

FASTMESH: Efficient Artistic Mesh Generation via Component Decoupling

Supplementary Material

1. Implementation Details

1.1. Dataset Filtering

For training, we used data from Objaverse [3], Objaverse-XL [4], and ShapeNet [1]. To enable the network to learn high-quality meshes, we filtered out samples that had either non-manifold geometry or undesirable structure.

Non-manifoldness Rather than eliminating all instances of non-manifoldness, we retained certain meshes where the issue was minor and did not negatively affect model performance, as illustrated in Fig. 1(a). To quantify the degree of non-manifoldness, we analyzed each mesh by checking the usage count of all edges, ensuring that each edge is referenced exactly twice across faces by following the definition of manifoldness. Instead of removing all non-manifold meshes, we filtered out only those with severe non-manifold characteristics—specifically, meshes where more than 10% of edges were used only once or more than twice, as shown in Fig. 1(b)

Undesirable Structure We removed meshes containing unnecessary coplanar faces, as shown in Fig. 1(c). These structures often introduce redundant vertices that interfere with training. To find them, we analyzed vertex normals and discarded more than half of the vertices that shared identical normals, which typically indicates large coplanar regions.

Remove duplicate meshes We also found a significant number of overlapping meshes in the Objaverse dataset. To eliminate redundancy, we recorded the vertex and face counts for each mesh and removed duplicates by identifying meshes with equivalent vertex-face count pairs.

1.2. Architecture

The autoregressive model for vertex generation is trained to predict the next token based on the full input sequence. The length of the vertex sequence indexed in a block-wise manner corresponds to the sum of the number of blocks and vertices, and thus, the maximum number of tokens is determined by the upper limits of both. Note that we follow the configuration of BPT [8] by setting the block size and the offset resolution to 8^3 and 16^3 , respectively. We employ absolute positional embeddings, where a learnable embedding vector is added at each position in the sequence. The vocabulary size is $8^3 + 16^3 + 2$, accounting for block indices, offset indices, and two special tokens (i.e., start and end tokens). The sequence is then passed through a transformer

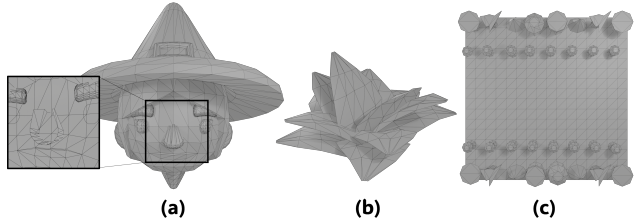


Figure 1. Examples from the dataset before and after filtering. (a) A non-manifold mesh with only minor defects is retained for training due to its limited impact on model performance. (b) A non-manifold mesh that is removed during filtering. (c) An undesirable structure with excessive faces used to represent a coplanar surface, also removed through filtering.

model with a causal mask using 24 transformer layers in the training phase. Lastly, the output features are projected to vocabulary-sized logits through a linear layer for next-token prediction. At the inference phase, the autoregressive model iteratively infers the next tokens and detokenizes the final sequence into discrete vertex positions. These initial positions are then refined by the fidelity enhancer, which has six transformer encoder layers to predict continuous offsets, improving the spatial accuracy of the generated geometry.

In the second stage, we reconstruct mesh faces using the given vertices and their spatial arrangement. To input the vertices into the transformer, we first convert the XYZ coordinates into three 256-dimensional vectors, resulting in a 768-dimensional representation. These vertex features are then processed through 24 layers of transformer blocks to capture geometric connectivity and spatial projection. To reduce the computational cost in the edge prediction, we compress the feature to 32×6 size, where 32 is the number of heads and 6 is the size of each vector. After that, we extract edge features by computing the Spacetime Distance [5] for each head, as described in Section 3.2 of the main paper. Each distance value is then used as the corresponding value for each dimension in the edge feature.

Lastly, we perform a prediction filtering process to refine the face generation. This filtering model is a fine-tuned version of the face generation network. During training, we ensured that all three types of attention masks—minimum candidate mask, minimum bandwidth mask, and no masking—were applied. To this end, we used a probabilistic ratio of 7:2:1, allowing the model to learn under diverse masking conditions while still being optimized for the minimum candidate setting. This sampling strategy encourages the model to generalize across different attention conditions while optimizing performance under the minimum candidate mask, which is used during final inference.

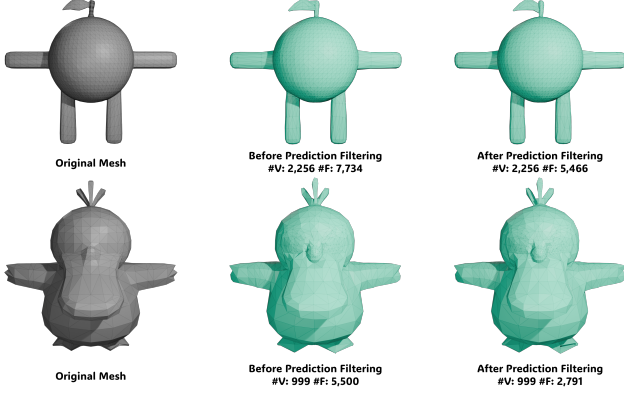


Figure 2. Results with and without prediction filtering. The post-processing reduces face count while preserving mesh structure.

The inference process consists of five iterative steps. In the first step, the initial adjacency matrix is generated using the face generation model. From the second step onward, the adjacency matrix is progressively refined through reordering and masking. Specifically, the second and third steps use the minimum bandwidth mask, while the fourth and fifth steps apply the minimum candidate mask to improve precision. This multi-step filtering process enables the model to correct early-stage errors and progressively eliminate unnecessary connections, thereby reducing the number of faces while preserving the original mesh shapes as shown in Fig 2.

1.3. Loss Function

FASTMESH utilizes two loss functions for vertex generation and one loss function for face generation. To train the autoregressive model in the first stage, we update the transformer parameters θ to maximize the likelihood of the sequence probability $P(x)$, as defined by the following equation:

$$P(x) = \prod_{t=1}^T P(x_t | x_{<t}, c; \theta), \quad (1)$$

where T is the number of tokens, x_t is the t -th token in the sequence, and c denotes the shape condition. This probability is maximized by minimizing the cross-entropy loss as follows:

$$\text{CELoss}(x) = - \sum_{t=1}^T \log P(x_t | x_{<t}, c, \theta) = - \log P(x). \quad (2)$$

In addition, we employ L1 loss to train the fidelity enhancer as a regression problem as follows:

$$\text{L1Loss}(v, \hat{v}) = \frac{1}{N} \sum_{i=1}^N |v_i - \hat{v}_i|, \quad (3)$$

where v and \hat{v} are predicted vertices and ground truth vertices, respectively.

1.4. Training Details

Our framework consists of three networks: an autoregressive model and a fidelity enhancer for vertex generation, and a face generation model. To accelerate training, we initialize the autoregressive model with pre-trained weights from MeshAnythingV2 [2]. We trained all models using the AdamW [6] optimizer with a weight decay of 0.99, for 400 epochs, with an additional 50 epochs for post-processing. The learning rate was linearly warmed up from 10^{-6} to 10^{-4} during the first epoch. After the warm-up phase, it was gradually decreased to 2×10^{-5} using cosine annealing scheduling. After completing the initial training of all components, we further fine-tuned the face generation model by applying prediction filtering. For evaluations and ablation studies, all experiments are conducted on a single A6000 GPU.

1.5. Sampling Strategy

In inference, we adopt a sampling strategy that balances diversity and stability in the autoregressive model (i.e., vertex generation) by controlling temperature, top- k , and top- p . Temperature controls randomness; top- k limits sampling to the k most probable candidates; and top- p ensures sampling from tokens with a cumulative probability $\geq p$. These factors directly affect sequence completeness and quality during generation. Empirically, we found temperature = 1.2, top- k = 100, and top- p = 0.9 to provide the best trade-off between mesh completeness and quality.

2. Detail Analysis in Toys4K dataset

We present a more detailed analysis of the Toys4K quantitative evaluation results shown in the main paper. Specifically, we break down the results by vertex count ranges and report the performance for each range in Table 1. As observed, our method consistently outperforms baselines in terms of Chamfer Distance (**CD**) across most categories, indicating high overall shape fidelity. However, in the 0–1000 vertex range, TreeMeshGPT and BPT slightly outperform our method in terms of Hausdorff Distance (**HD**). This suggests that autoregressive methods are highly precise when modeling short sequences, as fewer tokens are needed to represent simple meshes. Since mesh complexity increases and the required sequence length grows, these models tend to struggle with maintaining structural accuracy as shown on the Table. In contrast, our method not only captures the overall shape effectively but also demonstrates robust performance even as the mesh complexity increases, maintaining high-quality outputs in longer sequences.

Table 1. Quantitative detail comparison on the Toys4K dataset. We evaluated meshes with 5,000 or fewer vertices, partitioned them into five groups in increments of 1,000, and compared performance across each group.

Methods	0-1000 (872)		1000-2000 (505)		2000-3000 (320)		3000-4000 (212)		4000-5000 (154)	
	CD	HD	CD	HD	CD	HD	CD	HD	CD	HD
MeshAnything	12.95	28.77	10.39	24.34	12.86	29.13	10.88	22.38	11.88	25.80
MeshAnythingV2	10.32	25.16	10.50	26.91	10.08	25.48	9.55	21.58	10.11	21.35
TreeMeshGPT	4.02	9.65	5.61	14.46	7.14	18.19	7.51	19.85	6.85	19.86
BPT	4.64	9.31	5.31	11.04	7.51	15.69	8.00	17.58	6.14	15.33
FASTMESH-V1K	3.94	9.88	4.09	10.12	4.24	10.58	4.36	11.56	4.24	11.28
FASTMESH-V4K	3.91	9.74	4.06	10.05	4.13	10.49	4.23	11.19	4.34	11.54

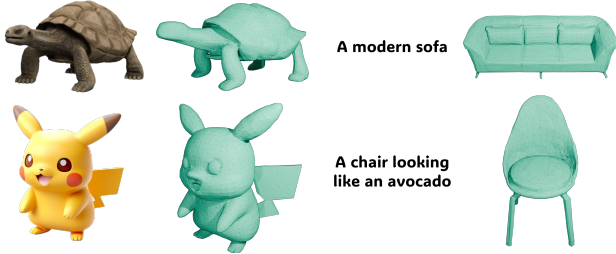


Figure 3. Examples of Image- and Text-to-3D Generation. The initial meshes are generated by TRELLIS [9], from which we extract point clouds to serve as shape conditions. (left: inputs, right: our results)

3. Applications and Additional Results

We utilize our framework in 3D generation models as a post-processing network to generate the results to be artistic meshes. We leverage TRELLIS [9] to generate initial meshes from a single image or text prompt, and sample the points to process the mesh generation in our framework. It can be seen that results are well-aligned with images and text prompts, producing structured artistic meshes as shown in Fig. 3. Furthermore, we additionally provide our results in Fig 4, 5 to further demonstrate that our method can accurately generate meshes across diverse cases.

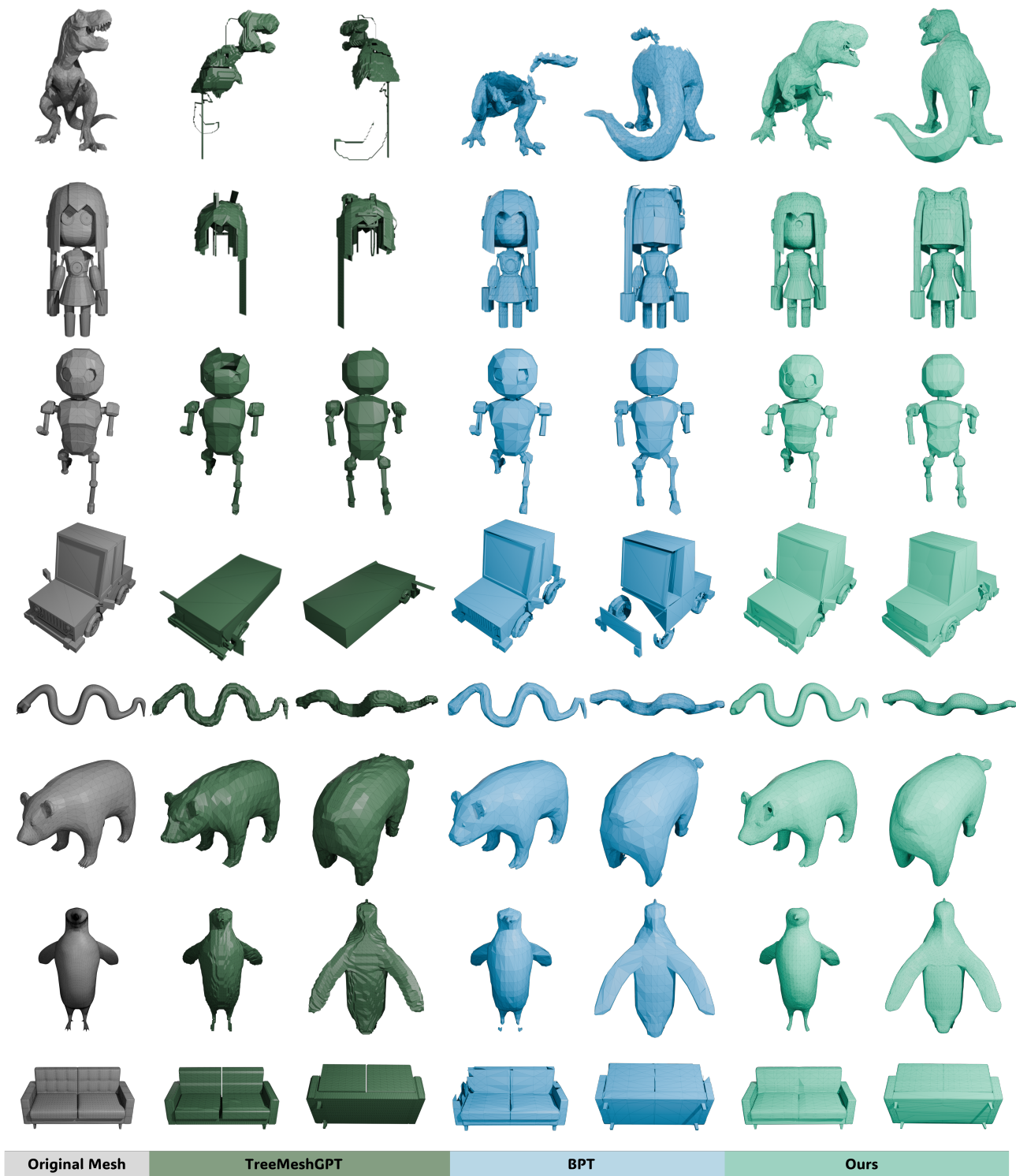


Figure 4. Additional qualitative comparison in Toys4K [7] dataset

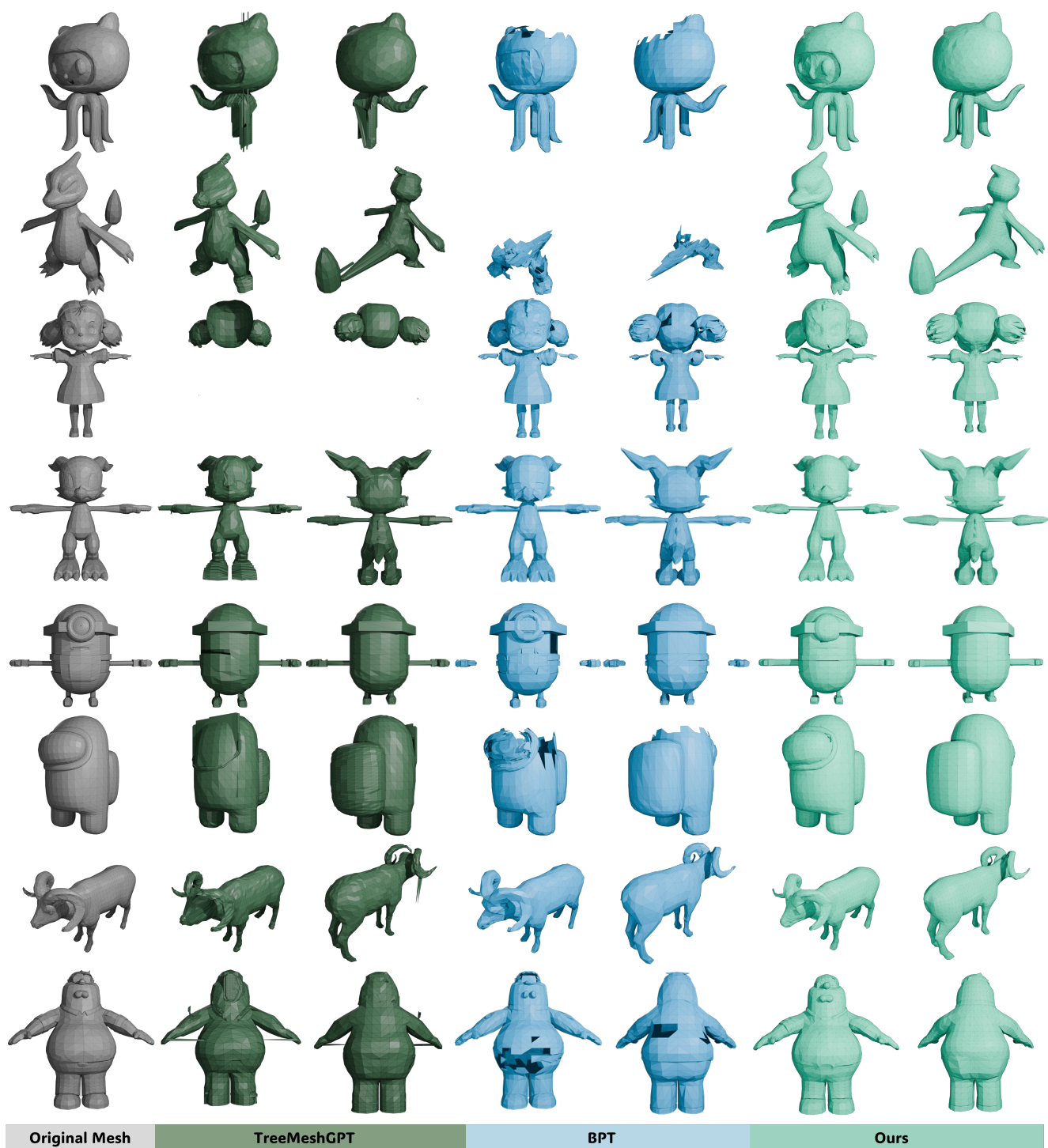


Figure 5. Additional qualitative comparison in Objaverse [3] and ObjaverseXL [4] dataset

References

- [1] 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. [1](#)
- [2] Yiwen Chen, Yikai Wang, Yihao Luo, Zhengyi Wang, Zilong Chen, Jun Zhu, Chi Zhang, and Guosheng Lin. MeshAnything V2: Artist-created mesh generation with adjacent mesh tokenization, 2024. [2](#)
- [3] 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. *arXiv preprint arXiv:2212.08051*, 2022. [1](#), [5](#)
- [4] Matt Deitke, Ruoshi Liu, Matthew Wallingford, Huong Ngo, Oscar Michel, Aditya Kusupati, Alan Fan, Christian Laforte, Vikram Voleti, Samir Yitzhak Gadre, Eli VanderBilt, Aniruddha Kembhavi, Carl Vondrick, Georgia Gkioxari, Kiana Ehsani, Ludwig Schmidt, and Ali Farhadi. Objaverse-xl: A universe of 10m+ 3d objects. *arXiv preprint arXiv:2307.05663*, 2023. [1](#), [5](#)
- [5] Marc T Law and James Lucas. Spacetime representation learning. In *The Eleventh International Conference on Learning Representations*, 2022. [1](#)
- [6] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. [2](#)
- [7] Stefan Stojanov, Anh Thai, and James M. Rehg. Using shape to categorize: Low-shot learning with an explicit shape bias. In *CVPR*, pages 1798–1808, 2021. [4](#)
- [8] Haohan Weng, Zibo Zhao, Biwen Lei, Xianghui Yang, Jian Liu, Zeqiang Lai, Zhuo Chen, Yuhong Liu, Jie Jiang, Chunchao Guo, Tong Zhang, Shenghua Gao, and C. L. Philip Chen. Scaling mesh generation via compressive tokenization. *CVPR*, 2025. [1](#)
- [9] Jianfeng Xiang, Zelong Lv, Sicheng Xu, Yu Deng, Ruicheng Wang, Bowen Zhang, Dong Chen, Xin Tong, and Jiaolong Yang. Structured 3d latents for scalable and versatile 3d generation. *arXiv preprint arXiv:2412.01506*, 2024. [3](#)