MeshXL: Neural Coordinate Field for Generative 3D Foundation Models

Sijin Chen[1,2,*], **Xin Chen**[2,†], **Anqi Pang**[2], **Xianfang Zeng**[2], **Wei Cheng**[2], **Yijun Fu**[2], **Fukun Yin**[1,2], **Zhibin Wang**[2], **Jingyi Yu**[3], **Gang Yu**[2], **Bin Fu**[2], **Tao Chen**[1,‡]
https://github.com/OpenMeshLab/MeshXL
[1]Fudan University [2]Tencent PCG [3]ShanghaiTech University
[†] project lead [‡] corresponding author

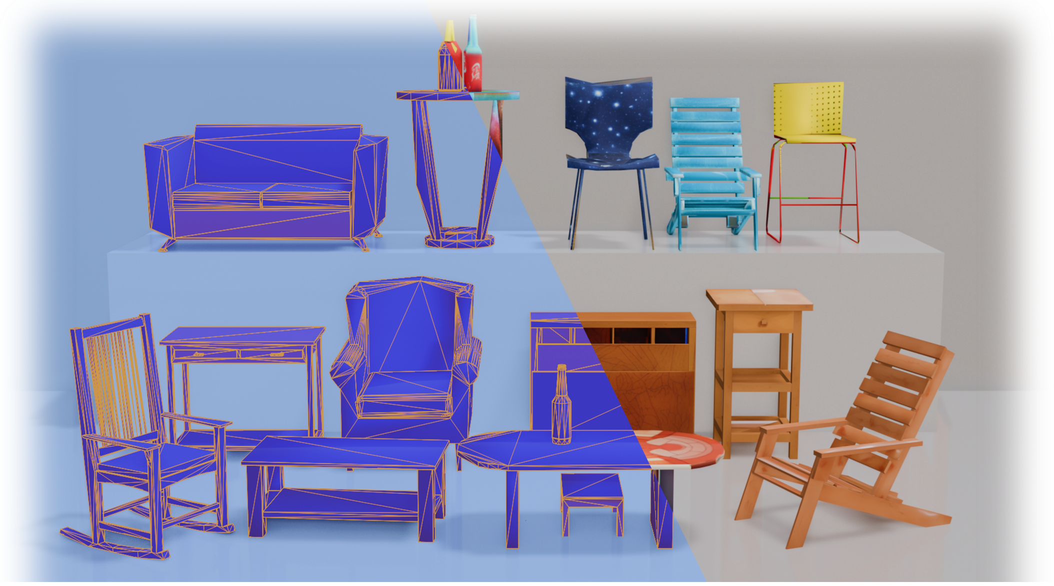


Figure 1: **MeshXL** can auto-regressively generate high-quality 3D meshes. We validate that **Neur**al **C**oordinate **F**ield (NeurCF), an explicit coordinate representation with implicit neural embeddings, is a simple-yet-effective sequence representation for large-scale mesh modelling.

Abstract

The polygon mesh representation of 3D data exhibits great flexibility, fast rendering speed, and storage efficiency, which is widely preferred in various applications. However, given its unstructured graph representation, the direct generation of high-fidelity 3D meshes is challenging. Fortunately, with a pre-defined ordering strategy, 3D meshes can be represented as sequences, and the generation process can be seamlessly treated as an auto-regressive problem. In this paper, we validate **Neur**al **C**oordinate **F**ield (NeurCF), an explicit coordinate representation with implicit neural embeddings, is a simple-yet-effective representation for large-scale sequential mesh modeling. After that, we present MeshXL, a family of generative pre-trained auto-regressive models that addresses 3D mesh generation with modern large language model approaches. Extensive experiments show that MeshXL is able to generate high-quality 3D meshes, and can also serve as foundation models for various down-stream applications.

[*] Research done when Sijin Chen was a research intern at Tencent PCG.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

1 Introduction

The generation of high-quality 3D assets [61, 77, 29] is essential for various applications in video games, virtual reality, and robotics. Among existing 3D representations [51, 38, 57, 61], 3D mesh represents 3D data as graphs, which possesses the flexibility and accuracy representing sharp edges as well as both flat and curved surfaces. However, the direct generation of high-quality 3D meshes is challenging, given 1) the unstructured graph representation and 2) the demand for estimating accurate spatial locations and connectivity within vertices.

To generate 3D meshes, many works adopt an indirect way by first producing data in other 3D representations, such as point clouds [97, 49, 54], SDF [88, 94], and multi-view images [46, 82, 30]. After that, they adopt re-meshing methods [37] to post-process the generated geometries. There are also attempts towards the direct generation of 3D polynomial meshes. PolyGen [53] adopts two separate decoder-only transformers for vertices generation and vertices connectivity prediction. MeshGPT [65] first builds a mesh VQVAE to first turn meshes into tokens, and then learns to generate the token sequences with a GPT model [59]. Meanwhile, PolyDiff [2] directly adopts discrete denoising diffusion [4] on the discretized mesh coordinates.

Though these methods have achieved initial success in creating 3D assets, they suffer from certain limitations. To preserve sufficient high-frequency information, point clouds and voxels requires dense samplings on the object surfaces, which inevitably leads to great redundancy while representing flat surfaces. The reconstruction-based methods [82, 30, 67], however, rely heavily on the accuracy of the multi-vew generation pipelines [46]. Additionally, the VQVAE-based 3D generation methods [88, 65] require sophisticated multi-stage training, which less favors learning from large scale data.

To tackle the above challenges and explore the potential of scaling up 3D generative pre-training, we introduce a simple-yet-effective way of 3D mesh representation, the **Neur**al **C**oordinate **F**ield (NeurCF). NeurCF represents the explicit 3D coordinates with implicit neural embeddings. We show that with a pre-defined ordering strategy, a 3D mesh can be represented by a one-and-only coordinate sequence, which further helps us formulate 3D mesh generation as an auto-regressive problem. After that, we present MeshXL, a family of generative pre-trained transformers [93, 59], for the direct generation of high-fidelity 3D meshes. Without resorting to intermediate 3D representations, NeurCF facilitates an end-to-end learning pipeline for the direct pre-training on large-scale 3D mesh data.

By organizing high-quality 3D assets from ShapeNet [9], 3D-FUTURE [22], Objaverse [17], and Objaverse-XL [16], we achieve a collection of over 2.5 million 3D meshes to support large-scale generative pre-training. Extensive experiments demonstrate that the NeurCF representation facilitates MeshXL to generate higher-quality 3D meshes. By training on the collection of large-scale 3D mesh data, MeshXL can achieve better performance with larger numbers of parameters (Fig. 3 and Tab. 5), and surpass prior arts on multiple categories in the ShapeNet dataset [9] (Tab. 3).

In summary, our contributions can be summarized as follows:

- We validate that Neural Coordinate Field is a simple-and-effective representation of 3D mesh, which is also friendly to large-scale auto-regressive pre-training.
- We present a family of MeshXLs that can be treated as strong base models for image-conditioned or text-conditioned 3D mesh generation tasks.
- We show that MeshXL surpasses state-of-the-art 3D mesh generation methods, and can produce delicate 3D meshes compatible with existing texturing methods.

2 Related Work

First, we present a concise review of existing 3D representations. Subsequently, we discuss related works on 3D generation and recent efforts in developing 3D foundation models.

3D Representations. Researchers have long sought for accurate and efficient methods to represent 3D data. **Point Cloud** [54, 57, 58, 89] captures the spatial positions of discrete points in the Euclidean space, which is preferred by various 3D sensors [15, 87, 66, 3, 7]. **Mesh** [53, 2, 65, 12] represents the 3D structure with graphs. By connecting the vertices with edges, mesh can also be interpreted into a set of polygons in the 3D space. Similar to point clouds, **3D Gaussians** [38, 68] also record the discrete Euclidean distribution in 3D space. However, each point is represented by a 3D Gaussian distribution function parameterized by its covariance matrix, color, and opacity. Given their fast convergence and rendering speed, 3D gaussians are often utilized for 3D reconstruction. **Neural Radiance Field** (NeRF) [51, 5] constructs a learnable volumetric function f using neural networks trained on multi-view images. Due to its derivability and flexibility, NeRF is also favored for 3D generative models [46, 99, 76, 56]. Additionally, there are other 3D representations such as multi-view images [74, 90, 100], voxel fields [61, 13, 45], and signed distance fields [94], among others [64, 88, 63]. In this paper, we consider the **Neural Coordinate Field** (NeurCF), an explicit spatial representation with implicit neural embeddings, and investigate its potential for scalable 3D asset generation.

3D Generation. With the exploration of various 3D representations and the collection of large-scale 3D datasets [17, 9, 16], researchers have also put much effort exploring the generation of high-fidelity 3D assets [42, 39]. The **G**enerative **A**dversarial **N**etwork (GAN) [25, 80, 1, 33] produces synthetic 3D data with a generator $\mathcal{G}$, and train a discriminator network $\mathcal{D}$ to distinguish the generated and real data. Additionally, the potential of **diffusion** models [54, 28, 62] in the direct generation of 3D data is also widely explored [97, 2, 54, 50, 47]. The key idea behind diffusion is to transform the desired data distribution into a simpler distribution (*e.g.* gaussian) and learn a desnoising model for the reverse process. Besides, researchers have also explored the potential of diffusion models in generating **multi-view** images [46, 16, 82, 43], and reconstruct them into 3D structures. In this paper, we mainly explore the **auto-regressive** methods for 3D generation. AutoSDF [52] and MeshGPT [65] learn to generate discrete tokens and reconstruct them into 3D representations with a VQVAE model [72]. PolyGen [53] adopts two decoder-only transformers that predict the location and connectivity of vertices, sequentially. In this paper, we explore the potential of an explicit sequential modelling method for 3D meshes, and present a family of generative pre-trained transformers, MeshXL, for high-fidelity 3D mesh generation.

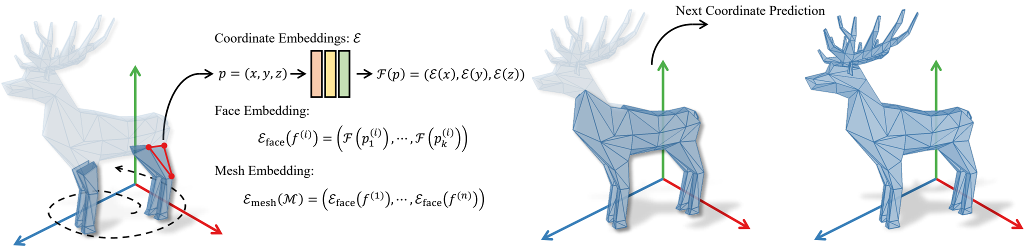
3D Foundation Models. The collection of large-scale high-quality 3D data [17, 16, 9, 81, 71, 21, 22] builds up the foundation for various 3D-related tasks [83, 27, 10, 41]. To explore the scaling effects in 3D learning, researchers have made great endeavors in building 3D foundation models for 3D understanding [96, 44, 98, 85, 86, 92, 100], reconstruction [30, 78, 67, 46, 16, 84, 73], and generation [61, 29, 65, 8]. With the introduction of large-scale 3D data in both variety and granularity [34, 41, 16], existing 3D foundation models are capable of generalizing to unseen concepts [100, 86, 44], generating high-fidelity 3D assets [88, 36, 65], responding to complex instructions [31, 10, 32, 41], and generating actions that interacts with the 3D environments [20, 79, 95]. In this paper, we present a fully end-to-end 3D mesh generation pipeline, explore the scaling effect for large-scale pre-training, and test whether our method can serve as a well-trained foundation model for various down-stream tasks.

3 Data

Data Sources. We provide details on the 3D data collections we use to train and evaluate our models. The whole data collection is built upon four widely-acknowledged 3D mesh datasets, *i.e.* ShapeNet V2 [9], 3D-FUTURE [22], Objaverse [17], and Objaverse-XL [16].

- **ShapeNet V2 [9]** collects about 51k 3D CAD models for 55 categories. We split the data in 9:1 for training and validation by each category.
- **3D-FUTURE [22]** present about 10k high-quality 3D mesh data for indoor furniture. However, because of the delicate design, the objects contain many faces. Therefore, only a small proportion of the data can be used to train our MeshXL models.
- **Objaverse [17]** is a large 3D data collection with more than 800k 3D objects for about 21k categories collected from Sketchfab. We split the data in 99:1 for training and validation, respectively.
- **Objaverse-XL [16]** further expand Objaverse [17] into a dataset with more than 10M 3D objects with additional data collected from GitHub, Polycam, Thingiverse, and Smithsonian. We split the Github and Thingiverse part of the Objaverse-XL dataset into 99:1 for training and validation.

Data collection and filtering. To organize existing datasets, we build up a filtering and pre-processing pipeline to ensure that the meshes meet our demand. We first collect meshes with fewer than 800 faces, and ensure that they have corresponding UV maps for rendering. After that, we render the 3D


Coordinate Embeddings: $\mathcal{E}$
$p = (x, y, z)$
$\mathcal{F}(p) = (\mathcal{E}(x), \mathcal{E}(y), \mathcal{E}(z))$
Face Embedding:
$\mathcal{E}_{\text{face}}(f^{(i)}) = \left(\mathcal{F}\left(p_1^{(i)}\right), \cdots, \mathcal{F}\left(p_k^{(i)}\right)\right)$
Mesh Embedding:
$\mathcal{E}_{\text{mesh}}(\mathcal{M}) = \left(\mathcal{E}_{\text{face}}(f^{(1)}), \cdots, \mathcal{E}_{\text{face}}(f^{(n)})\right)$
Next Coordinate Prediction

(a) Neural Coordinate Field

(b) Auto-regressive Mesh Generation

Figure 2: **Mesh Representation.** We present the **Neur**al **C**oordinate **F**ield (NeurCF) to encode the discretized coordinates in the Euclidean space. Benefiting from NeurCF and a pre-defined ordering strategy, our proposed MeshXL can directly generate the unstructured 3D mesh auto-regressively.

meshes, and discard those not center-aligned or occupying less than 10% of the rendered image. For those 3D objects with more than 800 but less than 20,000 faces, we use planar decimation to simplify the meshes. Finally, we achieve approximately 2.5 million 3D meshes (Tab. 1).

Planar Decimation Pipeline. To ensure the quality of the decimated 3D meshes, we make sure either a lower Hausdorff distance $\delta_{\text{hausdorff}}$ [65] or a similar rendered views [11].

Collecting text-mesh pairs. We render the 3D meshes with 12 different views and use CogVLM [75] to annotate 1) the front view and 2) the concatenated multi-view image. Then, we adopt the Mistral-7B-Instruct model [35] with in-context examples to generate a fused mesh caption.

Data Statistics. We present the data statistics of our large-scale 3D mesh collection in Tab. 1. After organizing and combing 3D assets from ShapeNet [9], 3D-FUTURE [22], Objaverse [17], and Objaverse-XL [16], we could achieve a total of 2.5 million 3D meshes.

Table 1: **Statistics for the Training Data and Validation Data.** After combining four data sources, our proposed MeshXL models are trained on approximately 2.5 million 3D meshes.

Dataset	Pre-training		Text-to-3D	
	Train	Val	Train	Val
ShapeNet [9]	16,001	1,754	15,384	1,728
3D-Future [22]	1,603	-	-	-
Objaverse [17]	85,282	854	83,501	820
Objaverse-XL [16]	2,407,337	15,200	1,347,802	13,579
Total	2,510,223	17,808	1,446,678	16,127

4 Neural Coordinate Field

Neural **C**oordinate **F**ield (NeurCF) is an explicit representation with implicit neural embeddings. To be specific, for a Euclidean 3D coordinate system, we can partition the vertices coordinates into an N^3 grid. Then, each discretized coordinate $p = (x, y, z)$ can be encoded with the coordinate embedding layer $\mathcal{E}$, where $\mathcal{F}(p) = (\mathcal{E}(x), \mathcal{E}(y), \mathcal{E}(z))$. Therefore, a k-sided polynomial face $f^{(i)}$ can be encoded with $\mathcal{E}_{\text{face}}(f^{(i)}) = (\mathcal{F}(p_1^{(i)}), \cdots, \mathcal{F}(p_k^{(i)}))$. For simplicity, we share the learnable coordinate embeddings $\mathcal{E}$ among axes.

Ordering. Due to the graph representation, the order of the mesh vertices and the order of the edges among them are permutation-invariant. A pre-defined ordering strategy is essential to facilitate the sequence modelling in MeshXL. We employ the same ordering strategy as PolyGen [53] and MeshGPT [65]. The mesh coordinates are first normalized into a unit cube based on the mesh's longest axis, and discretized into unsigned integers. Within each face, the vertices are cyclically permuted based their coordinates (z-y-x order, from lower to higher), which helps to preserve the direction of normal vectors. Then, we order these faces based on the permuted coordinates (lower to high). To this end, we can represent each 3D mesh with a **one-and-only** coordinate sequence, aiding large-scale generative pre-training on a large collection of 3D mesh data. With the NeurCF

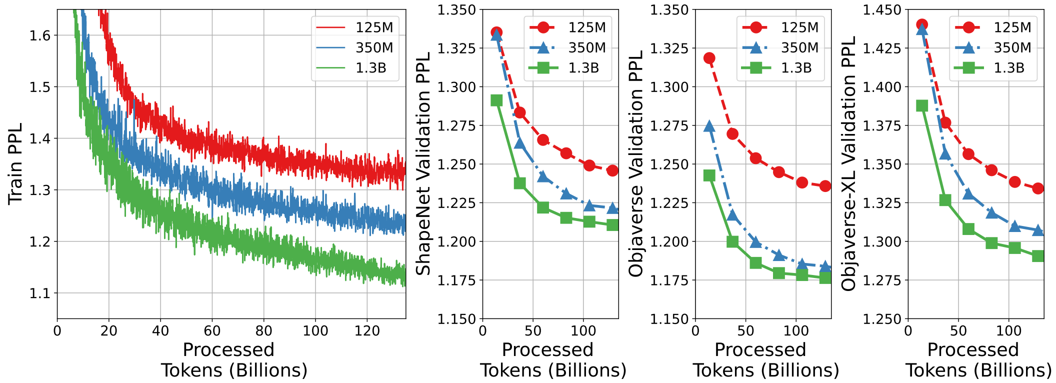


Figure 3: **Training and Validation Perplexity (PPL) for MeshXL Models.** We train all the models from scratch on 150 billion tokens. We observe that the performance grows with model sizes.

representation, an n-faced 3D k-sided polynomial mesh can be represented as the coordinate sequence $\mathcal{M} \in \mathbb{Z}^{n \times k \times 3}$, and further be encoded into $\mathcal{E}_{\text{mesh}} = (\mathcal{E}_{\text{face}}(f^{(1)}), \cdots, \mathcal{E}_{\text{face}}(f^{(n)}))$.

A Sequential Mesh Representation. One direct way to represent the 3D meshes is to directly reshape $\mathcal{M}$ into a vector with $(n \cdot k \cdot 3)$ tokens. As a special case, an n-faced triangular mesh can be represented by a vector with $9n$ tokens. Meanwhile, our representation can also be expanded to hybrid polynomial mesh representations with the proper introduction of separate tokens. For example, we can generate triangles within "<tri> $\cdots$ </tri>" and quadrilaterals within "<quad> $\cdots$ </quad>" also in one sequence. To identify the start and end of a mesh sequence, we add a <bos> ("begin-of-sequence") token before the mesh sequence and an <eos> ("end-of-sequence") token after.

Comparisons. Since we represent each coordinate with learnable embeddings, NeurCF is an end-to-end trainable representation for unstructured 3D meshes. Comparing to the decoupled vertex and polygon representation in PolyGen [53], NeurCF only requires one coordinate sequence for each 3D mesh. Additionally, NeurCF is storage and computation efficient comparing to voxel fields ($O(N^3)$), since it can easily scale up the resolution with a complexity of $O(N)$.

5 Method

We first present the architecture and training objective for MeshXL models. Then, we show that MeshXL models can take an additional modality as the condition for controllable 3D assets generation. After this, we investigate the effects of scaling.

Architecture. In Sec. 4, we present a simple-yet-effective way to represent a 3D mesh into a sequence. Therefore, the learning of 3D mesh generation can be formulated as an auto-regressive problem, which can be seaminglessly addressed by modern **L**arge **L**anguage **M**odel (LLM) approaches. In our paper, we adopt the decoder-only transformers using the OPT [93] codebase as our base models. To adapt the pre-trained OPT models to our *next-coordinate prediction* setting, we fine-tune the whole model with newly-initialized coordinate and position embeddings.

Generative Pre-Training. We train MeshXL models using the standard next-token prediction loss. Given the trainable weights θ and an $|s|$-length sequence s, the generation loss is calculated as:

$$\mathcal{L}_{\text{MeshXL}}\left(\theta\right) = -\sum_{i=1}^{|s|} \log P\left(s_{[i]} | s_{[1,\cdots,i-1]}; \theta\right). \tag{1}$$

For each mesh sequence, we add a <bos> token before the mesh tokens, and an <eos> token after to identify the ending of a 3D mesh. During inference, we adopt the top-k and top-p sampling strategy to produce diverse outputs.

$\mathcal{X}$-to-Mesh Generation. Here we mainly consider generating 3D meshes from images and texts. We first turn the extra conditions into tokens with pre-trained encoders [18, 19]. To align the additional text/image feature with the mesh coordinate field, we adopt the Q-Former architecture [40] to compress the encoded feature into a fixed-length of 32 learnable tokens as the prefix of the MeshXL model. The overall training objective for the conditional mesh generation is shown in Eq. (2):

$$\mathcal{L}_{\mathcal{X}\text{-to-mesh}}\left(\theta\right) = -\sum_{i=1}^{|s|} \log P\left(s_{[i]} | s_{[1,\cdots,i-1]}; \mathcal{X}\right). \tag{2}$$

During inference, the model predicts the mesh tokens after the fixed-length prefix.

Scaling Up. We present MeshXL in various sizes, including 125M, 350M, and 1.3B. The detailed hyperparameters for training different models can be found in Tab. 2. To better analyze the scaling effects, we train all models from scratch on 150 billion tokens. We provide both training curve and validation perplexity for different models in Fig. 3. One can see that as the number of parameters grows, the model achieves a lower validation perplexity, indicating a higher probability to produce the validation data.

Table 2: **Hyperparameters for different MeshXL Base Models.** We present three MeshXL models with 125M, 350M, and 1.3B parameters, respectively.

Hyperparameters	MeshXL(125M)	MeshXL(350M)	MeshXL(1.3B)
# Layers	12	24	24
# Heads	12	16	32
d_{model}	768	1,024	2,048
d_{FFN}	3,072	4,096	8,192
Optimizer	AdamW(β_1=0.9, β_2=0.999)		
Learning rate	1.0×10^{-4}	1.0×10^{-4}	1.0×10^{-4}
LR scheduler	Cosine	Cosine	Cosine
Weight decay	0.1	0.1	0.1
Gradient Clip	1.0	1.0	1.0
Number of GPUs	8	16	32
# GPU hrs (A100)	1,944	6,000	23,232

6 Experiments

We first briefly introduce the data, metrics, and implementation details in Sec. 6.1. Then, we provide evaluations and comparisons on the generated meshes ($cf.$ Sec. 6.2) and ablations ($cf.$ Sec. 6.3). We also provide visualization results in Sec. 6.4.

6.1 Data, Metrics, and Implementation Details

Data. We pre-train the base model with 2.5 million 3D meshes collected from the combination of ShapeNet [9], 3D-FUTURE [22], Objaverse [17], and Objaverse-XL [16]. We use planar decimation on meshes with more than 800 faces following MeshGPT [65] and RobustLowPoly [11]. For generative mesh pre-training, we randomly rotate these meshes with degrees from (0°, 90°, 180°, 270°), and adopt random scaling along each axis within range $[0.9, 1.1]$ for data augmentation.

Metrics. We follow the standard evaluation protocols in MeshGPT [65] and PolyDiff [2] to measure the quality of the generated meshes with the following metrics. We use Coverage (COV) to quantify the diversity of the generated meshes, which is sensitive to mode dropping but cannot be used to assess the generation quality. Minimum Matching Distance (MMD) calculates the average distance between the reference set and their closest neighbors in the generated set, but is not sensitive to low-quality results. The 1-Nearest Neighbor Accuracy (1-NNA) quantifies the quality and diversity between the generation set and the reference set, whose optimal value is 50%. We also adopt the Jensen-Shannon Divergence (JSD) score. Among all the above metrics, we use Chamfer Distance to measure the similarity between two samples. We also render the generated meshes and adopt the Frechet Inception Distance (FID) and Kernel Inception Distance (KID) on the rendered images for feature-level evaluation. We multiply the MMD, JSD, and KID scores by 10^3.

Implementation. We conduct all the experiments on a cluster consisting 128 A100 GPUs. We train our models under bfloat16 with the ZeRO-2 strategy [60] using the AdamW [48] optimizer with a

Table 3: **Quantitative Comparisons with Prior Arts on ShapeNet [9].** We scale MMD, JSD, KID by 10^3. MeshXL can produce diverse and high-quality 3D meshes.

Category	Methods	COV↑	MMD↓	1-NNA	JSD↓	FID↓	KID↓
Chair	PolyGen [53]	7.79	16.00	99.16	228.80	63.49	43.73
	GET3D [23]	11.70	15.92	99.75	155.25	67.84	42.10
	MeshGPT [65]	42.00	4.75	69.50	55.16	39.52	8.97
	MeshXL (125M)	50.80	**3.11**	56.55	9.69	28.15	1.48
	MeshXL (350M)	50.80	3.17	**55.80**	9.66	28.29	**1.39**
	MeshXL (1.3B)	**51.60**	3.23	**55.80**	**9.48**	**9.12**	1.84
Table	PolyGen [53]	44.00	3.36	67.20	25.06	54.08	14.96
	GET3D [23]	16.80	10.39	91.90	226.97	67.65	34.62
	MeshGPT [65]	34.30	6.51	75.05	92.88	53.75	7.75
	MeshXL (125M)	51.21	2.96	57.96	**12.82**	42.55	**0.92**
	MeshXL (350M)	49.70	3.07	**56.10**	13.64	43.43	1.27
	MeshXL (1.3B)	**52.12**	**2.92**	56.80	14.93	**22.29**	2.03
Bench	PolyGen [53]	31.15	4.01	83.23	55.25	70.53	12.1
	MeshGPT [65]	34.92	2.22	68.65	57.32	52.47	6.49
	MeshXL (125M)	54.37	1.65	**43.75**	16.43	**35.31**	**0.82**
	MeshXL (350M)	53.37	1.65	42.96	**15.41**	36.35	0.96
	MeshXL (1.3B)	**56.55**	**1.62**	39.78	15.51	35.50	1.60
Lamp	PolyGen [53]	35.04	7.87	75.49	96.57	65.15	12.78
	MeshGPT [65]	41.59	4.92	61.59	61.82	47.19	5.19
	MeshXL (125M)	**55.86**	5.06	48.24	43.41	34.61	**0.84**
	MeshXL (350M)	53.52	**4.18**	**49.41**	**34.87**	**25.94**	1.92
	MeshXL (1.3B)	51.95	4.89	47.27	41.89	31.66	0.99

learning rate decaying from 10^{-4} to 10^{-6} and a weight decay of 0.1. The detailed hyperparameters for different models can be found in Tab. 2. To train our base models, we load the weights from the pre-trained OPT models [93] and initialize the word embeddings and positional embeddings from scratch. Without further specification, we generate 3D meshes with the top-k and top-p sampling strategy with $k = 50$ and $p = 0.95$.

6.2 Evaluations and Comparisons

We provide quantitative as well as qualitative comparisons on both unconditional and conditional 3D mesh generation on public benchmarks.

Unconditional Generation. We evaluate MeshXL as well as other baseline methods using the ShapeNet [9] data in Tab. 3. We split the data by 9:1 for training and validation by each category. For evaluation, we fine-tune our pre-trained base model and sample 1,000 meshes for each category. Among the listed methods, we reproduce the MeshGPT [65] with a GPT2-medium model (355M) [59]. With a similar number of parameters, Mesh-XL (350M) out-performs MeshGPT by a large margin, showing a higher COV score, a lower MMD score, and a closer 1-NNA score to 50%. This indicates that MeshXL can produce diverse and high-quality 3D meshes.

Table 4: **User Study.** Compared to baseline methods, the meshes generated by MeshXL are better aligned with human preference in terms of both geometry and designs.

Methods	Quality↑	Artistic↑	Triangulation↑
PolyGen [53]	2.53	2.72	3.15
GET3D [23]	3.15	2.46	3.15
MeshXL	**3.96**	**3.45**	**3.72**
Reals	4.08	3.33	3.75

User Study. To evaluate how well the generated 3D meshes align with human preference, we perform user studies on the chair category in Tab. 4 with several baseline methods [53, 23]. We recruit and instruct the participants to score each mesh from 0 to 5 (higher is better) based on its 1) **quality**: the smoothness of object surfaces and completeness of the mesh, 2) **artistic**: how much do you believe this object is designed and created by artists, and 3) **triangulation**: how well do the connectivity among vertices aligns with the models created by professional designing software [14]. As a baseline evaluation, we also ask the participants to score the ground truth 3D geometries sampled from the

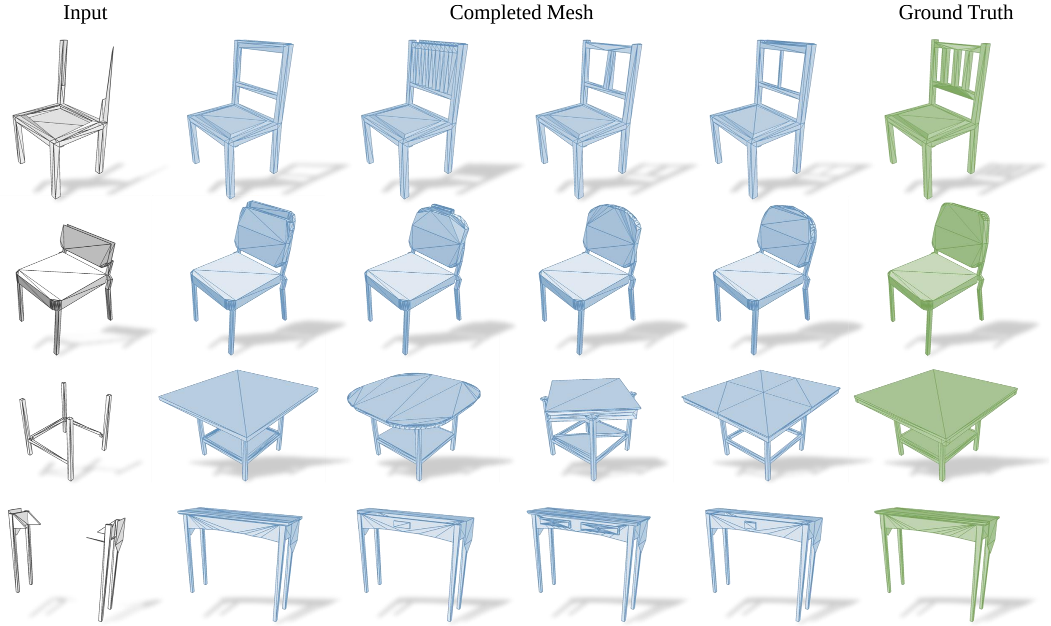


Figure 4: **Evaluation of Partial Mesh Completion.** Given some partial observation of the 3D mesh (white), MeshXL is able to produce diverse object completion results (blue).

ShapeNet data. We have collected a total of 434 valid responses. The results show that the 3D meshes created by MeshXL are consistently preferred by human in all dimensions.

6.3 Ablation Studies

Necessity of Mesh VQVAE. Comparing to MeshGPT [65], MeshXL is an end-to-end trainable model that produces 3D meshes with *next-coordinate prediction*. We show in Tab. 3 that, MeshXL out-performs MeshGPT with similar numbers of parameters. We also show that MeshXL can produce high quality 3D meshes with both sharp edges and smooth surfaces in Fig. 7. Furthermore, MeshXL can save the effort training a vector quantized mesh tokenizer [65, 72], which further facilitates generative pre-training on large scale datasets.

Effectiveness of Model Sizes. To analyze whether pre-training a larger model benefits 3D mesh generation, we evaluate MeshXL base models with different sizes on the Objaverse [17] dataset in Tab. 5. We observe that as the model size grows, the generated samples exhibits a larger COV, smaller JSD score, and a closer 1-NNA to 50%, which indicates an improved diversity and quality.

Table 5: **Effectiveness of Model Sizes on Objaverse.** As the model size grows, MeshXL achieves a closer 1-NNA to 50%, a larger COV and a smaller JSD, indicating better diversity and quality.

Method	COV↑	MMD↓	1-NNA	JSD↓	FID↓	KID ↓
MeshXL (125M)	39.76	5.21	67.34	26.03	17.32	4.48
MeshXL (350M)	40.79	5.20	65.68	23.71	15.14	3.33
MeshXL (1.3B)	**42.86**	**4.16**	**61.56**	**20.99**	**12.49**	**2.94**

Shape Completion. To analysis whether our method is capable of producing diverse outputs, we adopt MeshXL (1.3B) model to predict the whole object given some partial observations. In practice, we use 50% of the object mesh as input, and ask the model to predict the rest 50% in Fig. 4. One can see that Mesh-XL is able to produce diverse and reasonable outputs given the partial observation of the 3D mesh.

$\mathcal{X}$-to-Mesh Generation. To adopt the MeshXL base models to the $\mathcal{X}$-to-mesh generation setting, we adopt the Q-Former [40] to encode the additional conditions as prefixes. We showcases several conditional generation results in Fig. 5. We show that MeshXL can generate high-quality 3D meshes given the corresponding image or text as the additional inputs.

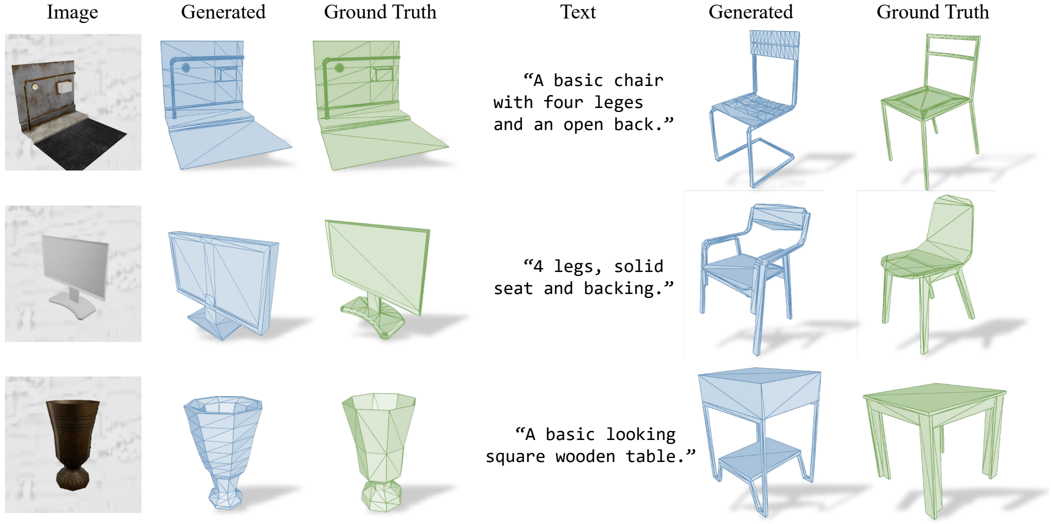


Image
Generated
Ground Truth
Text
Generated
Ground Truth
“A basic chair with four leges and an open back.”
“4 legs, solid seat and backing.”
“A basic looking square wooden table.”

Figure 5: **Evaluation of $\mathcal{X}$-to-mesh generation.** We show that MeshXL can generate high-quality 3D meshes given the corresponding image or text as the additional inputs.

Texturing. We adopt Paint3D [91], a coarse-to-fine texture generation pipeline, to generate textures for the 3D meshes produced by MeshXL in Fig. 6. We show that 3D meshes produced by MeshXL can easily fit in existing texturing methods to produce high-quality 3D assets.

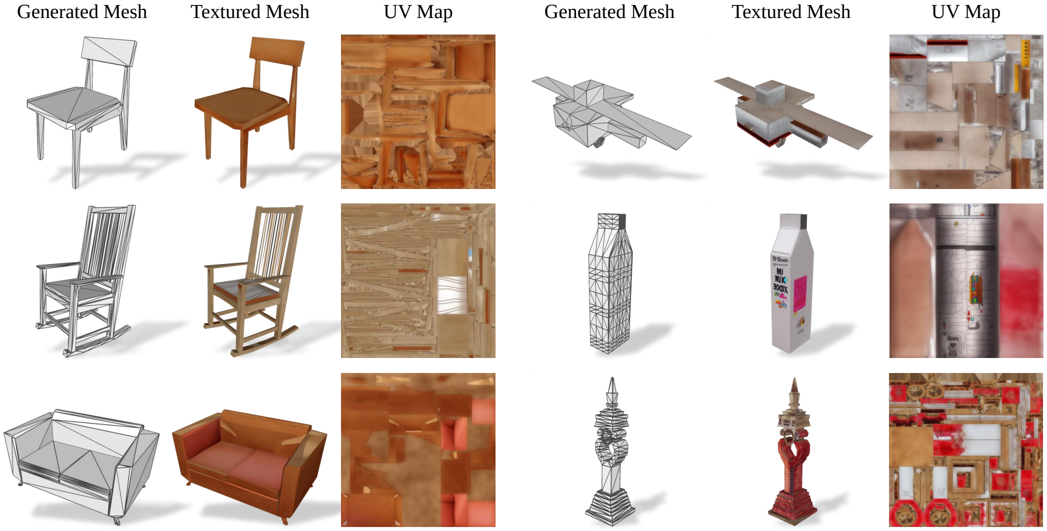


Figure 6: **Texture Generation for the Generated 3D Meshes.** We adopt Paint3D [91] to generate textures for 3D meshes produced by MeshXL.

6.4 Visualizations

We provide qualitative comparisons on the generated meshes in Fig. 7. MeshXL is able to produce high-quality 3D meshes with both sharp edges and smooth surfaces. We also visualize the normal vectors to compare the smoothness of object surfaces. The results show that 3D meshes generated by GET3D [23] have rough surfaces with tens of thousands of triangles, while MeshXL depicts the 3D shape with much smoother surfaces and less triangles.

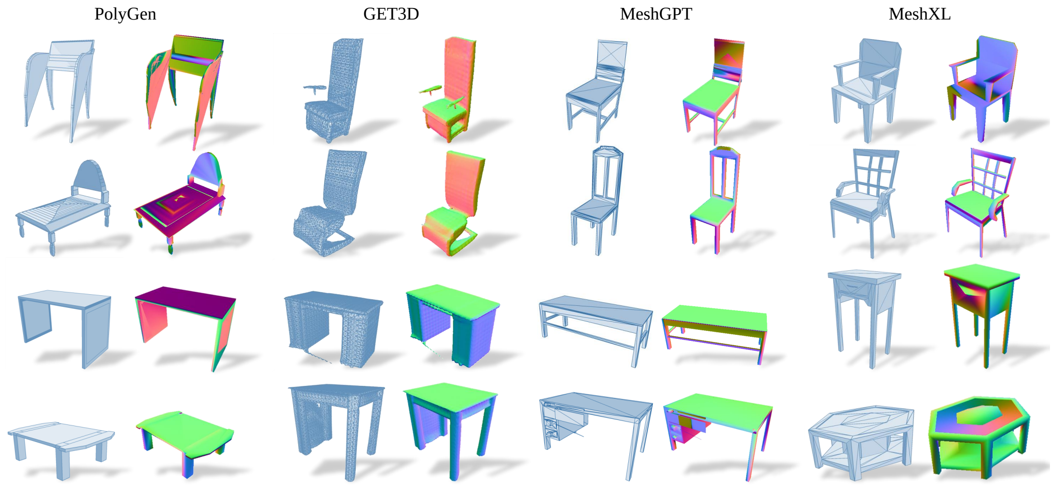


Figure 7: **Qualitative comparisons.** We visualize the generated meshes and normal vectors. MeshXL is able to produce high-quality 3D meshes with both sharp edges and smooth surfaces.

7 Discussions

Difference with PolyGen [53]. PolyGen treats 3D mesh data as a vertex sequence and a face sequence. PolyGen first generates the ordered vertices with the vertex transformer, then predicts the connectivity among vertices with the face transformer. Comparing to PolyGen, our proposed MeshXL adopts a more straightforward approach that *models the 3D mesh as a one-and-only coordinate sequence*, which further supports the direct and end-to-end pre-training on a large collection of 3D data.

Difference with MeshGPT [65]. MeshGPT consists of a mesh VQVAE [72] and a decoder-only transformer [59]. MeshGPT first learns a mesh VQVAE to quantize the 3D meshes into discrete tokens. After that, MeshGPT trains a decoder-only transformer to generate the discrete tokens for 3D mesh reconstruction. In comparison, our proposed MeshXL is an end-to-end method that learns the neural representation of coordinates and outputs 3D meshes directly.

Extensibility. Our method, MeshXL, is built upon the concept of auto-regressive methods. Therefore, our method is not restricted to the decoder-only transformers [59, 93, 69, 70], and can also be extended to other causal language models (*i.e.* Mamba [26], RWKV [55], and xLSTM [6]).

8 Limitations, Future Work, and Conclusions

Limitations and Future Work. The main drawback of MeshXLs is the inference time. During sampling, MeshXL will generate 7,200 tokens for an 800-faced 3D mesh, which takes a relatively long time because of the auto-regressive process. As for future works, recent endeavors on the RNN-related methods [6, 55, 26] and multiple tokens prediction for LLMs [24] might open up great opportunities in saving the inference cost.

Conclusion. We validate that NeurCF, an explicit coordinate representation with implicit neural embeddings, is a simple-and-effective representation of 3D meshes. By modelling the 3D mesh generation as an auto-regressive problem, we seek help from modern LLM approaches and present a family of generative pre-trained models, MeshXL, for high-fidelity 3D mesh generation. We show that MeshXL performs better given larger-scale training data and increased parameters. Extensive results show our proposed MeshXL can not only generate high-quality 3D meshes, but also exhibits great potential serving as base models for conditional 3D assets generation.

Acknowledgement

This work is supported by National Natural Science Foundation of China (No. 62071127), National Key Research and Development Program of China (No. 2022ZD0160101), Shanghai Natural Science Foundation (No. 23ZR1402900), Shanghai Municipal Science and Technology Major Project (No.2021SHZDZX0103).

References

[1] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. Learning representations and generative models for 3d point clouds. In *International conference on machine learning*, pages 40–49. PMLR, 2018.

[2] Antonio Alliegro, Yawar Siddiqui, Tatiana Tommasi, and Matthias Nießner. Polydiff: Generating 3d polygonal meshes with diffusion models. *arXiv preprint arXiv:2312.11417*, 2023.

[3] Iro Armeni, Ozan Sener, Amir R Zamir, Helen Jiang, Ioannis Brilakis, Martin Fischer, and Silvio Savarese. 3d semantic parsing of large-scale indoor spaces. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1534–1543, 2016.

[4] Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. Structured denoising diffusion models in discrete state-spaces. *Advances in Neural Information Processing Systems*, 34:17981–17993, 2021.

[5] Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5855–5864, 2021.

[6] Maximilian Beck, Korbinian Pöppel, Markus Spanring, Andreas Auer, Oleksandra Prudnikova, Michael Kopp, Günter Klambauer, Johannes Brandstetter, and Sepp Hochreiter. xlstm: Extended long short-term memory. *arXiv preprint arXiv:2405.04517*, 2024.

[7] Jens Behley, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Cyrill Stachniss, and Jurgen Gall. Semantickitti: A dataset for semantic scene understanding of lidar sequences. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9297–9307, 2019.

[8] Ziang Cao, Fangzhou Hong, Tong Wu, Liang Pan, and Ziwei Liu. Difftf++: 3d-aware diffusion transformer for large-vocabulary 3d generation. *arXiv preprint arXiv:2405.08055*, 2024.

[9] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

[10] Sijin Chen, Xin Chen, Chi Zhang, Mingsheng Li, Gang Yu, Hao Fei, Hongyuan Zhu, Jiayuan Fan, and Tao Chen. Ll3da: Visual interactive instruction tuning for omni-3d understanding, reasoning, and planning. *arXiv preprint arXiv:2311.18651*, 2023.

[11] Zhen Chen, Zherong Pan, Kui Wu, Etienne Vouga, and Xifeng Gao. Robust low-poly meshing for general 3d models. *ACM Transactions on Graphics (TOG)*, 42(4):1–20, 2023.

[12] Zhiqin Chen, Andrea Tagliasacchi, and Hao Zhang. Bsp-net: Generating compact meshes via binary space partitioning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 45–54, 2020.

[13] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3075–3084, 2019.

[14] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018.

[15] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5828–5839, 2017.

[16] Matt Deitke, Ruoshi Liu, Matthew Wallingford, Huong Ngo, Oscar Michel, Aditya Kusupati, Alan Fan, Christian Laforte, Vikram Voleti, Samir Yitzhak Gadre, et al. Objaverse-xl: A universe of 10m+ 3d objects. *Advances in Neural Information Processing Systems*, 36, 2024.

[17] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13142–13153, 2023.

[18] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[19] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[20] Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, et al. Palm-e: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*, 2023.

[21] Huan Fu, Bowen Cai, Lin Gao, Ling-Xiao Zhang, Jiaming Wang, Cao Li, Qixun Zeng, Chengyue Sun, Rongfei Jia, Binqiang Zhao, et al. 3d-front: 3d furnished rooms with layouts and semantics. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10933–10942, 2021.

[22] Huan Fu, Rongfei Jia, Lin Gao, Mingming Gong, Binqiang Zhao, Steve Maybank, and Dacheng Tao. 3d-future: 3d furniture shape with texture. *International Journal of Computer Vision*, pages 1–25, 2021.

[23] Jun Gao, Tianchang Shen, Zian Wang, Wenzheng Chen, Kangxue Yin, Daiqing Li, Or Litany, Zan Gojcic, and Sanja Fidler. Get3d: A generative model of high quality 3d textured shapes learned from images. *Advances In Neural Information Processing Systems*, 35:31841–31854, 2022.

[24] Fabian Gloeckle, Badr Youbi Idrissi, Baptiste Rozière, David Lopez-Paz, and Gabriel Synnaeve. Better & faster large language models via multi-token prediction. *arXiv preprint arXiv:2404.19737*, 2024.

[25] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[26] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

[27] Ziyu Guo, Renrui Zhang, Xiangyang Zhu, Yiwen Tang, Xianzheng Ma, Jiaming Han, Kexin Chen, Peng Gao, Xianzhi Li, Hongsheng Li, et al. Point-bind & point-llm: Aligning point cloud with multi-modality for 3d understanding, generation, and instruction following. *arXiv preprint arXiv:2309.00615*, 2023.

[28] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[29] Fangzhou Hong, Jiaxiang Tang, Ziang Cao, Min Shi, Tong Wu, Zhaoxi Chen, Tengfei Wang, Liang Pan, Dahua Lin, and Ziwei Liu. 3dtopia: Large text-to-3d generation model with hybrid diffusion priors. *arXiv preprint arXiv:2403.02234*, 2024.

[30] Yicong Hong, Kai Zhang, Jiuxiang Gu, Sai Bi, Yang Zhou, Difan Liu, Feng Liu, Kalyan Sunkavalli, Trung Bui, and Hao Tan. Lrm: Large reconstruction model for single image to 3d. *arXiv preprint arXiv:2311.04400*, 2023.

[31] Yining Hong, Haoyu Zhen, Peihao Chen, Shuhong Zheng, Yilun Du, Zhenfang Chen, and Chuang Gan. 3d-llm: Injecting the 3d world into large language models. *Advances in Neural Information Processing Systems*, 36:20482–20494, 2023.

[32] Jiangyong Huang, Silong Yong, Xiaojian Ma, Xiongkun Linghu, Puhao Li, Yan Wang, Qing Li, Song-Chun Zhu, Baoxiong Jia, and Siyuan Huang. An embodied generalist agent in 3d world. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2024.

[33] Moritz Ibing, Isaak Lim, and Leif Kobbelt. 3d shape generation with grid-based implicit functions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13559–13568, 2021.

[34] Baoxiong Jia, Yixin Chen, Huangyue Yu, Yan Wang, Xuesong Niu, Tengyu Liu, Qing Li, and Siyuan Huang. Sceneverse: Scaling 3d vision-language learning for grounded scene understanding. *arXiv preprint arXiv:2401.09340*, 2024.

[35] Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

[36] Biao Jiang, Xin Chen, Wen Liu, Jingyi Yu, Gang Yu, and Tao Chen. Motiongpt: Human motion as a foreign language. *Advances in Neural Information Processing Systems*, 36, 2024.

[37] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, volume 7, 2006.

[38] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4):1–14, 2023.

[39] Chenghao Li, Chaoning Zhang, Atish Waghwase, Lik-Hang Lee, Francois Rameau, Yang Yang, Sung-Ho Bae, and Choong Seon Hong. Generative ai meets 3d: A survey on text-to-3d in aigc era. *arXiv preprint arXiv:2305.06131*, 2023.

[40] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International conference on machine learning*, pages 19730–19742. PMLR, 2023.

[41] Mingsheng Li, Xin Chen, Chi Zhang, Sijin Chen, Hongyuan Zhu, Fukun Yin, Gang Yu, and Tao Chen. M3dbench: Let's instruct large models with multi-modal 3d prompts. *arXiv preprint arXiv:2312.10763*, 2023.

[42] Xiaoyu Li, Qi Zhang, Di Kang, Weihao Cheng, Yiming Gao, Jingbo Zhang, Zhihao Liang, Jing Liao, Yan-Pei Cao, and Ying Shan. Advances in 3d generation: A survey. *arXiv preprint arXiv:2401.17807*, 2024.

[43] Minghua Liu, Ruoxi Shi, Linghao Chen, Zhuoyang Zhang, Chao Xu, Xinyue Wei, Hansheng Chen, Chong Zeng, Jiayuan Gu, and Hao Su. One-2-3-45++: Fast single image to 3d objects with consistent multi-view generation and 3d diffusion. *arXiv preprint arXiv:2311.07885*, 2023.

[44] Minghua Liu, Ruoxi Shi, Kaiming Kuang, Yinhao Zhu, Xuanlin Li, Shizhong Han, Hong Cai, Fatih Porikli, and Hao Su. Openshape: Scaling up 3d shape representation towards open-world understanding. *Advances in Neural Information Processing Systems*, 36, 2024.

[45] Minghua Liu, Chao Xu, Haian Jin, Linghao Chen, Mukund Varma T, Zexiang Xu, and Hao Su. One-2-3-45: Any single image to 3d mesh in 45 seconds without per-shape optimization. *Advances in Neural Information Processing Systems*, 36, 2024.

[46] Ruoshi Liu, Rundi Wu, Basile Van Hoorick, Pavel Tokmakov, Sergey Zakharov, and Carl Vondrick. Zero-1-to-3: Zero-shot one image to 3d object. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9298–9309, 2023.

[47] Zhen Liu, Yao Feng, Michael J Black, Derek Nowrouzezahrai, Liam Paull, and Weiyang Liu. Meshdiffusion: Score-based generative 3d mesh modeling. *arXiv preprint arXiv:2303.08133*, 2023.

[48] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[49] Shitong Luo and Wei Hu. Diffusion probabilistic models for 3d point cloud generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2837–2845, 2021.

[50] Zhaoyang Lyu, Jinyi Wang, Yuwei An, Ya Zhang, Dahua Lin, and Bo Dai. Controllable mesh generation through sparse latent point diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 271–280, 2023.

[51] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020.

[52] Paritosh Mittal, Yen-Chi Cheng, Maneesh Singh, and Shubham Tulsiani. Autosdf: Shape priors for 3d completion, reconstruction and generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 306–315, 2022.

[53] Charlie Nash, Yaroslav Ganin, SM Ali Eslami, and Peter Battaglia. Polygen: An autoregressive generative model of 3d meshes. In *International conference on machine learning*, pages 7220–7229. PMLR, 2020.

[54] Alex Nichol, Heewoo Jun, Prafulla Dhariwal, Pamela Mishkin, and Mark Chen. Point-e: A system for generating 3d point clouds from complex prompts. *arXiv preprint arXiv:2212.08751*, 2022.

[55] Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Huanqi Cao, Xin Cheng, Michael Chung, Matteo Grella, Kranthi Kiran GV, et al. Rwkv: Reinventing rnns for the transformer era. *arXiv preprint arXiv:2305.13048*, 2023.

[56] Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv preprint arXiv:2209.14988*, 2022.

[57] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[58] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017.

[59] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[60] Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. Zero: Memory optimizations toward training trillion parameter models. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–16. IEEE, 2020.

[61] Xuanchi Ren, Jiahui Huang, Xiaohui Zeng, Ken Museth, Sanja Fidler, and Francis Williams. Xcube ($\mathcal{X}^3$): Large-scale 3d generative modeling using sparse voxel hierarchies. *arXiv preprint arXiv:2312.03806*, 2023.

[62] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

[63] Tianchang Shen, Jun Gao, Kangxue Yin, Ming-Yu Liu, and Sanja Fidler. Deep marching tetrahedra: a hybrid representation for high-resolution 3d shape synthesis. *Advances in Neural Information Processing Systems*, 34:6087–6101, 2021.

[64] J Ryan Shue, Eric Ryan Chan, Ryan Po, Zachary Ankner, Jiajun Wu, and Gordon Wetzstein. 3d neural field generation using triplane diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20875–20886, 2023.

[65] Yawar Siddiqui, Antonio Alliegro, Alexey Artemov, Tatiana Tommasi, Daniele Sirigatti, Vladislav Rosov, Angela Dai, and Matthias Nießner. Meshgpt: Generating triangle meshes with decoder-only transformers. *arXiv preprint arXiv:2311.15475*, 2023.

[66] Shuran Song, Samuel P Lichtenberg, and Jianxiong Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 567–576, 2015.

[67] Jiaxiang Tang, Zhaoxi Chen, Xiaokang Chen, Tengfei Wang, Gang Zeng, and Ziwei Liu. Lgm: Large multi-view gaussian model for high-resolution 3d content creation. *arXiv preprint arXiv:2402.05054*, 2024.

[68] Jiaxiang Tang, Jiawei Ren, Hang Zhou, Ziwei Liu, and Gang Zeng. Dreamgaussian: Generative gaussian splatting for efficient 3d content creation. *arXiv preprint arXiv:2309.16653*, 2023.

[69] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[70] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[71] Mikaela Angelina Uy, Quang-Hieu Pham, Binh-Son Hua, Thanh Nguyen, and Sai-Kit Yeung. Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1588–1597, 2019.

[72] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

[73] Peng Wang, Hao Tan, Sai Bi, Yinghao Xu, Fujun Luan, Kalyan Sunkavalli, Wenping Wang, Zexiang Xu, and Kai Zhang. Pf-lrm: Pose-free large reconstruction model for joint pose and shape prediction. *arXiv preprint arXiv:2311.12024*, 2023.

[74] Tai Wang, Xiaohan Mao, Chenming Zhu, Runsen Xu, Ruiyuan Lyu, Peisen Li, Xiao Chen, Wenwei Zhang, Kai Chen, Tianfan Xue, et al. Embodiedscan: A holistic multi-modal 3d perception suite towards embodied ai. *arXiv preprint arXiv:2312.16170*, 2023.

[75] Weihan Wang, Qingsong Lv, Wenmeng Yu, Wenyi Hong, Ji Qi, Yan Wang, Junhui Ji, Zhuoyi Yang, Lei Zhao, Xixuan Song, et al. Cogvlm: Visual expert for pretrained language models. *arXiv preprint arXiv:2311.03079*, 2023.

[76] Zhengyi Wang, Cheng Lu, Yikai Wang, Fan Bao, Chongxuan Li, Hang Su, and Jun Zhu. Prolificdreamer: High-fidelity and diverse text-to-3d generation with variational score distillation. *Advances in Neural Information Processing Systems*, 36, 2024.

[77] Zhenwei Wang, Tengfei Wang, Gerhard Hancke, Ziwei Liu, and Rynson W.H. Lau. Themestation: Generating theme-aware 3d assets from few exemplars. 2024.

[78] Xinyue Wei, Kai Zhang, Sai Bi, Hao Tan, Fujun Luan, Valentin Deschaintre, Kalyan Sunkavalli, Hao Su, and Zexiang Xu. Meshlrm: Large reconstruction model for high-quality mesh. *arXiv preprint arXiv:2404.12385*, 2024.

[79] Hongtao Wu, Ya Jing, Chilam Cheang, Guangzeng Chen, Jiafeng Xu, Xinghang Li, Minghuan Liu, Hang Li, and Tao Kong. Unleashing large-scale video generative pre-training for visual robot manipulation. *arXiv preprint arXiv:2312.13139*, 2023.

[80] Jiajun Wu, Chengkai Zhang, Tianfan Xue, Bill Freeman, and Josh Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. *Advances in neural information processing systems*, 29, 2016.

[81] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015.

[82] Jiale Xu, Weihao Cheng, Yiming Gao, Xintao Wang, Shenghua Gao, and Ying Shan. Instantmesh: Efficient 3d mesh generation from a single image with sparse-view large reconstruction models. *arXiv preprint arXiv:2404.07191*, 2024.

[83] Runsen Xu, Xiaolong Wang, Tai Wang, Yilun Chen, Jiangmiao Pang, and Dahua Lin. Pointllm: Empowering large language models to understand point clouds. *arXiv preprint arXiv:2308.16911*, 2023.

[84] Yinghao Xu, Zifan Shi, Wang Yifan, Hansheng Chen, Ceyuan Yang, Sida Peng, Yujun Shen, and Gordon Wetzstein. Grm: Large gaussian reconstruction model for efficient 3d reconstruction and generation. *arXiv preprint arXiv:2403.14621*, 2024.

[85] Le Xue, Mingfei Gao, Chen Xing, Roberto Martín-Martín, Jiajun Wu, Caiming Xiong, Ran Xu, Juan Carlos Niebles, and Silvio Savarese. Ulip: Learning a unified representation of language, images, and point clouds for 3d understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1179–1189, 2023.

[86] Le Xue, Ning Yu, Shu Zhang, Junnan Li, Roberto Martín-Martín, Jiajun Wu, Caiming Xiong, Ran Xu, Juan Carlos Niebles, and Silvio Savarese. Ulip-2: Towards scalable multimodal pre-training for 3d understanding. *arXiv preprint arXiv:2305.08275*, 2023.

[87] Chandan Yeshwanth, Yueh-Cheng Liu, Matthias Nießner, and Angela Dai. Scannet++: A high-fidelity dataset of 3d indoor scenes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12–22, 2023.

[88] Fukun Yin, Xin Chen, Chi Zhang, Biao Jiang, Zibo Zhao, Jiayuan Fan, Gang Yu, Taihao Li, and Tao Chen. Shapegpt: 3d shape generation with a unified multi-modal language model. *arXiv preprint arXiv:2311.17618*, 2023.

[89] Wang Yu, Xuelin Qian, Jingyang Huo, Tiejun Huang, Bo Zhao, and Yanwei Fu. Pushing the limits of 3d shape generation at scale. *arXiv preprint arXiv:2306.11510*, 2023.

[90] Xianggang Yu, Mutian Xu, Yidan Zhang, Haolin Liu, Chongjie Ye, Yushuang Wu, Zizheng Yan, Chenming Zhu, Zhangyang Xiong, Tianyou Liang, et al. Mvimgnet: A large-scale dataset of multi-view images. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9150–9161, 2023.

[91] Xianfang Zeng. Paint3d: Paint anything 3d with lighting-less texture diffusion models. *arXiv preprint arXiv:2312.13913*, 2023.

[92] Renrui Zhang, Ziyu Guo, Wei Zhang, Kunchang Li, Xupeng Miao, Bin Cui, Yu Qiao, Peng Gao, and Hongsheng Li. Pointclip: Point cloud understanding by clip. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8552–8562, 2022.

[93] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

[94] Zibo Zhao, Wen Liu, Xin Chen, Xianfang Zeng, Rui Wang, Pei Cheng, Bin Fu, Tao Chen, Gang Yu, and Shenghua Gao. Michelangelo: Conditional 3d shape generation based on shape-image-text aligned latent representation. *Advances in Neural Information Processing Systems*, 36, 2024.

[95] Haoyu Zhen, Xiaowen Qiu, Peihao Chen, Jincheng Yang, Xin Yan, Yilun Du, Yining Hong, and Chuang Gan. 3d-vla: A 3d vision-language-action generative world model. *arXiv preprint arXiv:2403.09631*, 2024.

[96] Junsheng Zhou, Jinsheng Wang, Baorui Ma, Yu-Shen Liu, Tiejun Huang, and Xinlong Wang. Uni3d: Exploring unified 3d representation at scale. *arXiv preprint arXiv:2310.06773*, 2023.

[97] Linqi Zhou, Yilun Du, and Jiajun Wu. 3d shape generation and completion through point-voxel diffusion. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5826–5835, October 2021.

[98] Haoyi Zhu, Honghui Yang, Xiaoyang Wu, Di Huang, Sha Zhang, Xianglong He, Tong He, Hengshuang Zhao, Chunhua Shen, Yu Qiao, et al. Ponderv2: Pave the way for 3d foundataion model with a universal pre-training paradigm. *arXiv preprint arXiv:2310.08586*, 2023.

[99] Joseph Zhu and Peiye Zhuang. Hifa: High-fidelity text-to-3d with advanced diffusion guidance. *arXiv preprint arXiv:2305.18766*, 2023.

[100] Xiangyang Zhu, Renrui Zhang, Bowei He, Ziyu Guo, Ziyao Zeng, Zipeng Qin, Shanghang Zhang, and Peng Gao. Pointclip v2: Prompting clip and gpt for powerful 3d open-world learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2639–2650, 2023.

A Appendix

A.1 More Visualization Results

Unconditional Results on ShapeNet. We visualize unconditional 3D mesh generation results for chair, table, lamp and bench in Fig. 8. One can see that MeshXL is able to produce diverse and high-quality 3D meshes.

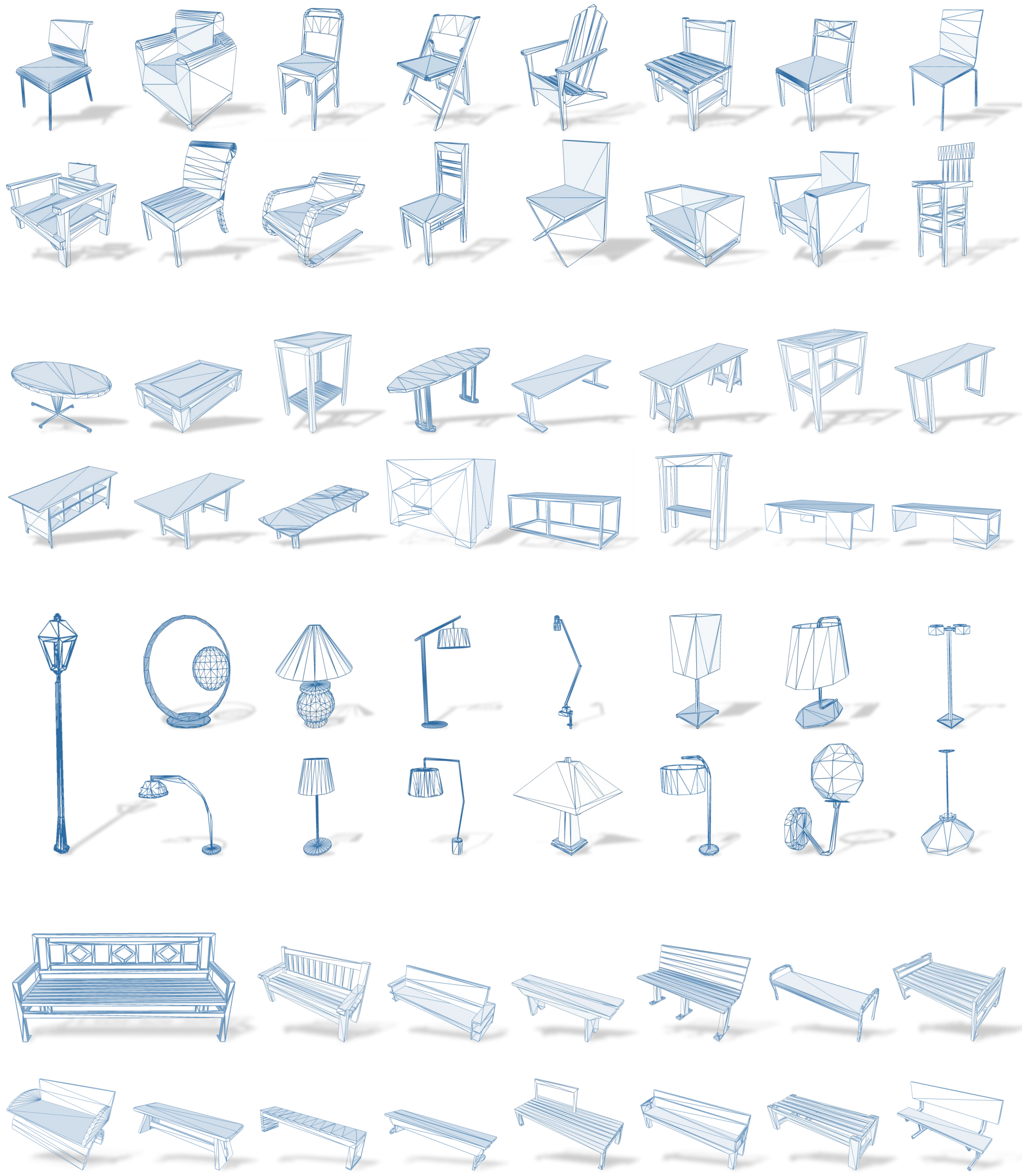


Figure 8: **Gallery results.** Additional generation results for chair, table, lamp, and bench.

Unconditional Generation on Objaverse. We visualize 3D meshes randomly sampled from MeshXL base model in Fig. 9. After training on a large-scale collection of 3D mesh data, MeshXL is able to produce diverse and high-quality 3D meshes.

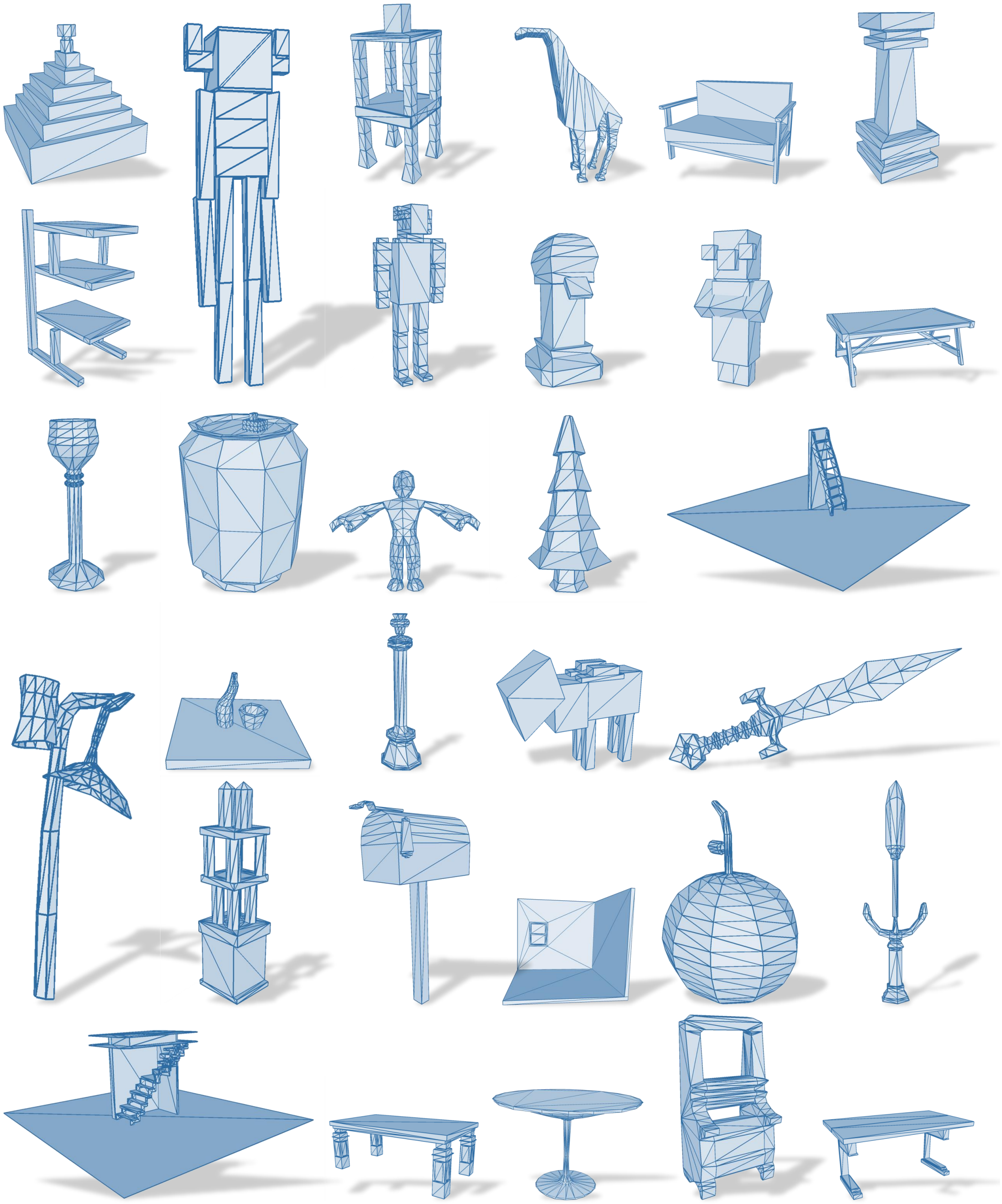


Figure 9: **Gallery results.** MeshXL is able to produce diverse 3D meshes with high quality.

A.2 Mesh Quality Assessment

How well is the triangulation. We evaluate the aspect ratio, face area, and number of faces for a better evaluation in Tab. 6. Though the meshes generated by our MeshXL have a higher average aspect ratio, we manage to achieve a smaller variance with much less 3D faces. This indicates the stability of our generation ability and the efficiency of the direct mesh representation. Since we train our MeshXLs only on triangular meshes, long-thin triangles inevitably exist in our training data. By co-training our MeshXLs on triangular meshes, 3D quads, and even hybrid representations, we could reduce the existence of long thin triangles for better generation quality.

Table 6: **Mesh Quality Assessment.** We evaluate the aspect ratio, face area and number of faces for the generated 3D meshes.

Method	Aspect Ratio		Face Area		Number of Faces	
	mean	std.	mean	std.	mean	std.
GET3D [23]	6.27	116.03	0.000	0.000	27251.80	11535.135
MeshXL (125M)	10.47	16.88	0.031	0.096	327.34	174.53
MeshXL (350M)	10.25	16.09	0.032	0.099	342.24	193.97
MeshXL (1.3B)	10.23	15.91	0.034	0.102	320.36	195.43

A.3 Inference Time Analysis

The inference cost of MeshXL is closely related to the numbers of generated faces and the model sizes. We perform inference cost analysis with a batch size of one using bfloat16 on a single RTX 3090. We carry out a an analysis (Tab. 7) on 1) the average inference time of generating a given number of triangles, and 2) the average inference time of generating 3D meshes.

Table 7: **Inference cost of MeshXL models.** We carry out inference cost analysis on time duration and memory usage under bfloat16 with a single RTX 3090.

num faces	MeshXL (125M)		MeshXL (350M)		MeshXL (1.3B)	
	time (s)	GPU mem. (GB)	time (s)	GPU mem. (GB)	time (s)	GPU mem. (GB)
100	6.30	1.59	11.30	2..98	12.08	8.41
200	12.50	1.65	22.70	3.20	24.03	9.17
400	25.21	1.85	45.81	3.78	48.09	11.17
800	49.88	2.28	92.19	5.74	96.49	21.66
avg.	29.49	-	44.65	-	49.43	-

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction of the paper clearly state the claims made, including the contributions made in the paper and important assumptions and limitations.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The paper discusses the limitations of the work performed by the authors, providing a balanced view of the study's scope and potential areas for improvement.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: All the theorems, formulas, and proofs in the paper have been numbered and cross-referenced, and all assumptions have been clearly stated or referenced in the statement of any theorems.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The paper has fully disclosed all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and conclusions of the paper.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We will open access to the code after the paper is accepted.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We have specified all the training and test details necessary to understand the results.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: The paper reports error bars and other appropriate information about the statistical significance of the experiments, ensuring the reliability and validity of the results.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: For each experiment, the paper provides sufficient information on the computer resource.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: The research conducted in the paper conforms in every respect with the NeurIPS Code of Ethics, ensuring ethical standards are upheld throughout the study.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The paper discusses both potential positive societal impacts and negative societal impacts of the work performed, providing a comprehensive evaluation of the study's broader implications.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [Yes]

    Justification: The paper describes safeguards that have been put in place for the responsible release of data or models that have a high risk for misuse, ensuring that appropriate measures are taken to mitigate potential risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The creators or original owners of assets used in the paper are properly credited, and the license and terms of use are explicitly mentioned and properly respected.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.

- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The new assets introduced in the paper are well documented, and the documentation is provided alongside the assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.