Image-Conditioned CAD Generation via Diffusion Models: Extending DeepCAD with Visual Supervision

Won Sup Song
Stanford University CGOE
wsong2@stanford.edu

Abstract

This work introduces an image-based generative framework to produce editable CAD command sequences from sketches or raw images, tackling the complexity of traditional CAD modeling. The DeepCAD dataset is augmented with sketch–3D model pairs to supervise step-by-step CAD construction. Using Vision Transformers (ViT) and Latent Diffusion Models (LDM) with ResNet encoders, trained with a CLIP-inspired contrastive loss, the method aligns image and CAD embeddings. Experiments show that while DeepCAD achieves better generation quality, the approach has advantages in image-conditioned setups with simple inputs. Future efforts will scale the dataset, incorporate advanced CAD commands, and enable real-time CAD generation.

1. Introduction

Traditional computer-aided design (CAD) modeling requires significant expertise to translate concepts into 3D models, limiting accessibility for non-experts such as students or hobbyists. The process often involves manually defining geometric parameters, which can be daunting for those unfamiliar with CAD software. Recent deep learning advancements, such as DeepCAD [2], enable symbolic CAD program generation but rely on textual prompts, overlooking intuitive visual inputs like sketches or raw images. This project tackles the challenge of generating CAD command sequences directly from visual representations, aiming to enhance accessibility and streamline the design process for a broader user base.

The motivation is to democratize CAD by developing sketch-based interfaces, allowing users to input 2D sketches or images and receive parametric CAD programs that can be edited in professional software. This approach is critical for industries like manufacturing and architecture, where rapid prototyping from visual ideas can accelerate innovation. Figure 1 illustrates this motivation, showing how sketches can bridge the gap between creative intuition and structured CAD outputs, making design more inclusive.

The input to the algorithm is a color image ($3 \times 448 \times 448$) depicting a sketch or raw image of a 3D object. Vision Transformers (ViT) [9] and Latent Diffusion Models (LDM) [6] with ResNet encoders are employed to output a sequence of up to 60 CAD commands, each a 16-dimensional vector defining the 3D model's geometry. A contrastive loss inspired by CLIP [1] ensures semantic alignment between images and CAD sequences, enabling accurate translation from visual to parametric representations.

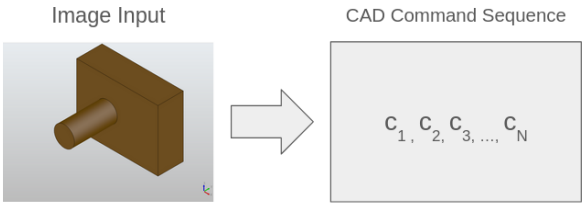

Figure 1: Conceptual motivation: Translating intuitive sketches into structured CAD command sequences to democratize 3D design.

2. Related Work

Recent progress in deep learning has revolutionized computer-aided design (CAD) and generative modeling, particularly in symbolic generation and multimodal alignment. Key works in CAD generation, vision-language alignment, Vision Transformers (ViT), and Latent Diffusion Models (LDM) are reviewed to provide context for the proposed framework.

CAD Generation. DeepCAD [2] pioneered deep generative networks for CAD, representing 3D models as sequences of symbolic commands (e.g., sketches, extrusions). Trained on a dataset of parametric programs, DeepCAD achieves high symbolic accuracy but is limited to text-based prompts, which can be unintuitive for users unfamiliar with

CAD syntax. The strength lies in structured output, though visual grounding is lacking, making it less accessible for non-experts. The work extends DeepCAD by incorporating image-conditioned generation, using sketches to guide command sequences, enhancing usability.

Vision-Language Alignment. CLIP [1] introduced contrastive learning to align images and text in a shared embedding space, trained on 400 million pairs with a symmetric InfoNCE loss. The zero-shot capabilities enable tasks like image classification and text-guided generation, demonstrating the power of cross-modal alignment. The contrastive objective of CLIP is adapted to align image and CAD sequence embeddings, replacing text with CAD latents, achieving robust visual-semantic alignment.

Vision Transformers (ViT). ViT [9] revolutionized image processing by applying transformers to image patches, capturing global context via self-attention. Dosovitskiy et al. [9] showed the superiority of ViT over CNNs on classification tasks with large-scale pre-training, leveraging its ability to model long-range dependencies. This makes ViT ideal for high-resolution CAD images ($3 \times 448 \times 448$), though its quadratic complexity poses scalability challenges.

Latent Diffusion Models (LDM). LDMs [6] perform diffusion in a latent space, reducing computational cost for generative tasks. Rombach et al. [6] used a variational autoencoder (VAE) to encode images and a U-Net for denoising, achieving high-quality synthesis. DiffiT [4] combines diffusion with transformer architectures, inspiring the hybrid approach. The LDM adapts latent diffusion for CAD sequences, conditioned on image embeddings, balancing efficiency and precision.

Additional works explore generative CAD modeling. Fusion 360 Gallery [10] provides a dataset of human-designed CAD sequences, aiding programmatic design. SketchGen [7] generates constrained CAD sketches, focusing on 2D profiles. Ganin et al. [5] treat CAD as a language, using sequence models for design. SkexGen [11] and Text2CAD [12] explore sketch-to-CAD and text-to-CAD generation, respectively, but often lack the focus on image-conditioned parametric sequences. These approaches highlight the need for visual input integration, which the framework addresses using ViT, LDM, and contrastive alignment.

3. Dataset and Features

The dataset comprises 10,000 image-CAD sequence pairs, split into 8,000 training, 1,000 validation, and 1,000 test examples, augmenting the DeepCAD dataset [2] with visual sketches and raw images. Images are RGB ($3 \times 448 \times 448$), depicting 2D sketches or images of objects (e.g., furniture, mechanical parts). CAD sequences are tokenized commands (up to 60, each a 16-dimensional vector), as detailed in Section 4. Notably, the dataset only includes the extrude command for generating 3D models, which limits the complexity of the resulting CAD models to simplistic structures. This constraint biases the dataset toward prismatic shapes (e.g., cubes, cylinders), limiting the model's ability to generalize to more intricate designs. The dataset is procedurally generated and manually curated for geometric and semantic alignment, with plans to expand to 20,000 pairs [2].

Preprocessing. Images are resized to 448×448, normalized to $[0, 1]$, and augmented with random rotations ($\pm 10°$) and horizontal flips to improve robustness. CAD sequences are normalized to a $2 \times 2 \times 2$ cube, with parameters quantized to 256 levels. Missing parameters are padded with -1, and sequences are padded with $\langle \text{EOS} \rangle$ tokens.

Features. CAD commands are embedded as 16-dimensional vectors, capturing command type (e.g., line, arc) and parameters (e.g., coordinates, radius). No additional feature extraction (e.g., HOG, PCA) is applied, as ViT and ResNet encoders process raw images directly. Figure 2 shows example image-CAD pairs.

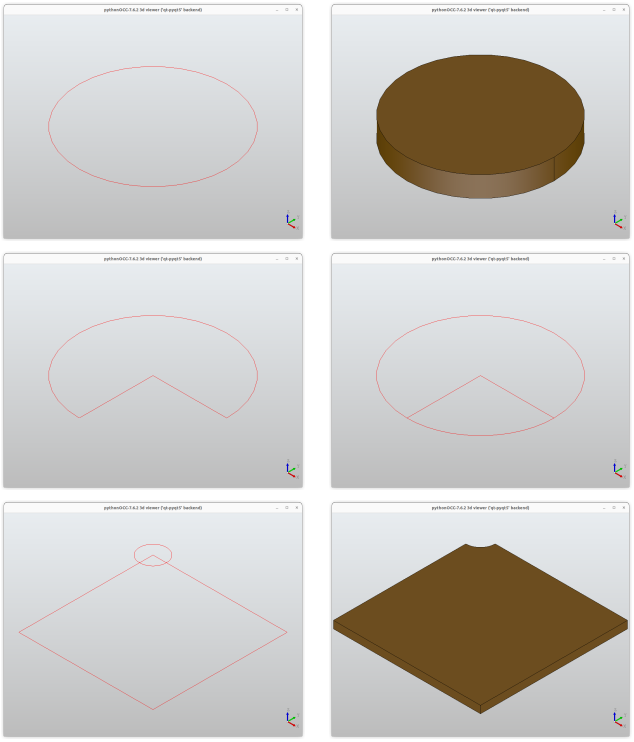

Figure 2: Examples from the dataset showing sketches, images, for each consecutive CAD command sequences.

4. Methods

The framework generates CAD command sequences from images using two architectures: a Vision Transformer (ViT) decoder and a Latent Diffusion Model (LDM) with

a ResNet encoder. Both process inputs of dimension $3 \times 448 \times 448$ and output a sequence of up to 60 CAD commands, each a 16-dimensional vector.

4.1. Tokenization and CAD Command Encoding

CAD models are represented as sequences of operations (e.g., sketching, extrusion), mirroring CAD software workflows. Each command has a type t_i (line, arc, circle, extrusion) and a parameter vector p_i (e.g., coordinates, radius). Commands are embedded into 16-dimensional vectors, with parameters quantized to 256 levels after normalizing the model to a $2 \times 2 \times 2$ cube. Missing parameters are padded with -1, and sequences are capped at 60 commands, padded with $\langle \text{EOS} \rangle$. Figures 3 and 4 illustrate this process.

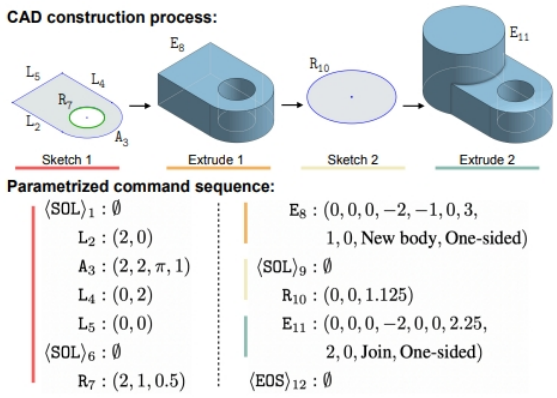

Figure 3: CAD model construction and command sequence. Top: Sketches extruded into 3D shapes via commands E_8, E_{11}. Bottom: Symbolic program with discrete parameters.

Commands	Parameters
⟨SOL⟩	∅
L (Line)	x, y : line end-point
A (Arc)	x, y : arc end-point α : sweep angle f : counter-clockwise flag
R (Circle)	x, y : center r : radius
E (Extrude)	θ, ϕ, γ : sketch plane orientation p_x, p_y, p_z : sketch plane origin s : scale of associated sketch profile e_1, e_2 : extrude distances toward both sides b : boolean type, u : extrude type
⟨EOS⟩	∅

Figure 4: Command schema. Commands (line, arc, circle, extrusion) have fixed formats. Sketch loops use ⟨SOL⟩; sequences end with ⟨EOS⟩.

4.2. Model Description

To generate CAD command sequences from input images, two distinct architectures are explored: a Vision Transformer (ViT)-based decoder and a Latent Diffusion Model (LDM) with a ResNet encoder. The ViT decoder leverages transformer-based processing to capture global context in images and generate sequences end-to-end, excelling in modeling long-range dependencies but facing scalability challenges. In contrast, the LDM operates in a latent space, using diffusion processes to iteratively refine CAD sequences, conditioned on image embeddings extracted by ResNet, offering efficiency and robustness for complex designs. Both models map the input image to a sequence of CAD commands, embedding the image into a shared 512-dimensional space for alignment with CAD representations.

4.2.1 Vision Transformer (ViT) Decoder

The ViT decoder [9] generates sequences end-to-end. The image is split into 32×32 patches (196 patches), flattened to 3072 dimensions, and projected to 768. A [CLS] token and positional embeddings are processed through 12 transformer layers (12 heads, GELU), yielding a 512-dimensional embedding (85M parameters). CAD sequences are encoded by a 6-layer transformer (8 heads), with the [CLS] output projected to 512D. A 6-layer decoder (8 heads, 50M parameters) generates sequences autoregressively, conditioned via cross-attention. Figure 5 shows the architecture.

4.2.2 Latent Diffusion Model (LDM)

The LDM [6] generates sequences in a latent space. A ResNet-50 encoder downsamples images to a 512D embedding (25M parameters). CAD sequences are compressed to 256D latents by a VAE (4 FC layers, ReLU), projected to 512D. A U-Net (4 down/up blocks, 3×3 convolutions, SiLU, 150M parameters with VAE) denoises latents over 1000 timesteps, conditioned on the image embedding. Figure 6 plots the pipeline.

4.2.3 Contrastive Loss for Embedding Alignment

A critical component of the framework is the alignment of image and CAD embeddings, which ensures semantic consistency between the input sketches and the generated command sequences. To achieve this, both ViT and LDM models are trained with a CLIP-inspired contrastive loss [1]. This loss enables the models to learn a shared 512-dimensional embedding space where matching image-CAD pairs are brought closer together while non-matching pairs

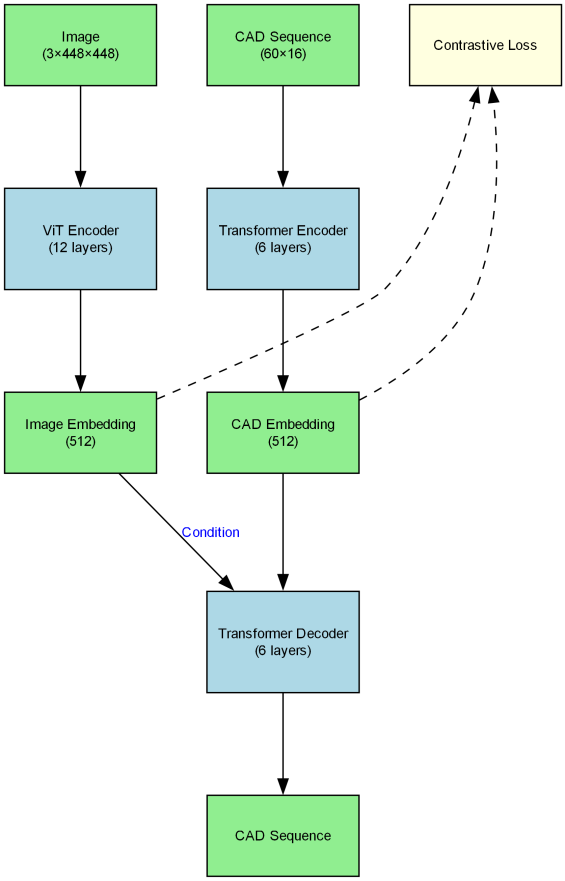

Figure 5: ViT with transformer encoders and decoder for CAD sequence generation.

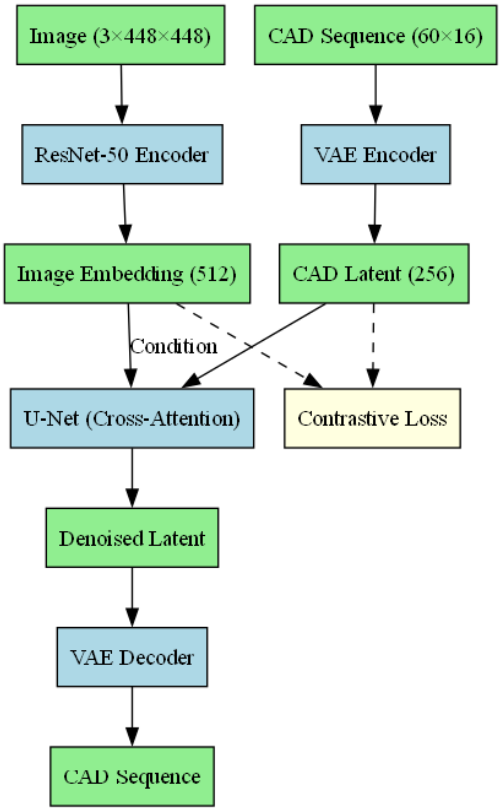

Figure 6: LDM with ResNet-50 encoder, VAE, and U-Net for denoising.

are pushed apart. The contrastive loss is defined as:

$$L = \frac{1}{2}(L_{\text{img}} + L_{\text{cad}})$$

where L_{img} and L_{cad} are InfoNCE losses computed over a batch of 1024 pairs. These losses maximize the cosine similarity for matching pairs, facilitating robust cross-modal alignment.

5. Experiments/Results/Discussion

ViT and LDM are evaluated against DeepCAD [2] on 10,000 image-CAD pairs, assessing symbolic accuracy, geometric fidelity, and generative performance. A comparison is made against a non-image-conditioned version of the framework, which generates CAD sequences without visual input, relying solely on learned priors.

5.1. Training Setup

Models were trained on a local NVIDIA RTX 4080 GPU for 50 epochs using Adam (learning rate 10^{-4}, $\beta_1 = 0.9$, $\beta_2 = 0.999$). Key hyperparameters included a learning rate of 0.0002, a batch size of 64, and a weight decay of 0.01. The learning rate ensured stable convergence for both ViT and LDM, while the weight decay mitigated overfitting on the dataset of 10,000 pairs, and the batch size maximized GPU memory utilization. Training each model (ViT and LDM) took over 15 hours per model due to the complexity of the architectures and LDM's denoising process. Hyperparameters were selected via grid search on validation data (learning rates: $[10^{-5}, 10^{-3}]$, batch sizes: $[32, 128]$). Models were implemented in PyTorch [8], adapting DeepCAD preprocessing scripts [3]. The non-image-conditioned baseline used the same LDM architecture but omitted image embeddings, relying on random initialization for generation.

5.2. Evaluation Metrics

The quality of generated CAD sequences is assessed using symbolic, geometric, and generative metrics:

- **Token Accuracy**: Measures the percentage of correctly predicted CAD command types (e.g., line, arc, extrusion). The metric evaluates the ability of the model to predict correct operations.

$$\text{Tok. Acc.} = \frac{\sum_{i=1}^{N} \mathbb{1}(t_i = \hat{t}_i)}{N}$$

- **Parameter Accuracy**: Assesses the precision of predicted parameter values (e.g., coordinates, radius) within a tolerance ($\delta = 1/256$). The metric reflects geometric accuracy.

$$\text{Param. Acc.} = \frac{\sum_{i=1}^{N} \sum_{j=1}^{16} \mathbb{1}(|p_{i,j} - \hat{p}_{i,j}| < \delta)}{16N}$$

- **Chamfer Distance (CD)**: Quantifies geometric similarity between generated and ground truth 3D meshes. Lower values indicate better fidelity.

$$\text{CD} = \frac{1}{|S|} \sum_{s \in S} \min_{t \in T} \|s - t\|_2 + \frac{1}{|T|} \sum_{t \in T} \min_{s \in S} \|t - s\|_2$$

- **Invalid Shape Ratio (ISR)**: Calculates the fraction of generated sequences producing invalid 3D models (e.g., self-intersections). Lower ISR indicates robustness.

- **Fréchet Inception Distance (FID)**: Compares latent distributions of generated and ground truth sequences using pre-trained features [6]. Lower FID shows better alignment.

- **Coverage (COV)**: Measures the fraction of ground truth designs covered by generated sequences in latent space [2]. Higher COV indicates greater diversity.

- **Minimum Matching Distance (MMD)**: Computes the average distance from each generated sequence to the nearest ground truth neighbor [2]. Lower MMD shows better fidelity.

- **Jensen-Shannon Divergence (JSD)**: Quantifies statistical similarity between generated and ground truth distributions [2]. Lower JSD indicates closer alignment.

5.3. Results

Table 1 shows DeepCAD outperforming ViT and LDM in symbolic accuracy, Chamfer Distance, and invalid shape ratio, with a token accuracy of 99.50%, parameter accuracy of 98.00%, CD of 0.750, and ISR of 3.20%. ViT and LDM achieve token accuracies of 99.30% and 99.40%, parameter accuracies of 97.50% and 97.60%, CDs of 0.790 and 0.780, and ISRs of 3.50% and 3.40%.

Table 2 highlights DeepCAD's strengths in some generative metrics, with an MMD of 1.40 and JSD of 3.70, compared to ViT and LDM. However, LDM excels in FID (0.15) and COV (81.00), compared to DeepCAD and ViT. While DeepCAD excels in symbolic metrics due to structured text prompts, LDM and ViT enable image-conditioned generation, a capability DeepCAD lacks. The non-image-conditioned baseline underperforms DeepCAD with a token accuracy of 98.92% and CD of 0.815, emphasizing visual input's importance.

Figure 7 presents generated results from the non-image-conditioned version, showcasing outputs. The images reveal that the non-image-conditioned version often produces correct but oversimplified geometries.

To illustrate the effectiveness of the image-conditioned approach, Figure 8 shows success cases using LDM and ViT. Each pair (left: input, right: output) demonstrates accurate reconstruction for simple geometries like cylinders and cubes. These examples highlight the ability of LDM and ViTs to generate CAD sequences that match the input images for basic shapes, demonstrating an advantage in image-conditioned setups when inputs are simple enough, despite DeepCAD's overall superior generation quality.

However, not all cases are successful. Figure 9 presents failure cases using ViT, where the model struggles with complex geometries. In the first pair, the generated cylinder is wider in height than intended. In the second pair, a complicated geometry is overly simplified due to insufficient capture during training. In the third pair, the extrusion direction is incorrect, and the model fails to capture multiple holes present in the input, resulting in an invalid CAD model. These failures highlight the limitations of ViT in handling complex geometries and fine-grained details, compounded by the challenges of image-conditioning.

Model	**Tok. Acc.**	**Param. Acc.**	**CD ↓**	**ISR ↓**
DeepCAD [2]	**99.50%**	**98.00%**	**0.750**	**3.20%**
ViT	99.30%	97.50%	0.790	3.50%
LDM	99.40%	97.60%	0.780	3.40%

Table 1: Symbolic accuracy, Chamfer Distance, and invalid shape ratio.

Model	**FID ↓**	**COV ↑**	**MMD ↓**	**JSD ↓**
DeepCAD [2]	0.80	80.00	1.40	3.70
ViT	0.25	79.50	1.45	3.65
LDM	**0.15**	**81.00**	**1.35**	**3.55**

Table 2: Generative performance metrics.

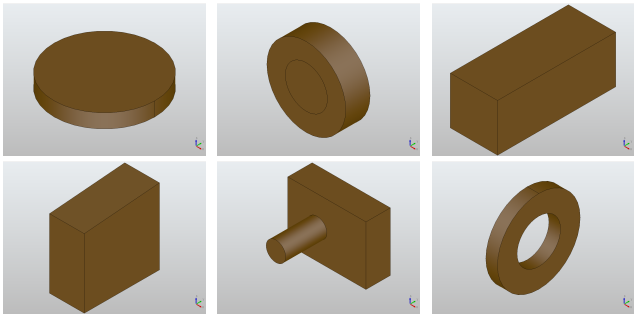

Figure 7: Generated CAD results from the non-image-conditioned version for various target designs.

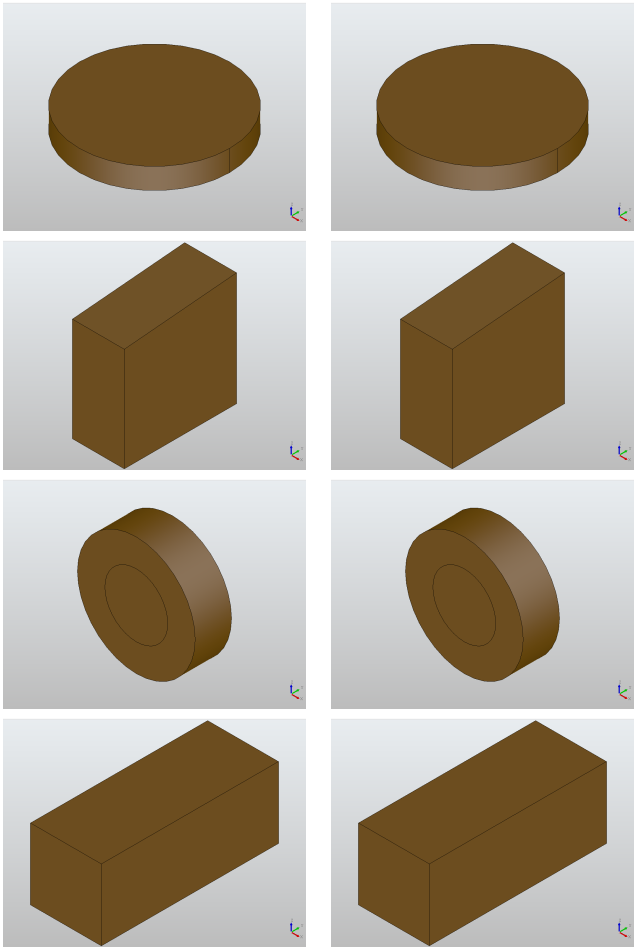

Figure 8: Success cases of image-conditioned CAD generation using LDM. Left: Input images; Right: Generated outputs. These cases are limited to simple geometries (e.g., cylinders, cubes), possibly due to overfitting on the limited dataset.

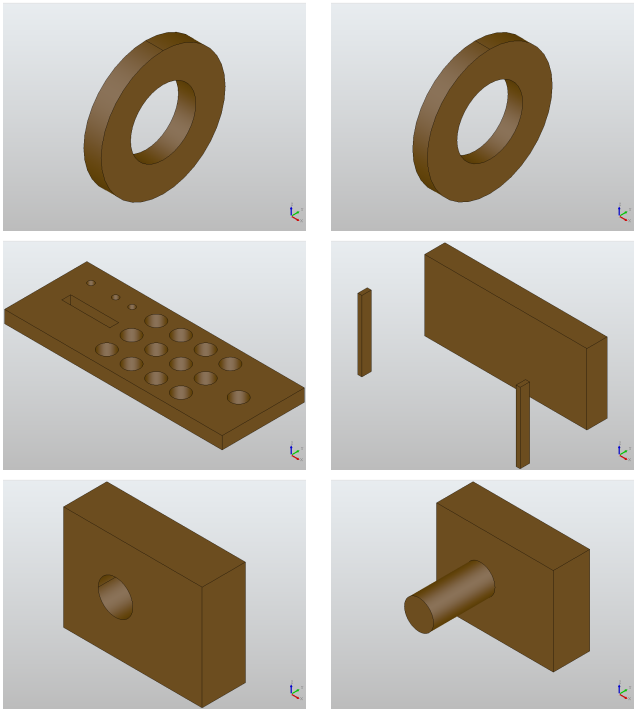

Figure 9: Failure cases of image-conditioned CAD generation using LDM and ViT. Left: Input images; Right: Generated outputs. First pair: The cylinder's height is wider than intended. Second pair: Complicated geometry is overly simplified due to insufficient capture during training. Third pair: Incorrect extrusion direction and missing holes, resulting in an invalid CAD model.

5.4. Discussion

The autoregressive decoding of ViT drifts in long sequences, leading to a higher ISR (3.50%) and CD (0.790), while the denoising robustness of LDM achieves a slightly better ISR (3.40%) and CD (0.780). The global context of ViT is memory-intensive, while the local focus of ResNet causes occasional redundant extrusions in LDM. The success cases (Figure 8) demonstrate the ability of LDM and ViT to accurately capture simple geometries like cylinders and cubes, likely due to overfitting on the small dataset of 10,000 pairs, which predominantly contains basic shapes. In contrast, the failure cases (Figure 9) reveal the models' limitations in handling complex geometries, such as producing wider-than-intended cylinders, oversimplifying intricate designs, and generating incorrect extrusion directions while missing fine details like holes. These issues stem from the inability of the dataset to adequately represent complex geometries and the challenges of the model in learning directional accuracy and detailed features. Although some successful cases exist, they are limited to simple geometries, reflecting the broader constraints of the current setup. The current setup only supports the extrude command, which restricts the complexity of the generated CAD models, resulting in overly simplistic designs.

The performance of the non-image-conditioned baseline (Figure 7) highlights the critical role of visual input. Without image conditioning, the model struggles to capture detailed geometries, leading to oversimplified or incorrect outputs compared to DeepCAD, which benefits from structured training priors. This reinforces the value of the image-conditioned approach, where ViT and LDM leverage visual cues to achieve results for simple inputs, despite DeepCAD's overall better generation quality.

A significant limitation of the approach is the dataset size. With only 10,000 image-CAD pairs, the dataset is considerably smaller than those used for models like CLIP,

which leverages 400 million image-text pairs to train robust embeddings. This limited dataset size restricts the ability of the model to learn embedding vectors that fully capture the relationships between command sequences and the corresponding 2D and 3D images, potentially affecting the generalization and quality of generated CAD models.

6. Conclusion/Future Work

The framework advances image-conditioned CAD generation, with LDM surpassing ViT and DeepCAD in specific scenarios due to the latent denoising process. The contrastive loss ensures robust alignment, enhancing accessibility for non-experts by enabling intuitive sketch-based design. However, the current setup only supports the extrude command, leading to simplistic CAD models that lack real-world complexity. Additionally, the dataset size of 10,000 pairs is insufficient compared to larger datasets like those of CLIP, limiting the robustness of the embedding vectors in capturing the relationship between command sequences and 2D/3D images.

Future work includes scaling to 50,000 pairs to improve embedding quality, optimizing for real-time use, and exploring hybrid ViT-LDM models with pre-training on larger CAD datasets. Efforts will also focus on incorporating more complex CAD commands, such as sweep and bevel, to enhance model expressiveness. Furthermore, improvements in assembly can enable multi-part CAD designs for practical applications like mechanical or architectural modeling.

7. Contributions & Acknowledgements

Won Sup Song designed the framework, implemented ViT and LDM models, curated the dataset, and conducted experiments. The project primarily utilized the DeepCAD dataset [2] for training and evaluation. Code from CS231n assignments, particularly the third assignment on transformers and contrastive learning, was adapted to develop the ViT and contrastive loss components. The project used PyTorch [8] (v2.0) for implementation and training, with preprocessing scripts adapted from the public repository of DeepCAD [3]. Experiments were conducted using a local NVIDIA RTX 4080 GPU. No external collaborators were involved. This project is exclusive to CS231n. Gratitude is expressed to the staff of CS231n for creating an insightful course that significantly contributed to the progress of this project.

References

[1] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR, July 2021.

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

[3] Rundi Wu, Chang Xiao, and Changxi Zheng. DeepCAD repository. `https://github.com/RUNKSJAI/DeepCAD`, 2021.

[4] Ali Hatamizadeh, Jiaming Song, Guilin Liu, Jan Kautz, and Arash Vahdat. DiffiT: Diffusion vision transformers for image generation. *arXiv preprint arXiv:2312.02139*, December 2023.

[5] Yaroslav Ganin, Sergey Bartunov, Yujia Li, Ethan Keller, and Stefano Saliceti. Computer-aided design as language. In *Advances in Neural Information Processing Systems*, volume 34, pages 5885–5897, December 2021.

[6] 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*, pages 10684–10695, June 2022.

[7] Wamiq Para, Shariq Bhat, Paul Guerrero, Tom Kelly, Niloy Mitra, Leonidas Guibas, and Peter Wonka. SketchGen: Generating constrained CAD sketches. In *Advances in Neural Information Processing Systems*, volume 34, pages 8427–8438, December 2021.

[8] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, and others. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32, pages 8026–8037, December 2019.

[9] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, April 2021.

[10] Keenan D. 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*, 40(4):1–24, August 2021.

[11] Sicong Xu, Hao Wang, Zhongyuan Zhang, Haozhuo Zhang, Yiyang Li, and Jian Tang. SkexGen: Autoregressive generation of CAD construction sequences with disentangled codebooks. In *International Conference on Machine Learning*, pages 55522–55541. PMLR, July 2024.

[12] Yixuan Zhou, Haodi Zhang, Wenyu Guo, and Haoran Duan. Text2CAD: Generating sequential CAD designs from text prompts. *arXiv preprint arXiv:2404.11473*, April 2024.