GENCAD-SELF-REPAIRING: FEASIBILITY ENHANCEMENT FOR 3D CAD GENERATION

Enrique Flores Medina
Department of Mechanical Engineering
Massachusetts Institute of Technology
Cambridge, MA 02139
Email: enriquef@mit.edu

Chikaha Tsuji
Department of Mechanical Engineering
Massachusetts Institute of Technology
Cambridge, MA 02139
Email: chikaha@mit.edu

Harshit Gupta
Department of Mechanical Engineering
Massachusetts Institute of Technology
Cambridge, MA 02139
Email: harshit1@mit.edu

ABSTRACT

With the advent of Generative AI, research surrounding its use in the creation of 3D models has emerged, allowing for the generation of CAD files by processing CAD images. One such model is GenCAD [1], which integrates an autoregressive transformer-based architecture with a contrastive learning framework, generating CAD command sequences from input images.

However, GenCAD cannot guarantee the generation of a "feasible" Boundary Representation (B-rep). Therefore, we propose a self-repairing pipeline that improves the feasibility rate of GenCAD using regression and guided diffusion. Simultaneously, this pipeline aims to not downgrade the accuracy of the generated geometries.

By significantly improving the feasibility of generated CAD files, this framework enables more robust applications in industries such as manufacturing, architecture, and product design.

I. INTRODUCTION

3D Modeling is a fundamental part of the complete product design process that allows for easy visualization of part shape, size, and configuration to be manufactured. However, the CAD modeling process is manual, intensive, and takes a lot of time and effort. To address this problem, and attenuate the time to design 3D models of products and parts, Generative AI has been particularly useful. Generative models have recently shown a huge improvement in this field wherein models utilize just the "image" and "text" embeddings to create fully functional feasible B-reps, satisfying design constraints and manufacturability. Additionally, existing off-the-shelf geometry kernels can seamlessly convert a parametric command sequence into a B-rep. However, there are some models that generate CAD programs in terms of parametric command sequences based on the CAD images. One such model is GenCAD, a generative model that employs an autoregressive transformer-based architecture with a contrastive learning framework, enhancing the generation of CAD programs from input images and providing a representation learning framework for multiple data modalities relevant to engineering designs [?]. GenCAD is distinguished by learning the distribution of design histories (or command sequences) that make up a B-rep and has superiority over other state-of-the-art generative CAD design models in terms of generating command sequences (a sequence of CAD commands and parameters design tokens), than B-rep files without design history.

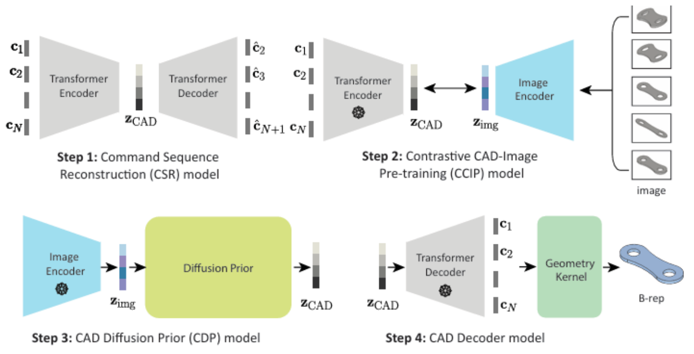

FIGURE 1. GENCAD ARCHITECTURE

However, GenCAD faces some limitations. One of the limitations is that it is particularly designed for simpler parts and geometries. Secondly, it cannot convert all of the input CAD images into CAD files, thus generating infeasible CAD command sequences. The objective of this study is to address the second limitation of the GenCAD by adding a self-repair pipeline on the infeasible CAD sequences to enhance the generation of valid CAD command sequences. By doing this, we are not only creating more feasible command sequences but also enhancing and increasing the existing GenCAD dataset. Additionally, to validate the accuracy of the CAD models generated from the infeasible design space, divergence metrics such as Maximum Mean Discrepancy (MMD) have been calculated among the point clouds of the corrected CADs and the ground truth.

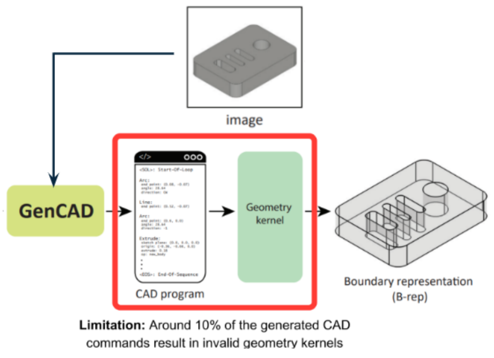
image
GenCAD
<SOL>: Start-Of-Loop
Arc:
Line:
Arc:
Extrude:
<EOS>: End-Of-Sequence
CAD program
Geometry kernel
Boundary representation (B-rep)
Limitation: Around 10% of the generated CAD commands result in invalid geometry kernels

FIGURE 2. GENCAD LIMITATION

The report is organized as follows. We discuss the related work in the current literature in Section 2. The methods used and the data set processed and gathered are discussed in Section 3. The results of the various methods used are discusssed in Section 4. Finally, we present our conclusion in Section 5, followed by acknowledgment in Section 6, and contributions from team members.

II. RELATED WORK

Recent advancements in generative AI have sparked interest in applying text-to-image models to engineering design, particularly for generating Computer-Aided Design (CAD) models. However, these models often struggle to produce feasible designs due to the complex requirements of engineering applications [2].

To address this challenge, researchers have proposed various methods. One approach involves using CAD images as prompts to improve design feasibility. By incorporating feasible CAD images into the generation process, text-to-image models like Stable Diffusion 2.1 can create more realistic and manufacturable designs [2]. Another method that utilizes both text and image encodings are encoder-decoder frameworks such as the CAD Translator method [3]. A method used for B-rep generation is the ComplexGen. ComplexGen views CAD reconstruction and generation as the detection of geometric features of different orders (vertices, edges, and surface patches) and their relationships. This method uses a two-step process, primitive design generation using a neural framework followed by optimization to recover a definite B-Rep chain for CAD design [4]. Some methods use an auto encoder that decomposes input images into depth, albedo, viewpoint, and illumination, and particularly work well for symmetric images [5]. Other models such as OpenECAD generates structured 2D sketches and 3D construction commands from images of 3D designs. These 3D construction command can pass through a 3D geometry kernel to create B-rep files [6].

An essential challenge in CAD research is the scarcity of comprehensive datasets. Among the few publicly accessible options, the ABC dataset is the largest, containing approximately one million CAD models sourced from various online repositories [7]. This dataset contains B-reps but does not provide any design histories or labels. Later, the DeepCAD dataset was developed [8]. The DeepCAD dataset includes design histories by extracting details from CAD designs within the Onshape repository. Similarly, the Fusion 360 dataset offers design histories along with assembly information and face segmentation. However, its small size limits its effectiveness for training models that generalize well [9]. The OpenECAD dataset uses a series of datasets that describe CAD operations via Python-like commands. Images include isometric, orthographic, and transparent views of 3D models [6].

While these methods show promise in generating more feasible CAD designs, challenges remain in ensuring structural validity and manufacturability. The GenCAD model does not ensure complete image to CAD transformation and outputs around 10% of the total B-Reps as infeasible. The methods used to address this limitation along with a description of the dataset used are discussed in the following sections.

III. METHODS

Six different methods were proposed to enhance GenCAD's geometry feasibility rate, all of which aimed at performing latent operations:

1. Self-repairing with an SSL* Regressor.
2. Self-repairing with a Regressor.
3. Guided Diffusion with a Classifier.
4. Guided Diffusion with an SSL* Regressor.
5. Guided Diffusion with a Classifier and an SSL* Regressor.
6. Guided Diffusion with a Classifier and an SSL* Regressor and self-repairing with an SSL* Regressor

*SSL: Self-Supervised Learning inspired regressor, explained further in section Furthermore, the methods took advantage of the labeled latent representations in the GenCAD-Self-Repairing dataset, which allowed for the use of supervised learning.

The details on the creation of the dataset are discussed first, following the fitting of different machine learning models for different purposes, and, finally, the rationale behind the methods utilized.

3.1. Dataset

The GenCAD-Self-Repairing dataset stems from the GenCAD dataset [?]. It was created by randomly selecting 133,617 images - from the total set of 840,947, given time and computing constraints - and then generating command sequences from them. As the command sequences were generated using the GenCAD model, the corresponding latent space representations, with shape (1 x 256), were saved. Then, a label was paired to them depending on whether the geometry kernel was able to convert the decoded sequence of commands into a B-rep, making each latent representation have a shape of (1 x 257). This process is shown in Figure 4.

Type	Count	Shape
Images	133,617	1x448x448
Valid Latent	123,809	1x257
Invalid Latent	9,808	1x257
Ground Truth Latent	26,723	1x256

TABLE 1. GENCAD-SELF-REPAIRING DATASET.

Finally, the corresponding ground truth command sequences from the selected images were extracted. Using the transformer encoder from GenCAD, they were converted into latent representations and saved as part of the dataset. For every five images, a single command sequence was extracted, given that they consist of deformations from the same underlying B-rep. This is shown in Figure 3.

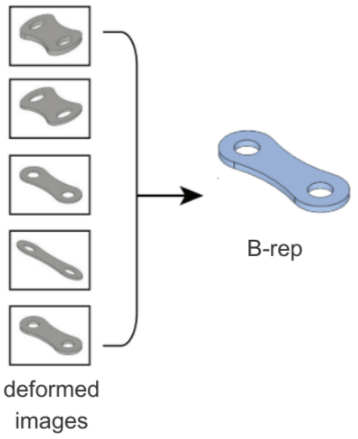

FIGURE 3. IMAGE SELECTION FOR THE DATASET.

3.2. Models

As discussed above, six different methods were used to improve the feasibility rate. These methods utilized three different supervised machine learning models trained on the GenCAD-Self-Repairing Dataset.

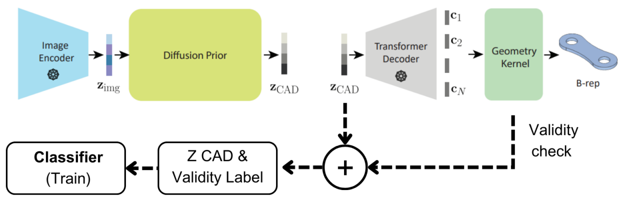

FIGURE 4. GENERATION OF DATASET AND CLASSIFIER TRAINING.

3.2.1. Classifier

The first model was a Multi-Layer Perceptron (MLP) Classifier. To achieve class balance and mitigate overfitting, the valid latent vectors were undersampled. The MLP Classifier was designed with an architecture consisting of three layers: 128, 64, and 1 neurons, respectively, with an input dimension of 256. Rectified Linear Unit (ReLU) activation functions were applied after the first two layers to introduce non-linearity, while a Sigmoid activation function was used in the output layer to produce probabilities for binary classification. The model was trained using an 80% training and 20% testing split, where the latent vectors served as inputs and their corresponding labels as outputs. The results of the classifier are shown in Table. 2, in addition to a confusion matrix in Fig. 5. Notably, this model demonstrated the separability of the classes by analyzing the latent vector representations, which is fundamental for the approach taken by GenCAD-Self-Repairing.

TABLE 2. MLP CLASSIFIER REPORT.

Class	Precision	Recall	F1-Score
1 = Valid	0.83	0.85	0.84
0 = Invalid	0.84	0.83	0.84
		Accuracy	0.84

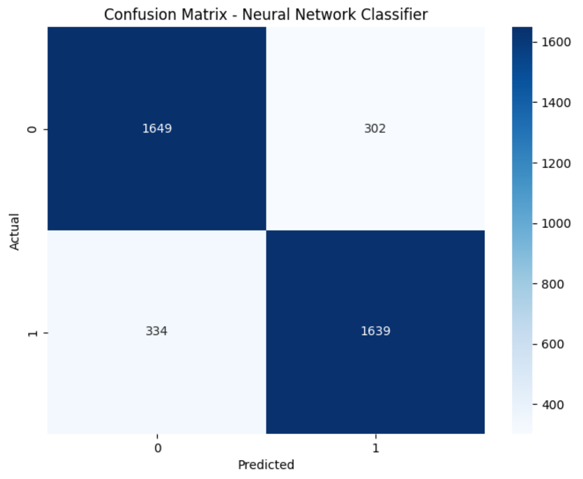

FIGURE 5. THE CONFUSION MATRIX OF THE MLP CLASSIFIER.

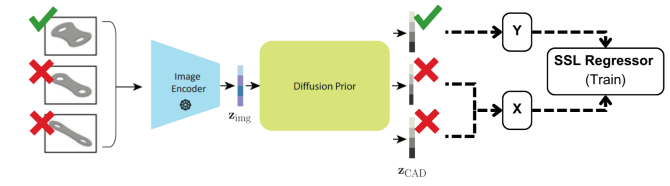

FIGURE 6. SSL-LIKE REGRESSOR INPUTS & OUTPUTS.

3.2.2 SSL-like Regressor

Two linear regressors were fitted for different purposes. The first one, shown in Figure 6, uses invalid latent representations from a given image as inputs, and the valid latent representations from that same image as outputs. In essence, this is trying to predict what the valid representation should be given an invalid representation.

It is important to note here that not all the images with invalid latent vectors were paired with at least one valid latent vector. After processing the data in this manner, there were a total of 9,029 pairs of invalid - valid latent vectors. After fitting the linear regressor 1 to the data, an $R^2 = 0.0685$ and $MSE = 0.0206$ were obtained.

This method, most notably, allowed for regression on the latent space without having to use the ground truth latent representations to define the desired outputs, in a fashion that resembles Self-Supervised Learning. Furthermore, this ensured almost no alterations in the structure of the latent space.

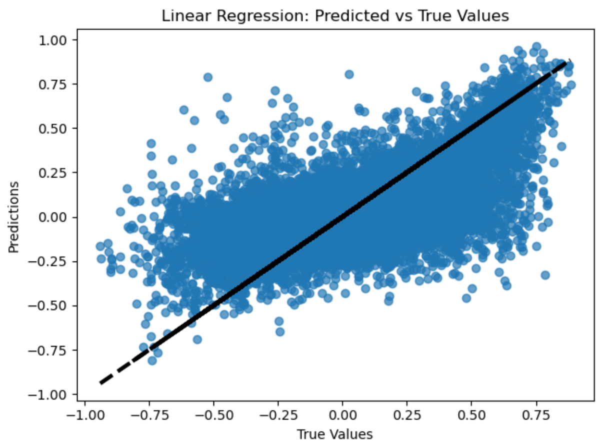

FIGURE 7. SSL-LIKE REGRESSOR RESULTS.

3.2.3 Regressor

The second regressor was trained using all the GenCAD latent vectors as inputs and the ground truth latent vectors as outputs, as shown in Figure 8. This regressor, in opposition to the first one, was trying to modify the latent structure to achieve higher overall point cloud accuracy (MMD score) when compared to the underlying point clouds while simultaneously improving the feasibility rate.

Given that there was an available ground truth latent vector for every image, 133,617 pairs of vectors were used to fit this regressor. The regressor had an $R^2 = 0.6301$ and an $MSE = 0.00414$.

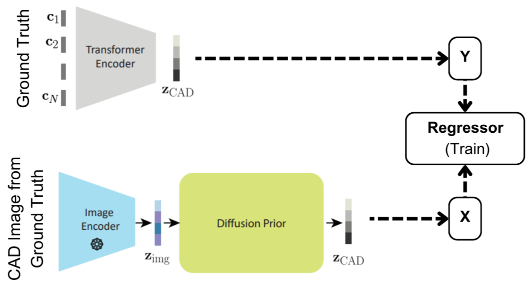

FIGURE 8. REGRESSOR INPUTS & OUTPUTS.

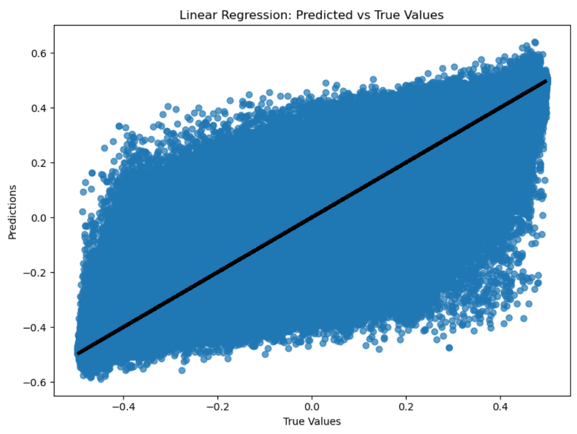

FIGURE 9. REGRESSOR RESULTS.

A key note here is that, since both regressors had 256 outputs, training different models and performing feature engineering were computationally expensive tasks. Hence, the decision of utilizing these regressors. However, an important next step, listed under future work, is the training of better regressors.

3.2.4 Guided Diffusion Denoising

Guided Diffusion Denoising is a key step in the proposed pipeline, leveraging both a classifier and a regressor to guide the denoising process in latent space to pull invalid representations to valid regions. This approach refines the denoising trajectory by incorporating gradient-based feedback, ensuring that the generated samples are closer to the desired distribution.

The diffusion model computes the denoised latent vector $\mathbf{z}_{t-1}$ at each timestep t through the following process:

$$\mu_t = E[\mathbf{z}_{t-1}|\mathbf{z}_t, \mathbf{z}_0], \quad (1)$$

$$\mathbf{z}_{t-1} = \mu_t + \sigma_t \varepsilon, \quad (2)$$

where μ_t is the posterior mean, σ_t is the posterior variance, and $\varepsilon \sim \mathcal{N}(0, \mathbf{I})$ is Gaussian noise.

Classifier Guidance: To ensure that the denoised sample aligns with the desired classification boundary, a classifier $C(\mathbf{z})$ predicts the infeasibility probability $P_{\text{infeasible}} = 1 - C(\mathbf{z})$. The gradient of this probability with respect to the current latent vector $\mathbf{z}_t$ is used to adjust the denoising direction:

$$\nabla_{\text{classifier}} = \nabla_{\mathbf{z}_t} \left(s_{\text{class}} \cdot P_{\text{infeasible}} \right), \quad (3)$$

where $s_{\text{class}} = 10$ is the guidance scale for the classifier. This gradient is subtracted from the posterior mean μ_t to encourage feasible solutions:

$$\mu_t' = \mu_t - \nabla_{\text{classifier}}. \quad (4)$$

Regressor Guidance: The regressor $R(\mathbf{z})$ predicts a target value that the latent vector $\mathbf{z}_t$ should approximate. To achieve this, a mean-squared error loss between the regressor output and $\mathbf{z}_t$ is computed:

$$\mathcal{L}_{\text{reg}} = \|R(\mathbf{z}_t) - \mathbf{z}_t\|_2^2. \quad (5)$$

The gradient of this loss is scaled by a guidance factor $s_{\text{reg}} = 10$ and used to guide the denoising process:

$$\nabla_{\text{regressor}} = \nabla_{\mathbf{z}_t} \left(s_{\text{reg}} \cdot \mathcal{L}_{\text{reg}} \right). \quad (6)$$

This gradient is also subtracted from μ_t:

$$\mu_t'' = \mu_t' - \nabla_{\text{regressor}}. \quad (7)$$

Final Update: The guided posterior mean μ_t'' is then used to compute the denoised sample:

$$\mathbf{z}_{t-1} = \mu_t'' + \sigma_t \varepsilon. \quad (8)$$

This guided denoising process ensures that the diffusion model generates samples that align with the desired distribution. Specifically, it focuses on producing samples within the feasible region by satisfying the constraints imposed by the classifier and regressor, thereby improving the overall feasibility and reliability of the generated commands.

3.2.5 Self-Repairing from Failures

Using both of the trained regressors, a pipeline to self-repair models was created. After an image was processed by GenCAD, if the geometry kernel deemed the generated command sequence as infeasible, the latent vector was used as input for the SSL regressor or the regular regressor in an attempt to predict a feasible latent vector. The corrected output was finally decoded again, with the generated command sequence now having a better chance of corresponding to a valid geometry. This pipeline is shown in Figure 10.

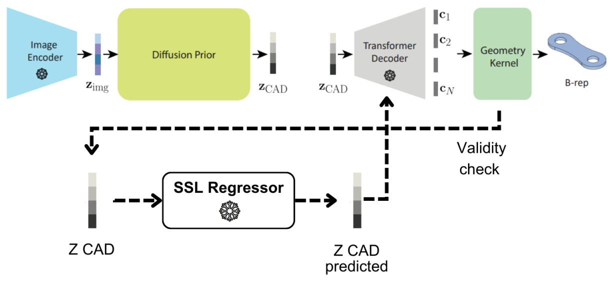

FIGURE 10. SELF-REPAIRING PIPELINE.

IV. RESULTS

The GenCAD's validation set, consisting of 8,515 B-rep images, was used to benchmark the six different proposed methods against GenCAD. First, a discussion and decision on evaluation metrics is presented.

4.1. Evaluation Metrics

The main goal was to achieve a higher feasibility rate. Hence, this is the first evaluation metric utilized. The formal definition is given by:

$$F = \frac{V}{V+I} \tag{9}$$

Where F is the feasibility rate, V is the total number of valid generated B-reps, and I is the total number of invalid generated B-reps.

The second goal was to maintain the accuracy of the generated geometries. Several metrics can be used for this goal, such as Jensen–Shannon Divergence (JSD) or Maximum Mean Discrepancy (MMD).

Both metrics measure the statistical similarity between the point distributions of two point clouds, but there are some important differences. JSD discretizes the data by computing a normalized histogram of the points in the point clouds, losing information in the process of binning. MMD, on the other hand, uses all the datapoints to compare the means between the distributions of the point clouds, being better suited for complex distributions, but also more computationally expensive.

Given our lack of knowledge on the underlying distributions, MMD is chosen as the accuracy metric. The kernel function utilized is the Gaussian RBF:

$$k(x,y) = e^{-\frac{||x-y||^2}{2\sigma^2}} \tag{10}$$

This is because we assume that close points are more highly correlated than points that are further apart. The MMD metric is defined as:

$$\mathrm{MMD} = \sqrt{\frac{1}{m^2}\sum_{i=1}^{m}\sum_{j=1}^{m} k(x_i,x_j) + \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n} k(y_i,y_j) - \frac{2}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n} k(x_i,y_j)} \tag{11}$$

4.2. Summary of Results

The Table 3 shows the Feasibility Rate and the average MMD Score for all the described methods over the validation set of images. The best results are shown in **bold letters**. (↓) indicates that lower is better, and (↑) indicates that higher is better.

Method	Feasibility (↑)	Mean MMD (↓)
GenCAD (baseline)	0.931	**0.180**
SSL Regressor	0.961	0.183
Regressor	0.931	0.181
Classifier Guidance	0.936	0.197
Regressor Guidance	0.936	0.198
Classifier and Regressor Guidance	0.954	0.199
Classifier and Regressor Guidance with SSL Regressor **(GenCAD-Self-Repairing)**	**0.970**	0.200

TABLE 3. SUMMARY OF RESULTS.

GenCAD (baseline) was able to convert 7,707 B-reps from 8,515 images, whereas GenCAD-Self-Repairing converted 8,239 B-reps from 8,515 images. This means that GenCAD-Self-Repairing was able to fix 66% (532/808) of the baseline infeasible images.

Furthermore, the decrease in MMD was only of 11% (0.2/0.18). Histograms of the average scores are shown to show the diversity of scores. Finally, a plot showing a 2-dimension Principal Component Analysis (PCA) on the generated latent space by GenCAD-Self-Repairing over the validation set, plotted on top of the GenCAD latent space over the same set, is shown in Figure 13.

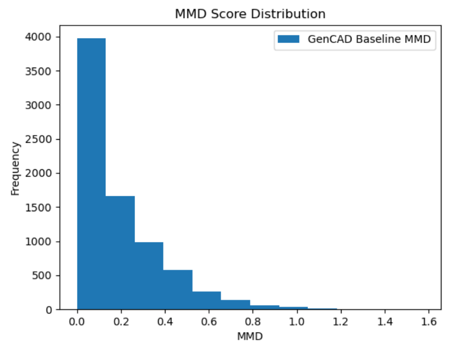

FIGURE 11. GENCAD BASELINE MMD SCORES.

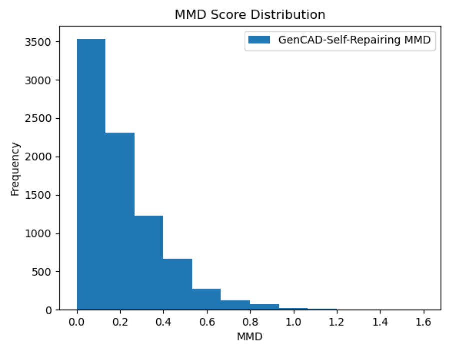

FIGURE 12. GENCAD-SELF-REPAIRING MMD SCORES.

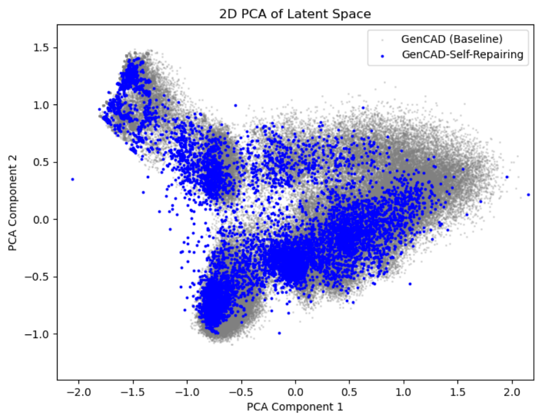

FIGURE 13. 2D PCA OF THE LATENT SPACE.

V. CONCLUSION and FUTURE WORK

Generative AI and design transformed the CAD modeling space with an increasing number of models and frameworks focusing on CAD generation from images, text embeddings, and command sequences. However, most of them lack a repair pipeline to lower the number of infeasible designs produced.

Among the various methods discussed, the "Classifier & Regressor" Diffusion Guidance with additional Regression repairing proves to be a highly efficient correcting pipeline converting $2/3$ of the infeasible designs into feasible designs and having a fairly decent MMD score as well, highlighting the similarities between the ground truth CAD models and the repaired generated CAD models.

An interesting point to be seen is that even with a low R^2 score of 0.06 on the SSL Regression model, a significant amount of infeasible CADs were converted into a feasible design, hinting at a higher efficiency of GenCAD's decoder architecture. From the latent representation, it can be observed that the corrected designs overlap fairly well over the original latent representation of the GenCAD model indicating the repair pipeline's effectiveness.

Some possible extensions and future work in this framework can be as follows. The accuracy of the generated CAD file with respect to the actual CAD file, after passing through the decoder can be evaluated with a different metric than MMD. A better regressor can be trained and tested for better prediction of feasible latent spaces from the infeasible latent space. Conditional Variational AutoEncoders (VAE) can also be tested instead of guided diffusion models, and performance can be compared.

VI. ACKNOWLEDGMENTS

We would like to thank Professor Faez Ahmed, Dr. Ferdous Alam, Lyle Regenwetter, Amin Heyrani Nobari, and Noah Bagazinski for their mentorship and advice throughout this project. Your support was truly outstanding.

VII. Team Contributions

Chikaha Tsuji: Created the GenCAD-Self-Repairing dataset, trained a classifier, implemented diffusion guidance, and evaluated the feasibility rates.

Harshit Gupta: Trained Regressors and Classifiers for feasibility prediction, calculated MMD and JSD metric scores for the various models, carried out data visualization, and designed the poster.

Enrique Flores Medina: Created the GenCAD-Self-Repairing dataset, trained the regressors and classifier, implemented the regressor self-repairing pipeline, performed a 2D PCA analysis of the results, evaluated MMD metrics for each method, and implemented a live demonstration of GenCAD-Self-Repairing.

REFERENCES

[1] Alam, M. F., and Ahmed, F., 2024. Gencad: Image-conditioned computer-aided design generation with transformer-based contrastive representation and diffusion priors.

[2] Chong, L., Rayan, J., Dow, S., Lykourentzou, I., and Ahmed, F., 2024. Cad-prompted generative models: A pathway to feasible and novel engineering designs.

[3] Li, X., Song, Y., Lou, Y., and Zhou, X., 2024. “CAD translator: An effective drive for text to 3d parametric computer-aided design generative modeling”. In ACM Multimedia 2024.

[4] Guo, H., Liu, S., Pan, H., Liu, Y., Tong, X., and Guo, B., 2022. “Complexgen: Cad reconstruction by b-rep chain complex generation”.

[5] Wu, S., Rupprecht, C., and Vedaldi, A., 2020. Unsupervised learning of probably symmetric deformable 3d objects from images in the wild.

[6] Yuan, Z., Shi, J., and Huang, Y., 2024. “Openecad: An efficient visual language model for editable 3d-cad design”. *Computers amp; Graphics,* ***124***, Nov., p. 104048.

[7] Koch, S., Matveev, A., Jiang, Z., Williams, F., Artemov, A., Burnaev, E., Alexa, M., Zorin, D., and Panozzo, D., 2019. Abc: A big cad model dataset for geometric deep learning.

[8] Wu, R., Xiao, C., and Zheng, C., 2021. Deepcad: A deep generative network for computer-aided design models.

[9] Willis, K. D. D., Pu, Y., Luo, J., Chu, H., Du, T., Lambourne, J. G., Solar-Lezama, A., and Matusik, W., 2021. Fusion 360 gallery: A dataset and environment for programmatic cad construction from human design sequences.