Sketch-to-CAD: A Deep Learning Approach for Predicting CAD Steps from Isometric Sketches

Benjamin Randoing
Department of Mechanical Engineering
Stanford University
bar39@stanford.edu

Ghadi Nehme
Department of Mechanical Engineering
Stanford University
ghadi@stanford.edu

Abstract

Generative artificial intelligence models present a growing technology with potential to transform standard practices in many disciplines. 3D Computer Aided Design (CAD) is the industry standard for many companies for rapid prototyping, professional design, and simulation under load. A core component of CAD workflows is a customizeable timeline with history of each step, such as sketches or extrudes, so edits upstream in the timeline will propagate through to appropriately modify the current design. However, CAD design is a laborious and time intensive process that often warrants sketching a preliminary 2D model. Here we show Sketch-to-CAD, a generative AI model to expedite CAD development by transforming a 2D sketch into a 3D CAD workflow with the appropriate timeline. Four encoder decoder model architectures were trained. CAD dataset for 2D input sketches and output cad vector instructions was leveraged from the ABC-dataset. Various combinations of CNN and transformer encoders with transformer and LSTM decoders dictate the model architectures, and the highest performing model featured a CNN encoder with a pretrained transformer decoder. Notably, the model featured command and parameter test accuracies of 93.50 and 68.30 % and resulted in saliency images that clearly focus on the input sketch lines. Accordingly, the Sketch2CAD model presents a disruptive technological advancement to support CAD design workflows. Expedited CAD design subsequently empowers engineering teams and corporations to iterate and design new products more often.

1. Introduction

Computer-Aided Design (CAD) is a prominent technology utilized across various engineering disciplines to facilitate 3D design, development, and simulation analysis. A singular CAD file comprises an amalgamation of operations, such as 2D sketches, extrudes, fillets, and chamfers. This CAD file effectively establishes a chronological framework, enabling editors to make upstream adjustments that seamlessly propagate throughout the design. While 3D design plays a pivotal role in prototyping and product development, the expeditious nature of 2D sketching empowers engineers to swiftly brainstorm and explore numerous design concepts. Consequently, it is customary to transform a 2D sketch into a CAD design. However, the process of converting an isometric 2D sketch into a comprehensive 3D model is arduous and time-consuming. By developing a pipeline for 3D design that provides an adjustable model timeline, one can facilitate the refinement of a 3D model, thereby enhancing the accessibility of 3D design and presenting an opportunity to revolutionize global workflows. In this context, we introduce a novel model that exhibits the ability to extract essential features from isometric sketches and generate the sequential steps necessary for constructing a CAD model. We will be exploring various architectures of CNN, Transformers and LSTM to encode and decode the isometric sketch into steps of a CAD model.

2. Related Work

Prior works have explored the use of sketching in the context of Computer Aided Design (CAD). One of the seminal works in this area is ”Sketch2CAD: Sequential CAD Modeling by Sketching in Context” [1], which develops an autoencoder latent space for generating CAD timelines for a 3D design. Subsequently, a generative adversarial network (GAN) is developed using the latent-GAN technique to randomly produce vectors is the latent space that result in realistic 3D CAD timelines. Additionally, a dataset of 178,238 CAD models and timelines was generated with designs that exclusively use simple sketch and extrude commands.

Another related work is ”A 2D Sketch-Based User Interface for 3D CAD” [2], which presents a user interface that allows users to search for and retrieve 3D CAD models based on sketches. This system typically use similarity metrics to match input sketches with existing models in a database.

Despite modern efforts to develop models for CAD specific applications, previous generative models for 3D design focus on non-customizeable formats [3]. OpenAI's Point-E produces point-cloud 3D outputs from complex prompts [4]. While point clouds, meshes, and .stl files enable finer and printable representations of 3D volumes, software for adjusting such designs does not exist.

3. Dataset and Features

We utilize a subset of the DeepCAD Dataset, which consists of 166,225 CAD models [5]. This dataset is derived from the ABC dataset [6], a collection of one million CAD models. Each CAD model in the DeepCAD Dataset is represented as a sequence of steps for sequentially building the sketches and extrusions of the model.

3.1. Input: Isometric Sketches

The input of our model is an isometric sketch of the CAD file. To obtain it, we load the STEP file associated with each CAD model in Fusion 360 [7] and export an isometric sketch of the part (Fig 1). Each image is inverted to represent sketched lines as positive signal. The image is normalized between 0 and 1. The initial image has size 1024 × 1024, it is resized to 256 × 256 and cropped to a size of 240 × 240, to reduce the size of the input.

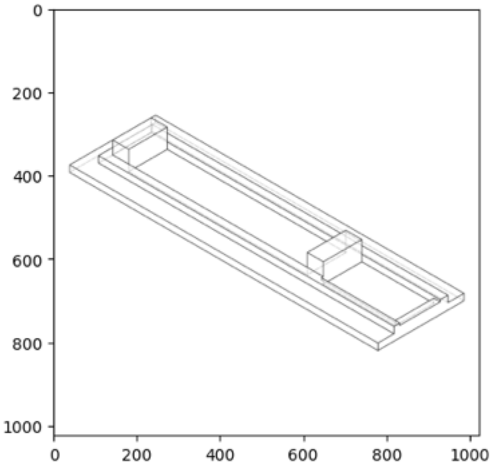
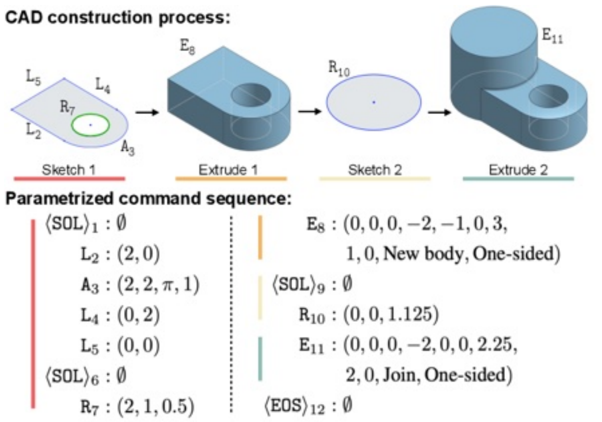


Figure 1. Example of a CAD drawing to be used as a proxy for hand drawn sketches during model development

3.2. Output: Steps of CAD model

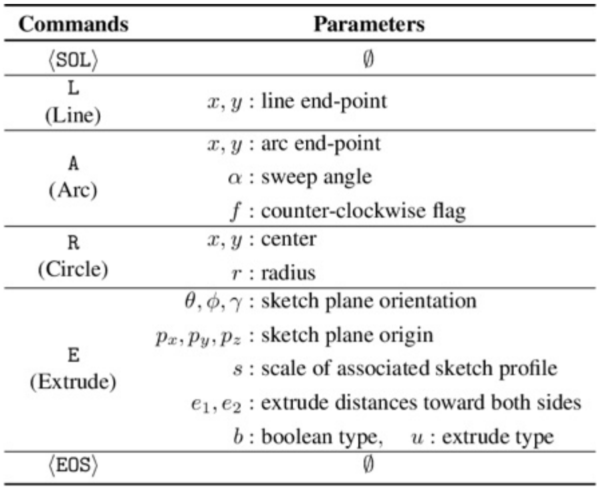
A CAD model, denoted as M, is defined by a series of curve commands and extrusion commands Fig. 2). In other words, M can be represented as a command sequence $M = [C_1, ..., C_{N_c}]$, where each C_i takes the form (t_i, p_i). Here, t_i represents the command type, and p_i represents the corresponding parameters. The commands and parameters are detailed in Figure 3.



$\langle \texttt{SOL} \rangle_1 : \emptyset$
$L_2 : (2, 0)$
$A_3 : (2, 2, \pi, 1)$
$L_4 : (0, 2)$
$L_5 : (0, 0)$
$\langle \texttt{SOL} \rangle_6 : \emptyset$
$R_7 : (2, 1, 0.5)$
$E_8 : (0, 0, 0, -2, -1, 0, 3,$ $1, 0,$ New body, One-sided)
$\langle \texttt{SOL} \rangle_9 : \emptyset$
$R_{10} : (0, 0, 1.125)$
$E_{11} : (0, 0, 0, -2, 0, 0, 2.25,$ $2, 0,$ Join, One-sided)
$\langle \texttt{EOS} \rangle_{12} : \emptyset$

Figure 2. A CAD model example specified by the commands in Fig. 3. (Top) the CAD model's construction sequence, annotated with the command types. (Bottom) the command sequence description of the model. [5]

After normalization steps, the final network-friendly representation of the CAD step is a 17 × 1 vector: $[t_i, x, y, \alpha, f, r, \theta, \phi, \gamma, p_x, p_y, p_z, s, e_1, e_2, b, u]$. Here, t_i is an integer between 0 and 5, and the other elements are integers between -1 and 255. A value of -1 indicates that the specific element is not used in that step. Representing the continuous variables of a CAD instruction (ie: x, y, or z, location) allows expressing each values with 8-bit numbers.



Commands	Parameters
$\langle \texttt{SOL} \rangle$	$\emptyset$
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
$\langle \texttt{EOS} \rangle$	$\emptyset$

Figure 3. CAD commands and their parameters. ⟨ SOL ⟩ indicates the start of a loop; ⟨ EOS ⟩ indicates the end of the whole sequence. [5]

Additionally, when creating CAD designs, it is important to maintain certain geometric relationships such as parallel and perpendicular sketch lines. However, directly generating continuous parameters through parameter regression can lead to errors that violate these relationships. To overcome this, parameter quantization is used to categorize parameters into specific levels, allowing the network to better uphold learned geometric relationships.

For this project, we will limit the number of steps to $N_c = 60$ and pad all step sequences to 60 using the empty step: $\langle$ EOS $\rangle$.

Thus, we have one classification task for the CAD commands with the number of classes equals 6 and and 16 other classification tasks for the CAD arguments with the number of classes equal 256. We can see in Figures 4 and 5, the data distribution across the classes.

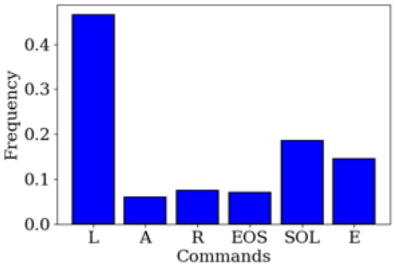


Figure 4. Frequency of the classes of each command in the dataset.

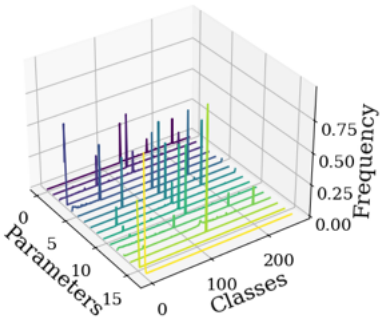


Figure 5. Frequency of the classes of each parameter in the dataset.

We observe a large data imbalance in the dataset. There are more Lines than any other CAD step (Fig. 4). We also observe that for almost each parameter, there is a dominant class that can be observed with the peaks in Fig. 5.

4. Methods

4.1. Evaluation

To evaluate our models, we calculate the Cross Entropy Loss between the ground truth sequence of CAD steps and the predicted sequence.

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

where $\ell(.,.)$ is the standard Cross-Entropy, N_p is the number of parameters (16) and β is a hyperparameter that compares the importance of both terms in the optimization process. For the commands and parameters that are not used in a step (labeled as -1), their contribution to the loss is ignored. Additionally, we developed saliency plots for a small subset (5 images) of the test set to identify which regions of the input images held the greater influence on the CAD sequence predictions.

Model evaluation includes a qualitative component of inspecting the resulting CAD file. Timelines and realistic extrusions are visualized to understand the extent to which the CAD sequences are realistic.

The quantitative evaluation will include the accuracy of the CAD commands and the accuracy of the CAD parameters by comparing them to the ground truth. We will also compare the number of CAD models that are perfectly predicted based on the ground truth.

4.2. Model Architectures

We explored several deep learning architectures for the Sketch-to-CAD task. The architectures we experimented with include Convolutional Neural Networks (CNNs), Transformers, and Long Short-Term Memory (LSTM) networks. We try different architectures of sketch images encoders and CAD steps decoders. The idea is to encode the input image into a latent vector z of dimension 256 and then input this vector into a decoder that will output the CAD steps.

4.2.1 Convolutional Neural Networks (CNNs) Encoder

CNNs are widely used for image-related tasks due to their ability to capture spatial dependencies in the data. We designed a CNN-based architecture that takes the input isometric sketch image and learns to extract relevant features for predicting the CAD steps. The CNN architecture consists of multiple convolutional layers followed by a fully connected layer.

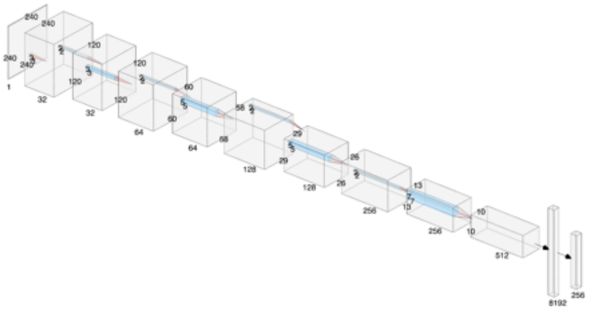


Figure 6. CNN Encoder Model Architecture

4.2.2 VGG Encoder

We implement a standard CNN architecture, VGG-19 depicted in Figure 7, as a starting point to encode the 2D isometric view sketch inputs. Once we obtain a fully-connected layer, we tune additional layers using the output ground truth data. The first model design featured a unfrozen VGG model with two supplemental fully-connected layers with batch normalization and dropout (p=0.5). The objective of the encoder layer is to create a latent vector that may be decoded using the DeepCAD transformer decoder from a pretrained autoencoder.

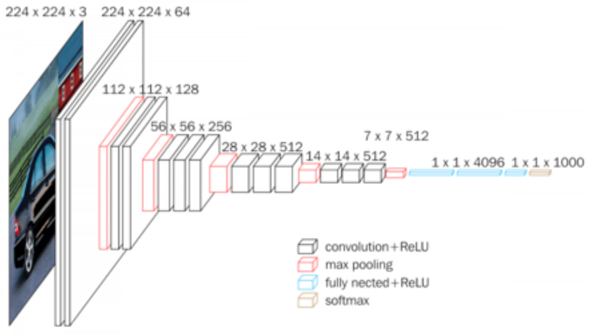


Figure 7. VGG-19 Encoder Model Architecture [8]

4.2.3 Transformers Encoder

Transformers have gained significant attention in natural language processing tasks due to their ability to capture long-range dependencies. We adapted the transformer architecture for our Sketch-to-CAD task by treating the input isometric sketch as a sequence of pixel patches. Each patch in the sketch image is treated as a token, and the transformer model learns to encode the spatial relationships between the pixel patches. The transformer architecture consists of multiple encoder layers, followed by a fully connected layer and a tanh activation (Fig. 8).

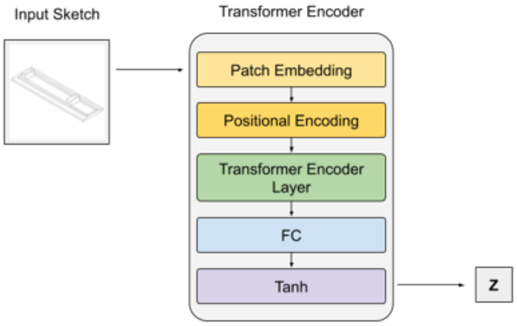


Figure 8. Transformer Encoder Model Architecture

4.2.4 Long Short-Term Memory (LSTM) Decoder

LSTM networks are a type of recurrent neural network (RNN) that are capable of learning long-term dependencies in sequential data. We designed an LSTM-based architecture for our Sketch-to-CAD task, which takes in the latent vector z. The LSTM network learns to process the sequential information and predict the corresponding CAD steps. We use a one-to-many LSTM architecture, we have 60 LSTM cells, each one predicting one step of the CAD sequence by taking in the hidden state of the previous cell and the latent vector z (Fig. 9).

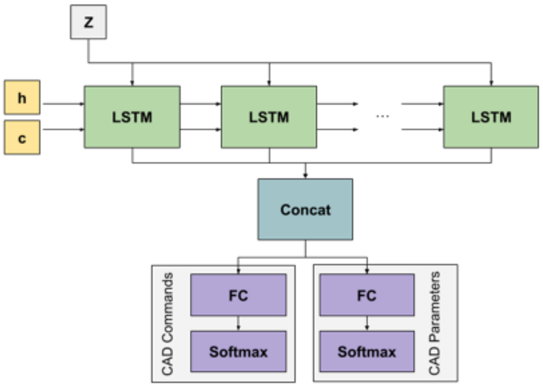


Figure 9. LSTM Decoder Model Architecture

4.2.5 Transformers Decoder

Built on Transformer blocks, the transformer decoder from the previously trained Deep CAD autoencoder depicted in Figure 10 is implemented [5].

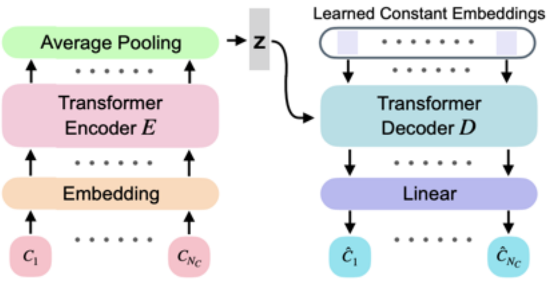


Figure 10. Deep CAD Autoencoder Architecture [5]

We will be using the decoder of this model and freeze its parameters during training. The decoder takes as input learned constant embedding while also attending to the latent vector z. Output from the last Transformer block is fed into a linear layer to predict a CAD command sequence $\hat{M} = [\hat{C}_1, \ldots, \hat{C}_{N_c}]$, including both the command type $\hat{t}_i$ and parameters $\hat{p}_i$ for each command. As opposed to the auto regressive strategy commonly used in natural language processing, Deep CAD adopts the feed-forward strategy. and the prediction of their model can be factorized as

$$p(\hat{M}|z,\theta) = \prod_{i=1}^{N_c} p(\hat{t}_i, \hat{p}_i|z,\theta)$$

where θ denotes network parameters of the decoder.

4.3. Model Implementation and Training

We decide to explore four combinations of these encoders and decoders:

- VGG Encoder and Transformer Decoder
- CNN Encoder and LSTM Decoder
- Transformer Encoder and Transformer Decoder
- CNN Encoder and Transformer Decoder

During training, we used the Adam optimizer. We employed early stopping based on the validation loss to prevent overfitting. The models were trained for a maximum of 50 epochs, with batch size set to 256. The learning rate of each model can be found in the following table:

Table 1. Learning Rates for the different models

Model	Learning Rate
VGG — Transformer	0.01
CNN — Transformer	0.001
Transformer — Transformer	0.00001
CNN — LSTM	0.005

5. Experiments, Results and Discussion

Each of the four model architectures was trained using a shuffled subset comprising 90% of the total data. The models were subsequently validated and tested using 5% of the data. For training, validation, and testing the predictive accuracy was obtained for the command and parameters. In each CAD step, a single command indicates whether the subsequent parameters are for a sketch, arc, circle, extrude, start token, or end token. The accuracies for both the commands and parameters in Table 2 indicate the model with a CNN encoder and transformer decoder resulted in the greatest accuracies for both commands and parameters.

5.1. Saliency Plots

Saliency plots of 3D CAD drawings provide a concise and informative visualization of the pixel components that have the greatest influence on the softmax logits. These plots offer a robust and clear representation of the relative importance of different pixels in determining the final output. By analyzing the saliency plots in 11, one can easily identify the specific areas within the 2D sketches that contribute to the decision-making process of the predictive models. The Transformer — Transformer model and CNN — Transformer model contain reasonable signal amplitude; however, the CNN — LSTM and VGG — Transformer models demonstrate limited signal amplitude in the saliency plot.

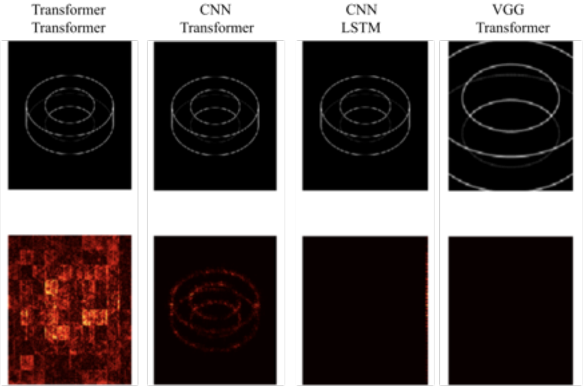


Figure 11. Comparative Saliency Visualizations

The saliency visualizations in Figure 11 highlight the ability of the model with a CNN encoder and Transformer decoder to recognize the attributes of the input image that correspond to a sketch. The VGG encoder model required cropped images to match the encoder architecture and featured the worst quantitative performance. Additionally saliency plots in Figure 12 confirm the model performance of the CNN - Transformer as it pertains to recognizing the important regions of the input image. Moreover, we also

Table 2. Command and Parameter Model Accuracies

Model	Train C	Train P	Val C	Val P	Test C	Test P
VGG — Transformer	76.44	50.39	75.59	50.41	75.81	50.58
CNN — Transformer	97.04	80.72	92.30	71.33	**93.50**	**68.30**
Transformer — Transformer	91.61	71.40	91.09	69.59	90.53	64.26
CNN — LSTM	78.28	59.95	78.74	59.86	78.13	55.88

observe the patterns of the patches that are used in the transformer encoder and how they influence the output.

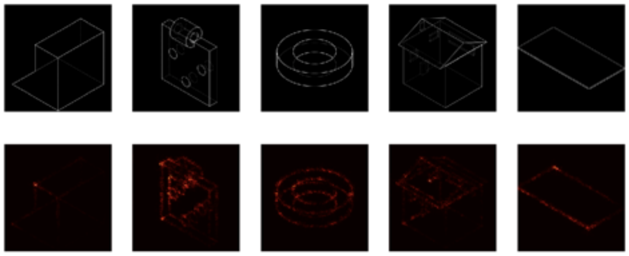


Figure 12. Saliency Visualizations for CNN — Transformer Model

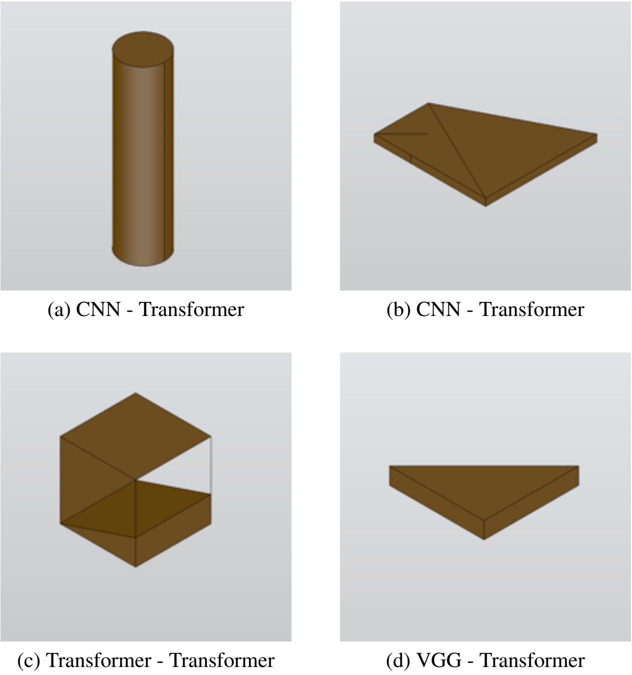
This explains the accuracies observed in Table 2. In fact, for the CNN-LSTM and VGG-Transformer model we observe that the model wasn't able to capture key features of the image which leads to low accuarcies compared to the other two models Transformer - Transformer and CNN - Transformer which were able to get a better extraction of features as seen in the saliency plots and thus, lead to better results.

5.2. CAD Predictions

After training each of the four model architectures, sample outputs were obtained at test time to visualize examples of CAD outputs. Successful CAD outputs depicted in Figure 13 were results of the three architectures with the DeepCAD transformer decoder.

Table 3. Number of fully accurate CAD part for the different models in the test dataset

Model	Fully accurate CAD parts
VGG — Transformer	0
CNN — Transformer	109
Transformer — Transformer	19
CNN — LSTM	0



(a) CNN - Transformer

(b) CNN - Transformer

(c) Transformer - Transformer

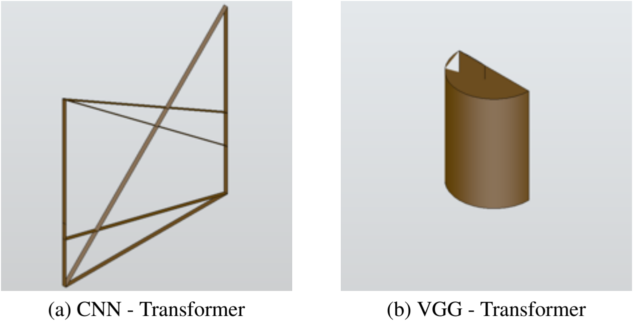
(d) VGG - Transformer

Figure 13. Successful CAD Predictions

Despite improvements in the accuracy of the models, notably the CNN encoder — Transformer decoder model, predicting CAD steps demonstrated marginal improvements due to the high degree of accuracy required to ensure the sketch lines form closed entities to be extruded. In the cases where the sketch lines do not form closed shapes, the predicted CAD as depicted in Figure 14 was an infeasible design. In addition to the unrealistic CAD predictive outputs from the three models with the DeepCAD transformer decoder, the LSTM decoder model resulted in identical failing predictions regardless of the input sketch. This might be due to the data imbalance observed in Fig. 5. In fact, we have parameters occurring very often, almost more than 70% of the time, which may have lead the model to converge towards these parameter values with high frequency peaks.

Thus, visually inspecting the generated CAD steps and looking at the number of succesful CAD parts generated (Table 3), we observe that the CNN - LSTM and VGG - Transformer models weren't able to generate good CAD parts. On the other side, we see that the Transformer -

Transformer and CNN - Transformer models were able to generate CAD parts made of a few elementary shapes like cylinders, cubes and triangles. Thus, these models have the potential of being able to be improved to potentially create more complex CAD models.


(a) CNN - Transformer

(b) VGG - Transformer

Figure 14. Unsuccessful CAD Predictions

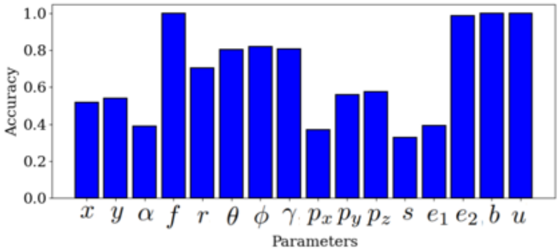


Figure 15. Parameters Accuracies for the CNN encoder - Transformer Decoder Model

We decide to take a closer look at the accuracies of each of the 16 parameters predicted for the best model we got, the CNN encoder - Transformer Decoder model. We observe that many parameters have very high accuracies, while others barely reach 40%. We see that parameters related to coordinates like the point coordinates (x, y) and the plane origin coordinates (p_x, p_y, p_z), and the parameter related to scaling s are the ones which are associated with the lowest accuracies (Fig. 15). This means that the failure we observe in our CAD parts is due to this inability of extracting coordinates of key points or the scaling of the different components by simply looking at a 2D sketch.

5.3. Limitations

While the CNN-Transformer model showed promising results, there are several limitations to consider:

CAD step representation: The CAD steps were represented as a sequence of discrete symbols in this study. This representation may introduce challenges in capturing fine-grained details and continuous variations in the CAD steps. Moreover, we observed a highly imbalanced data where we have parameters occurring almost more than 70% of the time, which lead to the LSTM model to fail by converging to one parameter value no matter the input.

Failure in detecting complex sequences of steps: The models failed in detecting complex sequences of steps and just was able to generate elementary shapes like cubes and cylinders. This was caused by the inability of the model to predict accurately coordinates of key points in the CAD steps like edges of lines or points of origin which will lead to complete failure of the CAD generation.

5.4. Future Directions

To address the limitations and further improve the Sketch-to-CAD models, several future directions can be explored:

Larger and more diverse dataset: Collecting a larger and more diverse dataset of isometric sketches and corresponding CAD steps would enhance the models' ability to generalize to a wider range of design scenarios and improve their performance.

Fine-grained CAD representations: Investigating alternative representations for CAD steps, such as continuous or vector-based representations, could capture finer details and provide more flexibility in generating accurate CAD outputs.

Deep Reinforcement Learning: It would be interesting to train a Deep RL model capable of based on a current state: CAD steps is able to take an action: CAD step and get a reward based on the new state. Some potential reward functions could be one that penalizes failing CAD sequences and one that encourages CAD sequences that generate sketches similar to the input image. In fact, this approach makes more sense in this context, given that there exist infinitely many representations that work for one sketch and thus, having a labeled CAD sequence for an image limits the capacity of the model to generalizes.

6. Conclusion

In conclusion, our study introduces Sketch-to-CAD, a generative AI model designed to expedite the process of CAD development by transforming 2D sketches into 3D CAD workflows with customizable timelines. To our knowledge, this is the very first paper that presents a method to predict an editable timeline of CAD steps based on sketches. We explored various deep learning architectures, including CNNs, Transformers, and LSTMs, to encode the isometric sketches and decode them into CAD steps.

The performance of the different algorithms varied, with some architectures demonstrating higher accuracy and more realistic CAD sequences than others. The CNN Encoder - Transformer Decoder architecture showed promising results in capturing relevant features from the input sketches, while generating realistic basic CAD parts. The other models weren't able to achieve good results since they converged to an optimal optima where the output is the same no matter the input which is due to data imbalance in the parameters distributions.

As next steps, we plan on incorporating techniques to address the data imbalances, such as oversampling or class weighting, which could enhance the performance and robustness of the models. Additionally, exploring ensemble methods, which combine the predictions of multiple models, might lead to improved accuracy and generalization capacity.

Furthermore, expanding the dataset by including a greater variety of CAD models and steps would provide a more comprehensive training set and enable the models to learn a wider range of design features and workflows. This could involve incorporating more complex CAD commands.

References

[1] Changjian Li, Hao Pan, Adrien Bousseau, and Niloy J Mitra. Sketch2cad: Sequential cad modeling by sketching in context. *ACM Transactions on Graphics (TOG)*, 39(6):1–14, 2020. 1

[2] Jiantao Pu, Kuiyang Lou, and Karthik Ramani. A 2d sketch-based user interface for 3d cad model retrieval. *Computer-aided design and applications*, 2(6):717–725, 2005. 1

[3] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion, 2022. 2

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

[5] 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, 2021. 2, 4, 5

[6] Sebastian Koch, Albert Matveev, Zhongshi Jiang, Francis Williams, Alexey Artemov, Evgeny Burnaev, Marc Alexa, Denis Zorin, and Daniele Panozzo. Abc: A big cad model dataset for geometric deep learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 2

[7] Autodesk. Fusion 360. CAD Software, 2014. 2

[8] https://wikidocs.net/165427. 4

7. Contributions

We met every week to work together on the project. We explored and collected the data together, discussed what experiments we were planning on doing, analyzed the outputs together and set our future directions. We contributed equally to each part of the project.