SolidGen: An Autoregressive Model for Direct B-rep Synthesis

최우진

2022/07/28

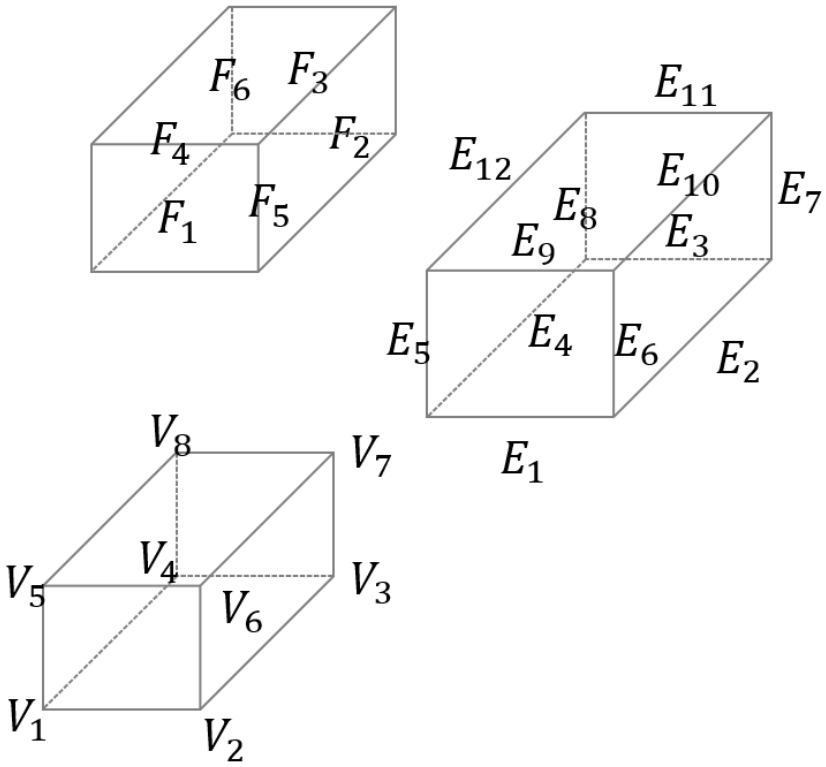
1. Introduction

- 거의 모든 object는 CAD를 통해서 생성한다.
- Boundary representation format (B-rep)을 통해 CAD object를 표현한다.
- 최근 연구는 B-rep 형식의 기본 3D 기하학 정보를 사용하는 대신 CAD 모델링 작업 시퀀스를 생성하는 데 초점을 맞추고 있다. [1, 2, 3]
- CAD에서 널리 사용되는 Fillet이나 Chamfer는 Sketch & Extrude에서는 사용하기 복잡하다.

[1] Karl D. D. Willis, Yewen Pu, Jieliang Luo, Hang Chu, Tao Du, Joseph G. Lambourne, Armando Solar-Lezama, and Wojciech Matusik. 2021. Fusion 360 Gallery: A Dataset and Environment for Programmatic CAD Construction from Human Design Sequences. ACM Transactions on Grap hics (TOG) 40, 4 (2021).
[2] Rundi Wu, Chang Xiao, and Changxi Zheng. 2021. DeepCAD: A Deep Generative Network for Computer-Aided Design Models. In IEEE International Conference on Comp uter Vision (ICCV). 6772–6782.
[3] Xianghao Xu, Wenzhe Peng, Chin-Yi Cheng, Karl D.D. Willis, and Daniel Ritchie. 2021. Inferring CAD Modeling Sequences Using Zone Graphs. In IEEE Conference on Comp uter Vision and Pattern Recognition (CVPR). 6062–6070.

1. Introduction

- **Boundary representation format (B-rep)**



Face		Edge		Vertex	
Face	Edges	Edge	Vertices	Vertex	Coordinate
F_1(앞)	$E_1E_6E_9E_5$	E_1	V_1V_2	V_1	$x_1y_1z_1$
F_2(오른)	$E_2E_7E_{10}E_6$	E_2	V_2V_3	V_2	$x_2y_2z_2$
F_3(뒤)	$E_3E_8E_{11}E_7$	E_3	V_3V_4	V_3	$x_3y_3z_3$
F_4(왼)	$E_4E_5E_{12}E_8$	E_4	V_4V_1	V_4	$x_4y_4z_4$
F_5(아래)	$E_1E_2E_3E_4$	E_5	V_5V_1	V_5	$x_5y_5z_5$
F_6(위)	$E_9E_{10}E_{11}E_{12}$	E_6	V_6V_2	V_6	$x_6y_6z_6$
		E_7	V_7V_3	V_7	$x_7y_7z_7$
		E_8	V_8V_4	V_8	$x_8y_8z_8$
		E_9	V_5V_6		
		E_{10}	V_6V_7		
		E_{11}	V_7V_8		
		E_{12}	V_8V_5		

1. Introduction

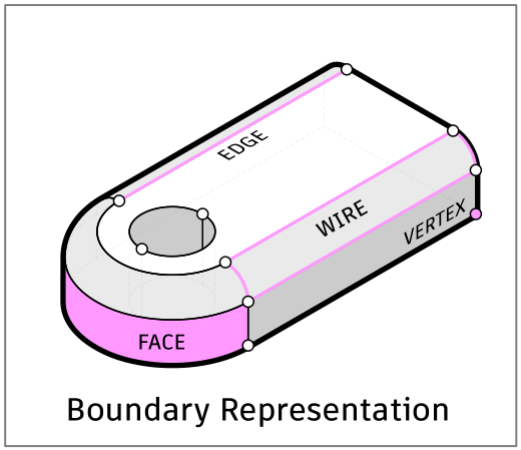
- **B-rep 합성 장점**
- **많은 데이터셋을 구하기 쉽다.**
  - 주로 과정을 저장하는 경우보다는 생성된 결과만 저장하는 경우가 많다.
- **기하학이랑 위상학은 비슷하게 표현되기 때문에 다음과 같은 경우도 B-rep으로 생성 가능하다.**
  - Freeform curve and surface (예: Bezier & non-uniform rational B-spline)
  - advanced topological structure (예: T-spline & Catmull-Clark subdivision mesh)
- **CAD의 몇가지 문제는 B-rep 형식으로만 해결 가능하다.**
  - 예: 데이터 교환으로 인해 생기는 평면이 사라지거나 잘린 경우에 생기는 구멍을 메우는 작업

1. Introduction

- **SolidGen Contibution**
- **CAD 시퀀스를 사용하지 않고 B-rep CAD model을 직접 합성하는 딥러닝 모델이다.**
- **새로운 표현 방식을 제시한다.**
  - Indexed Boundary Representation
  - B-rep을 숫자 배열로 변경하여 ML에 적용하였다.
  - 다시 원본 B-rep으로 완벽하게 복원 가능하다.
- **정량적 평가와 정성적 평가 모두 진행한다.**
  - 아무 조건 없이 모델을 생성 하는 경우
  - Class를 조건으로 제공해서 모델을 생성하는 경우

2. Representation

- **Boundary Representation**
- **B-rep 데이터 구조는 face, wire, edge, vertex으로 이뤄져 있다.**
- **B-rep 데이터 구조는 solid modeling에서는 효율적이지만,**
  머신 러닝을 통해 학습하기에는 복잡한 표현이다.
- **최근 B-rep 인코딩[4, 5, 6]에는 상당한 진전이 있었지만,**
  생성기반 작업을 위한 B-rep 디코딩은 여전히 미해결 과제이다.

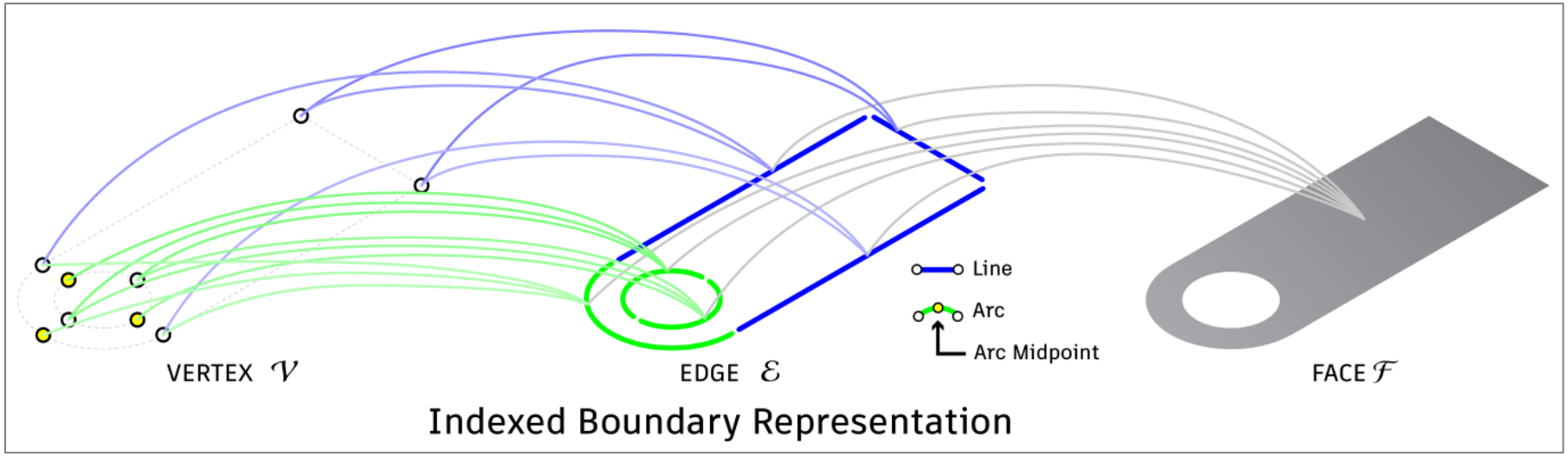


Boundary Representation

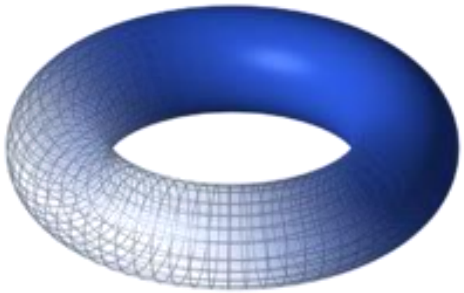
[4] Pradeep Kumar Jayaraman, Aditya Sanghi, Joseph G. Lambourne, Karl D.D. Willis, Thomas Davies, Hooman Shayani, and Nigel Morris. 2021. UV-Net: Learning From Boundary Representations. In IEEE Conference on Comp uter Vision and Pattern Recognition (CVPR). 11703–11712.
[5] Joseph G. Lambourne, Karl D.D. Willis, Pradeep Kumar Jayaraman, Aditya Sanghi, Peter Meltzer, and Hooman Shayani. 2021. BRepNet: A Topological Message Passing System for Solid Models. In IEEE Conference on Comp uter Vision and Pattern Recognition (CVPR). 12773–12782.
[6] Karl DD Willis, Pradeep Kumar Jayaraman, Hang Chu, Yunsheng Tian, Yifei Li, Daniele Grandi, Aditya Sanghi, Linh Tran, Joseph G Lambourne, Armando Solar-Lezama, et al. 2021. JoinABLe: Learning Bottom-up Assembly of Parametric CAD Joints. arXiv:2111.12772 (2021).

2. Representation

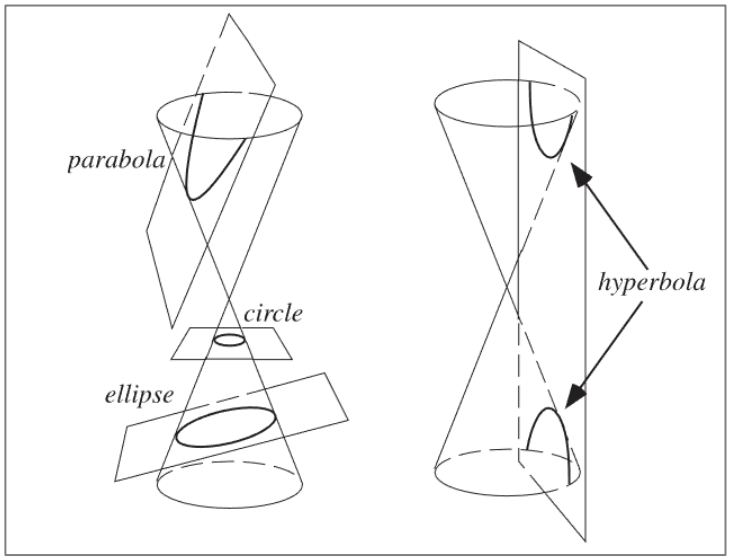
- **Indexed Boundary Representation**
- **B-rep 데이터를 ML에 사용하기 적합한 숫자 배열로 변경한다.**
- **Curve (line & arc)와 Surface (plane, cylinder, cone, sphere, torus)로 제한한다.**
- **B-spline과 Conic Section은 사용하지 않는다.**



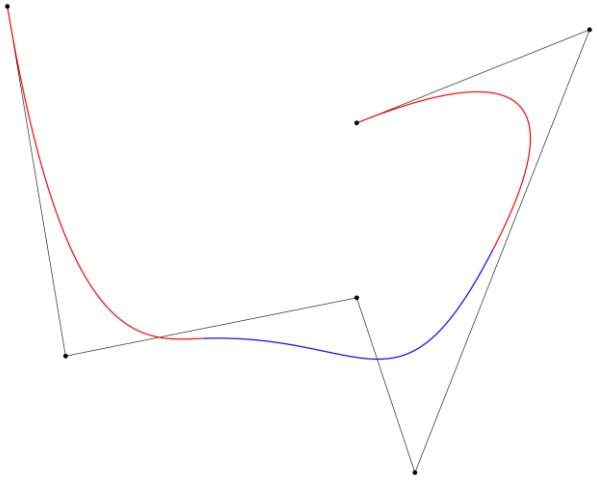
2. Representation



torus



Conic Section



B-spline

3. SolidGen Architecture

- **PolyGen[7]과 CurveGen[8]을 기반으로 SolidGen을 만들었다.**
  - PolyGen은 n각형의 mesh의 분포를 학습하는 모델
  - CurveGen은 2D Sketch의 분포를 학습하는 모델
- **SolidGen에서 학습하고자 하는 분포**

$$p(\mathcal{B}) = p(\mathcal{V}, \mathcal{E}, \mathcal{F}). \quad (1)$$

$$p(\mathcal{B}) = p(\mathcal{F}|\mathcal{E}, \mathcal{V})p(\mathcal{E}|\mathcal{V})p(\mathcal{V}), \quad (2)$$

- B-reps: $\mathcal{B}$
- Vertices: $\mathcal{V}$
- Edges: $\mathcal{E}$
- Faces: $\mathcal{F}$

- **추가적으로 조건을 사용하고자 하는 경우 (z: 조건)**

$$p(\mathcal{B}) = p(\mathcal{F}|\mathcal{E}, \mathcal{V})p(\mathcal{E}|\mathcal{V})p(\mathcal{V}|z)p(z)$$

[7] Charlie Nash, Yaroslav Ganin, SM Ali Eslami, and Peter Battaglia. 2020. Polygen:An autoregressive generative model of 3d meshes. In International Conference on Machine Learning (ICML). PMLR, 7220–7229.
[8] Karl DD Willis, Pradeep Kumar Jayaraman, Joseph G Lambourne, Hang Chu, and Yewen Pu. 2021. Engineering sketch generation for computer-aided design. In IEEE Conference on Comp uter Vision and Pattern Recognition Workshop s (CVPR Workshop). 2105–2114.

3. SolidGen Architecture

- **Vertex Model**
- **B-rep 데이터를 ML에 사용하기 적합한 숫자 배열로 변경한다.**
- **Vertex $\mathcal{V}$를 사전순으로 정렬(x, y, z 순서)하여 1D list으로 평탄화하고 끝에 <EOS>을 추가한다.**

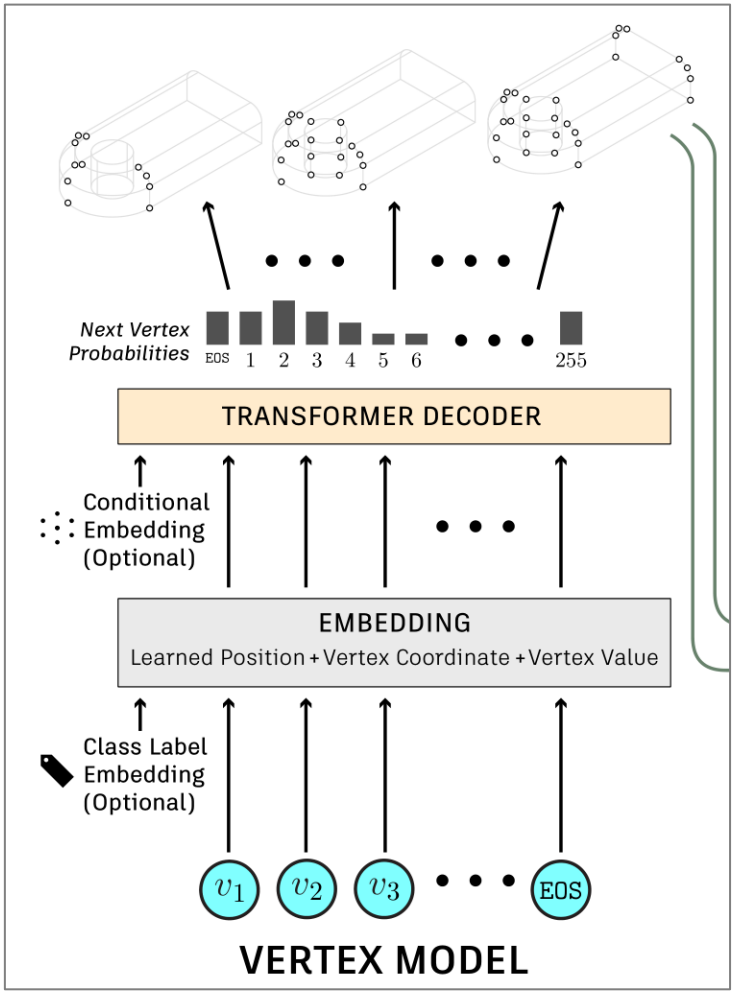
$$\mathcal{V}^{\text{seq}} = \{v_1, v_2, \ldots, v_{|\mathcal{V}^{\text{seq}}|-1}, < EOS >\} \ (|\mathcal{V}^{seq}| = (|\mathcal{V}| \times 3) + 1)$$

- **모델은 다음과 같은 분포를 학습한다.**
- **단, $\theta_{\mathcal{V}}$는 Transformer Decoder의 학습 가능한 매개변수이다.**

$$p(\mathcal{V}^{\text{seq}}; \theta_{\mathcal{V}}) = \prod_{t=1}^{|\mathcal{V}^{\text{seq}}|} p(v_t \mid v_1, v_2, \ldots, v_{t-1}; \theta_{\mathcal{V}}), \qquad (3)$$

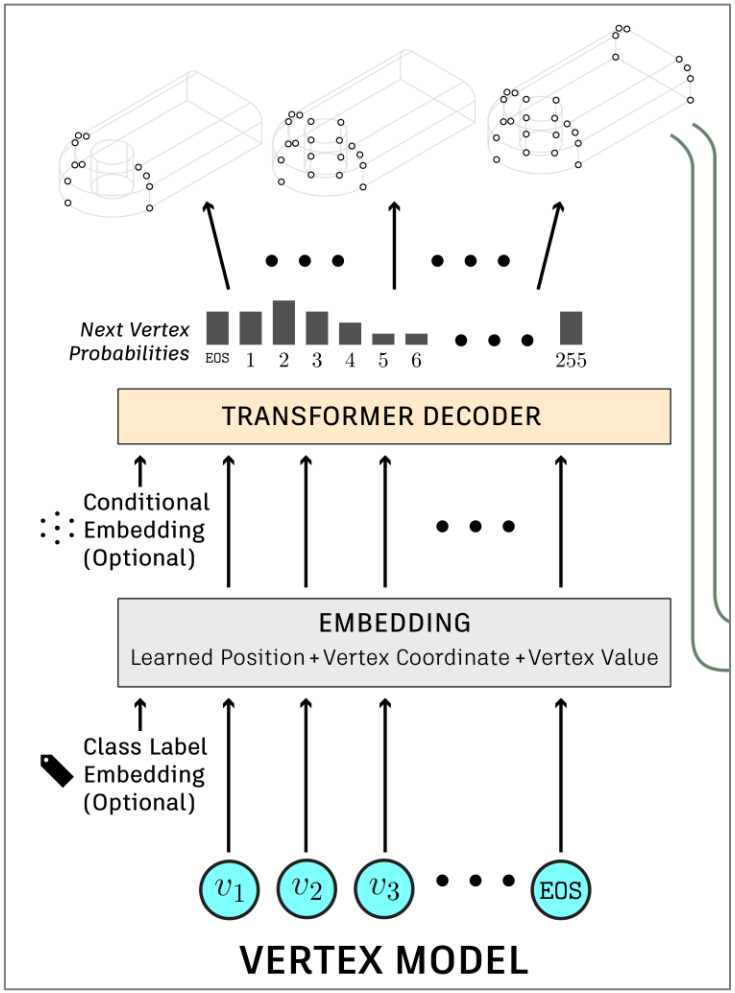
3. SolidGen Architecture

- **Vertex Model**
- **Positional Embedding**
  - 토큰이 속한 Vertex index의 위치를 나타낸다.
- **Coordinate Embedding**
  - 토큰이 x, y, z 좌표인지 여부를 나타낸다.
- **Value Embedding**
  - 8비트로 양자화된 x, y, z 좌표값을 나타낸다.
  - <EOS>토큰까지 포함해서 $2^8 + 1$차원으로 양자화한다.
- **Class Label Embedding (Optional)**
  - Class Label을 임베딩을 한다.
  - 추후 위에 임베딩하고 Concat을 통해 결합한다.



3. SolidGen Architecture

- **Vertex Model**
- 각 단계에서 모델은 다음 토큰에 대한
  모든 가능한 꼭짓점 위치의 확률 분포를 예측한다.
- 각 단계마다 $2^8 + 1$차원의 정규화 되지 않은
  로그 확률이 출력된다.



3. SolidGen Architecture

- **Edge Model**
- **B-rep 데이터를 ML에 사용하기 적합한 숫자 배열로 변경한다.**
- **Edge $\mathcal{E}$를 1D list으로 평탄화하고 새 에지 시작을 <NEW_EDGE>, 마지막을 <EOS>으로 추가한다.**

$$\mathcal{E}^{seq} = \{e_1, e_2, < NEW\_EDGE >, \dots, e_{|\mathcal{E}^{seq}|-1}, < EOS >\}$$

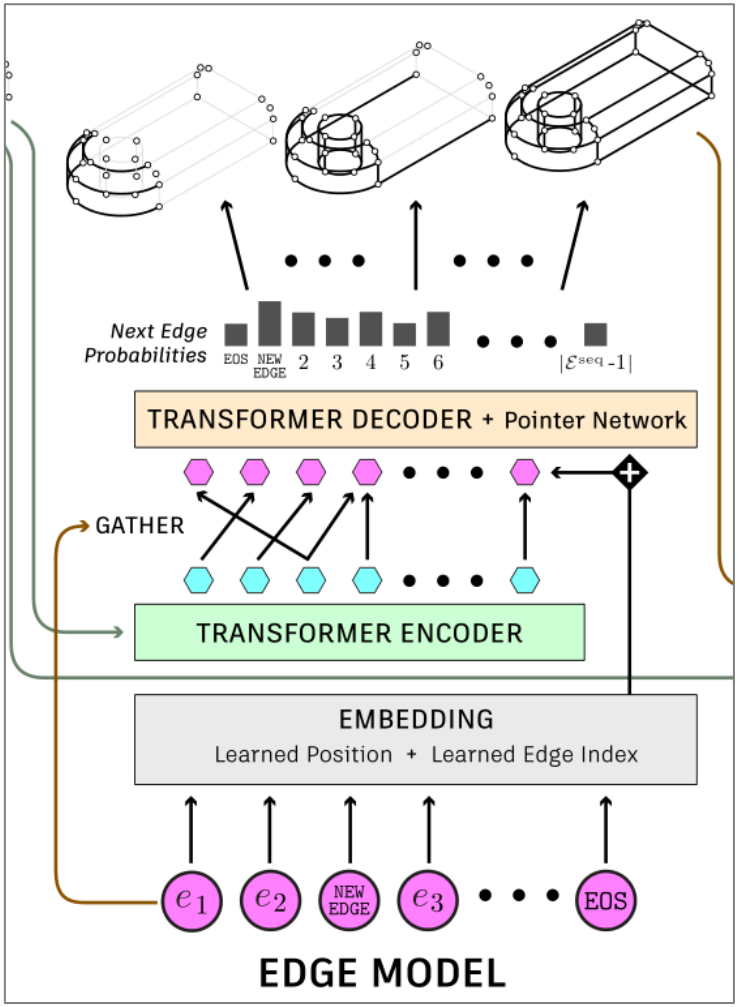
$$|\mathcal{E}^{seq}| = \sum_{E \in \mathcal{E}} (|E| + 1)$$

- **모델은 다음과 같은 분포를 학습한다.**
- **단, $\theta_\mathcal{E}$는 신경망의 학습 가능한 매개변수이다.**

$$p(\mathcal{E}^{\text{seq}}|\mathcal{V}; \theta_\mathcal{E}) = \prod_{j=1}^{|\mathcal{E}^{\text{seq}}|} p(e_j \mid e_1, e_2, \dots, e_{j-1}|\mathcal{V}; \theta_\mathcal{E}), \qquad (4)$$

3. SolidGen Architecture

- **Edge Model**
- **Value Embedding**
  - 8비트로 양자화된 x, y, z 좌표값을 나타낸다.
  - <EOS>토큰까지 포함해서 $2^8 + 1$차원으로 양자화한다.
  - $\mathcal{V} = \{(x_i, y_i, z_i)\}_{i=0}^{|\mathcal{V}|-1}$
  - $\mathbf{h}_{\mathcal{V}} = \{\text{Embed}_x(x_i) + \text{Embed}_y(y_i) + \text{Embed}_z(z_i)\}_{i=0}^{|\mathcal{V}|-1}, \quad (5)$
- **Transformer Encoder**
  - Vertex, <EOS>, <NEW_EDGE>의 임베딩 값을 입력으로 준다.
  - $\mathbf{h}_{\text{inp}} = T_{\text{enc}}^{\mathcal{E}}(\mathbf{h}_{\text{<EOS>}} \parallel \mathbf{h}_{\text{<NEW\_EDGE>}} \parallel \mathbf{h}_{\mathcal{V}}),$
  - $h_{inp}: (|\mathcal{V}| + 2, 256)$ 차원

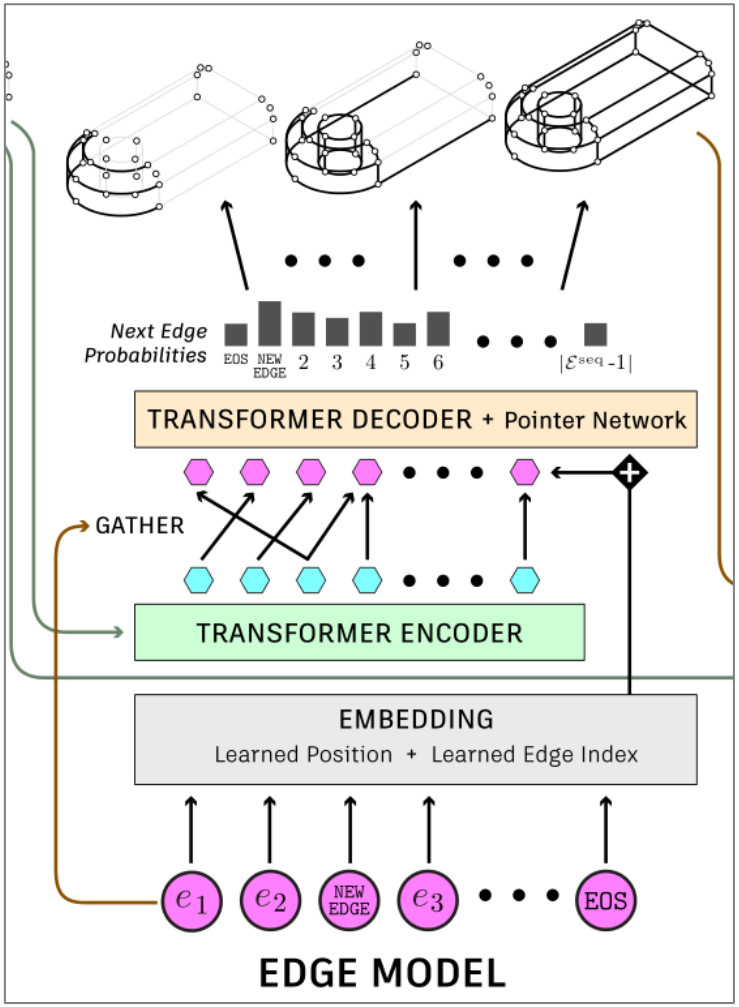


3. SolidGen Architecture

- **Edge Model**
- **Positional Embedding & Edge Index Embedding**
  - 각 토큰의 위치를 임베딩한다.
  - 각 토큰을 해당 에지에 매핑한다.
- **Edge Embedding**
  - $\mathcal{E}_{<t}^{seq}$의 vertex index에 상응하는 h_{inp}을 추출한다.
  - $\mathbf{h}_{\mathcal{E}_{<t}^{\text{seq}}} = \{\mathbf{h}_{\text{inp}}[e_j]\}_{j=1}^{t-1}$. ($|\mathcal{E}_{<t}^{seq}| \times 256$ 차원)
- **Transformer Decoder**

$$\mathbf{p}_t = T_{\text{dec}}^{\mathcal{E}}(\mathbf{h}_{\mathcal{E}_{<t}^{\text{seq}}}),$$

$$p(e_t = k \mid \mathcal{E}_{<t}^{\text{seq}}, \mathcal{V}) = \text{softmax}_k(\mathbf{p}_t \cdot \mathbf{h}_{\text{inp}}[k]).$$



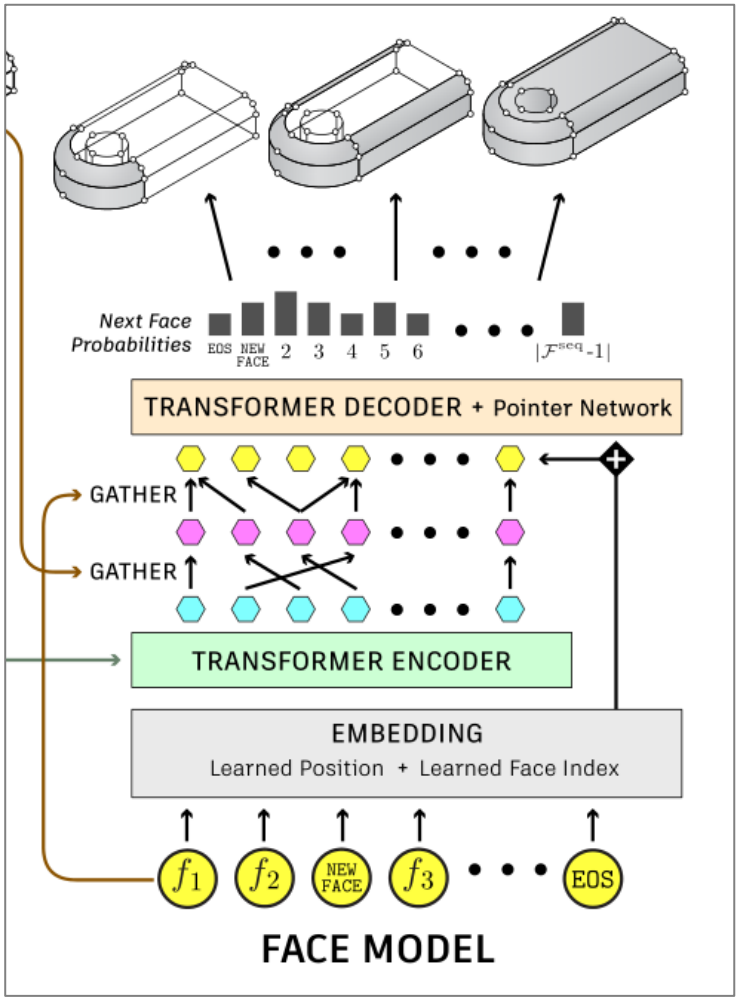
3. SolidGen Architecture

- **Face Model**
- B-rep 데이터를 ML에 사용하기 적합한 숫자 배열로 변경한다.
- Face $\mathcal{F}$를 1D list으로 평탄화하고 새 면 시작을 <NEW_FACE>, 마지막을 <EOS>으로 추가한다.
- 모델은 다음과 같은 분포를 학습한다.
- 단, $\theta_{\mathcal{F}}$는 신경망의 학습 가능한 매개변수이다.

$$p(\mathcal{F}^{\text{seq}}|\mathcal{E},\mathcal{V};\theta_{\mathcal{F}}) = \prod_{t=1}^{|\mathcal{F}^{\text{seq}}|} p(f_t \mid f_1, f_2, \ldots, f_{t-1}|\mathcal{E},\mathcal{V};\theta_{\mathcal{F}}), \quad (6)$$

3. SolidGen Architecture

- **Face Model**
- **Value Embedding**
  - $\mathbf{h}_{\mathcal{E}} = \{\sum_{v \in E} \mathbf{h}_{\mathcal{V}}[v]\}_{E \in \mathcal{E}}$.
- **Face Embedding**
  - $\mathbf{h}_{\mathcal{F}^{\text{seq}}_{<t}} = \{\mathbf{h}_{\mathcal{E}}[f_k]\}_{k=1}^{t-1}$.
- **Transformer Encoder**
  - Vertex, <EOS>, <NEW_EDGE>의 임베딩 값을 입력으로 준다.
  - $\mathbf{h}_{\text{inp}} = T^{\mathcal{F}}_{\text{enc}}(\mathbf{h}_{\texttt{<EOS>}} \parallel \mathbf{h}_{\texttt{<NEW\_FACE>}} \parallel \mathbf{h}_{\mathcal{E}})$.
  - h_{inp}: $(|\mathcal{V}| + 2, 256)$ 차원



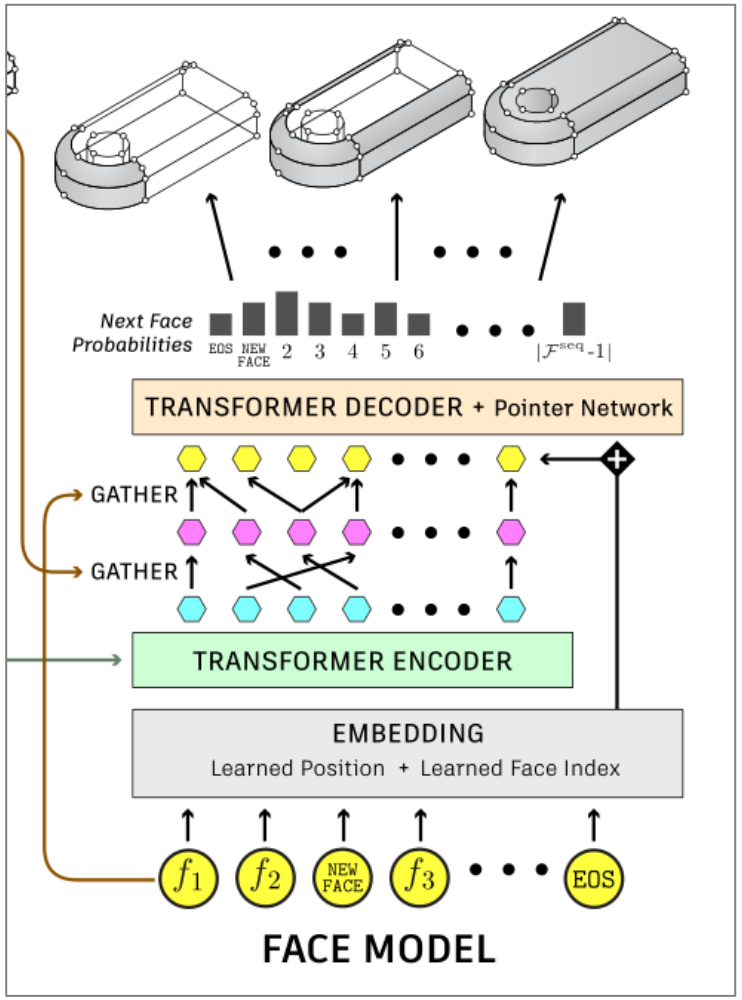
3. SolidGen Architecture

- **Face Model**
- **Transformer Decoder**

$$\mathbf{p}_t = T^{\mathcal{F}}_{\text{dec}}(\mathbf{h}_{\mathcal{F}^{\text{seq}}_{<t}}),$$

$$p(f_t = k \mid \mathcal{F}^{\text{seq}}_{<t}, \mathcal{E}, \mathcal{V}) = \text{softmax}_k(\mathbf{p}_t \cdot \mathbf{h}_{\mathcal{E}}[k]).$$

h_{inp}



4. Masking Invalid Outputs

- **Invalid Outputs**
  - 모델이 출력한 vertex 시퀀스의 길이 - 1(<EOS>)이 3으로 나눠지지 않는 경우
  - <NEW_EDGE> 토큰이 line, arc 뒤에 나오지 않은 경우
  - 동일한 edge에 속하는 edge token이 유일하지 않은 경우 (인덱스가 겹치는 경우)
  - 각 face이 edge들로 닫힌 boundary가 아닌 경우

5. Experiments

Implementation Details

- Pytorch로 구현, Adam optimizer 사용, Learning Rate의 초기값: 10^{-4}
- Teacher-forcing을 사용하기 때문에 3가지 모델 모두 병렬로 학습 가능하다.
- Solid Modeling kerne은 OpenCascade/pythonOCC을 사용한다.

5. Experiments

Datasets

Parametric Variations (Pvar) Dataset

- SolidGen을 Class 조건을 추가해서 testing하기 위한 데이터셋
- 60개의 template Solid(즉, Class)이며, sketch 차원, 확장 거리, 확장된 wire를 조절할 수 있는 parameter가 있다.
- 이러한 parameter을 변경해서 기하학적으로 다르지만 위상은 거의 동일한 Solid Sample 생성한다.
- 총 120,000 models 생성, 각 template Solid마다 2,000개 생성한다.
- 데이터를 90% (학습) / 5% (테스트) / 5% (검증) 으로 나눠서 사용한다.

DeepCAD Dataset

- 기존 데이터에 중복이 상당 부분이 있어서 해시 기반으로 중복을 제거한다.
- 간단한 모델 (< 8 faces)이나 복잡한 모델 (> 130 faces)는 제거하였다.
- 64,449개 학습 데이터 / 3,578개 검증 데이터 / 3,577개 테스트 데이터로 나눠서 사용한다.

5. Experiments

Metrics

Valid

- (1) 모든 face는 적어도 하나의 삼각형은 포함해야 한다.
- (2) wire의 각 edge는 제대로 정렬되어야 한다.
- (3) wire는 교차를 하지 말아야 한다.

Novel

- Valid 중에서 training set에 없는 경우이며, 값이 작을수록 모델이 학습 데이터를 기억하는 것을 의미한다.

Unique

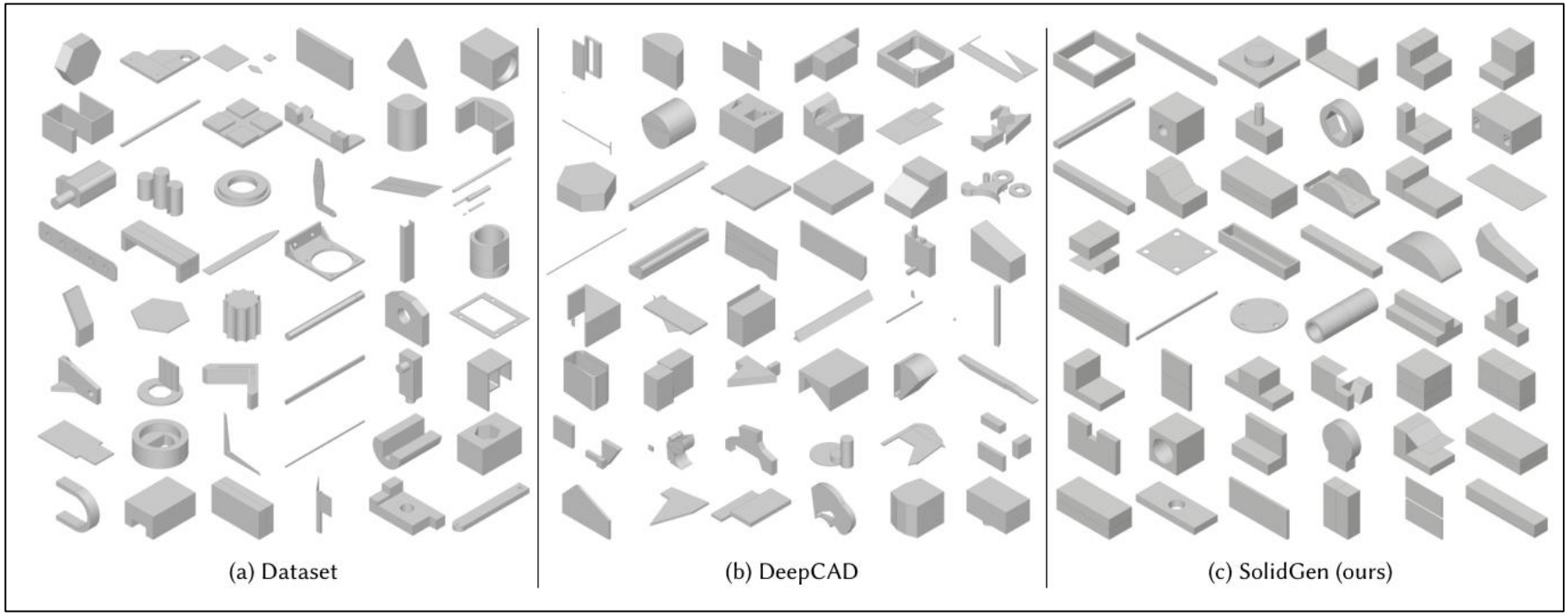
- Valid 중에서 서로 겹치지 않는 경우이며, 값이 작을수록 생성 데이터의 분산이 작은 것을 의미한다.

Accuracy

- Class를 조건으로 주고 생성한 경우에만 측정한다.

5. Experiments

- **Unconditional Generation**
- **DeepCAD와 비교 (정성적 평가)**



5. Experiments

Unconditional Generation

DeepCAD와 비교 (정량적 평가)

- DeepCAD 데이터셋을 이용하여 학습한 모델을 통해 1000 랜덤 샘플 생성
- Valid가 높은 이유: (1) 최종 B-rep를 index B-rep을 통해 기존보다 더 잘 표현하였다.
  (2) 마스킹을 통해 불필요한 출력을 일부 제거하였다.
  (3) B-rep을 여러 요소로 분해해서 어려운 문제를 단순화하여 표현해서 학습하였다.
- Novel, Unique가 낮은 이유:
  - DeepCAD의 유효한 샘플 중 많은 수가 노이즈가 많고 비현실적이어서 독특하기 때문이다.
- 결과

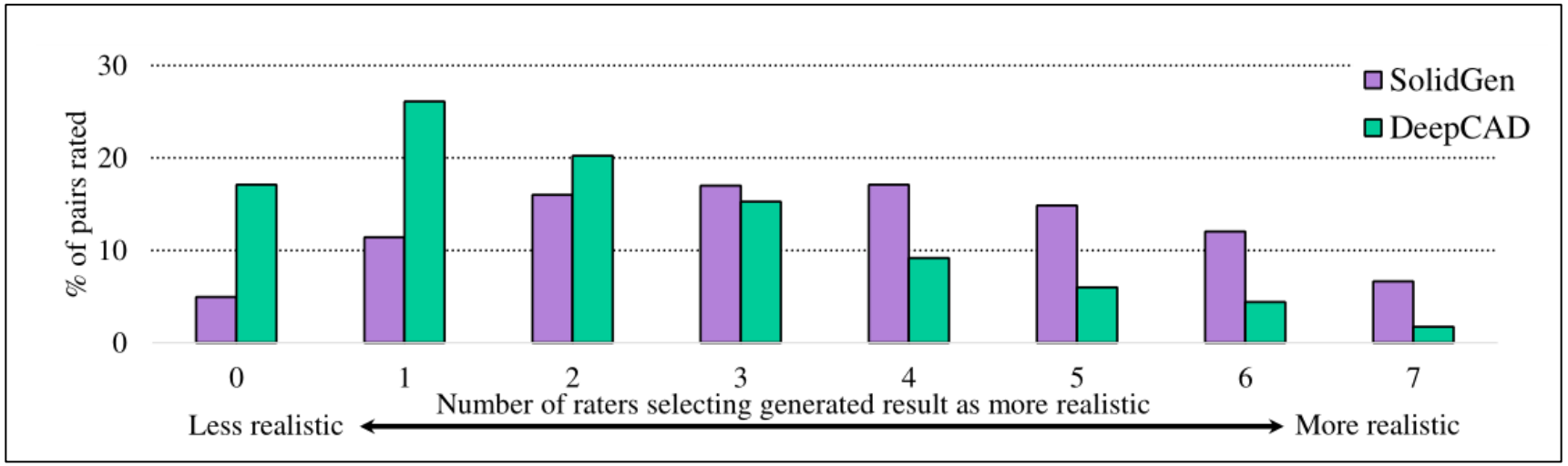
Method	Valid (↑)	Novel (↑)	Unique (↑)
Ours	92.133	86.638	89.146
DeepCAD	72.067	99.214	99.861

5. Experiments

Unconditional Generation

Human Perceptual Evaluation

- 생성된 샘플을 Amazon의 Mechanical Turk service를 통해 인지 평가를 진행하였다.
- 인지 평가 작업자에게 SolidGen이나 DeepCAD으로 생성된 CAD와 학습 데이터의 랜덤 선택된 CAD를 비교하여 둘 중 어느 것이 더 현실적인지 평가하도록 한다.



5. Experiments

Conditional Generation

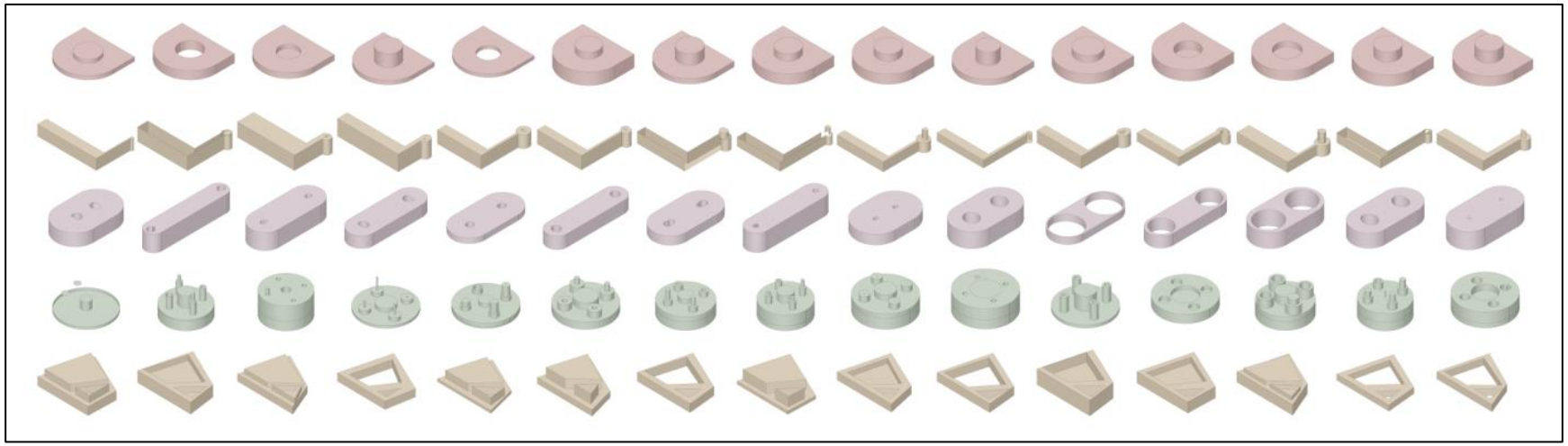
- PVar Dataset을 통해 학습 및 평가한다.
- Unconditional Generation에서 사용한 모델을 동일하게 사용한다.
- 40개의 Class를 사용하여 2400개의 샘플을 생성한 결과

Valid (↑)	Novel (↑)	Unique (↑)	Accuracy (↑)
99.167	90.294	98.655	75.198

5. Experiments

Conditional Generation

- PVar Dataset을 통해 학습 및 평가한다.
- Unconditional Generation에서 사용한 모델을 동일하게 사용한다.
- 40개의 Class를 사용하여 2400개의 샘플을 생성한 결과



SolidGen: An Autoregressive Model for Direct B-rep Synthesis

감사합니다