UniSketch: A Unified Framework for Parametric Sketch Generation and Constraint Prediction

Jing Lin[1], **Fazhi He**[2,*], **Rubin Fan**[3]
[1]Key Laboratory of Aerospace Information Security and Trusted Computing, Ministry of Education, School of Cyber Science and Engineering, Wuhan University, China
[2]School of Artificial Intelligence, Wuhan University, China
[3]School of Computer Science, Wuhan University, China
fzhe@whu.edu.cn

Abstract

In modern Computer-Aided Design (CAD), parametric sketches play a crucial role by capturing both the geometric structure and design intent through constraints. However, existing deep learning–based sketch methods remain restricted to simple geometric primitives and limited constraint types, hindering their application to complex real-world engineering tasks. To address this gap, we introduce the UniSketch dataset, comprising 3,836,290 sketches. It offers a comprehensive and diverse collection of 7 types of geometric primitives and 23 types of 2D constraints, all represented as unified vector sequences suitable for deep learning applications. Leveraging the UniSketch dataset, we propose a unified multi-task Transformer framework as a true foundation model for parametric sketch modeling, supporting diverse core tasks like image-to-sketch generation, constraint prediction, and unconditional sketch synthesis. Furthermore, the generated sketches can be efficiently converted to CAD-compatible formats, enabling seamless integration with industrial CAD system for re-editing and reusing. The experimental results show that UniSketch outperforms existing methods in multiple tasks, demonstrating its versatility and practical value in industrial CAD applications.

Code — https://github.com/fazhihe/UniSketch
Datasets — https://huggingface.co/unisketch

Introduction

Parametric CAD tools are essential in industries like mechanical engineering, aerospace, and architecture (Otey et al. 2018; Zhang and Luo 2009). Parametric sketches serve as the foundation of 3D modeling, where geometric primitives and constraints define the initial shape and structure of a part (Shah 1998; Camba, Company, and Naya 2022; Company et al. 2020).

Despite their importance, automated sketch modeling faces challenges due to limited datasets and methods. Most existing work handles only simple primitives and constraints, while real-world industrial designs involve complex primitives such as ellipses and conic curves, and higher-order constraints like symmetry and offset.

To address these issues, we introduce the UniSketch dataset, a large-scale, high-quality dataset with diverse primitives and a full range of 2D constraints, represented as unified vector sequences compatible with deep learning.

However, high-quality data alone is insufficient. Existing methods are typically single-task, limiting their adaptability and performance. Moreover, most constraint prediction methods rely solely on primitive parameters without leveraging sketch images.

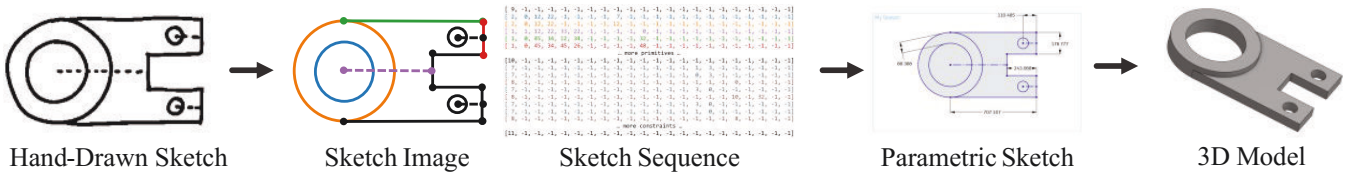
To this end, we propose UniSketch, a unified multi-task Transformer framework as a general-purpose foundation model for CAD sketches. By handling both primitives and constraints with a unified vector format, it simplifies the model architecture. UniSketch consists of a Vision Transformer (ViT) encoder (Dosovitskiy et al. 2020), a gated Transformer decoder (Vaswani et al. 2017), and dual output heads. This simple yet effective architecture supports several tasks, including image-to-sketch generation, constraint prediction, unconditional sketch synthesis, and others. The gated mechanism enables the decoder to support both image-driven and purely sequence-based modeling. For example, image features can be used to improve constraint prediction from primitive parameters.

The output of UniSketch is a structured parametric sketch sequence that can be converted to CAD-compatible formats for real-time editing (see Figure 1).

We evaluate UniSketch on four tasks: image-to-sketch generation, constraint prediction, unconditional generation, and sketch completion. Results demonstrate it outperforms existing methods, highlighting its versatility and practical applicability.

The main contributions of this paper are as follows:

- We introduce the UniSketch dataset, a large-scale parametric sketch dataset featuring diverse geometric primitives and 2D constraints, tailored for deep learning applications.
- We propose UniSketch, a foundational unified Transformer framework that supports multimodal inputs and enables flexible multi-task modeling.
- Extensive experiments demonstrate superior performance and practical applicability, with generated sketches convertible to CAD-compatible formats for further editing.



(a) CAD Workflow Integration

[9, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1]	**Control flag:**	**<SOSK>**
[2, 0, 12, 22, -1, -1, -1, -1, 7, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1]	**Primitive 0:**	a circle
[2, 0, 12, 22, -1, -1, -1, -1, 12, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1]	**Primitive 1:**	a circle
[1, 1, 12, 22, 33, 22, -1, -1, -1, -1, 0, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1]	**Primitive 2:**	a line
[1, 0, 45, 34, 12, 34, -1, -1, -1, -1, -1, 32, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1]	**Primitive 3:**	a line
[1, 0, 45, 34, 45, 26, -1, -1, -1, -1, -1, 48, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1]	**Primitive 4:**	a line
… more primitives …		
[10, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1]	**Control flag :**	**<EOP>**
[7, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 3, -1, -1, -1, -1, -1, -1]	**Select:**	the center of primitive 1
[7, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 0, 3, -1, -1, -1, -1, -1, -1]	**Select:**	the center of primitive 0
[8, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 0, -1, -1, -1, -1, -1]	**Constraint:**	coincident
[7, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 3, 0, -1, -1, -1, -1, -1, -1]	**Select:**	the whole primitive 3
[8, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 10, -1, 32, -1, -1, -1]	**Constraint:**	length
[7, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 3, 0, -1, -1, -1, -1, -1, -1]	**Select:**	the whole primitive 3
[7, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 1, 0, -1, -1, -1, -1, -1, -1]	**Select:**	the whole primitive 1
[8, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 8, -1, -1, -1, -1, -1]	**Constraint:**	tangent
… more constraints …		
[11, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1]	**Control flag:**	**<EOSK>**

(b) Sketch Sequence Example

Figure 1: (a) CAD workflow integration. UniSketch integrates into the CAD workflow by predicting a constraint-based sketch sequence from an image. The sequence can be adapted for reusing in CAD systems to support subsequent 3D modeling. (b) Sketch sequence example. The left side shows the vector sequence of a sketch, while the right side illustrates the meaning of each vector. The colors of the primitive vectors correspond to the visual appearance of the primitives in the sketch image.

Related Work

CAD Datasets

Parametric 3D Dataset Large-scale 3D datasets typically represent geometry as meshes, point clouds, or B-Reps (Chang et al. 2015; Wu et al. 2015; Sun et al. 2018; Koch et al. 2019), but often lack records of human-driven modeling operations. DeepCAD (Wu, Xiao, and Zheng 2021) and Fusion 360 Gallery (Willis et al. 2021b) include modeling semantics and operation histories, yet sketches are treated as intermediate steps, lacking constraint annotations and diverse primitive types, which limits their use in sketch-specific modeling research.

Parametric Sketch Dataset Dedicated sketch datasets have gained attention as sketches play a critical role in CAD modeling. CADL (Ganin et al. 2021) uses Protocol Buffers, which is not directly suitable for deep learning. SketchGraphs (Seff et al. 2020) collects a large number of real-world sketches from Onshape and converts them into geometric constraint graphs. While the converted graphs have limited primitive and constraint types, the raw JSON data contains the full information and supports flexible preprocessing and extension. Vitruvion (Seff et al. 2021) and SketchGen (Para et al. 2021) introduce sequential representations based on SketchGraphs but do not expand primitive coverage or support numerical and higher-order constraints. HPSketch (Fan et al. 2025) provides a more diverse set of primitives and constraints; however, its synthetic origin limits its reflection of real user intent, and its loop-based representation enforces sketch closure, reducing flexibility for incomplete or open designs.

Sketch Inference with Constraints

Some sketch generation methods (e.g., CurveGen and TurtleGen (Willis et al. 2021a)) rely on automatic constraint solvers, which establish geometric relationships based on physical tolerances and predefined priorities, but lack flexibility to adapt to user intent. Recent learning-based frameworks infer both primitives and their associated constraints.

SketchGraphs (Seff et al. 2020) represents sketches as graphs with primitives and constraints as nodes and edges, predicting constraints from a given primitive set using a GNN (Scarselli et al. 2008), but cannot infer geometric parameters, limiting performance when initial layouts are imprecise. SketchGen (Para et al. 2021), CADL (Ganin et al. 2021), and Vitruvion (Seff et al. 2021) model sketches as token sequences using Transformer-based architectures with pointer networks (Vinyals, Fortunato, and Jaitly 2015). While generating structured sketches with explicit parameters and constraint references, they support only a limited range of primitives and constraints, and constraint references rely on indices, reducing readability.

Image-Based Sketch Inference

Generative networks such as DeepSVG (Carlier et al. 2020), Im2Vec (Reddy et al. 2021), and other related approaches (Egiazarian et al. 2020; Ellis et al. 2018) have advanced vector graphic synthesis but often rely on domain-specific architectures, limiting their applicability to CAD sketches.

Recent CAD-focused works include Vitruvion (Seff et al. 2021), which separates modeling into two tasks: predicting primitives from raster images and inferring constraints; PICASSO (Karadeniz et al. 2025), which predicts sequences of parameterized primitives from sketches without constraints; and DAVINCI (Karadeniz et al. 2024), which jointly predicts primitives and constraints from images but does not generalize to tasks like predicting constraints from a given primitive sequence. Overall, these methods are largely single-task and lack a unified multi-modal framework.

UniSketch Dataset

Data Source and Construction

The UniSketch dataset builds upon SketchGraphs (Seff et al. 2020), which extracts a large collection of parametric CAD sketches created by users from the public Onshape platform.

As noted by (Ganin et al. 2021), sketch construction is similar to natural language modeling, where selecting the next primitive or constraint is akin to generating the next word in a sentence. Furthermore, (Seff et al. 2021) demonstrated that shuffling the order of primitives significantly degrades network performance, indicating that the true sequence order contains meaningful structural information. Therefore, we reprocessed the original `JSON` files and transformed them into a sequence format more suitable for Transformer-based networks, with richer primitive and constraint types. This makes the UniSketch dataset the most comprehensive parametric sketch dataset.

The dataset construction process includes the following steps:

Filtering and Reconstruction Unsupported primitives (splines, text, and images) and constraints involving external references (e.g., projected, pierce) are both removed. To control sketch complexity, only samples with 4 to 16 primitives and total sequence lengths not exceeding 200 are retained. Each sketch is converted into a structured vector sequence. We define the minimal parameter set required for each operation. Redundant parameters are removed by setting default values. The sequence of operations for each sketch is then represented as fixed-length vectors and stored in `HDF5` format (Folk et al. 2011) (with a `.h5` file extension) for efficient loading and training.

Normalization and Quantization To ensure geometric consistency and numerical stability, sketches are globally translated and scaled so that their coordinates and numerical parameters fall within a fixed range. These values are then quantized into 64 equally spaced integer bins, mapped to the $[0, 64)$ range. Additional details are provided in the supplementary material.

Deduplication and Visualization Hashing is applied to the quantized sequences to remove duplicates, ensuring the uniqueness of each sketch. The primitive parts of the sketches are rendered, with construction primitives represented as dashed lines and regular primitives as solid lines. The sketches are saved as 128x128 pixel images.

The final processed UniSketch dataset contains 3,836,290 sketches. The distributions of primitive and constraint types and other information can be found in the supplementary material. If the filtering thresholds are relaxed, a larger dataset can be obtained, offering even more complex sketch data.

Sequence Representation

In our dataset, each sketch is represented as a unified vector sequence, integrating both geometric primitives and constraint information. The sequence contains three types of vectors: primitive vectors (representing parameterized geometric primitives), constraint vectors (defining the selection and application of geometric/numerical constraints), and control flags (indicating structural boundaries such as sketch start, primitive end, and sketch end). As shown in Table 1, the vector consists of 22 fields, with the first bit indicating the vector type, and the remaining bits used to record various parameters.

Primitive Encoding The dataset supports seven types of primitives: point, line, circle, arc, ellipse, elliptical arc, and conic curve. All primitives are encoded in a unified format, with different types occupying predefined parameter bits. For example, the vector representing an `ellipse` includes the coordinates of its center, the lengths of the major and minor axes, and the rotation angle, which are placed in the 3rd, 4th, 9th, 10th, and 11th positions, respectively. To maintain structural consistency, unused fields are filled with -1. Detailed field definitions for each primitive type are provided in the supplementary material.

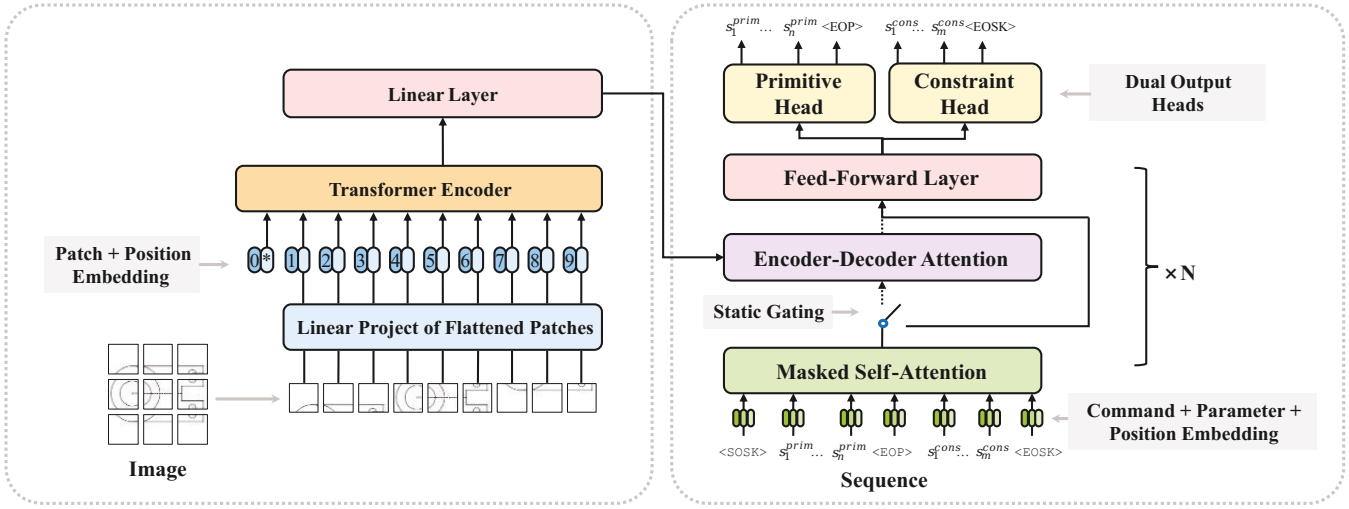
Constraint Encoding The dataset supports many 2D constraint types, totaling 23 types, including coincident, parallel, tangent, equal, angle, radius, rho, and others. The constraint vector contains selection operations (using multi-level indexing to locate reference) and constraint operations (specifying the type and required parameters). Specifically, in the selection operation, the 15th bit is the primitive index, the 16th bit indicates the subcomponent type (whole, start, end, center and internal). When the subcomponent type is internal, the 17th bit indicates the subcomponent index such as the control points of a conic curve. If no third-level index is needed, the field is set to -1. In the constraint operation, the 18th bit specifies the constraint type, and bits 19 to 22 might store values of angle, length, direction, or rho, depending on the constraint type. Full definitions can be found in the supplementary material.

Control Flag Encoding Control flag vectors mark structural boundaries within the sequence to ensure correct parsing of the sketch hierarchy. There are three types: the start of sketch (`<SOSK>`), end of primitive (`<EOP>`), and end of sketch (`<EOSK>`). The first bit indicates the control type and the remaining bits are set to -1.

An example is shown in Figure 1, where part (a) displays the rendered sketch image and part (b) shows the corresponding vector sequence. This representation offers several advantages. It adopts a unified format, where all operations are encoded as fixed-length vectors with parameter padding

Operation	t_{op}	c	x_1	y_1	x_2	y_2	x_3	y_3	r_1	r_2	α	p_{start}	p_{end}	ρ	s_1	s_2	s_3	t_c	v_c	l_c	θ_c	ρ_c
Point	t_{op}	c	x_1	y_1																		
Line	t_{op}	c	x_1	y_1	x_2	y_2					α											
Circle	t_{op}	c	x_1	y_1					r_1													
Arc	t_{op}	c	x_1	y_1					r_1			p_{start}	p_{end}									
Ellipse	t_{op}	c	x_1	y_1					r_1	r_2	α											
Elliptical arc	t_{op}	c	x_1	y_1					r_1	r_2	α	p_{start}	p_{end}									
Conic	t_{op}	c	x_1	y_1	x_2	y_2	x_3	y_3				p_{start}	p_{end}	ρ								
Selection	t_{op}														s_1	s_2	s_3					
Constraint	t_{op}																	t_c	v_c	l_c	θ_c	ρ_c

Table 1: Vector representation fields for sketch operations.



(a) Vision Transformer Encoder (b) Transformer Decoder

Figure 2: Overall architecture of UniSketch. The network consists of a ViT encoder and a gated Transformer decoder. The encoder extracts visual features from the input sketch image, while the decoder generates the primitive and constraint tokens in sequence. The cross-attention modules in the decoder are conditionally activated depending on the presence of the image input, enabling both image-sequence and sequence-only input.

handled via a masking mechanism. This ensures structural consistency and facilitates network training. It is also extensible: new primitive or constraint types can be supported by reusing or extending existing fields without modifying the overall structure. In addition, the representation is human-readable and interpretable, which supports manual construction. Finally, it can be converted back to `JSON` format and integrated into CAD platforms such as Onshape via `API`, enabling downstream design and editing workflows (see Figure 1).

UniSketch Framework

We present UniSketch, a foundational unified Transformer framework for parametric CAD sketch modeling. UniSketch integrates a ViT encoder, a Transformer decoder with dual output heads, and a static gating mechanism. This design supports several core tasks, including generating sketch sequences from images, predicting constraints from primitive sequences with or without image guidance, unconditional sketch synthesis and sketh completion. The architecture overcomes the task-specific limitations of conventional CAD networks and significantly improves generality.

Network Architecture

As illustrated in Figure 2, UniSketch consists of four main components: a ViT encoder, a Transformer decoder, a static gating controller, and dual output heads.

ViT Encoder The encoder follows the standard ViT (Dosovitskiy et al. 2020) design. The input is a grayscale sketch image $I \in \mathbb{R}^{H\times W\times 1}$, which is divided into non-overlapping $P \times P$ patches. Each patch is linearly projected into a d-dimensional embedding, and position embeddings are added before feeding the tokens into multiple Transformer self-attention layers. The resulting visual token sequence is denoted as $Z = \text{ViT}(I) \in \mathbb{R}^{L\times d}$, where L is the number of patches. To match the decoder input dimension, a lightweight linear bridge layer is applied:

$$Z' = \text{Linear}(Z), \tag{1}$$

Image input	Sequence input	Task
✓	$S_{\text{in}} = [\texttt{<SOSK>}]$	Image-to-sketch generation
✓	$S_{\text{in}} = [\texttt{<SOSK>}, s_1^{\text{prim}}, \ldots, s_n^{\text{prim}}, \texttt{<EOP>}]$	Image-guided constraint prediction
×	$S_{\text{in}} = [\texttt{<SOSK>}]$	Unconditional sketch synthesis
×	$S_{\text{in}} = [\texttt{<SOSK>}, s_1^{\text{prim}}, \ldots, s_n^{\text{prim}}, \texttt{<EOP>}]$	Constraint prediction from primitives
×	$S_{\text{in}} = [\texttt{<SOSK>}, s_1^{\text{prim}}, \ldots, s_k^{\text{prim}}], k < n$	Sketch completion

Table 2: Overview of supported tasks and input configurations

which serves as the cross-attention context for the decoder.

Transformer Decoder The decoder follows a standard Transformer (Vaswani et al. 2017) architecture and autoregressively generates the target sequence $S = [s_1, s_2, \ldots, s_T]$. Each token s_t is embedded as: $e(s_t) = e_{\text{cmd}}(s_t) + e_{\text{param}}(s_t) + e_{\text{pos}}(s_t)$, where the embeddings capture command type, parameter values, and positional information, respectively. The decoder stacks multiple layers of self-attention and feed-forward networks. Depending on the gating state, it optionally incorporates visual context from the encoder through cross-attention layers. The conditional generation probability is:

$$P(S \mid x) = \prod_{t=1}^{T} P(s_t \mid s_{<t}, x), \tag{2}$$

where $x = Z'$ (if the encoder is active) or $x = \emptyset$ (if disabled).

Static Gating Mechanism A static gate controls whether the decoder attends to the visual context. When the gate is enabled, the decoder uses cross-attention to incorporate image features. When disabled, it relies solely on the input sequence and self-attention, functioning as a decoder-only architecture (Radford et al. 2018). This mechanism requires no additional parameters but provides flexible support for both image-guided and sequence-only tasks, improving network generalization and adaptability.

Dual Output Heads Sketch sequences include both of geometric primitives and constraints, which exhibit different structures and parameter distributions. To address this, we introduce two separate linear output heads: one for primitive tokens and one for constraint tokens. During generation, the decoder dynamically selects the appropriate head based on the current position t and the index of the primitive end flag $t_{\texttt{<EOP>}}$. Let h_t be the hidden state at step t:

$$s_t = \begin{cases} \text{PrimitiveHead}(h_t), & t \le t_{\texttt{<EOP>}} \\ \text{ConstraintHead}(h_t), & t > t_{\texttt{<EOP>}} \end{cases} \tag{3}$$

This task-aware output separation allows the decoder to handle the semantic and statistical differences between primitives and constraints, while maintaining a shared latent space. This design enhances both network compactness and overall performance.

Task Formulation

Network Inputs The network accepts two types of inputs: (1) Image input I: A grayscale sketch image. When provided, the encoder is enabled and its output serves as visual context. (2) Sequence input S_{in}: A partial or complete primitive sequence.

Network Output A complete sketch sequence: $S_{\text{out}} = [\texttt{<SOSK>}, s_1^{\text{prim}}, \ldots, s_n^{\text{prim}}, \texttt{<EOP>}, s_1^{\text{cons}}, \ldots, s_m^{\text{cons}}, \texttt{<EOSK>}]$, where s_i^{prim} are primitive tokens, s_i^{cons} are constraint tokens, and <SOSK>, <EOP>, <EOSK> are control tokens indicating the start of the sketch, end of primitives, and end of the sketch, respectively.

UniSketch reformulates sketch modeling as a sequence generation task. This unified formulation allows UniSketch to flexibly handle diverse modalities and tasks. Different input configurations correspond to different tasks, as summarized in Table 2.

Training Strategy

During training, it is crucial to accommodate both image-sequence and sequence-only input scenarios. However, this multi-modal design introduces challenges in training stability. Specifically, strategies such as staged or periodically alternating training often lead to catastrophic forgetting, where the model overfits to the currently dominant task and degrades in performance on the other. This causes oscillations and hinders convergence. To address this issue, we adopt a mixed-batch training strategy: for each training batch, we randomly include the image input with a probability of $p = 0.5$. This encourages the model to adapt to both input modes within a unified parameter space, improving both stability and generalization.

We adopt a teacher forcing strategy during training. At each decoding step t, the network is fed the ground-truth token $s_{<t}$ alonog with the visual context x (if applicable), and predicts the next token s_t. The objective is to maximize the conditional generation probability $P(S \mid x)$, as defined in Section .

Since constraint tokens typically dominate the sequence, the network may bias toward constraint prediction, impairing primitive modeling performance. To address this, we employ a weighted loss strategy to balance primitive and constraint learning. Consequently, we not only assign different weights to the cross-entropy losses of command types and parameter values for each output vector but also apply distinct weights to primitive and constraint vectors.

Experiments

Experimental Setup

Dataset For comparative experiments, we train and evaluate the UniSketch framework on the Vitruvion dataset (Seff

Method	Teacher forcing			Autoregressive		
	$\text{Acc.}_{\text{prim}}$ ↑	Precision ↑	Recall ↑	F1 score ↑	$\text{Acc.}_{\text{prim}}$ ↑	$\text{Acc.}_{\text{param}}$ ↑
$\text{Vitruvion}_{\text{prim}}$	43.97/41.73	24.52/17.27	25.44/17.99	24.77/17.46	26.18/18.53	62.92/56.51
UniSketch	**51.95/52.54**	**26.74/32.39**	**28.89/33.25**	**27.50/32.57**	**31.09/35.71**	**73.81/74.98**

Table 3: Performance comparison on image-to-sequence generation

Method	Teacher forcing		Autoregressive		
	$\text{Acc.}_{\text{cons}}$ ↑	$\text{Acc.}_{\text{param}}$ ↑	Precision ↑	Recall ↑	F1 score ↑
$\text{Vitruvion}_{\text{cons}}$	74.69	88.94	-	-	-
SketchGraphs	-	-	52.34	51.99	52.16
UniSketch	**79.02**	**91.42**	**78.34**	**62.03**	**66.00**

Table 4: Performance comparison on constraint prediction

et al. 2021), derived from Sketchgraphs, as baseline networks do not support complex primitives and constraints. The Vitruvion dataset, providing both precise images and noisy hand-drawn simulated images for sketches, is converted to our format with a 64-bin quantization strategy. For ablation studies, we use the UniSketch dataset, which supports diverse primitives and constraints. To reduce training costs, we randomly sample 1,000,000 sketches from the UniSketch dataset, creating a benchmark set split into training (900k), validation (50k), and test (50k) sets.

Baselines We conduct comparative studies against the Vitruvion (Seff et al. 2021) and SketchGraphs (Seff et al. 2020) networks to validate the effectiveness of our approach.

Vitruvion employs two separate networks corresponding to our tasks: one for primitive generation from images and another for constraint prediction from primitive sequences. We evaluate the primitive generation network under both teacher forcing (TF) and autoregressive (AR) settings, while the constraint prediction network is evaluated only under TF. In TF, the model receives ground-truth tokens at each decoding step, whereas in AR, the model generates the full sequence based solely on its previous predictions.

The SketchGraphs baseline is built on a graph neural network and inherently operates autoregressively, as it does not have access to ground-truth outputs during inference. Its predictions at each step depend solely on previously predicted elements. For consistency, we therefore evaluate UniSketch in the AR setting on the constraint prediction task when comparing to SketchGraphs.

Evaluation Metrics For primitive modeling, we report primitive-level accuracy under the TF setting, which requires an exact match of the entire primitive, including both its type and parameters. Under TF, the model can iteratively refine each prediction by using ground-truth inputs at subsequent steps. Moreover, the decoding granularities of the compared methods differ fundamentally, making parameter-level metrics inherently biased toward models that decode at a finer resolution. For this reason, parameter-level metrics are not directly compared in the TF setting. In the AR setting, we first perform Hungarian matching between predicted and ground-truth primitives to align corresponding elements. Based on the matched pairs, we then compute parameter-level accuracy together with primitive-level accuracy, precision, recall, and F1 score.

For constraint prediction, we use constraint-level accuracy and parameter-level accuracy under the TF strategy since they share the same granularity; under the AR strategy, we report precision, recall, and F1 score. All numerical results in tables are reported as percentages (%) unless otherwise specified.

Image-to-Sequence Generation

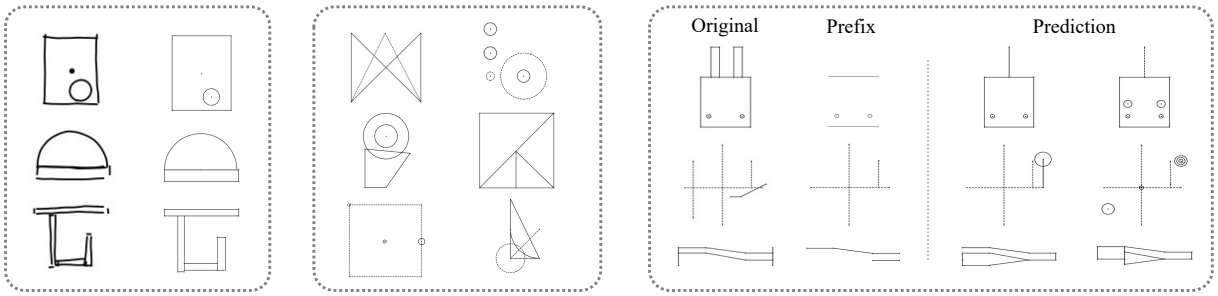
We conducted a comparison of UniSketch and the primitive network of Vitruvion (Seff et al. 2021) on both original and noisy sketch images, where results for original/noisy inputs are reported in the x/y format. As shown in Table 3, UniSketch outperforms Vitruvion across all metrics, reflecting its stronger ability to generate complete and coherent primitives. The slightly higher performance of UniSketch on noisy sketches might result from mild noise acting as a regularizer, encouraging the model to rely more on stable geometric structures rather than low-level appearance features. The reported results are based on strict exact matching of primitive parameters. In practice, small deviations within a reasonable threshold would still yield correct or acceptable primitives, meaning the effective performance is higher than these strict scores suggest. Several examples of generated sketches are shown in Figure 3.

Constraint Prediction

We also compared UniSketch with the constraint networks of Vitruvion (Seff et al. 2021) and SketchGraphs (Seff et al. 2020). As shown in Table 4, UniSketch consistently outperforms both baselines across all metrics, demonstrating its strength in unified geometric and parametric reasoning. As expected, autoregressive decoding performs worse than teacher forcing because of error accumulation.

Method	Image-to-sequence generation		Constaint prediction	
	$\text{Acc.}_{\text{prim}}\uparrow$	$\text{Acc.}_{\text{param}}\uparrow$	$\text{Acc.}_{\text{cons}}\uparrow$	$\text{Acc.}_{\text{param}}\uparrow$
$\text{UniSketch}_{\text{mixed-batch}}$	**36.61**	**71.37**	64.84	82.68
$\text{UniSketch}_{\text{img-seq}}$	29.71	64.63	**64.94**	**82.73**
$\text{UniSketch}_{\text{seq-only}}$	29.97	64.57	64.80	82.62
$\text{UniSketch}_{\text{seq-only}} - D$	29.58	64.37	64.72	82.60

Table 5: Ablation study of UniSketch components



(a) Image to Sequence (b) Unconditional Sketch Synthesis (c) Sketch Completion

Figure 3: Several examples of the three tasks: (a) image-to-sketch generation, (b) unconditional sketch synthesis, and (c) sketch completion.

Unconditional Sketch Synthesis and Sketch Completion

We further evaluate our model on two generative tasks: unconditional sketch synthesis and sketch completion. For both tasks, we adopt top-k sampling strategies to generate sketch sequences, either from the <SOSK> token or from partially completed sketches. Figure 3 shows representative examples of generated sketches. Detailed results and additional examples are provided in the supplementary material.

Ablation Studies

We performed ablation experiments on the UniSketch dataset to assess the importance of various design components by progressively removing key architectural elements as shown in Table 5. Here, D denotes the dual-head architecture. The overall performance is lower than that in Table 3 and Table 4, due to the greater diversity and complexity of the UniSketch dataset, which makes the prediction task more difficult.

Input Modality Gating Mixed-batch training improves the adaptability of UniSketch to different input modalities and facilitates parameter sharing across tasks. Interestingly, training primitives using only the image-sequence setting performs worse than alternating training, and even slightly worse than sequence-only training in primitive modeling. This may seem counterintuitive, since images theoretically offer richer information to support generation. This setting achieves the best performance in constraint prediction, showing that visual cues help capture geometric relationships. One possible explanation is that image features are relatively complex. The current network capacity may be insufficient to fully capture these features, resulting in suboptimal convergence. In contrast, mixed-batch training enables the model to first learn a reasonably effective solution from the simpler sequence-only supervision, which is then further refined through image-based guidance. This also underscores the potential advantage of multi-task learning.

Dual Output Heads Unifying primitive and constraint predictions into a single decoding head resulted in a decline in prediction performance for both primitives and constraints, indicating that task separation is beneficial for modeling their distinct features.

Conclusion and Future Work

In this paper, we present UniSketch, a foundation unified framework for parametric sketch generation and constraint prediction. To support a wide range of sketch tasks, we construct the UniSketch dataset, which covers common constraint types in mainstream CAD system. UniSketch flexibly handles different input modalities, including images and primitive sequences, and produces complete structured sketches. Extensive experiments demonstrate its effectiveness and generality across various tasks.

Limitations and Future Work A current limitation of our work is the lack of support for spline-based primitives, which are typically defined by multiple control points. As future work, we plan to extend the primitive vocabulary by modeling splines and their control points as separate primitive types. In addition, we aim to expand our dataset by incorporating additional sources, including real-world industrial part sketches, to further enhance the diversity and representativeness of the data. Finally, we plan to refine constraint evaluation by incorporating solvers to assess global satisfaction states.

Acknowledgements

This work is supported by the National Natural Science Foundation of China under Grant No 62072348. The numerical calculations in this paper have been done on the supercomputing system in the Supercomputing Center of Wuhan University.

References

Camba, J. D.; Company, P.; and Naya, F. 2022. Sketch-based modeling in mechanical engineering design: Current status and opportunities. *Computer-Aided Design*, 150: 103283.

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: 16351–16361.

Chang, A. X.; Funkhouser, T.; Guibas, L.; Hanrahan, P.; Huang, Q.; Li, Z.; Savarese, S.; Savva, M.; Song, S.; Su, H.; et al. 2015. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*.

Company, P.; Naya, F.; Contero, M.; and Camba, J. D. 2020. On the role of geometric constraints to support design intent communication and model reusability. *Comput-Aided Des Appl*, 17(1): 61–76.

Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Egiazarian, V.; Voynov, O.; Artemov, A.; Volkhonskiy, D.; Safin, A.; Taktasheva, M.; Zorin, D.; and Burnaev, E. 2020. Deep vectorization of technical drawings. In *European conference on computer vision*, 582–598. Springer.

Ellis, K.; Ritchie, D.; Solar-Lezama, A.; and Tenenbaum, J. 2018. Learning to infer graphics programs from hand-drawn images. *Advances in neural information processing systems*, 31.

Fan, R.; He, F.; Liu, Y.; and Lin, J. 2025. A history-based parametric CAD sketch dataset with advanced engineering commands. *Computer-Aided Design*, 103848.

Folk, M.; Heber, G.; Koziol, Q.; Pourmal, E.; and Robinson, D. 2011. An overview of the HDF5 technology suite and its applications. In *Proceedings of the EDBT/ICDT 2011 workshop on array databases*, 36–47.

Ganin, Y.; Bartunov, S.; Li, Y.; Keller, E.; and Saliceti, S. 2021. Computer-aided design as language. *Advances in Neural Information Processing Systems*, 34: 5885–5897.

Karadeniz, A. S.; Mallis, D.; Mejri, N.; Cherenkova, K.; Kacem, A.; and Aouada, D. 2024. DAVINCI: A Single-Stage Architecture for Constrained CAD Sketch Inference. In *British Machine Vision Conference*.

Karadeniz, A. S.; Mallis, D.; Mejri, N.; Cherenkova, K.; Kacem, A.; and Aouada, D. 2025. Picasso: A feed-forward framework for parametric inference of cad sketches via rendering self-supervision. In *2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 6475–6484. IEEE.

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. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 9601–9611.

Otey, J.; Company, P.; Contero, M.; and Camba, J. D. 2018. Revisiting the design intent concept in the context of mechanical CAD education. *Computer-aided design and applications*, 15(1): 47–60.

Para, W.; Bhat, S.; Guerrero, P.; Kelly, T.; Mitra, N.; Guibas, L. J.; and Wonka, P. 2021. Sketchgen: Generating constrained cad sketches. *Advances in Neural Information Processing Systems*, 34: 5077–5088.

Radford, A.; Narasimhan, K.; Salimans, T.; Sutskever, I.; et al. 2018. Improving language understanding by generative pre-training.

Reddy, P.; Gharbi, M.; Lukac, M.; and Mitra, N. J. 2021. Im2vec: Synthesizing vector graphics without vector supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 7342–7351.

Scarselli, F.; Gori, M.; Tsoi, A. C.; Hagenbuchner, M.; and Monfardini, G. 2008. The graph neural network model. *IEEE transactions on neural networks*, 20(1): 61–80.

Seff, A.; Ovadia, Y.; Zhou, W.; and Adams, R. P. 2020. Sketchgraphs: A large-scale dataset for modeling relational geometry in computer-aided design. *arXiv preprint arXiv:2007.08506*.

Seff, A.; Zhou, W.; Richardson, N.; and Adams, R. P. 2021. Vitruvion: A generative model of parametric cad sketches. *arXiv preprint arXiv:2109.14124*.

Shah, J. J. 1998. Designing with parametric cad: Classification and comparison of construction techniques. In *International workshop on geometric modelling*, 53–68. Springer.

Sun, X.; Wu, J.; Zhang, X.; Zhang, Z.; Zhang, C.; Xue, T.; Tenenbaum, J. B.; and Freeman, W. T. 2018. Pix3d: Dataset and methods for single-image 3d shape modeling. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2974–2983.

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.

Vinyals, O.; Fortunato, M.; and Jaitly, N. 2015. Pointer networks. *Advances in neural information processing systems*, 28.

Willis, K. D.; Jayaraman, P. K.; Lambourne, J. G.; Chu, H.; and Pu, Y. 2021a. Engineering sketch generation for computer-aided design. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2105–2114.

Willis, K. D.; Pu, Y.; Luo, J.; Chu, H.; Du, T.; Lambourne, J. G.; Solar-Lezama, A.; and Matusik, W. 2021b. Fusion 360 gallery: A dataset and environment for programmatic cad construction from human design sequences. *ACM Transactions on Graphics (TOG)*, 40(4): 1–24.

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*, 6772–6782.

Wu, Z.; Song, S.; Khosla, A.; Yu, F.; Zhang, L.; Tang, X.; and Xiao, J. 2015. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1912–1920.

Zhang, Y.; and Luo, X. 2009. Design intent information exchange of feature-based CAD models. In *2009 WRI World Congress on Computer Science and Information Engineering*, volume 3, 11–15. IEEE.