A. Appendix

A.1. Background of VQ-VAE

We use the VQ-VAE to obtain discrete representations for faces and edges in the B-rep. Given a geometric input $\mathbf{x} \in \mathbb{R}^{H\times W\times C}$, the VQ-VAE learns to encode it using a set of discrete codebook vectors drawn from a learnable codebook. Specifically, the encoder E_ϕ maps $\mathbf{x}$ to a feature map $\mathbf{z}_e \in \mathbb{R}^{h\times w\times n_q}$, where h and w denote the spatial resolution of the downsampled feature map, and n_q is the dimensionality of each latent vector. Each element $\mathbf{z}_{e,ij}$ in this feature map represents the latent vector located at spatial position (i, j). To obtain a discrete representation, every latent vector $\mathbf{z}_{e,ij}$ is quantized to its nearest entry in a codebook $Z = \{\mathbf{e}_k\}_{k=1}^K$, where K is the size of the codebook and each codeword $\mathbf{e}_k \in \mathbb{R}^{n_q}$ is a learnable prototype vector in the latent space. The quantization function $q(\cdot)$ replaces each latent vector with its closest codeword according to the squared Euclidean distance:

$$\mathbf{z}_{q,ij} = q(\mathbf{z}_{e,ij}) = \mathbf{e}_k, \quad k = \arg\min_{\mathbf{e}_k\in Z} \|\mathbf{z}_{e,ij} - \mathbf{e}_k\|_2^2 \quad (7)$$

This process effectively embeds the encoder output feature tensor into a discrete index grid of size $h \times w$, where each grid element corresponds to one codebook entry. The resulting discrete latent grid is then used by the decoder for reconstruction. The overall forward pass of the VQ-VAE is given by:

$$\hat{\mathbf{x}} = D_\theta(\mathbf{z}_q) = D_\theta(q(\mathbf{z}_e)) = D_\theta(q(E_\phi(\mathbf{x}))) \quad (8)$$

where E_ϕ and D_θ denote the encoder and decoder, respectively. The encoder maps the input to a continuous latent feature map, which is then quantized to the nearest entries in a learnable codebook, yielding discrete latent codes. The decoder reconstructs the input from these quantized representations.

Training Objective. During training, the encoder E_ϕ, decoder D_θ, and codebook Z are jointly optimized by minimizing the following loss:

$$\mathcal{L} = \|\mathbf{x} - \hat{\mathbf{x}}\|_2^2 + \|\mathrm{sg}[E_\phi(\mathbf{x})] - \mathbf{z}_q\|_2^2 + \beta\,\|E_\phi(\mathbf{x}) - \mathrm{sg}[\mathbf{z}_q]\|_2^2 \quad (9)$$

where $\mathrm{sg}[\cdot]$ denotes the *stop-gradient* operator, which prevents gradients from backpropagating through its argument. The loss comprises three components:

- **Reconstruction loss** $\|\mathbf{x} - \hat{\mathbf{x}}\|_2^2$, ensuring that the reconstructed sample $\hat{\mathbf{x}}$ remains faithful to the input geometry $\mathbf{x}$;
- **Codebook loss** $\|\mathrm{sg}[E_\phi(\mathbf{x})] - \mathbf{z}_q\|_2^2$, which updates the codebook embeddings toward the encoder outputs;
- **Commitment loss** $\beta\|E_\phi(\mathbf{x}) - \mathrm{sg}[\mathbf{z}_q]\|_2^2$, which penalizes large deviations between the encoder outputs and the assigned codewords, encouraging the encoder to commit to discrete representations.

Here, β is a weighting hyperparameter, typically set to 0.25.

Optimization Strategy. In practice, the codebook can be updated either through the explicit codebook loss in Equation 9 or through an exponential moving average (EMA), which improves stability by smoothing embedding updates over training iterations. However, both approaches mainly optimize the codewords that are actively selected during quantization, while rarely used codewords receive little training unless reactivated.

In our training method, the codebook is updated not only through standard gradient based updates of active codewords, but also through a targeted dynamic reinitialization mechanism inspired by CVQ-VAE [47] that reactivates persistently unused entries. Throughout training, the system monitors the assignment statistics of all codewords and reinitializes those with consistently low utilization by drawing representative feature vectors from a historical feature buffer and directly replacing the corresponding inactive entries. In addition, we replace the conventional Euclidean distance nearest neighbor search in Equation 7 with cosine similarity based matching, which improves the generalization capability of the learned geometric representations.

Interpretation. The VQ-VAE effectively transforms high-dimensional continuous data into a discrete index representation, forming a finite vocabulary of latent tokens. This discrete latent space provides a symbolic abstraction of geometry, enabling autoregressive models, such as Transformer, to model geometric data analogously to natural language. Each latent index can thus be viewed as a *Geometry Token* capturing local shape features, serving as the fundamental unit for downstream sequence modeling.

A.2. Ablation Study of Face-Edge VQ-VAE

In this section, we conduct an ablation study to analyze how the size of the VQ-VAE codebook influences reconstruction performance, in terms of both geometric fidelity and B-rep validity. VQ-VAE discretizes continuous encoder features into a latent space composed of K learnable codebook vectors, and the choice of K determines the representational capacity of this space, thereby affecting reconstruction quality and training stability. To isolate the effect of codebook size, we experiment with four configurations, specifically $K = 512,\ 1,024,\ 2,048$, and $4,096$, while keeping all other hyperparameters fixed. Experiments are performed on the DeepCAD dataset. Model performance under each setting is assessed using two metrics:

- **Valid**: For each codebook size, we randomly sample 3,000 B-rep instances from the test set and use the same subset across all configurations. Given the predicted UV-sampled features together with the associated ground-truth geometric information (e.g., 3D positions), we apply the post-processing pipeline to reconstruct B-rep models and compute the proportion of reconstructions deemed

Table 6. Ablation study of codebook size in VQ-VAE.

Codebook Size	Valid ↑	Train Loss ↓	Val Loss ↓
512	69.8%	9.2×10^{-5}	2.0×10^{-5}
1024	73.2%	7.0×10^{-5}	1.0×10^{-5}
2048	76.7%	4.7×10^{-5}	9.0×10^{-6}
4096	87.5%	1.6×10^{-5}	3.0×10^{-6}

valid.

- **Reconstruction Loss**: The geometry reconstruction loss on the training and validation sets (i.e., Train Loss and Val Loss), which measures the fidelity of the reconstructed shapes with respect to the input.

As shown in Table 6, increasing the codebook size consistently improves reconstruction fidelity and reduces the B-rep invalid rate. When the codebook size reaches $K = 4,096$, the model obtains the best overall performance, exhibiting the highest valid rate and stable training dynamics without signs of overfitting. Although larger codebooks may theoretically provide higher representational capacity, they are prone to poor utilization and unstable quantization behavior during VQ-VAE training. Moreover, excessively large codebooks expand the discrete vocabulary used for downstream autoregressive modeling, which can hinder sample quality and introduce additional computational cost. Therefore, we adopt $K = 4,096$ for the DeepCAD dataset as a balanced configuration that achieves strong reconstruction accuracy, high B-rep validity, stable quantization behavior, and manageable complexity for generative modeling.

A.3. Ablation Design of 3D Position Tokenization

This section investigates several deep learning architectures for vector quantization of bounding boxes, where each bounding box is represented as a six-dimensional vector $\mathbf{b} = [x_{\min}, y_{\min}, z_{\min}, x_{\max}, y_{\max}, z_{\max}] \in [-1, 1]^6$. To evaluate and compare the accuracy of different quantization strategies for bounding boxes, we structure our ablation studies around two paradigms: *convolution-based spatial expansion* schemes and *Transformer-based quantization* schemes.

In the *convolution-based spatial expansion* paradigm, to leverage the powerful convolutional architecture of VQ-VAE and adapt its 2D convolutions, we repeat the bounding box vector across spatial grids of varying resolutions and evaluate model performance under two configurations:

- **Scheme 1** (32×32 **High-Res**): We reshape $\mathbf{b}$ into a $(2, 3)$ matrix, then repeat it to $(32, 3)$, and finally tile it into $\mathbf{b}' \in \mathbb{R}^{32\times32\times3}$. We then feed $\mathbf{b}'$ into a five-block U-Net encoder with channel widths $32, 64, 128, 256, 512$, where the first four blocks perform downsampling and the last block extracts features without further downsampling, yielding a latent feature map $\mathbf{z}_e \in \mathbb{R}^{2\times2\times128}$.
- **Scheme 2** (16×16 **Low-Res**): Similarly, we construct a lower-resolution input $\mathbf{b}'' \in \mathbb{R}^{16\times16\times3}$ and employ a lighter four-block U-Net backbone with channel widths $64, 128, 256, 512$, also yielding a latent feature map $\mathbf{z}_e \in \mathbb{R}^{2\times2\times128}$.

For both convolution-based schemes, we apply a 1×1 convolution to project $\mathbf{z}_e$ to 64 dimensions to match the codebook, perform nearest-neighbor lookup against a learnable codebook of size $4,096$ and dimensionality 64 to obtain the quantized representation $\mathbf{z}_q$, and reconstruct the input via a symmetric decoder.

In the *Transformer-based quantization* schemes, we directly perform feature extraction and vector quantization on the bounding box representation.

- **Scheme 3 (Single-code quantization).** The entire bounding box is treated as a single atomic unit. A multilayer perceptron (MLP) maps $\mathbf{b}$ to a 64-dimensional embedding vector, which is then processed by an 8-layer standard Transformer encoder. The resulting feature is quantized into a single discrete codeword selected from the codebook. The decoder consists of a symmetric stack of Transformer encoder blocks, followed by an MLP that reconstructs the bounding box.
- **Scheme 4 (Multi-code quantization).** To explicitly capture both the independence and the inter-axis correlations among the X, Y, and Z dimensions, we reshape $\mathbf{b}$ into a $(3, 2)$ matrix, corresponding to the minimum and maximum values along each axis. An MLP embeds this matrix into a tensor of shape $(3, 64)$, which is subsequently processed by the same 8-layer Transformer encoder architecture. Each of the three 64-dimensional features is independently quantized, producing three discrete codewords from the shared codebook. The decoder adopts a symmetric architecture mirroring that of the encoder, comprising a symmetric stack of Transformer encoder blocks followed by an MLP to reconstruct the original bounding box.

All Transformer-based schemes use an embedding dimension of 64, a feed-forward hidden dimension of 256, and a codebook of size 4,096 with 64-dimensional codewords.

We evaluate the performance of the four schemes under two metrics:

- **Acc**: Bounding box reconstruction accuracy, defined as the fraction of samples for which all six components of the reconstructed bounding box $\hat{\mathbf{b}}$ lie within an absolute tolerance ϵ of the ground-truth $\mathbf{b} \in [-1, 1]^6$. Formally,

$$\begin{aligned} \text{Acc} &= \frac{1}{|\mathcal{D}|} \sum_{\mathbf{b}\in\mathcal{D}} s(\mathbf{b}, \hat{\mathbf{b}}), \\ s(\mathbf{b}, \hat{\mathbf{b}}) &= \begin{cases} 1, & \text{if } |\hat{b}_i - b_i| \le \epsilon \quad \forall i \in [6], \\ 0, & \text{otherwise.} \end{cases} \end{aligned} \tag{10}$$

Table 7. Ablation study of 3D position tokenization schemes.

Method	Acc ↑	Train Loss ↓	Val Loss ↓
Scheme 1	98.4%	2.5×10^{-5}	1.5×10^{-5}
Scheme 2	99.4%	2.0×10^{-5}	8.0×10^{-6}
Scheme 3	96.4%	4.6×10^{-5}	3.3×10^{-5}
Scheme 4	97.6%	3.9×10^{-5}	2.5×10^{-5}

We set $\epsilon = 0.01$, with $|\mathcal{D}|$ denoting the size of the dataset.

- **Reconstruction Loss**: The geometry reconstruction loss on the training and validation sets (i.e., Train Loss and Val Loss), measured as the ℓ_2 distance between the input bounding box and its VQ-VAE reconstruction. This metric quantifies the overall fidelity of shape reconstruction.

After extensive hyperparameter tuning, we report the performance of the four position tokenization schemes on the two metrics described above. The results are presented in Table 7. For deep learning-based methods, the best reconstruction accuracy under $\epsilon = 0.01$ reaches 99.4%. In comparison, our proposed *uniform scalar quantization* partitions the interval $[-1, 1]$ into L discrete levels with a fixed step size of $2/(L-1)$. Accordingly, the maximum reconstruction error is theoretically bounded by half of this step, i.e., $1/(L-1)$. Therefore, setting $L = 101$ guarantees 100% reconstruction accuracy when $\epsilon = 0.01$. Moreover, as L increases, the *uniform scalar quantization* scheme can achieve reconstruction accuracy that surpasses deep learning-based approaches, highlighting the superior precision and reliability of our method. Balancing precision and vocabulary size, we choose $L = 2,048$ for our final setting.

A.4. Qualitative Analysis of Block Sequentialization

In our holistic token sequence representation, each geometric primitive (i.e., face or edge) is represented by a fixed-length token block. The full sequence is constructed by concatenating these blocks in a specific order. To better preserve causal ordering and maintain the inherent local geometric structure of the B-rep model during autoregressive generation, we explored different strategies for ordering these blocks. Specifically, our strategy involves two key objectives: 1) Face block ordering: Faces sharing an edge should be positioned close to each other in the sequence, and 2) Edge block ordering: Each edge should be positioned close to its associated face in the sequence. Thus, we aim to minimize the block-level distances between topologically related primitives.

We denote the sets of face blocks and edge blocks by $\mathcal{F} = \{f_1, \ldots, f_{N_f}\}$ and $\mathcal{E} = \{e_1, \ldots, e_{N_e}\}$, respectively. For each face block f and edge block e, let $p_F(f) \in \mathbb{N}$ and $p_E(e) \in \mathbb{N}$ represent the starting token positions of their corresponding blocks. The distance between any two blocks is defined as the absolute difference between their starting positions, representing their relative separation within the causal context window. Based on this notion of distance, we formulate the following two objectives:

- **Face block ordering loss**:

$$L_F = \sum_{(f_i, f_j) \in \mathcal{A}_F} \left| p_F(f_i) - p_F(f_j) \right| \tag{11}$$

where $\mathcal{A}_F = \{(f_i, f_j) \mid f_i \text{ and } f_j \text{ share an edge}\}$. This loss measures the total sequential distance between all pairs of adjacent faces.

- **Edge block ordering loss**:

$$L_E = \sum_{e \in \mathcal{E}} \sum_{f \in \mathcal{F}(e)} \left| p_E(e) - p_F(f) \right| \tag{12}$$

where $\mathcal{E}$ is the set of all edge blocks, and $\mathcal{F}(e)$ denotes the set of face blocks incident to edge block e. This loss measures the total sequential distance between each edge block and the blocks of all its incident faces.

Analysis of face block ordering loss. Minimizing the Face block ordering loss L_F (Equation 11) corresponds to the classical *Minimum Linear Arrangement* (MLA) problem on the face adjacency graph, which seeks a linear ordering of nodes that minimizes the total distance between adjacent pairs. However, standard MLA formulations assume symmetric context access and do not account for the unidirectional dependency inherent in autoregressive generation. In a decoder-only architecture with causal masking, a face block can attend only to previously generated blocks. This results in an asymmetric information flow: a face generated later can condition on its earlier neighbors, but not vice versa. Under this constraint, the conventional MLA objective is no longer fully appropriate. To accommodate this causal structure, we introduce a structural constraint: faces should be ordered in *descending degree* (i.e., faces with higher degree are placed earlier in the sequence). This ensures that faces with richer connectivity appear early, thereby maximizing the opportunity for their adjacent faces to attend to them during subsequent generation steps. Consequently, the face block ordering problem becomes a *Minimum Linear Arrangement problem with a descending-degree constraint*, preserving the core MLA objective while explicitly incorporating the causal prior required by autoregressive modeling.

Analysis of edge block ordering loss. In our sequence layout, all face blocks appear before a single *SEP* token, followed by all edge blocks. Each face block consists of a fixed number of k_f tokens, and each edge block consists of a fixed number of k_e tokens. Let p_{SEP} denote the token position of the *SEP* token.

For any edge block e and an incident face block f, the block-level distance between their starting positions decom-

poses as:

$$L_E = \underbrace{\sum_{e\in\mathcal{E}} \sum_{f\in\mathcal{F}(e)} \big(p_E(e) - p_{\text{SEP}}\big)}_{\text{edge-to-SEP}} + \underbrace{\sum_{e\in\mathcal{E}} \sum_{f\in\mathcal{F}(e)} \big(p_{\text{SEP}} - p_F(f)\big)}_{\text{face-to-SEP}} \tag{13}$$

Here, the total block-level distance between incident face blocks f and edge blocks e decomposes into two aggregated components: the *face-to-SEP* term, which quantifies the total distance of the face blocks to the separator, and the *edge-to-SEP* term, which quantifies the total distance of the edge blocks to the separator.

We observe that the edge-to-SEP distances form an arithmetic sequence, since all edge blocks are placed contiguously right after the *SEP* token. Consequently, the sum of these distances is:

$$\text{edge-to-SEP} = N_e + k_e \cdot \frac{(N_e - 1)N_e}{2} \tag{14}$$

Here, N_e denotes the number of edge blocks and k_e is the size (i.e., number of tokens) of each edge block, both of which are fixed constants for a given input. The first edge block is at distance 1 from *SEP*, and each subsequent block is shifted by k_e positions. Since this sum depends only on the constants N_e and k_e, the edge-to-SEP contribution to the edge block ordering loss L_E (Equation 12) remains unchanged under any permutation of the edge blocks. Consequently, L_E is determined solely by the ordering of the face blocks.

For face-to-SEP, we observe that a face block f of degree $\deg(f)$ contributes its distance to *SEP* exactly $\deg(f)$ times, once for each incident edge. This yields the expression:

$$\text{face-to-SEP} = \sum_{f\in\mathcal{F}} \deg(f) \cdot (p_{\text{SEP}} - p_F(f)) \tag{15}$$

where $\mathcal{F}$ denotes the set of all face blocks. Moreover, when generating any edge block, the full set of faces is already available as context, so the causal dependency for edge prediction is naturally satisfied. Therefore, to reduce L_E, faces with higher degree should be placed closer to *SEP*, which means positioning them later in the face block sequence to effectively minimize the weighted sum of their distances.

Balanced Ordering Strategies. Based on our analyses of the Face and Edge block ordering losses, we find that the two objectives impose mutually conflicting constraints on where high-degree faces should be placed in the sequence:

- Minimizing L_F corresponds to a minimum linear arrangement (MLA) problem with a *high-degree-first* constraint: high-degree faces should be placed early in the sequence.
- Minimizing L_E requires placing high-degree faces late in the sequence, near *SEP*, to minimize their weighted distance contribution.

This leads to a multi-objective constrained minimum linear arrangement problem, for which we propose the following heuristic strategies:

- **Prioritize minimizing** L_F:
  - *Spectral Ordering (SS)*: Perform spectral ordering of faces using the Fiedler vector, which is the second smallest eigenvector of the graph Laplacian and a standard technique for approximating solutions to the MLA problem. After obtaining the initial ordering, compute the average degree of faces in the first and second halves of the sequence. If the second half exhibits a higher average degree, reverse the entire sequence to place high-degree faces earlier.
- **Prioritize minimizing** L_E:
  - *Degree-Ascending Ordering (DEG-A)*: Sort faces by degree in ascending order so that low-degree faces come first and high-degree faces appear last. This ordering strictly minimizes the edge–face distance term in L_E.
- **Balanced strategies** that jointly account for both the Face block ordering loss (L_F) and Edge block ordering loss (L_E):
  - *Centroid-based Ordering (ZYX)*: Sort faces lexicographically by their centroid coordinates in the order Z, then Y, then X;
  - *Breadth-First Search (BFS)*: Start from the highest-degree face and perform a BFS that preferentially expands toward lower-degree neighboring faces.
  - *Depth-First Search (DFS)*: Start from the highest-degree face and similarly prioritize visiting lower-degree neighbors during traversal.

The ordering of edge blocks does not affect the values of L_F or L_E. However, to avoid excessively large differences between $p_E(e)$ and $p_F(f)$ that could destabilize the attention mechanism in autoregressive modeling, we adopt the *maximum adjacent face index ascending* strategy (MAX-IDX-A). Specifically, for each edge, we define its sorting key as the maximum index among all its adjacent faces in the face sequence, and then sort all edges in ascending order of this key. This approach encourages each edge to appear close to its neighboring faces in the sequence, thereby improving the distribution of edge–face positional offsets. The experimental results for the face and edge block ordering strategies can be found in Section 5.5.