in section 3.1, the parameters of each CAD command can be a combination of discrete and continuous values. To unify these types of parameters, we quantize their values into 256 levels and express them using 8-bit integers similar to Wu et al. (2021). Next, we organize each command vector onto a continuous space by adding an embedding layer before the encoder network. The output of the embedding layer is a d_z-dimensional vector which is fed into the transformer encoder. We use $d_z = 256$ for all experiments reported in this study. The transformer encoder produces a latent representation of the CAD commands, $\mathbf{z}_{\text{CAD},t} \in \mathbb{R}^{d_z}$ where $1 \leq t \leq N$ and N is the padded sequence length. As our goal is to train a generative model using the learned representation, we create a single latent vector, $\mathbf{z}_{\text{CAD}} \in \mathbb{R}^{d_z}$, for each sequence by average pooling the sequence of latent vectors from the CAD encoder output. Constant embeddings are learned during training to project this latent vector into again a sequence of latent vectors before feeding into the decoder network. Next, the latent vector, $\mathbf{z}_{\text{CAD}}$, goes through the transformer decoder and then another embedding layer to project the continuous values into the CAD command parameters. The embedding layer is a combination of both the CAD command embedding and the positional encoding of the sequence. Here we use the sinusoidal positional encoding from the original transformer implementation (Vaswani et al., 2017).

3.3.2 Contrastive CAD-Image Pre-training (CCIP): Joint representation learning of the CAD program and CAD-image

As our goal is to develop an image-conditional CAD generative model, the second step of our framework focuses on learning the joint distribution of the latent space of the CAD images and their corresponding CAD command sequences. For this purpose, we utilize the trained CAD encoder model from Step 1 to create CAD latents[2]. To generate image latents, $\mathbf{z}_{\text{img}} \in \mathbb{R}^{d_z}$, we use a ResNet-18 based architecture as the image

[2]CAD latents denote the embedding vectors obtained from encoding the CAD command sequences, whereas image latents represent the image embeddings obtained from encoding the images

encoder. We have experimented with larger ResNet architectures but found ResNet-18 to be sufficient for our dataset. First, we preprocess the input images by resizing them into 256×256 pixels, center cropping and then normalizing with $\mathcal{N}(0.5, 0.5)$. Then the images are passed through four layers with increasing dimensionality: $64, 128, 256$ and 512. Each of these layers contains two blocks of convolutional layers. We use several dropout layers in each encoder block of the image encoder model. The output of the image encoder is $512 \times 8 \times 8$ which is projected onto the d_z-dimensional space using a linear layer. During training, we keep the CAD encoder frozen. Note that the training dataset contains several scaled versions of the same CAD model. This image augmentation helps the model to learn the underlying representation of similar types of CAD geometries.

3.3.3 CAD Diffusion Prior (CDP): Conditional generative model for CAD

One of the most critical components of our framework is the diffusion prior model that generates CAD latents conditioned on image latents. In essence, the diffusion prior model is a conditional latent denoising diffusion probabilistic model (Ho et al., 2020). The forward diffusion process is standard where the CAD latent, $\mathbf{z}_{\text{CAD}}$, is destroyed with Gaussian noise sequentially to create the noised CAD latent, $\mathbf{z}$. During the denoising process, we concatenate the noised CAD latent, $\mathbf{z}$ with image latent, $\mathbf{z}_{\text{image}}$ and use a projection layer before feeding into the denoising model. For the denoising model, we use a ResNet-MLP architecture similar to Gorishniy et al. (2021) where several blocks of ResNet-MLP blocks are used with a normalization and linear head as shown in Figure 4. For completeness, we also develop deterministic prior in addition to the diffusion prior. The deterministic prior is straightforward and deterministically predicts $\mathbf{z}_{\text{CAD}}$ from $\mathbf{z}_{\text{image}}$. For this study, we use a simple ResNet-MLP architecture as our deterministic prior model. Note that the CDP model is a generative model while the deterministic prior model only maps $\mathbf{z}_{\text{image}}$ to $\mathbf{z}_{\text{CAD}}$ deterministically. The complete architecture of the CDP model is shown in Figure 4.

3.3.4 CAD Decoder model: Generating CAD sequences from latent representations

Finally, we need a decoder model to convert the generated CAD latents from step 3 into CAD command sequences. To this end, we propose to utilize the pre-trained decoder part of the CSR model from Step 1 instead of training a decoder model from scratch. The use of a pre-trained decoder model is particularly useful for scaling our approach to large-scale datasets. During inreference time, we would generate $\mathbf{z}_{\text{CAD}}$ from the diffusion prior model and feed it to the frozen CAD decoder model to generate the CAD command sequences $\mathbf{c}_1, \ldots, \mathbf{c}_N$.

4 EXPERIMENTS

4.1 Training details

Command Sequence Reconstruction (CSR) model: For both the encoder and the decoder models, we use the standard transformer encoder and decoder architecture respectively. Each transformer model consists of four self-attention layers and eight attention heads. The output of the final encoder layer goes through another *tanh* layer to create the latent vectors, $\mathbf{z}_{\text{CAD}}$. In total, there are 6.72 million trainable parameters in our model. To train the autoencoder model, we use the following loss:

$$\mathcal{L} = \sum_{i=1}^{N_c} \ell(\hat{t}_i, t_i) + \beta \sum_{i=1}^{N_c} \sum_{j=1}^{N_p} \ell(\hat{\mathbf{p}}_{ij}, \mathbf{p}_{ij}), \tag{1}$$

where N_c is the number of CAD commands, N_p is the number of parameters in each CAD command, t_i is the command type and $\mathbf{p}_{ij}$ is the parameters in each command. Recall that, in this study $N_c = 6$ and $N_p = 16$ due to the modeling of the CAD sequence in section 3.1. Here use cross-entropy loss as $\ell(\cdot, \cdot)$ in equation 1 with a weight parameter β that regulates the relative weight of the loss of the command type and their parameters, respectively. The loss combines the CAD operation type loss and the corresponding parameter value loss.

Contrastive CAD-Image Pre-training (CCIP) model: For the CCIP model, we follow the standard contrastive learning procedure and use the normalized temperature-scaled cross entropy loss (Chen et al., 2020). The main idea is to maximize the similarity between a CAD image, $\mathbf{I} \in \mathbb{R}^{1\times448\times448}$ and its corresponding CAD command sequence $\{\mathbf{c}_i\}_{i=1}^{B}$ in a given batch of B training data pairs $(\mathbf{I}, \{\mathbf{c}_i\})$. Given a batch of B example pairs, we have $2B$ data points in total for the contrastive prediction. By considering the positive pair as the only positive data sample, we consider the rest of the $2(B-1)$ samples as negative data pairs. If the similarity between the data pairs is expressed as cosine similarity, $\text{sim}(\mathbf{u}, \mathbf{v}) = \mathbf{u}^T\mathbf{v}/||\mathbf{u}||\mathbf{v}||$, then the contrastive loss can be defined as the following

$$\ell_{i,j} = -\log \frac{\exp(\text{sim}(\mathbf{z}_{\text{CAD},i}, \mathbf{z}_{\text{image},j})/\tau)}{\sum_{k=1}^{2B} \mathbb{1}_{[k\neq i]} \exp(\text{sim}(\mathbf{z}_{\text{CAD},i}, \mathbf{z}_{\text{image},k})/\tau)}, \tag{2}$$

where $\mathbb{1}[k \neq 1] \in \{0, 1\}$ is the indicator function and τ is the temperature parameter. The final contrastive loss is then calculated as $\mathcal{L} = \frac{1}{2B}\sum_{k=1}^{B}[\ell(2k-1, 2k) + \ell(2k, 2k-1)]$.

For our study, we experiment with two types of image encoders; ResNet-18 and ResNet-35 (He et al., 2016), while keeping the CAD encoder frozen. In the CCIP model, there are 28.22 million trainable parameters in total.

CAD Diffusion Prior (CDP) model: Traditionally diffusion models are used for image generation tasks and thus require a 2D U-net architecture for the denoising process. In our case, the diffusion model is trained on the vector representation of the latent space. We experiment with two different denoising models to replace the U-net architecture for this purpose: an MLP-based architecture and an MLP with residual connections, ResNet-MLP. We find the ResNet-MLP model to be more effective for the denoising model and use the model for all results reported in this paper.

4.2 Training

The CSR model is trained using a learning rate of 1×10^{-3} and a batch size of 512. We use the Adam optimizer with a scheduler to warm up the learning rate from zero to 1×10^{-3} for the first 2000 steps. The CSR model is trained for 1000 epochs. The CCIP is trained using the Adam optimizer with decoupled weight decay (Loshchilov & Hutter, 2017) with the ReduceLROnPlateau scheduler and an initial learning rate of 1×10^{-3}. The CCIP model is trained for 300 epoch. For the CDP model, both the diffusion and de-noising process use 500 timesteps in our model. The CDP model is trained for 1 million timesteps with a fixed learning rate of 1×10^{-5} where the gradient is accumulated every 2 steps. A maximum gradient norm of 1.0 is also used for training. All models are trained on an 80GB NVIDIA A100 GPU. Additional training details can be found in the Appendix.

4.3 Evaluation and baselines

4.3.1 Command Sequence Reconstruction (CSR) model:

As our transformer-based CSR model takes the sequence of CAD commands as input, we directly compare the CSR model against the DeepCAD autoencoder (Wu et al., 2021) to show how autoregressive training helps in obtaining a more robust latent space. We mainly evaluate the reconstruction performance of each CAD command sequence in terms of three metrics as proposed by Wu et al. (2021): the mean accuracy of the CAD command generation μ_{cmd}, the mean accuracy of the CAD parameter value generation μ_{param}, the mean chamfer distance, μ_{CD}, and the ratio of invalid shapes IR for the generated CAD against the ground truth. The mean CAD command generation accuracy and the mean command parameter generation accuracy for a set of reconstructed CAD models, $\mathcal{G}$, are defined as the following,

$$\begin{aligned}
\mu_{\text{cmd}} &= \frac{1}{|\mathcal{G}|}\sum_{k=1}^{|\mathcal{G}|}\frac{1}{N_c}\sum_{i=1}^{N_c}\mathbb{1}[t_i = \hat{t}_i], \\
\mu_{\text{param}} &= \frac{1}{|\mathcal{G}|}\sum_{k=1}^{|\mathcal{G}|}\frac{1}{\sum_{i=1}^{N_c}\mathbb{1}[t_i = \hat{t}_i]N_p}\sum_{i=1}^{N_c}\sum_{j=1}^{N_p}|\mathbf{p}_{i,j} - \hat{\mathbf{p}}_{i,j}| < \eta \cdot \mathbb{1}[t_i = \hat{t}_i],
\end{aligned} \tag{3}$$

where $\mathbb{I}[\cdot]$ is the indicator function, N_c is the total number of CAD commands in the sequence, t_i and $\hat{t}_i$ are ground truth and predicted CAD commands respectively, $\mathbf{p}_{i,j}$ and $\hat{\mathbf{p}}_{i,j}$ are the ground truth and predicted jth parameter value of the ith CAD command respectively. As mentioned in section 3.3.1, the parameter values are quantized into 8-bit integers.

As we cannot directly calculate the chamfer distance from the B-rep data, we convert each 3D solid into a point cloud using a fixed number of points, 2000 in this case. Once we have the point clouds of both the reconstructed, $\mathcal{G}$ and the ground truth $\mathcal{S}$ shapes, we can calculate the chamfer distance, CD. Finally, we can calculate the mean chamfer distance as the following, $\mu_{CD} = \frac{1}{|\mathcal{G}|}\sum_{k\in\mathcal{G}} CD(k, \mathcal{S})$. We also evaluate these metrics with respect to the sequence length of the CAD commands. Intuitively, higher-length sequences should be more challenging to reconstruct due to the complexity of the CAD commands.

4.3.2 Contrastive CAD-Image Pre-training (CCIP) model:

Due to the lack of existing literature on using a learning-based approach for image conditional CAD program generation, we evaluate the performance of the proposed CCIP model in terms of **image-based CAD retrieval** tasks. This is an important problem in the engineering design domain due to the wide applications of CAD retrieval from large CAD databases. The retrieval task is formulated as the following: given a CAD-image, $\mathbf{I}$, the model should identify the CAD program, $\{c_i\}$, that aligns with this image from a set of n_b CAD programs. Drawing motivation from image-based retrieval literature (Koh et al., 2024), we sample several batches of CAD examples, $n_b = 10, 128, 1024, 2048$, from the test dataset. Among these n_b examples, we randomly select one sample and utilize its CAD-image as the retrieval input to the CCIP model. Next, we find the CCIP similarity between the $n_b \times n_b$ examples and pick the CAD example corresponding to the highest cosine similarity. As a baseline, we compare our methods against the image-to-image search of CAD models. For this purpose, we utilize a ResNet-18 model with pre-trained ImageNet weights.

4.3.3 CAD Diffusion Prior (CDP) model:

To evaluate the image conditional generation, we compare our method against unconditional generation. For this purpose, we train an unconditional latent diffusion model from scratch as a baseline generative model. In addition to the unconditional model, we also compare the CDP model against other two baselines from literature–the latent GAN (l-GAN) model for CAD program generation Wu et al. (2021) and the SkexGen model for B-rep generation (Xu et al., 2022). For direct comparison with the baselines, we report the metrics proposed in Achlioptas et al. (2018) for point-cloud generative models. Specifically, we create two sets of point clouds: one for ground truth shapes $\mathcal{G}$ and one for the generated shapes $\mathcal{S}$. Next, we calculate the following: Coverage (COV) which measures the diversity of the shapes in $\mathcal{G}$, Minimum matching distance (MMD) which measures the fidelity of generated shapes, and Jensen-Shannon Divergence (JSD) which calculates the statistical distance metric between the two distributions. Details of these metrics can be found in the Appendix.

Finally, we evaluate the image-to-CAD conversion by calculating the FID score of the generated CAD programs. We create CAD programs from image inputs for the entire test data and extract their latent vectors as ground truth. Next, we create the CAD latents using conditional, unconditional, and deterministic MLP-Prior models. If the ground truth embeddings and the generated embeddings have the normal distributions $\mathcal{N}(\mu_{\mathcal{S}}, \Sigma_{\mathcal{S}})$ and $\mathcal{N}(\mu_{\mathcal{G}}, \Sigma_{\mathcal{G}})$ respectively then the FID score can be calculated as, $FID = ||\mu_{\mathcal{S}} - \mu_{\mathcal{G}}||_2^2 + \mathrm{tr}\left(\Sigma_{\mathcal{S}} + \Sigma_{\mathcal{G}} - 2(\Sigma_{\mathcal{S}}\Sigma_{\mathcal{G}})^{1/2}\right)$. Intuitively, the FID score calculates how aligned the generated CAD programs are when compared to the test CAD-images.

5 RESULTS

5.1 CAD sequence reconstruction

We directly compare our autoregressive CSR model with the DeepCAD transformer autoencoder model. As shown in Table 1, our model outperforms the DeepCAD transformer autoencoder in every metric: mean

command accuracy μ_{cmd}, mean parameter accuracy μ_{param}, mean chamfer distance μ_{CD}, and invalid ratio IR.

Table 1: Shape encoding performance of the proposed autoregressive encoder-decoder model. Arrows indicate whether higher is better or lower is better. Bold numbers represent the overall best performance.

Method	$\mu_{\text{cmd}}(\uparrow)$	$\mu_{\text{param}}(\uparrow)$	$\mu_{\text{CD}}(\downarrow)$	IR$(\downarrow)$
DeepCAD	99.36	97.59	0.783	3.44
GenCAD	**99.51**	**97.78**	**0.762**	**3.32**

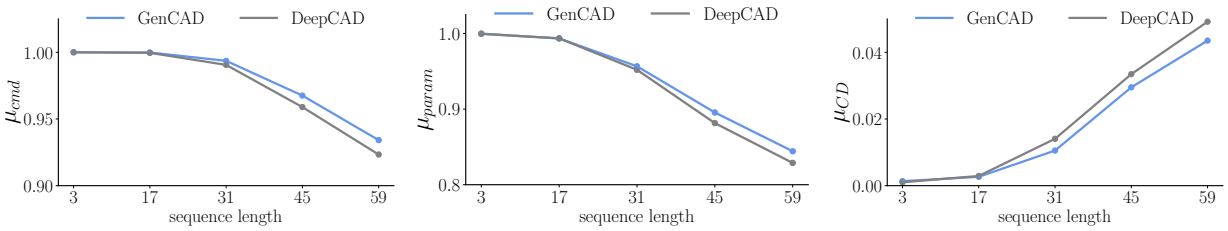

Figure 5: Performance of GenCAD-autoencoder corresponding to the length of CAD command sequences

Most importantly, our model is more accurate for higher lengths of CAD command sequences as shown in Figure 5. We anticipate that the high accuracy for longer sequence lengths is due to the autoregressive encoding of the latent space. By utilizing autoregressive training, we allow the model to encode more information about the CAD command sequence. Some qualitative examples of our model are shown in Figure 6. Our model can reconstruct the CAD programs more accurately for relatively complex shapes when compared to DeepCAD.

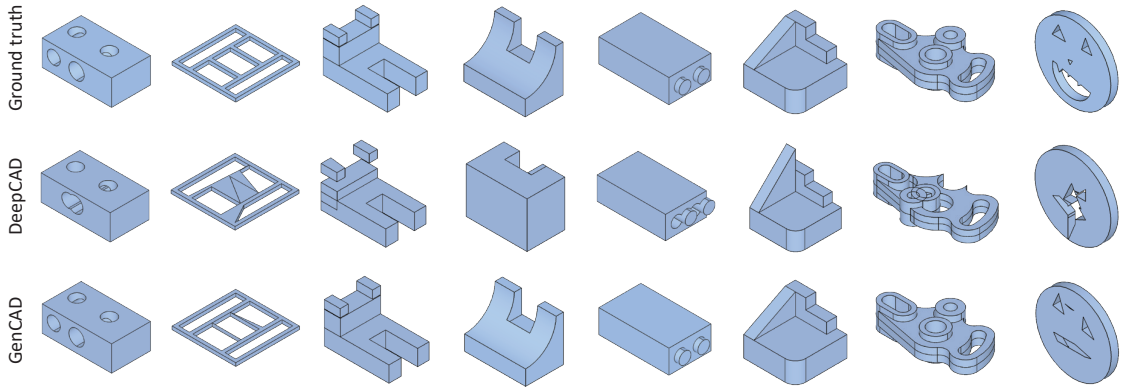

Figure 6: **3D CAD shape encoding performance:** We show the qualitative examples against DeepCAD which also uses a transformer-based model for CAD command reconstruction but does not use autoregressive training. Our method gives more accurate CAD command prediction for a test dataset that is close to the ground truth.

5.2 Image-based CAD retrieval

As there are no existing image-based CAD retrieval methods, we compare our model against an image-to-image similarity metric. Using a pre-trained ResNet-18 architecture, we extract the image latents and find the cosine similarity between the ground truth and the randomly sampled CAD images. We also provide random guesses as a simple baseline for comparison. As shown in Table 2, our model outperforms other

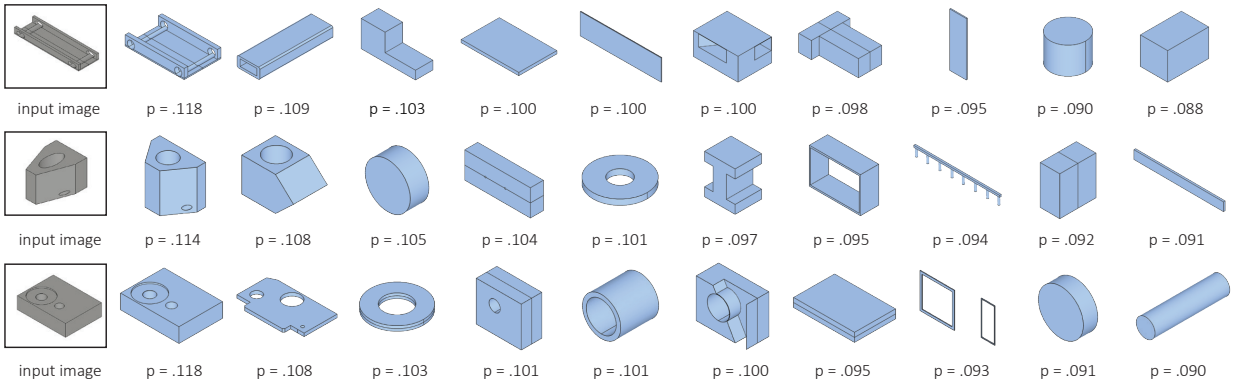

Figure 7: **Image-based CAD retrieval:** Qualitative performance of GenCAD-CCIP model is shown in terms of image-based CAD retrieval task for randomly sampled batches of 10 examples from the test dataset. The input CAD-image is shown on the left most column and the retrieved CAD programs are shown in the rest of the columns. We also provide the probability assigned by the CCIP model for each retrieval. The multimodal latent space helps in the accurate retrieval of CAD programs from input CAD images by assigning higher probabilities to similar CAD programs and lower probabilities to the rest of the CAD programs.

Table 2: Image-based CAD retrieval performance of GenCAD. $n_b = 10, 128, 1024$ and 2048 are repeated $1000, 10, 3$ and 3 times respectively. Arrows indicate whether higher is better or lower is better. Bold numbers represent the overall best performance.

Method	$R_{B=10}(\uparrow)$	$R_{B=128}(\uparrow)$	$R_{B=1024}(\uparrow)$	$R_{B=2048}(\uparrow)$
random	10.06 ± 0.19	N/A	N/A	N/A
ResNet-18 (pre-trained)	77.70 ± 0.00	19.26 ± 0.00	5.21 ± 0.45	3.91 ± 0.64
GenCAD	$\mathbf{98.49 \pm 3.93}$	$\mathbf{91.41 \pm 0.6}$	$\mathbf{70.28 \pm 0.79}$	$\mathbf{60.77 \pm 0.99}$

baselines by a large margin. Note that our model is most accurate for small batches of CAD programs and the accuracy drops when the number of CAD programs increases. Our model is also almost 62% accurate in retrieving CAD programs based on provided input images from a relatively large collection of CAD programs, i.e., 2048. Overall, the CCIP model is more than 15× accurate in retrieving the correct CAD program when compared to image-tp-image search. Some qualitative examples of retrieval results are shown in Figure 7 and Figure 8. As the CAD programs are randomly sampled within the batch, a diverse number of shapes are present which makes the retrieval challenging. It is easy to see that our model provides higher probability values to the correct CAD program based on the image input. Additionally, similar shapes are assigned relatively higher probabilities compared to shapes that are different. This is more clearly seen in Figure 8 where we retrieve the top 5 CAD programs from a collection of 2048 CAD programs based on the provided input image. Our model extracts shapes that are similar to the input image and also provides diversity within the shape.

5.3 Unconditional CAD generation

As mentioned in section 4.3, we compare our unconditional CAD generation model against the DeepCAD latent GAN (l-GAN) model and the SkexGen model. Below we provide the performance in terms of coverage (COV), Maximum Mean Discrepancy (MMD), and Jensen Shannon Divergence (JMD). Our unconditional model outperforms both the DeepCAD l-GAN model and the SkexGen model in terms of both COV and MMD metrics as shown in Table 3. A few qualitative examples of generated 3D shapes using the latent diffusion model are shown in Figure 9. Note that our model can generate complex CAD programs resulting in realistic 3D shapes.

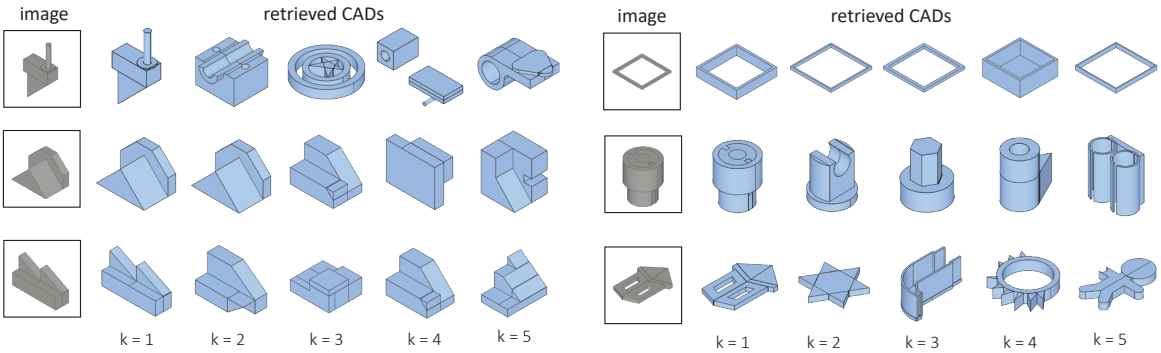

Figure 8: **Image-based CAD retrieval from large CAD database:** Qualitative performance of GenCAD-CCIP model is shown in terms of image-based retrieval of CAD programs from large database. Here, we retrieve top-k, where $k = 5$, CAD programs from the entire test dataset of CAD programs based on input images. The first column shows the input image and the rest of the columns show retrieved CAD based on the image. Row three shows that GenCAD can retrieve CAD programs when there are multiple duplicates of the same CAD program in the database.

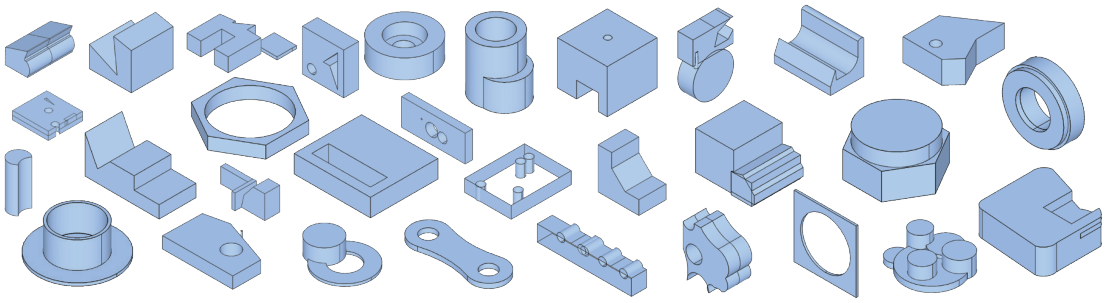

Figure 9: GenCAD (unconditional) 3D CAD generation.

5.4 Image conditional CAD generation

Using the trained diffusion prior, our model can predict CAD latents from image latents which are then fed into the pre-trained CAD decoder to create the CAD program. In Table 3, we show that our conditional diffusion generative model outperforms DeepCAD l-GAN, SkexGen and also our unconditional diffusion model in terms of COV, MMD and JSD metrics. Finally, to quantify the alignment of the generated samples compared to the input images we compare the FID score between the distribution of cad latents for the entire test data. As DeepCAD is the only available model that generates CAD program, we directly compare our results against DeepCAD model.

As an additional baseline, we also compare our results against the deterministic-prior model from 4.3.3. Our results indicate that the conditional diffusion prior model provides the lowest FID score when compared to

Table 3: 3D CAD (B-rep) shape generation performance. Arrows indicate whether higher is better or lower is better. Bold numbers represent the overall best performance.

Method	type	COV (↑)	MMD (↓)	JSD (↓)
DeepCAD (Wu et al., 2021)	unconditional	78.13	1.45	3.76
SkexGen (Xu et al., 2022)	unconditional	78.17	1.55	4.89
GenCAD	unconditional	78.27	1.44	3.94
GenCAD	conditional	**81.37**	**1.38**	**3.49**

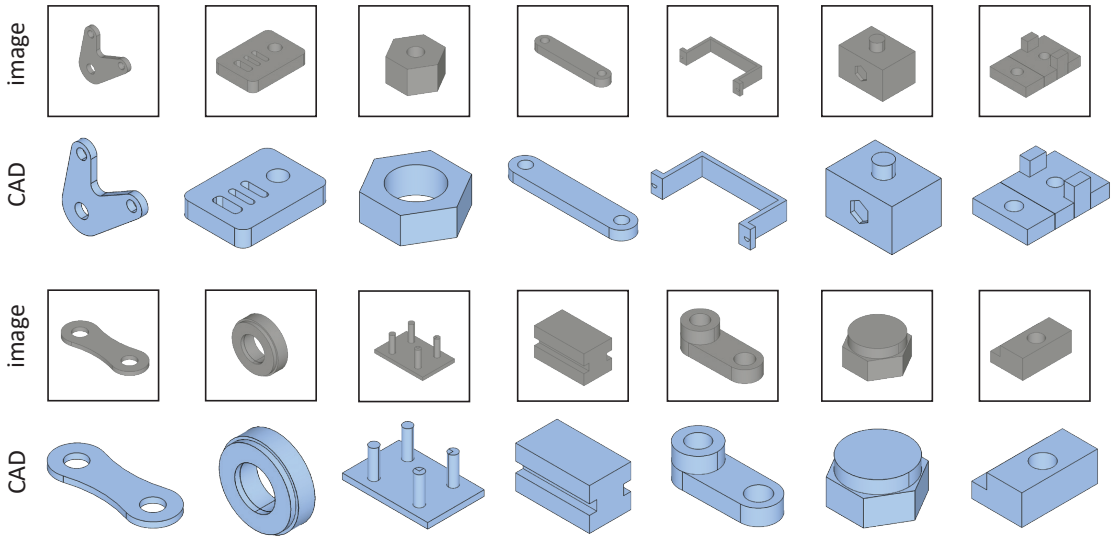

Figure 10: **GenCAD (Conditional) 3D CAD generation:** Qualitative examples of image-to-CAD generation capability of GenCAD using the conditional latent diffusion prior model.

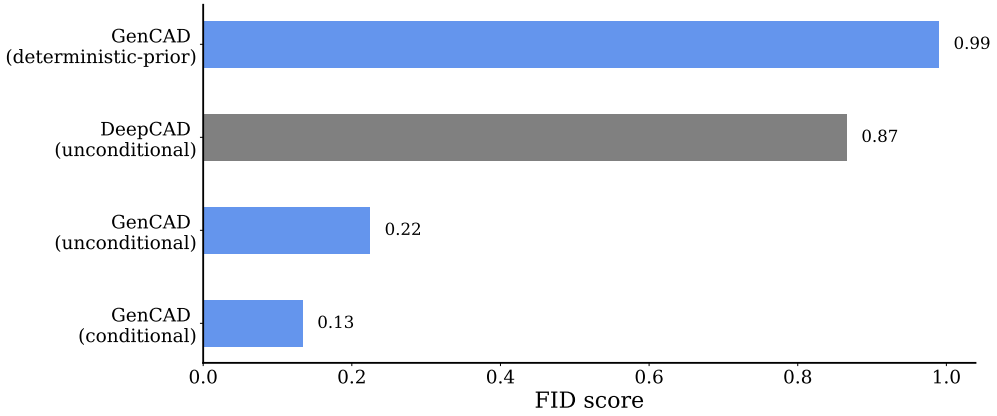

Figure 11: FID score of the generated dataset and the corresponding test dataset (lower is better).

the l-GAN, deterministic-prior, and unconditional diffusion model as shown in Figure 11. Note that the deterministic prior has the highest FID score and the unconditional diffusion model outperforms the l-GAN model. This result shows that the diffusion prior model can generate CAD programs that are close to the ground truth distribution. While the l-GAN model can generate unconditional CAD programs, these CAD models are not well aligned with the test dataset and the latent diffusion model is in-between the prior model and the l-GAN model. We also provide qualitative examples of our model in Figure 10.

Finally, we show the diversity of our diffusion prior model in Figure 12 where we sample multiple CAD programs for the same image input. Our model can generate similar CAD programs while providing useful variation in the parameter space.

6 LIMITATIONS

Although GenCAD shows impressive image-to-CAD performance, there are some limitations to the current version of this model. The CAD programs used in this study are comparatively simpler than industrial design tasks. The limited CAD vocabulary used in this study needs to be extended by including more sophisticated

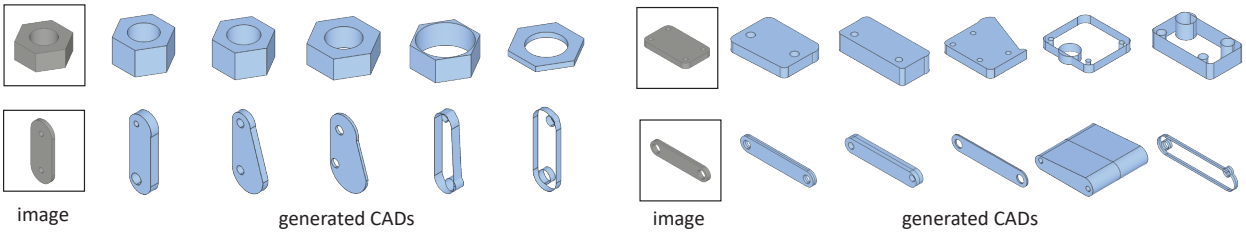

Figure 12: **Diversity of generated CAD:** Several samples drawn from the latent conditional model of the GenCAD are shown conditioned on the input image.

CAD tokens such as revolve operation, edge operation (e.g., fillets/chamfers), and other sketch operations. Additionally, GenCAD cannot guarantee the generation of a valid CAD similar to most of the generative CAD models reported in other studies. Drawing motivation from recent NLP literature, a step forward can be the addition of feedback from a CAD verification tool such as a geometry kernel. Building such a framework is non-trivial and will have the potential to create user-specified CAD programs. Finally, the images used in this study are mostly isometric and noise-free CAD-images. Future studies should focus on generating CAD programs from real-world object images with noisy backgrounds and non-isometric views.

7 CONCLUSION

CAD program generation is a challenging task for AI models that have the potential to automate industrial design pipelines. While unconditional generation of CAD programs has been studied, a more important problem is to generate grounded CAD programs based on user intention. We provided a first-of-its-kind AI model, GenCAD, that can generate CAD programs for complex 3D shapes both unconditionally and also based on image inputs. We argue that the image-aligned CAD program generation has the potential to be directly utilized for design task acceleration. We also show the application of GenCAD in image-based CAD program retrieval tasks. In summary, GenCAD solves two long-standing challenges in the 3D CAD community: it can retrieve a CAD program from large databases using an image and also generate the CAD program of the image. The current limitation of GenCAD is its dataset which includes relatively simpler CAD commands. In future studies, we aim to create a more sophisticated and real-world CAD dataset that can enhance the capabilities of GenCAD.

References

Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. Learning representations and generative models for 3d point clouds. In *International conference on machine learning*, pp. 40–49. PMLR, 2018.

Antonio Alliegro, Yawar Siddiqui, Tatiana Tommasi, and Matthias Nießner. Polydiff: Generating 3d polygonal meshes with diffusion models. *arXiv preprint arXiv:2312.11417*, 2023.

Holly K Ault. Using geometric constraints to capture design intent. *Journal for Geometry and Graphics*, 3 (1):39–45, 1999.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Weijuan Cao, Trevor Robinson, Yang Hua, Flavien Boussuge, Andrew R Colligan, and Wanbin Pan. Graph representation of 3d cad models for machining feature recognition with deep learning. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, volume 84003, pp. V11AT11A003. American Society of Mechanical Engineers, 2020.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pp. 1597–1607. PMLR, 2020.

Andrew R Colligan, Trevor T Robinson, Declan C Nolan, Yang Hua, and Weijuan Cao. Hierarchical cadnet: Learning from b-reps for machining feature recognition. *Computer-Aided Design*, 147:103226, 2022.

Dov Dori and Karl Tombre. From engineering drawings to 3d cad models: are we ready now? *Computer-Aided Design*, 27(4):243–254, 1995.

Dov Dori and Liu Wenyin. Automated cad conversion with the machine drawing understanding system: concepts, algorithms, and performance. *IEEE Transactions on Systems, Man, and Cybernetics-part A: systems and humans*, 29(4):411–416, 1999.

Yutong Feng, Yifan Feng, Haoxuan You, Xibin Zhao, and Yue Gao. Meshnet: Mesh neural network for 3d shape representation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pp. 8279–8286, 2019.

Yaroslav Ganin, Sergey Bartunov, Yujia Li, Ethan Keller, and Stefano Saliceti. Computer-aided design as language. *Advances in Neural Information Processing Systems*, 34:5885–5897, 2021.

Christophe Geuzaine and Jean-François Remacle. Gmsh: A 3-d finite element mesh generator with built-in pre-and post-processing facilities. *International journal for numerical methods in engineering*, 79(11): 1309–1331, 2009.

Yury Gorishniy, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. Revisiting deep learning models for tabular data. *Advances in Neural Information Processing Systems*, 34:18932–18943, 2021.

Can Gümeli, Angela Dai, and Matthias Nießner. Roca: Robust cad model retrieval and alignment from a single image. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4022–4031, 2022.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

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

Pradeep Kumar Jayaraman, Aditya Sanghi, Joseph G Lambourne, Karl DD Willis, Thomas Davies, Hooman Shayani, and Nigel Morris. Uv-net: Learning from boundary representations. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11703–11712, 2021.

Pradeep Kumar Jayaraman, Joseph G Lambourne, Nishkrit Desai, Karl DD Willis, Aditya Sanghi, and Nigel JW Morris. Solidgen: An autoregressive model for direct b-rep synthesis. *arXiv preprint arXiv:2203.13944*, 2022.

Heewoo Jun and Alex Nichol. Shap-e: Generating conditional 3d implicit functions. *arXiv preprint arXiv:2305.02463*, 2023.

Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4401–4410, 2019.

Sebastian Koch, Albert Matveev, Zhongshi Jiang, Francis Williams, Alexey Artemov, Evgeny Burnaev, Marc Alexa, Denis Zorin, and Daniele Panozzo. Abc: A big cad model dataset for geometric deep learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9601–9611, 2019.

Jing Yu Koh, Daniel Fried, and Russ R Salakhutdinov. Generating images with multimodal language models. *Advances in Neural Information Processing Systems*, 36, 2024.

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*, pp. 12773–12782, 2021.

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

Jharna Majumdar and AG Seethalakshmy. A cad model based system for object recognition. *Journal of Intelligent and Robotic Systems*, 18:351–365, 1997.

Vijay Nagasamy and Noshir A Langrana. Engineering drawing processing and vectorization system. *Computer Vision, Graphics, and Image Processing*, 49(3):379–397, 1990.

Setsuo Ohsuga. Toward intelligent cad systems. *Computer-aided design*, 21(5):315–337, 1989.

Onshape. *NVIDIA CUDA Compute Unified Device Architecture Programming Guide*. NVIDIA Corporation, 2007.

Wamiq Para, Shariq Bhat, Paul Guerrero, Tom Kelly, Niloy Mitra, Leonidas J Guibas, and Peter Wonka. Sketchgen: Generating constrained cad sketches. *Advances in Neural Information Processing Systems*, 34: 5077–5088, 2021.

Les A Piegl. Ten challenges in computer-aided design. *Computer-aided design*, 37(4):461–470, 2005.

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

Can Qin, Haoxuan You, Lichen Wang, C-C Jay Kuo, and Yun Fu. Pointdan: A multi-scale 3d domain adaption network for point cloud representation. *Advances in Neural Information Processing Systems*, 32, 2019.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *International conference on machine learning*, pp. 8821–8831. Pmlr, 2021.

Ali Razavi, Aaron Van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. *Advances in neural information processing systems*, 32, 2019.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10684–10695, 2022.

Chitwan Saharia, Jonathan Ho, William Chan, Tim Salimans, David J Fleet, and Mohammad Norouzi. Image super-resolution via iterative refinement. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(4):4713–4726, 2022.

Michał J Tyszkiewicz, Pascal Fua, and Eduard Trulls. Gecco: Geometrically-conditioned point diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 2128–2138, 2023.

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

Peng-Shuai Wang, Yang Liu, Yu-Xiao Guo, Chun-Yu Sun, and Xin Tong. O-cnn: Octree-based convolutional neural networks for 3d shape analysis. *ACM Transactions On Graphics (TOG)*, 36(4):1–11, 2017.

Kevin J Weiler. *Topological structures for geometric modeling (Boundary representation, manifold, radial edge structure)*. Rensselaer Polytechnic Institute, 1986.

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.

Karl DD Willis, Pradeep Kumar Jayaraman, Hang Chu, Yunsheng Tian, Yifei Li, Daniele Grandi, Aditya Sanghi, Linh Tran, Joseph G Lambourne, Armando Solar-Lezama, et al. Joinable: Learning bottom-up assembly of parametric cad joints. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15849–15860, 2022.

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*, pp. 6772–6782, 2021.

Jiale Xu, Weihao Cheng, Yiming Gao, Xintao Wang, Shenghua Gao, and Ying Shan. Instantmesh: Efficient 3d mesh generation from a single image with sparse-view large reconstruction models. *arXiv preprint arXiv:2404.07191*, 2024a.

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. *arXiv preprint arXiv:2207.04632*, 2022.

Xiang Xu, Joseph G Lambourne, Pradeep Kumar Jayaraman, Zhengqing Wang, Karl DD Willis, and Yasutaka Furukawa. Brepgen: A b-rep generative diffusion model with structured latent geometry. *arXiv preprint arXiv:2401.15563*, 2024b.