Proceedings of the ASME 2025 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference
IDETC/CIE 2025
August 17–20, 2025, Anaheim, CA, USA

IDETC/CIE 2025-168988

IMAGE2CADSEQ: COMPUTER-AIDED DESIGN SEQUENCE AND KNOWLEDGE INFERENCE FROM PRODUCT IMAGES

Xingang Li, Zhenghui Sha*
Walker Department of Mechanical Engineering
The University of Texas at Austin
Austin, Texas 78712
Email: zsha@austin.utexas.edu

ABSTRACT

Computer-aided design (CAD) tools empower designers to design and modify 3D models through a series of CAD operations, commonly referred to as a CAD sequence. In scenarios where digital CAD files are inaccessible, reverse engineering (RE) has been used to reconstruct 3D CAD models. Recent advances have seen the rise of data-driven approaches for RE, with a primary focus on converting 3D data, such as point clouds, into 3D models in boundary representation (B-rep) format. However, obtaining 3D data poses significant challenges, and B-rep models do not reveal knowledge about the 3D modeling process of designs. To this end, our research introduces a novel data-driven approach based on representation learning to infer CAD sequences from product images, coined as Image2CADSeq. These sequences can then be translated into B-rep models using a solid modeling kernel. Unlike B-rep models, CAD sequences offer enhanced flexibility to modify individual steps of model creation, providing a deeper understanding of the construction process of CAD models. One unique contribution of this paper is the development of a multi-level evaluation framework for model assessment, so the predictive performance of the Image2CADSeq model can be rigorously evaluated. The model was trained on a specially synthesized dataset, and various neural network architectures were explored to optimize the performance. The experimental and validation results show the great potential of our model in data-driven reverse engineering of 3D CAD models from 2D images.

1 INTRODUCTION

Computer-aided design (CAD) systems can significantly reduce design time by avoiding the need for traditionally required labor-intensive manual drawings [1]. Contemporary CAD systems such as Fusion 360, SOLIDWORKS, and OnShape enable designers to create and modify CAD models. However, in certain scenarios, the CAD model of a product may not be readily available due to various factors, including outdated documentation, lack of digital records, and commercial reasons. Reverse engineering (RE) is employed to overcome these obstacles, utilizing measurement and analysis tools to reconstruct CAD models [2, 3].

Integrating RE with CAD systems can not only allow designers to leverage the advantages of existing products while incorporating their own innovative ideas and improvements but can also be used for design knowledge restoration and management [2, 3]. However, the traditional RE process faces two major limitations. First, it focuses on reconstructing 3D models rather than CAD sequences. Compared to 3D models, a CAD sequence provides access to the historical construction process and associated design knowledge and it facilitates geometry modification using parametric modeling. Second, the process has been performed primarily manually, making it labor-intensive and time-consuming. Recently, researchers have explored data-driven methods, such as converting 3D point clouds [4,5] or voxels [6, 7] into CAD models. Nevertheless, these 3D input data are often challenging to acquire due to inaccessibility and unavailability. 3D scanning could be a solution, yet quality is often

*Corresponding author.

unsatisfactory and cost is an unavoidable factor to consider when acquiring specialized equipment and expertise.

Compared to point clouds or voxels, images are easier to acquire, particularly given the popularity of mobile devices nowadays. Thus, our question arises: Can we reverse engineer CAD sequences directly from 2D images, supporting designers to interpret and edit CAD models in design? Our literature review has indicated a scarcity of research exploring the know-how to this question. Therefore, we are motivated to develop a data-driven reverse engineering approach that can generate a sequence of CAD operations based on a single image, referred to as "Image2CADSeq" hereafter for brevity.

The contributions of this study lie in four aspects. First, this study pioneers the prediction of CAD sequences from a single image input, utilizing a target-embedding variational autoencoder (TEVAE) architecture [8]. Second, we developed a data synthesis pipeline based on Fusion 360 Gallery domain-specific language (DSL), generating synthetic data that mimic real-world images and CAD models. Third, we developed a matrix representation for the Fusion 360 Gallery DSL which is of great value for geometry learning research using the Fusion 360 Gallery dataset [9] Lastly, a new multi-level evaluation framework, consisting of 10 performance metrics (see Section 3.5 for details), provides a comprehensive assessment scheme that can be generally applied to similar CAD sequence or program inference tasks. We anticipate that the Image2CADSeq model can potentially revolutionize CAD systems by simplifying model reconstruction, enabling both experienced and novice designers to contribute actively, fostering collaboration, education, and design democratization.

2 LITERATURE REVIEW

In this section, we present a review of deep learning methods specifically tailored for CAD sequence generation and prediction, which are most relevant to our work.

Significant progress has been made in the generation of CAD sequences for the reconstruction of 3D models particularly through *Sketch-and-Extrude* modeling operations. Recently, there have been methods [10, 11] for generative models specifically designed for the unconditional generation of CAD sequences. These models aim to autonomously create CAD sequences without relying on specific conditions or inputs. Specifically, Wu et al. [10] presented the first generative model, DeepCAD, that learns from sequences of CAD modeling operations to produce editable CAD designs. By drawing an analogy between CAD operations and natural language, the authors propose to utilize a transformer architecture [12] aiming to leverage the capabilities of transformer models in understanding and generating sequences, adapting them to the context of CAD design operations. Unconditional generative models, such as works [10, 11], that do not rely on input conditions (e.g., text, sketches, or images) indeed serve as valuable tools for randomly generating a multitude of designs, offering inspiration and exploration of diverse possibilities. However, these models cannot directly incorporate designers' intent into the generation process due to the lack of user input. Consequently, the designs generated can deviate from the expectations or specific requirements of the designers. This discrepancy highlights the need for mechanisms that allow designers to guide or influence the output, ensuring that the generated designs align more closely with their intent and preferences.

To that end, several methods have been introduced to allow the generation of CAD sequences given the target of B-rep models [9,13], voxels [6,7], point clouds [4,5], and sketches [14,15]. Particularly, Fusion 360 Gym [9] was developed to reconstruct a CAD model given a B-Rep model, utilizing a face-extrusion technique that relies on existing planar faces within the B-Rep model. However, despite the potential for CAD sequence generation, the face-extrusion method differs significantly from the more natural sketch-extrusion method commonly used by human designers. Moreover, this technique is ineffective when confronted with a lack of available planar or profile data in the input data, such as images. Our work aims to fill a research gap in the existing literature by focusing on the task of generating CAD sequences from images and we proposed a multi-level evaluation system to comprehensively evaluate the prediction of the CAD sequences.

3 METHODOLOGY

The flowchart depicted in Figure 1 illustrates the proposed approach of Image2CADSeq. Our objective is to harness the power of deep learning to predict a CAD sequence—a series of CAD operations characterized by specific operation types and their corresponding parameters—from an image. The image could be a rendering from a CAD model or a real-world photograph of a 3D object.

We employ a CAD program as a representational tool for CAD sequences. A CAD program enables designers to programmatically script their designs in a specialized scripting environment. The CAD program is then streamlined into a vectorized representation for neural network modeling. This representation can facilitate not only the development of our neural network's architecture but also the creation of a data-synthesis pipeline tasked with generating the training data for the neural network model. In addition, given the complexity of the Image2CADSeq task, we develop a multi-level evaluation system that rigorously assesses our neural network models' performance on the prediction of CAD program (i.e., the operation sequence, type, and parameters), thereby ensuring the reliability and accuracy of our approach.

3.1 DESIGN REPRESENTATION OF CAD PROGRAMS

In this study, we employ a domain-specific language (DSL), namely Fusion 360 Gallery (Gallery for conciseness) [9], as a particular use case of the CAD program to demonstrate our approach. While Gallery DSL currently only supports the *Sketch* and *Extrude* operations [9], our approach can be easily extended should more operation types be available in the future.

One essential step when using neural networks to effectively interpret CAD programs involves the development of an effective and efficient representation for each CAD operation and the entire CAD program. However, there are three major challenges: 1) Different CAD programs comprise varying numbers of operations. 2) Different CAD operations involve different numbers of parameters. 3) Parameters can be either continuous or discrete values. To tackle these challenges, we developed a design representation with a unified data structure following the method introduced in a few studies [10, 16]. We identified 10 variables $(t, I, x, y, \alpha, r, [I], d, O, s)$ from the Gallery DSL, detailed in Table 1. In what follows, we elaborate on our approach to handling these variables in vector representations.

(1) $t \in \{0,1,2,3,4,5,6\}$ represents the operation types with $0-4$ representing *add_sketch*, *add_line*, *add_arc*, *add_circle*, and *add_extrude*. The values 5 and 6 are used to represent the start (SOP) and the end (EOP) of a CAD program, which are not typical CAD operations but are included for the learning process to indicate a complete CAD program as required by a transformer model [10, 12, 16].

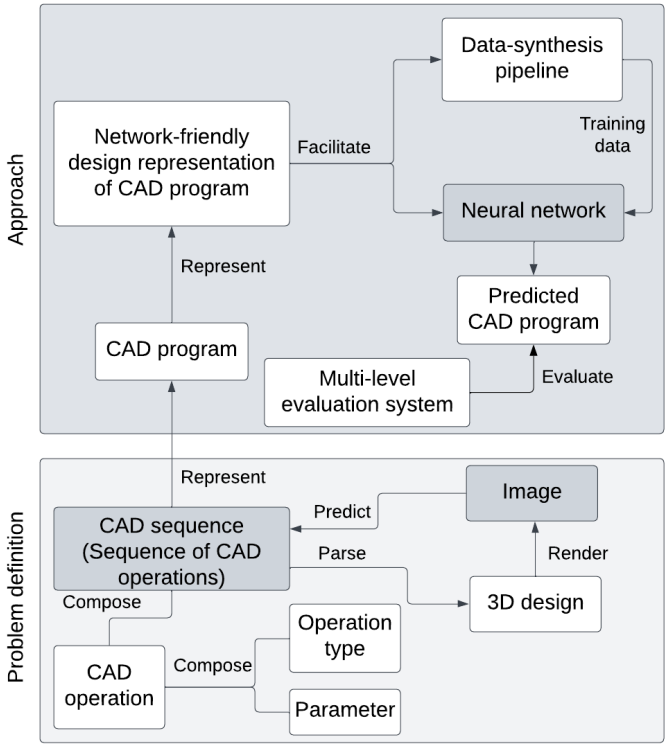


FIGURE 1. Approach overview

TABLE 1. Variables for the vectorized design representation of Gallery DSL

		Variable	Characteristic	Value range
Operation type		t	Discrete	{0, 1, 2, 3, 4, 5, 6}
Operation parameters	Identifier of sketch plane	I	Discrete	{0, 1, 2}
	End point x	x	Continuous	[-1, 1]
	End point y	y	Continuous	[-1, 1]
	Sweep angle	α	Continuous	[-1, 0) or (0, 1] (*180)
	Radius	r	Continuous	(0, 1]
	Identifier of profile	$[I]$	Discrete	{0, 1, 2, ...}
	Extrusion distance	d	Continuous	[-1, 0) or (0, 1]
	Boolean operations	O	Discrete	{0, 1, 2, 3}
Auxiliary factor	Scale factor	s	Discrete	0~255 (e.g., 10)

(2) $I \in \{0,1,2\}$ indicates the *Sketch Plane* using one of the canonical planes: "XY", "XZ", or "YZ".

(3,4) x and y are the coordinates of the endpoint for *Line* and *Arc*, while they represent the center point when the operation type is *Circle*. We excluded the start point required by *Line* and *Arc* from the design representation by obtaining it from the precedent curve to make sure all curves are connected one after another, making the vectorized representation more compact. There are two extra considerations for this setting: (i) If one curve has no precedent, we default its start point to the origin $(0,0)$ when parsing the design representation. (ii) For *Arc* that requires a center point instead of an endpoint, we calculate the coordinates for the center point based on its start point, endpoint, and sweep angle.

(5) α represents the sweep angle of an *Arc*.

(6) r is the radius of a *Circle*.

(7) $[I]$ represents the profile index in the *Sketch*.

(8) d represents the signed distance of the depth for *Extrude*.

(9) $O \in \{0,1,2,3\}$ is used to indicate the Boolean operations: join, cut, intersect, or add, respectively.

(10) s is an auxiliary factor that can be used to scale a CAD model.

In addition, to standardize the treatment of both continuous and discrete parameters, we discretize continuous parameters through *quantization*. This involves: (a) Confining continuous values to a subset of $[-1,1]$ (e.g., $(0,1]$ for radius and $[-1,1]$ for endpoint x and y; (b) Dividing each range into 256 equal seg-

ments, enabling representation as 8-bit integers (i.e., $0-255$); (c) For the sweep angle (α) in radians, we multiply it by 180 for encoding; (d) Handling scale factor (s): Although the scale factor can be a non-negative continuous value, we limit it to 256 levels for consistency with other continuous values' quantization. Consequently, the 10 variables can encode both the operation type and its associated parameters. From the 10 variables, a fixed-dimensional vector can be formalized as a unified design representation for each CAD operation, and the unused parameters will be filled with values of -1.

The subsequent consideration involves standardizing CAD programs of varying sizes (the number of CAD operations involved). To achieve a consistent data structure across all CAD programs, we used a treatment, called *maximum program length*. Then, CAD programs shorter than this maximum length are extended by appending end marks (*EOP*) until they reach the predetermined length.

In this study, we use 7 variables to construct a 7-dimensional vector $[t, I, x, y, \alpha, r, d]$ for each CAD operation (i.e., one step/line in a CAD sequence). Additionally, we assign default values to the other three variables $[I], O$, and s, setting them as 0, 3, and 10 representing the $0th$ profile in *Sketch* (i.e., closed 2D shapes formed by CAD operations such as *Line*, *Circle*, and *Arc*), *Boolean* operation *Add*, and a scale factor of 10, correspondingly. These variables are chosen because we focus on single object creation from a single *Sketch* profile. Among these variables, different ones can be selected, which can influence the complexity of the data structure and thus the complexity of designs. For example, all 10 variables can be added to enable the creation of CAD models that are formed by multiple *Sketch* profiles and *Boolean* operations (e.g., join, cut, intersect, and add). In addition, the maximum length for a CAD program is set to 10. As a result, the design representation of a CAD program will be a matrix, namely, the feature matrix. Mathematically, the feature matrix denoted as P, is expressed as $P = \left[\mathbf{o^1}, \mathbf{o^2}, \ldots, \mathbf{o^{N_c}}\right]^T \in \mathbb{R}^{10\times 7}$, where $\mathbf{o^i} \in \mathbb{R}^7$ is a CAD operation vector, and $N_c = 10$ is the sequence length of the CAD program. Refer to Figure 3 for an example of how a cylinder is converted to a feature matrix, including the quantization of its parameters as explained earlier.

3.2 NEURAL NETWORK ARCHITECTURE

We developed the Image2CADSeq model, utilizing a target-embedding representation learning method [17,18], as illustrated in Figure 2. It features an encoder-decoder network for Stage 1 (S1), which is for unsupervised learning and enables the efficient encoding of target objects (i.e., matrix representation of CAD programs) within a latent space. An additional encoder is integrated into Stage 2 (S2), focusing on supervised learning to regress the previously learned latent space using feature objects (i.e., images) as input.

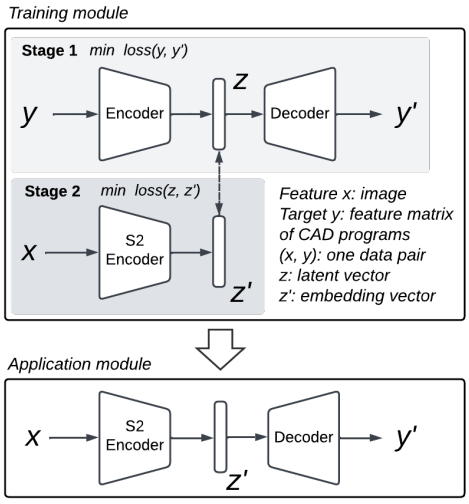


FIGURE 2. Image2CADSeq model using a target-embedding representation learning method

The Image2CADSeq model employs a two-stage training strategy [8, 19]. In Stage 1, the focus is on independent training of the encoder-decoder network. The objective is to minimize the reconstruction loss between the actual matrix feature of CAD programs (y) and its reconstructed equivalent (y'). Completing this stage involves fixing the learnable parameters of the neural network model and saving the learned model, thereby capturing a latent space of y. Stage 2 shifts the focus to independent training of the S2 encoder by minimizing the difference between the latent vector, derived from the learned latent space, and the embedding vector produced by the S2 encoder using an image as input. Importantly, each image used in this stage is directly associated with its feature matrix from Stage 1. This image and its corresponding feature matrix are associated with the same 3D object, and they form one data pair. The alignment of the latent vector with the embedding vector is performed specifically for these data pairs, ensuring that the S2 encoder training is precisely tuned to the corresponding images. This approach ensures a cohesive and targeted learning process. We present a novel data synthesis pipeline to generate training data pairs in Section 3.3.

After training the Image2CADSeq model, the S2 encoder is integrated with the decoder from S1, creating the application module. This module is capable of predicting a feature matrix given an image input. Subsequently, this feature matrix can be translated into a CAD program using the Gallery DSL. Finally, the CAD program can be parsed into a 3D object by utilizing Fusion 360 software.

Two strategies were used for Stage 1 of the Image2CADSeq model: 1) a baseline AE and 2) a VAE to compare their performance and explore a better architecture for the Image2CADSeq model. The corresponding architecture of the model formed

by these two strategies is called target embedding autoencoder (TEA) and target-embedding variational autoencoder (TEVAE), respectively. For Strategy 1, we obtained the AE model by modifying the transformer-based AE from DeepCAD [10]. It uses a typical reconstruction loss coupled with a regularization loss. Reconstruction loss ensures accurate reconstruction of input features in the output, while regularization loss prevents overfitting to the training data. For Strategy 2, we further modified the AE architecture to form a VAE model and added a KL-divergence loss to the loss function.

We utilized ResNet18 [20] for the Stage 2 Encoder in both strategies. In addition, a dropout layer is positioned between the encoder and the embedding vector layer to prevent overfitting to the training data, thus maintaining its efficacy on unseen data. We utilized a regression loss between the embedding vector of the image and its corresponding latent vector obtained from the latent space in Stage 1 and a regularization loss to promote the generalizability of the Stage 2 encoder.

3.3 DATA SYNTHESIS

With the proposed design representation of the CAD programs and Fusion 360 software, we introduce an automatic data synthesis pipeline, as illustrated in Figure 3. This method is tailored to generate training data pairs comprising feature matrices of CAD programs and the corresponding images, essential for training the Image2CADSeq model. The process begins with preparing a list of shape templates, such as cylinders, employing the **Sketch-and-Extrude** paradigm of the Gallery DSL. For example, a cylinder can only be formed by drawing a circle and subsequently extruding that shape; it cannot be made by sketching a rectangle and revolving it. For these basic template shapes, we establish a series of template sequences of CAD operations (e.g., *add_line*, *add_circle*). One template shape may correspond

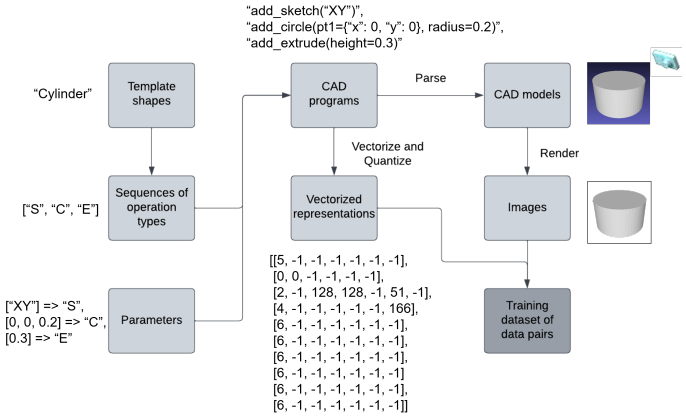
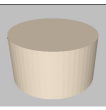


FIGURE 3. Synthesis pipeline for the training dataset of image and vectorized CAD sequence, exemplified using a cylinder model

TABLE 2. Template shapes for the data synthesis

Template Shape (*Sketch*)+ *Extrue*	Template sequence of operation types	Example
TS 1 (*Circle*)	["S", "C", "E"]	
TS 2 (Triple *Line*)	["S", "L", "L", "L", "E"]	
TS 3 (Triple *Arc*)	["S", "A", "A", "A", "E"]	
TS 4 (Double *Arc* + single *Line*)	["S", "L", "A", "A", "E"] ["S", "A", "L", "A", "E"] ["S", "A", "A", "L", "E"]	
TS 5 (Single *Arc* + Double *Line*)	["S", "A", "L", "L", "E"] ["S", "L", "A", "L", "E"] ["S", "L", "L", "A", "E"]	

to several template CAD sequences (see Table 2 for more details). The corresponding parameter values of these operations are then generated based on the range specified in Table 1. By integrating these template operations with their respective parameters, a complete CAD program is formulated, which is then translated into 3D CAD models through Fusion 360 software. These models are then rendered to obtain their images. Additionally, the CAD programs are vectorized and quantized to derive their feature matrices, as discussed in Section 3.1. An image paired with its feature matrix, both derived from the same CAD program, constitutes a data pair. The method is exemplified using a cylinder model in Figure 3.

In this study, we use simple shapes instead of complex shape datasets to ensure a controlled and interpretable evaluation of the model performance with the proposed multi-level comprehensive evaluation system. This ensures that our approach is robust and transparent, laying a solid foundation for future extensions to more complex datasets. We developed a collection of 5 template shapes (TS), as depicted in Table 2. We used *add_line* (L), *add_arc* (A), *add_circle* (C) to create the *Sketch*, and applied the *Extrude* operation to generate the 3D shapes. TS 1-3 each correspond to unique sequences of operation types, while TS 4 and 5 are associated with three varied sequences. An example for each template shape is also presented.

We synthesized 2,000 different shapes corresponding to every sequence type outlined in Table 2, except for the sequence of TS 1, for which we synthesized 6,000 shapes. This was taken to ensure a balanced dataset in terms of both the length of sequences and the number of shapes for each template shape category. Consequently, this led to the creation of 22,000 CAD mod-

els. For imaging purposes, all these 3D models were rendered using a uniform perspective camera positioned at $(20.0, 20.0, 20.0)$ looking towards the origin $(0.0, 0.0, 0.0)$ and all the images are in the resolution of 512×512 pixels. This process resulted in the image set $X = \{x_k\}_{k=1}^{22000}$. The corresponding feature matrices $Y = \{y_k\}_{k=1}^{22000}$ were also saved during synthesis. The final training dataset was $\{x_k, y_k\}_{k=1}^{22000}$.

3.4 TRAINING DETAILS

The dataset was divided into three subsets: train, validation, and test set with a proportion of 8:1:1. The validation set was used to monitor the training process. We employed a grid search strategy in Stage 1 to find optimal hyperparameters of the neural network models. Using such a search, a latent dimension size of 256 proved optimal for both models, resulting in the lowest reconstruction loss for the test set data among trials with dimensions of 64, 128, 256, and 512. Other hyperparameters include 500 epochs of training with a batch size of 512, the Adam optimizer, and a learning rate of 0.001. Moving to Stage 2, we initiated training with a pre-trained ResNet18 model [20] that possesses a broad comprehension of various images. The S2 encoder was trained for 50 epochs using the Adam optimizer with a learning rate of 0.0001 and a batch size of 128. A dropout ratio of 0.4 was applied in the dropout layer.

3.5 EVALUATION METRICS

In the Image2CADSeq task, there are three key elements: the image, the CAD program, and the 3D CAD model. The CAD program consists of a sequence of CAD operations, each involving an operation type and its associated parameters. To assess the effectiveness of our approach, we have developed a set of evaluation metrics, as illustrated in Table 3.

For the evaluation of 3D CAD models and images, we utilize established metrics such as the intersection over union (IoU) and mean squared error (MSE), respectively, as shown in Table 3(a). Some other metrics, such as dice similarity [21] and Hausdorff distance [22], can also be utilized to understand the performance of the model from different perspectives. However, assessing the quality of CAD programs poses a challenge due to the scarcity of suitable metrics in the literature. To address this gap, we introduce a novel evaluation system for CAD programs based on the proposed matrix representation, detailed in Table 3(b). To comprehensively evaluate the information loss between the predicted CAD programs with the ground truth, this system incorporates both hierarchical (H1-3) and double-layered (L1,2) aspects as shown below, facilitating a multi-dimensional assessment of CAD program prediction.

1. Hierarchies:
   - H1: Sequence evaluation – Evaluates the accuracy of the entire CAD program and the specific order in which certain CAD operation types follow.
   - H2: Sequence-based operation type evaluation – Examines the accuracy of each individual operation type within the sequence.
   - H3: Set-based operation type evaluation – Assesses the operation types as a collective set, without considering the sequential order. Even if the operation type sequence varies, a prediction is considered superior if it accurately predicts a higher number of operation types due to the preservation of information.
2. Layers:
   - L1: Operation type layer – Evaluates the accuracy of the CAD operation types.
   - L2: Parameter layer – Assesses the accuracy of the parameters associated with each CAD operation type.

Recall that a feature matrix P can be expressed as $P = \left[\mathbf{o^1}, \mathbf{o^2}, \ldots, \mathbf{o^{N_c}}\right]^T$. The vector $\mathbf{o^i} \in \mathbb{R}^7$ is a CAD operation vector which can be noted as $\mathbf{o^i} = \begin{bmatrix} t \\ \mathbf{p} \end{bmatrix}$, where t is an integer indicating the operation type, and $\mathbf{p}$ is a vector of integers representing the corresponding parameters (see Section 3 for more details). In what follows, we explain the evaluation metrics using the same notation.

ACP. Accuracy of CAD programs (ACP) is calculated by $\text{ACP} = \frac{1}{\mathbf{N}} \sum_{\mathbf{i=1}}^{\mathbf{N}} \mathbb{I}(\mathbf{P^i} = \widehat{\mathbf{P^i}})$, representing the ratio of the predicted CAD programs that are precisely aligned with the ground truth ones, where N is the total number of test data for evaluation, P^i denotes the ground truth CAD program, while $\hat{P}^i$ represents the corresponding predicted CAD program, $\mathbb{I}(\cdot)$ is the indicator function that returns 1 if the condition is true, and 0 otherwise.

ASOT & EDSOT. Two metrics are defined for evaluating the sequence of operation types: 1) accuracy of the sequence of operation types (ASOT) and 2) edit distance of the sequence of operation types (EDSOT). ASOT assesses the proportion of predicted CAD sequences with operation types (without considering the associated operation parameters) that match exactly the ground truth, as defined by $\text{ASOT} = \frac{1}{\mathbf{N}} \sum_{\mathbf{i=1}}^{\mathbf{N}} \mathbb{I}(\mathbf{P^i}[:, \mathbf{1}] = \widehat{\mathbf{P^i}}[:, \mathbf{1}])$. In addition, $P^i[:, 1]$ represents the first column of P^i which is the sequence of operation types in a CAD program, and similarly for $\hat{P}^i[:, 1]$.

$$M[i,j] = \begin{cases} \max(i,j) & \text{if} \min(i,j) = 0, \\ \min \begin{cases} M[i-1,j]+1 \\ M[i,j-1]+1 \\ M[i-1,j-1]+1_{(a_i \neq b_j)} \end{cases} & \text{otherwise.} \end{cases} \quad (1)$$

TABLE 3. Comprehensive evaluation metrics for the image, the CAD program, and the 3D CAD model

Table (a): Evaluation metrics for the 3D CAD models and images

Evaluation focus	Metrics	Interpretation
CAD programs to 3D models	Parsing rate	Successful rate of parsing the CAD programs to 3D models
Localization and reconstruction of 3D models	Intersection over union (IoU)	Given the successful parsing, the degree of similarity between the reconstructed 3D CAD models and the ground truth models
Image	Mean squared error (MSE)	The measurement of the average squared difference between the pixel values of the rendered image and ground truth image

Table (b): Evaluation metrics for the CAD programs

Evaluation focus		Metrics		Interpretation
		L1: Operation type	L2: Parameter	
H1: Sequence evaluation	The CAD program	Accuracy of the CAD programs (ACP)		The ratio of the predicted CAD programs that precisely align with the ground truth ones
	The sequence of CAD operation types	Accuracy of the sequence of operation types (ASOT)	-	The ratio of the predicted CAD operation type sequences that exactly match the ground truth ones
		Edit distance of the sequence of operation types (EDSOT)	-	The level of similarity between the predicted operation type sequence and the ground truth
H2: Sequence-based Operation type-wise evaluation	The CAD operation type	Accuracy of the operation types (AOT)	-	The proportion of CAD operation types in the predicted sequences that align with their corresponding operation types in the ground truth
		-	Accuracy of the parameter[1] (AP^1)	The agreement of associated parameters when the CAD operation type is correctly predicted considering the sequential order
H3: Set-based operation type evaluation	The multiset of CAD operation types	Multiset similarity of operation types (MSOT)	-	The similarity of predicted CAD operation types in a CAD program to those in the ground truth CAD program without considering the order
		-	Accuracy of the parameter[2] (AP^2)	The agreement of associated parameters when the CAD operation type is correctly predicted without considering the order

In the case of EDSOT, it measures the level of similarity between the predicted CAD operation type sequence and the ground truth. While there exist various metrics to calculate the edit distance, we utilize the Levenshtein distance as shown in Equation (1), which is commonly employed to compare sequential data in applications, such as computational biology [23]. Given two strings a and b of lengths m and n, respectively, the Levenshtein distance $L(a,b)$ can be calculated using dynamic programming. We define a matrix M of size $(m+1)\times(n+1)$, where $M[i,j]$ represents the minimum number of operations (i.e., insertions, deletions or substitutions) required to transform the substring $a[1:i]$ into the substring $b[1:j]$. After calculating the values for all entries of the matrix M, the Levenshtein distance is given by $L(a,b) = M[m,n]$.

AOT. The accuracy of the operation types, denoted as AOT, is computed as the proportion of CAD operation types in the predicted sequences that align with their corresponding operation types in the ground truth, taking into account the order us-

ing AOT $= \frac{\sum_{i=1}^{N}\sum_{j=1}^{l^i}\mathbb{I}(\mathbf{P^i}[j,1]=\widehat{\mathbf{P}}^{\mathbf{i}}[j,1])}{\sum_{i=1}^{N}|(\mathbf{P^i}[:,1])|}$. In addition, the function $|\cdot|$ is employed to determine the length of a sequence, and l^i is defined as $\min(|P^i[:,1]|,|\widehat{P}^i[:,1]|)$, representing the number of operation types that need to be compared in a sequence for the ith data point of the test data for evaluation.

AP^1. The accuracy of parameter^1 (AP^1) is determined by assessing the agreement of associated parameters when the CAD operation type is correctly predicted, considering the sequential order, as defined in Equation (2). Conditions (c1-3) serve as the input criteria for the indicator functions. This metric function serves as the second layer beneath the first layer, AOT, indicating that parameter evaluation occurs exclusively when the operation type is accurately predicted (i.e., c1). For c2, recall our use of 8-bit integers (i.e., $0-255$) to represent the parameter values. Regarding c3, the parameter η denotes the permissible tolerance for differences between the predicted parameters and their ground truth values. For example, given a specific permissive tolerance $\eta \in [0,255]$, if a ground truth parameter value is $z \in [0,255]$, to be counted as a correct prediction, the predicted parameter value $\hat{z}$ must satisfy the following conditions: $|\hat{z}-z| \leq \eta$ and $\hat{z} \in [0,255]$. Furthermore, the summation over k is a consequence of each CAD operation vector $\mathbf{o} \in \mathbb{R}^7$ having its first dimension representing the operation type, while the subsequent dimensions (i.e., dimensions 2-7) pertain to the associated parameters.

$$
\begin{aligned}
\text{AP}^1 &= \frac{\sum_{i=1}^{N}\sum_{j=1}^{l^i}\sum_{k=2}^{7}(\mathbb{I}(\text{c1})\cdot\mathbb{I}(\text{c2})\cdot\mathbb{I}(\text{c3}))}{\sum_{k=2}^{7}\sum_{i=1}^{N}|(P^i[:,1])|} \\
&\text{c1}: P^i[j,1] = \widehat{P}^i[j,1] \\
&\text{c2}: 0 \leq \widehat{P}^i[j,k] \leq 255 \\
&\text{c3}: |P^i[j,k] - \widehat{P}^i[j,k]| \leq \eta
\end{aligned}
\tag{2}
$$

MSOT. The multiset similarity of operation types (MSOT) is a metric that compares the similarity of predicted CAD operation types in a CAD program to those in the ground truth CAD program, without taking the sequential order into account. In mathematics, a set is defined as a collection of elements where the order of these elements is irrelevant and duplicate elements are not permitted. Conversely, a multiset follows a similar principle as a set, but it allows the inclusion of repeated elements. Thus, a set can be seen as a special case of multiset where each element occurs only once. To implement the MSOT, we adapted two commonly used metrics: Tanimoto coefficient (TC) and cosine similarity (CS) in the Cheminformatics field for carrying out molecular similarity calculations [24]. As the order of the elements in a multiset is not concerned in this case, we can represent a multiset as a vector, where each element corresponds to the count of a particular element in the multiset. For instance, in this study, we have a universe of elements for all the operation types $\{0,1,2,3,4,5,6\}$, a multiset of a triangular prism $\{5,0,1,1,1,4,6\}$ can be represented by a vector $[1,3,0,0,1,1,1]$ with each number representing the count of a particular operation type. Denote $\mathbf{a}$ and $\mathbf{b}$ as the vectors of two multisets and the TC between $\mathbf{a}$ and $\mathbf{b}$ can then be calculated as $\text{TC}(\mathbf{a},\mathbf{b}) = (\mathbf{a}\cdot\mathbf{b})/(||\mathbf{a}||^2 + ||\mathbf{b}||^2 - \mathbf{a}\cdot\mathbf{b})$, where $\mathbf{a}\cdot\mathbf{b}$ denotes the dot product between the two vectors (sum of the element-wise multiplication), $||\cdot||$ denotes the Euclidean norm. CS measures the cosine of the angle between two vectors and CS between $\mathbf{a}$ and $\mathbf{b}$ is calculated as $\text{CS}(\mathbf{a},\mathbf{b}) = (\mathbf{a}\cdot\mathbf{b})/(||\mathbf{a}|| \times ||\mathbf{b}||)$.

AP^2. Similar to AP^1, the accuracy of parameter^2 (AP^2) serves as the second layer in the evaluation of CAD operations which can be similarly calculated using Equation (2). Notably, in AP^2, the assessment does not take into account the order of operations. In this study, an operation type can occur multiple times in a CAD program, such as the *Line* operation in a triangular prism. This introduces a challenge regarding which instance of the *Line* operation in the predicted CAD program should be matched with the corresponding instance in the ground truth CAD program for parameter comparison. Despite this challenge, AP^2 retains practical significance, particularly in scenarios where there are no repeated elements in the CAD operations. In addition, it is essential to maintain AP^2 to preserve the integrity of the evaluation system.

4 RESULTS AND DISCUSSION

4.1 EVALUATION OF CAD PROGRAMS

Figure 4 provides a comparison of the Image2CADSeq model's performance, evaluated under two different architectures: TEA and TEVAE. The results of the TEVAE model are detailed in Figure 4 (b). It displays high accuracy in most metrics similar to the baseline performance of the TEA model in Figure 4 (a) but surpasses its performance in ACP and AP^1. Particularly, the ACP metric shows a significant improvement in the TEVAE model and achieves a higher value at $n = 3$ (does not decrease to zero as in (a)). The AP^1 metric also reveals an upward trend, settling at a higher value than previously seen with the TEA model. For a more complete comparison of the two cases, we summarize the results of all metrics at $n = 6$ in Table 4. This summary demonstrates that the TEVAE model surpasses the TEA counterpart, yet the improvement is marginal. This motivates us to perform an in-depth evaluation of the model performance in other

TABLE 4. Results when evaluated at the first 6 operations of the CAD programs

	ACP(↑)	ASOT(↑)	AOT(↑)	AP1(↑)	MSC-TC(↑)	MSC-CS(↑)	EDSOT(↑)
TEA	0.000	0.961	0.991	0.489	0.990	0.994	-0.053
TEVAE	**0.004**	**0.967**	**0.993**	**0.541**	**0.992**	**0.996**	**-0.042**

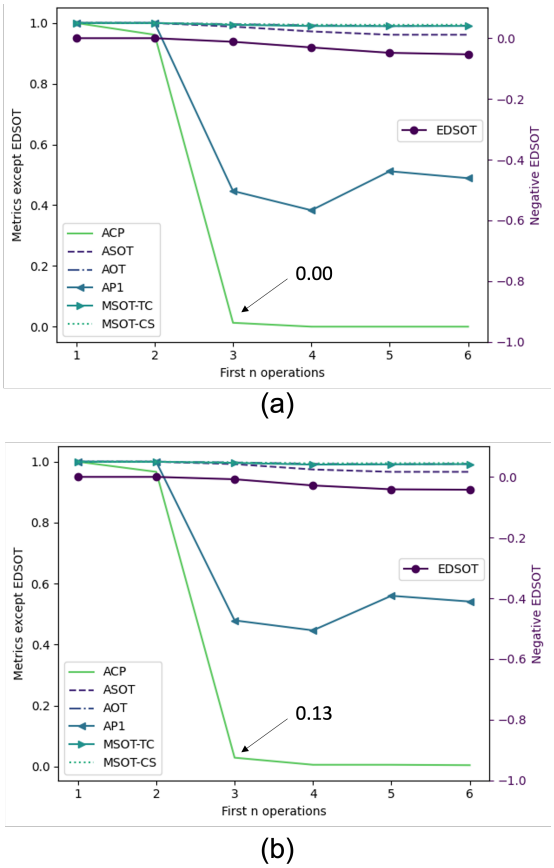


FIGURE 4. Evaluation of the Image2CADSeq model's performance using two distinct architectures: TEA (a) and TEVAE (b)

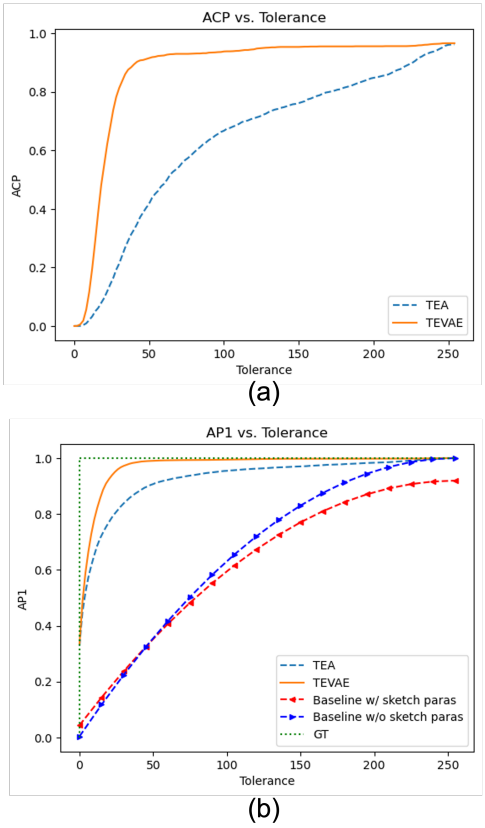


FIGURE 5. Overall parameter accuracy versus the tolerance levels evaluated using (a) ACP and (b) AP1 for the three cases.

aspects beyond the CAD program evaluation, as outlined in the following sections.

4.2 THE OVERALL PARAMETER ACCURACY

The models demonstrate high accuracy in predicting the sequence of CAD operations but are less precise in parameter prediction. To facilitate a clearer comparison between the two cases with respect to parameter prediction accuracy, we have included Figure 5 to illustrate the relationship between parameter accuracy and tolerance (of the difference between the ground truth (GT) parameter value and the predicted value) using metrics ACP and AP^1.

Especially, in Figure 5 (b), the blue dashed line with triangle markers represents the AP^1 value achieved by randomly guessing parameters given a specific tolerance, but without considering the *Sketch* parameter (i.e., the identifier of the sketch plane I) whose values are not allowed for tolerance. We derived the equation for the random model in Equation (3). The equation can be simplified to $AP^1 = (-\eta^2 + 511\eta + 256)/65536$. This line acts as a baseline to evaluate the model's effectiveness in accurately predicting parameters if the sketch parameter is not considered.

$$AP^1 = \frac{1}{256}\left(\frac{(2\eta+1)(256-2\eta)+2\left(\frac{2\eta(1+2\eta)}{2}-\frac{\eta(1+\eta)}{2}\right)}{256}\right) \quad (3)$$

$$AP^1 = \frac{1}{3}\cdot\frac{11}{91}+\frac{(-\eta^2+511\eta+256)}{65536}\cdot\frac{80}{91} \quad (4)$$

To consider the *Sketch* parameter I, we need to take into account the characteristics of the dataset (i.e., the ratio of each parameter taken among all possible parameters in the design representation of the CAD programs as outlined in Section 3.1). Accordingly, we obtain Equation (4) plotted as the red dashed line with triangle markers in Figure 5 (b). Note that the number of the sketch parameter takes $\frac{11}{91}$ of all parameters. Additionally, a green dotted line is used to indicate the ideal scenario where the

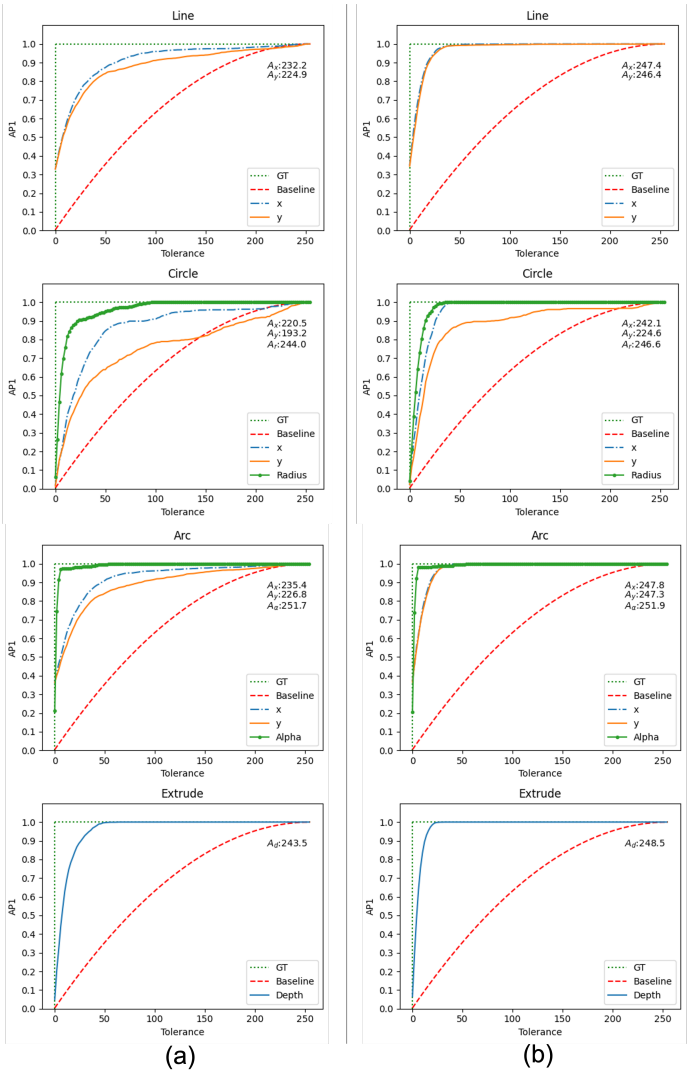


FIGURE 6. The variation of AP1 versus tolerance for the operation parameters for the *Line*, *Circle*, *Arc*, and *Extrude* operations, in that order. Column (A) TEA and Column (b) TEVAE.

parameters are perfectly predicted with zero tolerance (i.e., GT). Other lines in Figures 5 (a) and (b) depict the corresponding metric values for different cases, providing a comprehensive view of the model's performance in parameter prediction.

In both (a) and (b) of Figure 5, we consistently see that the metric values increase with rising tolerance levels. A notable point in Figure 5 (a) is that the ACP values for all two cases reach their highest at a tolerance of 255 and the corresponding values are 0.961 and 0.967 for each case, in accordance with the ASOT values presented in Table 4. This can be interpreted as the result that when we evaluate the entire CAD program in terms of ACP given that all parameters are accurately predicted, we are essentially assessing ASOT. In Figure 5 (b), a crucial observation is that both lines exceed the baseline of the random guess of parameters. This indicates that the models are effectively learning parameter prediction.

Additionally, a significant observation in both figures is how differently the models respond to changes in tolerance. Specifically, the TEVAE model exhibits a faster increase for ACP and AP1 to changes in tolerance in contrast to the TEA model, suggesting that the TEVAE model excels in parameter prediction compared to the TEA model.

4.3 PARAMETER ACCURACY OF OPERATION TYPES

To gain more insight into how the models perform in parameter prediction, we plotted the variation of AP^1 versus the tolerance for the operation parameters for each CAD operation type in Figure 6.

Spanning columns (a) to (b), the rows in each column show variations in AP^1 against tolerance levels ($\eta = 0 - 255$) for specific parameters, corresponding to different CAD operations, *Line*, *Circle*, *Arc*, and *Extrude*. These results look into the model's adaptability and accuracy across various CAD operations. Each figure includes a red dashed line representing the baseline as defined in Equation (3). In addition, the green dotted line illustrates the perfect prediction of the parameters with zero tolerance. The other lines show the AP^1 for specific parameters related to the respective CAD operations. To facilitate a more quantitative comparison of how well the parameters are predicted, we also computed the area under the curve (AUC) for each parameter, as indicated in the upper right corner of each figure. Generally, the parameters of *Circle* are particularly harder for the model to predict compared to other operations (i.e., *Line*, *Arc*, and *Extrude*) in each case. In addition, the coordinates x and y of *Line*, *Circle*, and *Arc* are more difficult to predict compared to other parameters such as the radius of a Circle and the Alpha angle of an *Arc*. This result indicates that the model is better at predicting shapes than the position of shapes.

Comparing the two cases, in Column (b), the results demonstrate a better performance compared to Column (a). All metric values not only surpass those in Column (a), but they also show a further deviation from the baseline, indicating a significant enhancement of the model's predictive performance. These values are closely approaching the ground truth (GT) line, underscoring the refined ability of the TEVAE model to learn from the training data.

4.4 EVALUATION OF 3D CAD MODELS AND IMAGES

Table 5 shows the summary of the parsing rate, intersection over union (IoU), and mean squared error (MSE) outlined in Figure 3 of the two cases. We perform an analysis of IoU and MSE by calculating the mean and standard deviation for each metric in the table. Aligning with our observations in the evaluation of the

TABLE 5. The mean and standard deviation of parsing rate, intersection over union (IoU), and mean squared error (MSE) of the two cases

	TEA	TEVAE
Parsing rate (↑)	0.55	**0.58**
IoU (↑)	0.214 ± 0.225	$\mathbf{0.431 \pm 0.255}$
MSE (↓)	0.022 ± 0.012	$\mathbf{0.017 \pm 0.011}$

CAD programs, as introduced in previous sections, the TEVAE achieves a better performance compared to the TEA model.

4.5 DISCUSSION

The effectiveness of the TEVAE architecture is demonstrated through improved performance across various metrics, significantly exceeding the results achieved in the TEA architecture. The superior performance of the TEVAE model can be largely attributed to the three advantages offered by VAEs. (1) Unlike traditional AEs, VAEs create a latent space that follows a well-defined and continuous distribution, such as the Gaussian distribution typically used. This design facilitates smoother interpolation between data points, enhancing the capture of meaningful variations in CAD designs. (2) The encoder in a VAE is more efficient in extracting relevant and prominent features from CAD programs than a standard AE. This efficiency stems from the VAE's focus on capturing the underlying data distribution, rather than merely replicating input data. (3) The inclusion of the KL-divergence term in the VAE's loss function helps reduce overfitting. It promotes the model to capture a broader data distribution rather than memorizing specific instances. This enhances TEVAE's generalizability on new, unseen data.

It is indeed a challenge in our research to further improve the prediction accuracy of operation parameters. An important observation from our experiments is the near-perfect reconstruction capability in Stage 1 training with an accuracy of the CAD program (ACP), up to 99.9%. However, the problem arises during Stage 2 training, which involves regressing the latent space learned in Stage 1 using images as input. The difficulty lies in aligning the latent representation of the image with that of the CAD programs. To address this issue, we plan to explore modal alignment techniques as introduced in the recent literature [18, 25]. They could offer a promising solution to unify different modalities in a single latent space to promote cross-modal synthesis.

Limitations: We have been focused on synthesizing simple geometrical shapes, such as cylinders and tri-prisms. While these basic geometries are fundamental to more complex designs, our focus on them has limited the network's capability to handle intricate, real-world design tasks. Recognizing this, we acknowledge the need to train the Image2CADSeq model with more diverse and complex datasets to tackle advanced design challenges. We plan to collect more sophisticated geometries that mirror the complexities encountered in actual design environments, often embodied as assemblies comprising multiple interconnected components. To achieve this, we plan to explore two primary strategies: (1) Enhancing our data synthesis pipeline: We intend to integrate a wider range of complex geometries into our current data synthesis pipeline. This expansion will allow the network to learn from a broader spectrum of shapes and structures, better preparing it for real-world applications. (2) Using real-world design datasets: Another avenue involves harnessing datasets that include historical CAD modeling process data. An example is the Autodesk Fusion 360 Gallery dataset [9], which offers a rich source of real-world design examples. Our objective here is to extract CAD sequences that correspond to more intricate designs. This approach will enable the network to learn from actual design processes, further enhancing its applicability to practical scenarios. We also recognize the limitations of the existing model, which relies on rendered images as input. This may lead to suboptimal performance with real-world images or photographs for real-world applications. To overcome these challenges, we aim to enhance the model's capability to handle a diverse range of images featuring various colors, textures, perspectives, and light. Specifically, implementing data augmentation techniques, such as introducing objects in different colors and under various lighting conditions or backgrounds, is a potential solution. It is anticipated to enhance the model's training dataset and improve its understanding of a broader range of real-world image data.

The result underscores the need for further enhancements in the Image2CADSeq model to improve its accuracy in inferring the CAD representations of images of real-world objects. In particular, enhancing the model's ability to accurately predict parameters is essential to improve the parsing rate. Predicting CAD sequences from images could transform the reverse engineering process and expedite conceptual design. Our research leads the way in this field and indicates the potential of Image2CADSeq. However, further investigation is needed to compare the proposed model with other models in the literature to more comprehensively understand its performance.

5 CONCLUSION

In this study, we have developed a novel Image2CADSeq model to predict CAD sequences from images. This network, particularly exemplified by the performance of the TEVAE model, aims to revolutionize design methodologies by enabling the conversion of images into operational CAD sequences. A CAD sequence offers more benefits than pure 3D CAD models, such as greater flexibility in modifying CAD operations and managing the historical process/knowledge of CAD model construction.

For training purposes, our focus is on synthesized data representing simple shape primitives. In addition, we propose an

evaluation framework that can comprehensively assess model performance. The results obtained are very promising, yet improvement can still be made. Therefore, our future efforts will be directed towards (1) Enhancing geometric complexity. We will expand the model's capabilities to encompass a broader spectrum of geometries. This expansion aims to align the model more closely with those in real-world design applications; (2) Incorporating diverse design data. A key area of development involves the integration of more varied and realistic design datasets. This can greatly facilitate the machine learning process; (3) Advancing training methodologies. We plan to explore innovative network architectures and training methodologies to improve the efficiency and adaptability of the model; (4) Incorporating industry standards. Engaging with industry experts will be crucial to guide the development of the model. Their insights will ensure that the model meets practical needs and adheres to industry standards.

In summary, the proposed approach has significant potential to lead to transformative changes in existing CAD systems, revolutionizing the product development cycle. Additionally, it has the potential to promote the democratization of design, allowing people with limited experience or expertise to actively participate in CAD. For example, this approach can help regular customers engage in product design and concept generation, promoting personalized design and creation and human-centered generative design [18, 26].

ACKNOWLEDGMENT

The authors gratefully acknowledge the financial support from the National Science Foundation through award 2207408.

DISCLAIMER

This paper presents the partial findings of our work in a preprint on arXiv (arXiv: 2501.04928), which was previously peer-reviewed for the Journal of Mechanical Design. The work in the preprint is currently under revision and will be submitted to JMD in the future. We confirm that this paper has not been published previously elsewhere. There is no existing or potential future copyright conflict, as arXiv does not claim copyright over posted preprints.

REFERENCES

[1] Rosato, D., and Rosato, D., 2003. "5 - computer-aided design". In *Plastics Engineered Product Design*, D. Rosato and D. Rosato, eds. Elsevier Science, Amsterdam, pp. 344–380.

[2] Varady, T., Martin, R. R., and Cox, J., 1997. "Reverse engineering of geometric models—an introduction". *Computer-aided design,* ***29***(4), pp. 255–268.

[3] Buonamici, F., Carfagni, M., Furferi, R., Governi, L., Lapini, A., and Volpe, Y., 2018. "Reverse engineering modeling methods and tools: a survey". *Computer-Aided Design and Applications,* ***15***(3), pp. 443–464.

[4] Uy, M. A., Chang, Y.-Y., Sung, M., Goel, P., Lambourne, J. G., Birdal, T., and Guibas, L. J., 2022. "Point2cyl: Reverse engineering 3d objects from point clouds to extrusion cylinders". In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 11850–11860.

[5] Ren, D., Zheng, J., Cai, J., Li, J., and Zhang, J., 2022. "Extrudenet: Unsupervised inverse sketch-and-extrude for shape parsing". In European Conference on Computer Vision, Springer, pp. 482–498.

[6] Lambourne, J. G., Willis, K., Jayaraman, P. K., Zhang, L., Sanghi, A., and Malekshan, K. R., 2022. "Reconstructing editable prismatic cad from rounded voxel models". In SIGGRAPH Asia 2022 Conference Papers, pp. 1–9.

[7] Li, P., Guo, J., Zhang, X., and Yan, D.-M., 2023. "Secadnet: Self-supervised cad reconstruction by learning sketch-extrude operations". In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 16816–16826.

[8] Li, X., Xie, C., and Sha, Z., 2022. "A predictive and generative design approach for three-dimensional mesh shapes using target-embedding variational autoencoder". *Journal of Mechanical Design,* ***144***(11), p. 114501.

[9] Willis, K. 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". *ACM Transactions on Graphics (TOG),* ***40***(4), pp. 1–24.

[10] Wu, R., Xiao, C., and Zheng, C., 2021. "Deepcad: A deep generative network for computer-aided design models". In Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 6772–6782.

[11] Xu, X., Willis, K. D., Lambourne, J. G., Cheng, C.-Y., Jayaraman, P. K., and Furukawa, Y., 2022. "Skexgen: Autoregressive generation of cad construction sequences with disentangled codebooks". In International Conference on Machine Learning, PMLR, pp. 24698–24724.

[12] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I., 2017. "Attention is all you need". *Advances in neural information processing systems,* ***30***.

[13] Xu, X., Peng, W., Cheng, C.-Y., Willis, K. D., and Ritchie, D., 2021. "Inferring cad modeling sequences using zone graphs". In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 6062–6070.

[14] Li, C., Pan, H., Bousseau, A., and Mitra, N. J., 2020. "Sketch2cad: Sequential cad modeling by sketching in context". *ACM Transactions on Graphics (TOG),* ***39***(6), pp. 1–

14.
[15] Li, C., Pan, H., Bousseau, A., and Mitra, N. J., 2022. "Free2cad: Parsing freehand drawings into cad commands". *ACM Transactions on Graphics (TOG),* ***41***(4), pp. 1–16.
[16] Carlier, A., Danelljan, M., Alahi, A., and Timofte, R., 2020. "Deepsvg: A hierarchical generative network for vector graphics animation". *Advances in Neural Information Processing Systems,* ***33***, pp. 16351–16361.
[17] Jarrett, D., and van der Schaar, M., 2020. "Target-embedding autoencoders for supervised representation learning". In International Conference on Learning Representations.
[18] Li, X., Wang, Y., and Sha, Z., 2023. "Deep learning methods of cross-modal tasks for conceptual design of product shapes: A review". *Journal of Mechanical Design,* ***145***(4), p. 041401.
[19] Mostajabi, M., Maire, M., and Shakhnarovich, G., 2018. "Regularizing deep networks by modeling and predicting label structure". In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 5629–5638.
[20] He, K., Zhang, X., Ren, S., and Sun, J., 2016. "Deep residual learning for image recognition". In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770–778.
[21] Dice, L. R., 1945. "Measures of the amount of ecologic association between species". *Ecology,* ***26***(3), pp. 297–302.
[22] Huttenlocher, D. P., Klanderman, G. A., and Rucklidge, W. J., 1993. "Comparing images using the hausdorff distance". *IEEE Transactions on pattern analysis and machine intelligence,* ***15***(9), pp. 850–863.
[23] Navarro, G., 2001. "A guided tour to approximate string matching". *ACM computing surveys (CSUR),* ***33***(1), pp. 31–88.
[24] Bajusz, D., Rácz, A., and Héberger, K., 2015. "Why is tanimoto index an appropriate choice for fingerprint-based similarity calculations?". *Journal of cheminformatics,* ***7***(1), pp. 1–13.
[25] Song, B., Zurita, N. S., Zhang, G., Stump, G., Balon, C., Miller, S., Yukish, M., Cagan, J., and McComb, C., 2020. "Toward hybrid teams: A platform to understand human-computer collaboration during the design of complex engineered systems". In Proceedings of the Design Society: DESIGN Conference, Vol. 1, Cambridge University Press, pp. 1551–1560.
[26] Demirel, H., Goldstein, M., Li, X., and Sha, Z., 2023. "Human-centered generative design framework: An early design framework to support concept creation and evaluation". *International Journal of Human-Computer Interaction*.