# 3DILG: Irregular Latent Grids for 3D Generative Modeling

**Anonymous Author(s)**
Affiliation
Address
email

## Abstract

We propose a new representation for encoding 3D shapes as neural fields. The representation is designed to be compatible with the transformer architecture and to benefit both shape reconstruction and shape generation. Existing works on neural fields are grid-based representations with latents being defined on a regular grid. In contrast, we define latents on irregular grids which facilitates our representation to be sparse and adaptive. In the context of shape reconstruction from point clouds, our shape representation built on irregular grids improves upon grid-based methods in terms of reconstruction accuracy. For shape generation, our representation promotes high-quality shape generation using auto-regressive probabilistic models. We show different applications that improve over the current state of the art. First, we show results of probabilistic shape reconstruction from a single higher resolution image. Second, we train a probabilistic model conditioned on very low resolution images. Third, we apply our model to category-conditioned generation. All probabilistic experiments confirm that we are able to generate detailed and high quality shapes to yield the new state of the art in generative 3D shape modeling.

## 1 Introduction

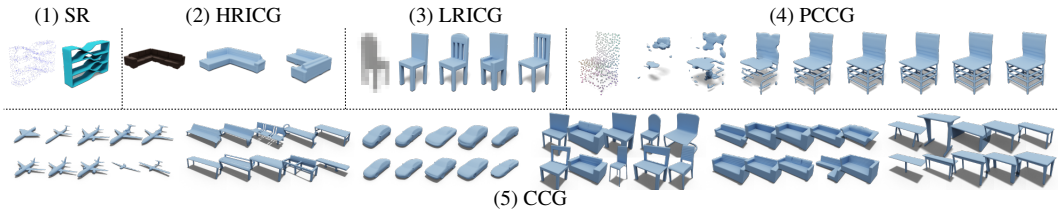


Figure 1: Irregular Latent Grids enable many applications: (1) shape reconstruction, (2) high-resolution-image-conditioned generation, (3) low-resolution-image-conditioned generation, (4) point-cloud-conditioned generation, and (5) category-conditioned generation. The data structure is especially suited for auto-regressive modeling (applications 2-5). For each of these applications the probabilistic approach enables sampling of many plausible models for a single query.

Neural fields for 3D shapes are popular in machine learning because they are generally easier to process with neural networks than other alternative representations, *e.g.*, triangle meshes or spline surfaces. Earlier works represent a shape with a single *global* latent [23, 11, 6, 30, 53]. This already gives promising results in shape autoencoding and shape reconstruction. However, shape details are often missing and hard to recover from a global latent. Later, *local* latent grids for shapes were

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

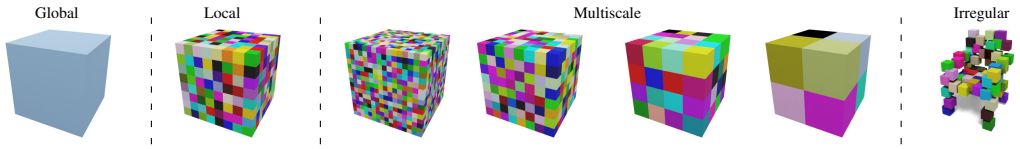


Figure 2: **Latent grids.** From left to right, we show single global latent (*e.g.*, OccNet [23]), latent grid (*e.g.* ConvONet [33]), multiscale latent grids (*e.g.* IF-Net [8]), and our irregular latent grid.

proposed [33, 20, 2]. A local latent mainly influences the shape (surface) in a local 3D neighborhood, thus perceiving shape details. However, in contrast to global latents, local latents require positional information about their location, *e.g.*, as implicitly defined by a regular grid. Furthermore, *multiscale* latent pyramid grids [8, 5] provide some performance improvement over basic grids. We illustrate the three current ways of modeling neural fields in Fig. 2.

In this paper, we set out to study irregular grids as 3D shape representation for neural fields. This extends previous grid-based representations, but allowing each latent to have an arbitrary position in space, rather than a position that is pre-determined on a regular grid. We do not want to give up the advantages of a fixed length representation, and therefore propose to encode shapes as a fixed length sequence of tuples $(\mathbf{x}_i, \mathbf{z}_i)_{i \in \mathcal{M}}$, where $\mathbf{x}_i$ are the 3D positions and $\mathbf{z}_i$ are the latents. The overall advantages of this representation are that it is fixed length, works well with transformer-based architectures using established building blocks, and that it can adapt to the 3D shape it encodes by placing latents at the most important positions. The representation also scales better than grid-based (and especially pyramid-based) representations to a larger number of latents. The number of latents in a representation is a factor that greatly influences the training time of transformer architectures, as the computation time is quadratic in the number of latents.

While our proposed representation brings some improvement for 3D shape reconstruction from point clouds, the improvement is very significant for 3D generative models that have been less explored. We believe that the most promising avenue for 3D shape generation is to follow recent image generative methods based on vector quantization and auto-regressive models using transformer, *e.g.* VQGAN [14]. These models operate on discrete latent vectors represented by indices [44]. During training, an autoregressive probabilistic model is trained to predict discrete indices. When doing inference (sampling), discrete indices are sampled from the autoregressive model and are decoded to images with a learned decoder. These models work well in the image domain, because a single image can comfortably be represented by medium size, *e.g.* $16 \times 16$ latent grids of 256 latents [14]. This does not directly scale to 3D shapes, as a $16 \times 16 \times 16$ grid is too large to be comfortably trained on 4-8 GPUs due to the quadratic complexity of the transformer architecture [46]. As a result, generative models based on low-resolution regular grids lead to artifacts in the details, whereas our representation yields a nice and clean surface (*e.g.* Fig. 9).

We show the following applications of our proposed representations (see Fig. 1). For 3D shape reconstruction, we show results for 3D reconstruction from a point cloud. For generative modeling, we show results for image-conditioned 3D shape generation, object category-conditioned 3D shape generation, and point-cloud-conditioned 3D shape generation. The generative model is probabilistic and can generate multiple different 3D shapes for the same conditioning information.

We summarize the contributions of our work as follows: 1) We propose irregular latent grids as 3D shape representation for neural fields. Our shape representation thereby extends existing works using a global latent, a local latent grid or a multi-scale grid. 2) We improve upon grid-based SOTA methods for 3D shape reconstruction for point-clouds. 3) We improve state-of-the art generative models for 3D shape generation, including image-conditioned generation, object category-conditioned generation and point cloud-conditioned generation.

## 2 Related Work

### 2.1 Neural shape representations

Shape analysis with neural networks processes shapes in different representations. Common representations include voxels [9, 10, 37] and point clouds [34, 35, 48, 49, 42]. More recently, researchers study shapes represented with neural fields [50], *e.g.*, signed distance functions (SDFs) or occupancy

(indicator) functions of shapes modeled by neural networks. Subsequently, meshes can be extracted by contouring methods such as marching cubes [22]. The methods have been called *neural implicit representations* [23, 24, 30, 15, 11, 6, 53] or coordinate-based representations [40]. We decided to use the term *neural fields* in this paper [50].

**Point cloud processing.** Earlier works for point cloud processing rely on multilayer perceptrons (MLPs), *e.g.*, PointNet [34]. PointNet++ [35] and DGCNN [48] extended the idea by employing a hierarchical structure to capture local information. Inspired by ViT [13], which treats images as a set of patches, PCT [17] and PT [54] propose a transformer backbone for point cloud processing. Both works introduce extra modules which are not standard transformers [46] anymore. Since our main goal is not to develop a general backbone for point clouds, we simply work with a *standard* transformer for shape autoencoding (first-stage training). Similarly, both PointBERT[52] and PointMAE [28] use standard transformers for point cloud self-supervised learning.

**Neural fields for 3D shapes.** One possible approach is to represent a shape with a single *global* latent, *e.g.*, OccNet [23], CvxNet [11] and Zhang *et al.* [53]. While this method is very simple, it is not suitable to encode surface details or shapes with complex topology. Later works studied the use of *local* latents. Compared to global latents, these representations represent shapes with multiple local latents. Each latent is responsible for reconstruction in a local region. ConvONet [33], as a follow-up work of OccNet, learns latents on a grid (2d or 3d). IF-Net [8], trains a model which outputs 3d latent grids of several different resolutions. The grids are then concatenated into a final representation.

### 2.2 Generative models

Generative models include generative adversarial networks (GANs) [16], variational autoencoders (VAEs) [21], normalizing flows (NFs) [36, 12] and auto-regressive models (ARs) [45]. Our shape representation is designed to be compatible with auto-regressive generative models. Thus we will mainly discuss related work in this area.

In the image domain, earlier AR works generate pixels one-by-one, *e.g.*, PixelRNN [45] and its follow-up works [43, 38]. Combining this idea with autoregressive transformers, similar approaches are applied to 3D data, *e.g.*, PointGrow [39] for point cloud generation and PolyGen [27] for polygonal mesh generation. Other use cases include floor plan generation [29] and indoor scene generation [47, 32].

An important ingredient of many auto-regressive models is vector quantization (VQ) that was originally integrated into VAEs. VQVAE [44] adopts a two-stage training strategy. The first stage is an autoencoder with vector quantization, where bottleneck latents are quantized to discrete indices. In the second stage, an autoregressive model is trained to predict the discrete indices. This idea is further developed in VQGAN [14] which builds the autoregressive model with a transformer [46]. Vector quantization, has been shown to be an efficient way for generating high resolution images. We study how VQ can be used in the 3D domain, specifically, *neural fields* for 3D shapes.

There are some works adopting VQVAE to the 3D domain. Canonical Mapping [7] proposes point cloud generation models with VQVAE. In concurrent work, AutoSDF [26] generalizes VQVAE to 3D voxels. However, the bottleneck latent grid resolution is low and it is difficult to capture details as in ConvONet and IF-Net. Increasing the resolution is the key for detailed 3D shapes reconstruction. On the other hand, higher resolution, means more difficult training of autoregressive transformers. Another concurrent work, ShapeFormer [51], shows a solution by sparsifying latent grids. However, ShapeFormer still only accepts voxel grids as input (point clouds need to be voxelized). Also, the sequence length varies for different shapes requiring the method to go through a whole dataset to find a maximum sequence length. Again, the maximum sequence length is dataset-dependent. Comparing to existing works, our method processes point clouds directly, learns fixed-size latents and outputs neural fields. A comparison summary can be found in Table 1.

## 3 Shape Reconstruction

Given an input point cloud $\{\mathbf{x}_i \colon \mathbf{x}_i \in \mathbb{R}^3\}_{i \in \mathcal{N}}$ (also in matrix form $\mathbf{X} \in \mathbb{R}^{N \times 3}$), our goal is to predict the indicator function $\mathcal{O}(\cdot) : \mathbb{R}^3 \to [0, 1]$ corresponding to the volume represented by the

Table 1: **Methods comparison.** All methods here take point clouds as input. If the column **Requiring Voxelization** says yes, it means the method takes voxels as input. The numbers shown in parenthesis are alternative choices of hyperparameters.

| | Local Latents | Requiring Voxelization | Output | Latent Resolution | Vector Quantization | Sequence Length | Comments |
|---|---|---|---|---|---|---|---|
| OccNet [23] | ✗ | ✗ | Fields | 1 | ✗ | 1 | |
| ConvONet [33] | ✓ | ✓ | Fields | $32^3(64^3)$ | ✗ | $32^3(64^3)$ | |
| IF-Net [8] | ✓ | ✓ | Fields | $128^3$ | ✗ | $128^3$ | Multiscale |
| AutoSDF [26] | ✓ | ✓ | Voxels | $8^3$ | ✓ | $8^3$ | |
| CanMap[7] | ✓ | ✗ | Point Cloud | 128 | ✓ | 128 | |
| ShapeFormer [51] | ✓ | ✓ | Fields | $16^3(32^3)$ | ✓ | variable | Sparsification |
| 3DILG (Ours) | ✓ | ✗ | Fields | 512 | ✓ | 512 | |

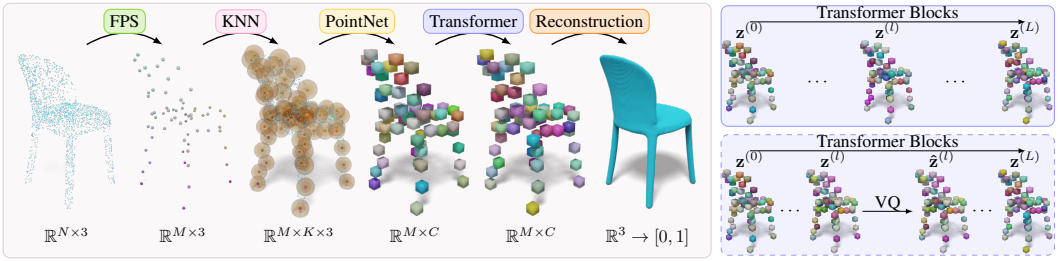


Figure 3: **Shape reconstruction from point clouds. Left:** the main pipeline. The framework can be used with (**Bottom Right**) or without (**Top Right**) vector quantization.

point cloud. Our representation is being generated in three steps (See Fig. 3): patch construction (Sec. 3.1), patch information processing (Sec. 3.2), and reconstruction (Sec. 3.3).

### 3.1 Patch Construction

We sub-sample the input point cloud via Farthest Point Sampling (FPS),

$$\boxed{\text{FPS}}\,(\{\mathbf{x}_i\}_{i\in\mathcal{N}}) = \{\mathbf{x}_i\}_{i\in\mathcal{M}} = \mathbf{X}_{\mathcal{M}} \in \mathbb{R}^{M\times 3}, \tag{1}$$

where $\mathcal{M} \subset \mathcal{N}$ and $|\mathcal{M}| = M$. Next, for each point in the sub-sampled point set $\{\mathbf{x}_i\}_{i\in\mathcal{M}}$, we apply the K-nearest neighbor (KNN) algorithm to find $K-1$ points to form a point patch of size $K$,

$$\forall i \in \mathcal{M}, \quad \boxed{\text{KNN}}\,(\mathbf{x}_i) = \{\mathbf{x}_j\}_{j\in\mathcal{N}_i}, \tag{2}$$

where $\mathcal{N}_i$ is the neighbor index set for point $\mathbf{x}_i$ and $|\mathcal{N}_i| = K-1$. Thus we have a collection of point patches, $\{(\mathbf{x}_i, \{\mathbf{x}_j\}_{j\in\mathcal{N}_i}) : |\mathcal{N}_i| = K-1\}_{i\in\mathcal{M}} = \mathbf{X}_{\mathcal{M},K} \in \mathbb{R}^{M\times K\times 3}$. We project each patch with a mini-PointNet-like [34] module to an embedding vector

$$\forall i \in \mathcal{M}, \quad \boxed{\text{PointNet}}\,(\mathbf{x}_i, \{\mathbf{x}_j\}_{j\in\mathcal{N}_i}) = \mathbf{e}_i \in \mathbb{R}^C, \tag{3}$$

where $C$ is the embedding dimension for patches.

### 3.2 Transformer

Furthermore, we build a transformer to learn local latents $\{\mathbf{z}_i\}_{i\in\mathcal{M}}$. The point coordinates $\{\mathbf{x}_i\}_{i\in\mathcal{M}}$ are converted to positional embeddings (PEs) [25], $\mathbf{p}_i = \text{PE}(\mathbf{x}_i) \in \mathbb{R}^C$. Our transformer takes as input the sequence of patch-PE pairs,

$$\boxed{\text{Transformer}}\,(\{(\mathbf{e}_i, \mathbf{p}_i)\}_{i\in\mathcal{M}}) = \{\mathbf{z}_i : \mathbf{z}_i \in \mathbb{R}^C\}_{i\in\mathcal{M}}. \tag{4}$$

The transformer is composed of $L$ blocks. We denote the intermediate outputs of all blocks as $\mathbf{z}_i^{(0)}, \mathbf{z}_i^{(1)}, \cdots, \mathbf{z}_i^{(l)}, \cdots, \mathbf{z}_i^{(L)}$, where $\mathbf{z}_i^{(0)}$ is the input $\mathbf{e}_i$ and $\mathbf{z}_i^{(L)}$ is the output $\mathbf{z}_i$.

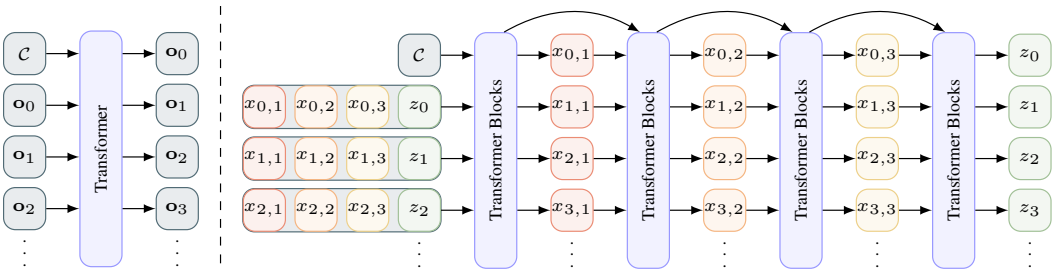


Figure 4: **Autoregressive Generative Models with Unidirectional Transformer. Left**: sequence prediction. **Right**: detailed visualizations with sequence element components.

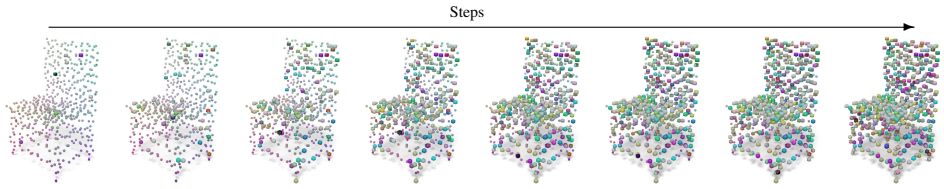


Figure 5: **Autoregressive Generative Models with Bidirectional Transformer. Cubes** (□) are predicted tokens. From left to right, we show 8 decoding steps.

**Vector Quantization.** This model can be used with vector quantization [44]. We simply replace the intermediate output at block $l$ with its closest vector in a dictionary $\mathcal{D}$,

$$\forall i \in \mathcal{M}, \quad \underset{\hat{\mathbf{z}}_i^{(l)} \in \mathcal{D}}{\arg\min} \left\| \hat{\mathbf{z}}_i^{(l)} - \mathbf{z}_i^{(l)} \right\|. \tag{5}$$

The dictionary $\mathcal{D}$ contains $|\mathcal{D}| = D$ vectors. As in VQVAE [44], the gradient with respect to $\mathbf{z}_i^{(l)}$ is approximated with the straight-through estimator [1].

### 3.3 Reconstruction

We expect that each latent $\mathbf{z}_i$ is responsible for shape reconstruction near the point $\mathbf{x}_i$. However, our goal is to estimate $\mathcal{O}(\cdot)$ for an arbitrary $\mathbf{x} \in \mathbb{R}^3$. We interpolate latent $\mathbf{z}_\mathbf{x}$ for point $\mathbf{x}$ with the Nadaraya-Watson estimator, $\mathbf{z}_\mathbf{x} = \frac{\sum_{i \in \mathcal{M}} \exp(-\beta \|\mathbf{x} - \mathbf{x}_i\|^2) \mathbf{z}_i}{\sum_{i \in \mathcal{M}} \exp(-\beta \|\mathbf{x} - \mathbf{x}_i\|^2)}$, where $\beta$ controls the smoothness of interpolation and can be fixed or learned. The final indicator is an MLP with a sigmoid activation, $\hat{\mathcal{O}}(\mathbf{x}) = \mathrm{Sigmoid}(\mathrm{MLP}(\mathbf{x}, \mathbf{z}_\mathbf{x}))$.

**Loss Function.** We optimize the estimated $\hat{\mathcal{O}}(\cdot)$ by comparing to ground-truth $\mathcal{O}(\cdot)$ via a binary cross entropy loss (BCE) and an additional commitment loss term [44],

$$\mathcal{L} = \mathcal{L}_{\text{recon}} + \lambda \mathcal{L}_{\text{commit}} = \mathbb{E}_{\mathbf{x} \in \mathbb{R}^3} \left[ \mathrm{BCE}(\hat{\mathcal{O}}(\mathbf{x}), \mathcal{O}(\mathbf{x})) \right] + \lambda \mathbb{E}_{\mathbf{x} \in \mathbb{R}^3} \left[ \mathbb{E}_{i \in \mathcal{M}} \left\| \mathrm{sg}(\hat{\mathbf{z}}_i^{(l)}) - \mathbf{z}_i^{(l)} \right\|^2 \right], \tag{6}$$

where $\mathrm{sg}(\cdot)$ is the stop-gradient operation. When $\lambda = 0$, it means we train the model without vector quantization.

## 4 Autoregressive Generative Modeling

With Vector Quantization (Eq. 5), we compress the bit size of intermediate latents $\{\hat{\mathbf{z}}_i^{(l)}\}_{i \in \mathcal{M}}$ to $\log_2 D$ where $D$ is the size of the dictionary $\mathcal{D}$. We denote the compressed index as $z_i \in \{0, 1, \ldots, D-1\}$. We also quantize point coordinates $\mathbf{x}_i$ to $(x_{i,1}, x_{i,2}, x_{i,3})$ where each entry is a 8-bit integer $\{0, 1, \ldots, 255\}$, As a result, we obtain a discrete representation of the 3D shape $\{(x_{i,1}, x_{i,2}, x_{i,3}, z_i)\}_{i \in \mathcal{M}}$.

### 4.1 Unidirectional Transformer

Autoregressive Generation with unidirectional transformer is a more classical approach. We often need to sequentialize an unordered set if the order is not defined yet. Specifically, we re-order the representations ascendingly by first component $x_{i,1}$, then by second component $x_{i,2}$, then by third component $x_{i,3}$,

$$\mathcal{S} = \{(x_{0,1}, x_{0,2}, x_{0,3}, z_0), (x_{1,1}, x_{1,2}, x_{1,3}, z_1), \ldots, (x_{M-1,1}, x_{M-1,2}, x_{M-1,3}, z_{M-1})\}. \quad (7)$$

Our goal to predict sequence elements one-by-one. A common way is to flatten the sequence $\mathcal{S}$ by concatenating the quadruplets, for examples, PointGrow [39] and PolyGen [27]. However, here we consider an approach generating *quadruplet-by-quadruplet*. We write $\mathbf{o}_i = (x_{i,1}, x_{i,2}, x_{i,3}, z_i)$. The likelihood of generating $\mathbf{o}_i$ autoregressively is

$$p(\mathcal{S} \mid \mathcal{C}) = \prod_{i=0}^{i=M-1} p(\mathbf{o}_i \mid \mathbf{o}_{<i}, \mathcal{C}), \quad (8)$$

where $\mathcal{C}$ is conditioned context. Similar to ATISS [32], we predict components of $\mathbf{o}_i = (x_{i,1}, x_{i,2}, x_{i,3}, z_i)$ autoregressively:

$$p(\mathbf{o}_i|\mathbf{o}_{<i}) = p(x_{i,1}|\mathbf{o}_{<i})p(x_{i,2}|x_{i,1}, \mathbf{o}_{<i})p(x_{i,3}|x_{i,2}, x_{i,1}, \mathbf{o}_{<i})p(z_i|x_{i,3}, x_{i,2}, x_{i,1}, \mathbf{o}_{<i}), \quad (9)$$

where the context $\mathcal{C}$ is omitted to avoid clutter. The model is shown in Fig. 4. Different from ATISS [32] which uses MLPs to decode different components, instead we continue to apply transformer blocks for component decoding.

### 4.2 Bidirectional Transformer

Bidirectional transformer for autoregressive generation is recently proposed by MaskGIT [4]. Here we show our model can also be combined with bidirectional transformer. However, here we consider a different task. We generate $\{z_i\}_{i\in\mathcal{M}}$ conditioned on $\{(x_{i,1}, x_{i,2}, x_{i,3})\}_{i\in\mathcal{M}}$. In training phase, we sample a subset $\{z_i\}_{i\in\mathcal{V}\subset\mathcal{M}}$ of $\{z_i\}_{i\in\mathcal{M}}$ as the input of the bidirectional transformer. We aim to predict $\{z_i\}_{i\in\mathcal{M}\setminus\mathcal{V}}$. The coordinates $\{(x_{i,1}, x_{i,2}, x_{i,3})\}_{i\in\mathcal{M}}$ are converted to positional embeddings (either learned or fixed) as condition. The likelihood of generating $\mathcal{M} \setminus \mathcal{V}$ is as follows,

$$\prod_{\mathcal{V}\subset\mathcal{M}} p(\{z_i\}_{i\in\mathcal{M}\setminus\mathcal{V}} \mid \{z_i\}_{i\in\mathcal{V}}, \{(x_{i,1}, x_{i,2}, x_{i,3})\}_{i\in\mathcal{M}}). \quad (10)$$

In practice, the bidirectional transformer takes as input all tokens except that $\{z_i\}_{i\in\mathcal{M}\setminus\mathcal{V}}$ are replaced by a special masked token $[m]$. When decoding (inference), we iteratively predict multiple tokens at the same time. Tokens are sampled based on their probabilities (transformer output). See [4] for a detailed explanation. We show visualization of decoding steps in Fig. 5.

## 5 Reconstruction Experiments

We set the size of the input point cloud to $N = 2048$. The number and the size of point patches are $M = 512$ and $K = 32$, respectively. In the case of Vector Quantization, there are $D = 1024$ vectors in the dictionary $\mathcal{D}$. Other details of the implementation can be found in the Appendix. We use the datset ShapeNet-v2 [3] for shape reconstruction. We split samples into train/val/test (48597/1283/2592) set. Following the evaluation protocol of [11, 53], we include three metrics, volumetric IoU, the Chamfer-L1 distance, and F-Score [41]. An evaluation on scene level reconstruction can be found in the appendix.

**Results** We compare our method with three existing works, OccNet [23], ConvONet [33] and IF-Net [8]. The results can be found in Table 2. We select 7 categories among 55 categories with largest training set (*table, car, chair, airplane, sofa, rifle, lamp*). Detailed results can be found in the Appendix. We show different choices of $M$. As we can see in this table, increasing $M$ from 64 to 512 gives performance boost. Even with the simplest model $M = 64$, our results are better than ConvONet's. The best results are achieved when setting $M = 512$. In this case, our results lead in most categories when comparing to IF-Net. The results of IF-Net are better than ours in terms of metric F-Score in some categories. However, we argue that in this case the metric F-score is saturated (values are close to 1), which making it hard to compare. We also show results after introducing vector quantization to our model in the same table. We can see that, vector quantization harms the performance slightly.

Table 2: **Shape reconstruction.** We train all models on ShapeNet-v2 (55 categories). All competitor methods are trained with the corresponding official released code. For our model, we set $N = 2048$ and $K = 32$. We select 7 categories to show. These categories have largest training set among 55 categories. We also show averaged metrics over all categories. The numbers shown in parenthesis are results of vector quantization.

| | | OccNet | ConvONet | IF-Net | 3DILG (Ours) | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | $M = 64$ | $M = 128$ | $M = 256$ | $M = 512$ |
| IoU ↑ | mean (selected) | 0.822 | 0.881 | 0.929 | 0.922(0.904) | 0.936(0.929) | 0.945(0.943) | **0.952**(0.950) |
| | mean (all) | 0.825 | 0.888 | 0.934 | 0.923(0.907) | 0.937(0.929) | 0.946(0.943) | **0.953**(0.950) |
| Chamfer ↓ | mean (selected) | 0.058 | 0.040 | 0.034 | 0.038(0.038) | 0.035(0.036) | 0.034(0.034) | **0.032**(0.030) |
| | mean (all) | 0.072 | 0.052 | 0.041 | 0.048(0.052) | 0.044(0.046) | 0.041(0.042) | **0.040**(0.040) |
| F-Score ↑ | mean (selected) | 0.898 | 0.951 | 0.975 | 0.959(0.948) | 0.968(0.964) | 0.972(0.969) | **0.976**(0.975) |
| | mean (all) | 0.858 | 0.933 | **0.967** | 0.942(0.926) | 0.955(0.948) | 0.963(0.958) | 0.966(0.965) |

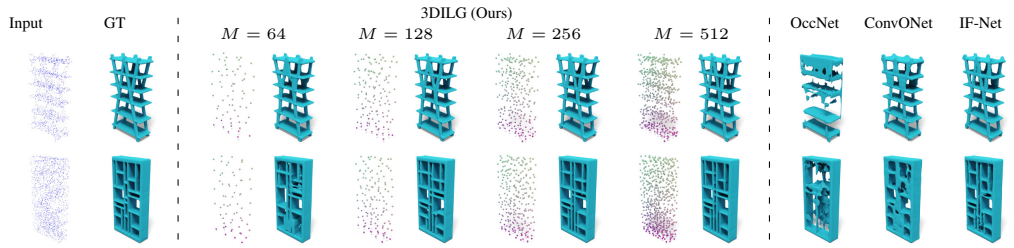


Figure 6: **Shape reconstruction.** The column **Input** shows input point clouds of size 2048. The column **GT** shows ground-truth meshes. We compare our results of different $M$ with ConvONet. We also show $\{\mathbf{x}_i\}_{i \in \mathcal{M}}$ obtained via Farthest Point Sampling.

**Qualitative Comparison** The qualitative results are shown in Fig. 6. From the visualization, it can be seen that the reconstruction quality increases as $M$ increases, especially for shapes with complex topology. OccNet, as a global latent method, often fails to recover complex structures. The recovering ability of ConvONet are much better than OccNet because of localized latents. By learning multiscale latents, IF-Net improves further upon ConvONet. Regarding our method, results of $M = 64$ are better than ConvONet. When $M = 128$, the results are comparable with IF-Net.

## 6 Generative Experiments

We introduce three experiments to show how our model can be combined with auto-regressive transformers for generative modeling. In Sec. 6.1, we show image-conditioned generation, a.k.a, probabilistic shape reconstruction from single image. In Sec. 6.2, we show how we generate samples given a category label (using the 55 ShapeNet categories). In Sec. 6.3, we further show generation conditioned on downsampled point clouds. Different from the first two tasks, the point-cloud-conditioned generation is based on a bidirectional transformer described in Sec 4.2.

### 6.1 Probabilistic Shape Reconstruction from a Single Image

We train a uni-directional transformer for this task (see Sec. 4.1). The context $\mathcal{C}$ is an image. To train this model, we render 40 images ($224 \times 224$) of different views for each shape in ShapeNet. The implementation of the uni-directional transformer is based on GPT. It contains 24 blocks, where each block has an attention layer with 16 heads and 1024 embedding dimension. When sampling, nucleus sampling [19] with top-$p$ (0.85) and top-$k$ (100) are applied to predicted token probabilities.

**High resolution images.** We show some generated samples when $\mathcal{C}$ are high resolution images in Fig. 7. We compare our results with a deterministic method, OccNet [23]. As we can see in the results, the deterministic method (OccNet) tends to create blurred meshes. However, probabilistic reconstruction (ours) is able to output detailed meshes.

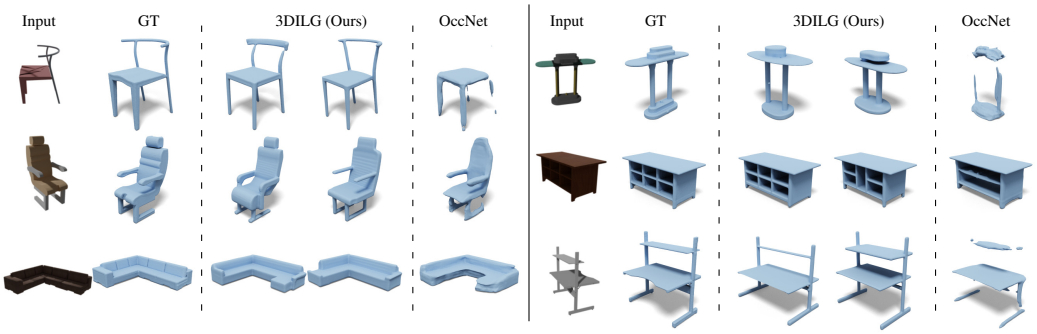


Figure 7: **Image-conditioned generation** ($224 \times 224$)**.** We sample 2 shapes for each input image, and compare them with OccNet.

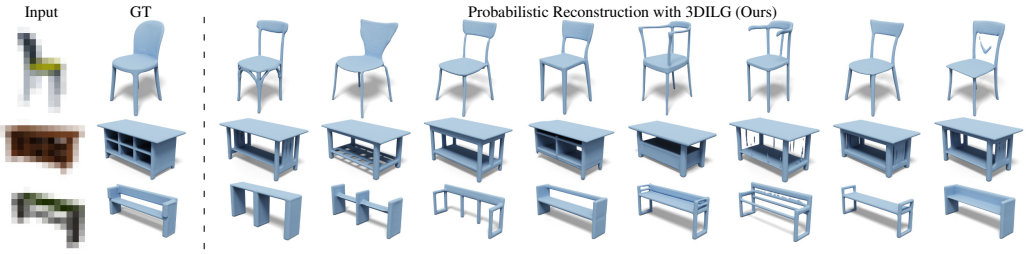


Figure 8: **Image-conditioned generation** ($16 \times 16$)**.** We sample 8 shapes for each input image.

**Low resolution images.** We also consider another more challenging task. The input images are down-sampled to low resolution ones ($16 \times 16$). In this case, the generative model has more freedom to find multiple plausible interpretations, including variations with different topology (see Fig. 8).

### 6.2 Category-Conditioned Generation

To generate shapes given a category label, we use the context $\mathcal{C}$ to encode the category label. We employ the uni-directional transformer based on GPT as in Sec. 6.1.We compare to our proposed baseline model that encodes shapes as a latent grid of resolution $8^3$. To make a fair comparison, we also extend ViT [13] to the 3D voxel domain in the first stage of training. In the second stage, we use the same uni-directional transformer for sequence prediction. Here the sequence length is $8^3 = 512$, which is the same as in our proposed model. The baseline model is named **Grid-**$8^3$. The comparison to the baseline model is in Fig. 9. We can see that the baseline is unable to generate high quality shapes. We argue that this is because the representation is not expressive enough to capture surface details. Furthermore, we show more generated samples of our model in Fig. 10. The three selected categories are *bookshelf, bench* and *chair*. For both the baseline and our model, we render 10 images of predicted shapes, and calculate the Fréchet Inception Distance (FID) [18, 31] between predictions and test sets. The metrics can be found in Table 3. A perceptual study on the quality of generated samples can be found in the appendix.

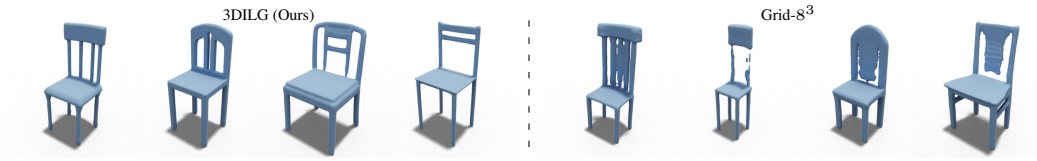


Figure 9: **Comparison of category-conditioned generation.** We compare our results (**Left**) with the baseline (**Right**).

Table 3: **FID ↓ for category-conditioned generation**. We compare our results with the baseline Grid-$8^3$. The 7 categories are the largest categories in ShapeNet.

| | Categories | | | | | | | mean |
|---|---|---|---|---|---|---|---|---|
| | table | car | chair | airplane | sofa | rifle | lamp | |
| Grid-$8^3$ | 72.396 | 95.566 | 58.649 | 42.009 | 58.092 | 59.456 | 87.319 | 67.641 |
| 3DILG (Ours) | **68.016** | **92.597** | **45.333** | **30.957** | **53.244** | **40.500** | **72.672** | **57.617** |

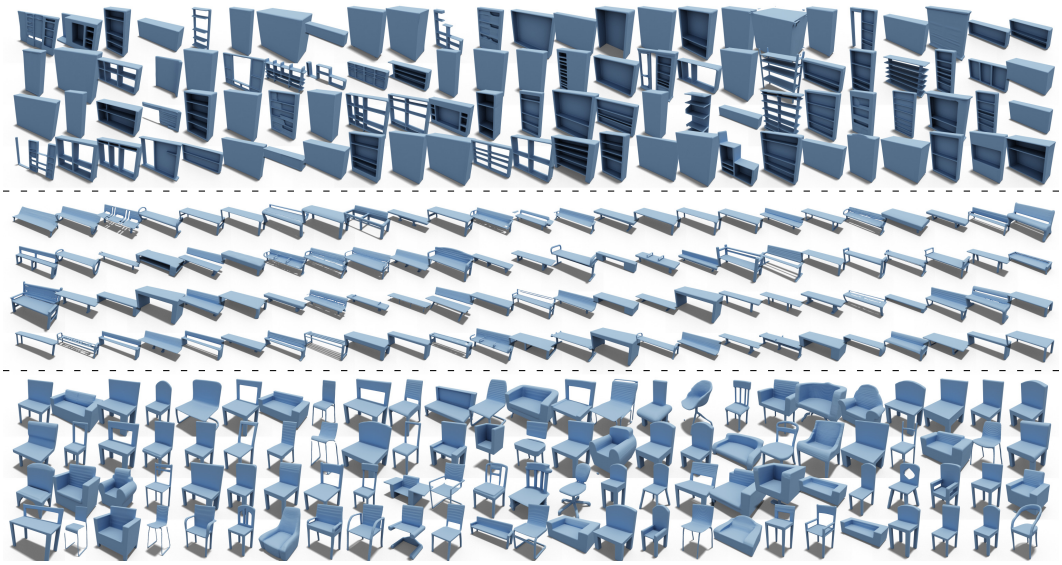

Figure 10: **Category-conditioned generation.** We choose 3 categories to show (*bookshelf, bench* and *chair*). We show 100 samples for each category.

### 6.3 Point Cloud Conditioned Generation

We train a bidirectional transformer described in Sec 4.2. The model takes as input $\{\mathbf{x}_i\}_{i \in \mathcal{M}}$. The results can be found in Fig. 11.

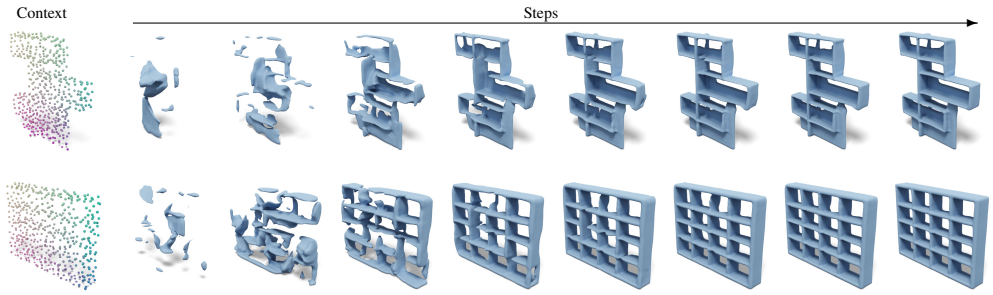


Figure 11: **Point cloud conditioned generation.** We show 8 decoding steps.

## 7 Conclusion

We studied neural fields for shapes and presented a new representation. Different from common approaches which define latents a on a regular grid or multiple regular grids, we position latents on an irregular grid. Comparing to existing works, the representation better scales to larger models, because the irregular grid is sparse and adapts to the underlying 3D shape of the object. In our results, we demonstrated an improvement in 3D shape reconstruction from point clouds and generative modeling conditioned on images, object category, or point-clouds over alternative grid-based methods. In future work, we suggest to explore other applications of our proposed representation, *e.g.*, shape completion, and extensions to textured 3D shapes and 3D scenes.

# References


[1] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013. (Cited on 5)

[2] Rohan Chabra, Jan E Lenssen, Eddy Ilg, Tanner Schmidt, Julian Straub, Steven Lovegrove, and Richard Newcombe. Deep local shapes: Learning local sdf priors for detailed 3d reconstruction. In *European Conference on Computer Vision*, pages 608–625. Springer, 2020. (Cited on 2)

[3] 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. (Cited on 6)

[4] Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T Freeman. Maskgit: Masked generative image transformer. *arXiv preprint arXiv:2202.04200*, 2022. (Cited on 6)

[5] Zhang Chen, Yinda Zhang, Kyle Genova, Sean Fanello, Sofien Bouaziz, Christian Häne, Ruofei Du, Cem Keskin, Thomas Funkhouser, and Danhang Tang. Multiresolution deep implicit functions for 3d shape representation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13087–13096, 2021. (Cited on 2)

[6] Zhiqin Chen, Andrea Tagliasacchi, and Hao Zhang. Bsp-net: Generating compact meshes via binary space partitioning. *arXiv preprint arXiv:1911.06971*, 2019. (Cited on 1, 3)

[7] An-Chieh Cheng, Xueting Li, Sifei Liu, Min Sun, and Ming-Hsuan Yang. Autoregressive 3d shape generation via canonical mapping. *arXiv preprint arXiv:2204.01955*, 2022. (Cited on 3, 4)

[8] Julian Chibane, Thiemo Alldieck, and Gerard Pons-Moll. Implicit functions in feature space for 3d shape reconstruction and completion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6970–6981, 2020. (Cited on 2, 3, 4, 6)

[9] Christopher B Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *European conference on computer vision*, pages 628–644. Springer, 2016. (Cited on 2)

[10] Angela Dai, Charles Ruizhongtai Qi, and Matthias Nießner. Shape completion using 3d-encoder-predictor cnns and shape synthesis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5868–5877, 2017. (Cited on 2)

[11] Boyang Deng, Kyle Genova, Soroosh Yazdani, Sofien Bouaziz, Geoffrey Hinton, and Andrea Tagliasacchi. Cvxnet: Learnable convex decomposition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 31–44, 2020. (Cited on 1, 3, 6)

[12] Laurent Dinh, David Krueger, and Yoshua Bengio. NICE: non-linear independent components estimation. In Yoshua Bengio and Yann LeCun, editors, *International Conference on Learning Representations (ICLR)*, 2015. (Cited on 3)

[13] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR*, 2021. (Cited on 3, 8)

[14] Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12873–12883, 2021. (Cited on 2, 3)

[15] Kyle Genova, Forrester Cole, Daniel Vlasic, Aaron Sarna, William T Freeman, and Thomas Funkhouser. Learning shape templates with structured implicit functions. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7154–7164, 2019. (Cited on 3)

[16] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in Neural Information Processing Systems*, 27:2672–2680, 2014. (Cited on 3)

[17] Meng-Hao Guo, Jun-Xiong Cai, Zheng-Ning Liu, Tai-Jiang Mu, Ralph R Martin, and Shi-Min Hu. Pct: Point cloud transformer. *Computational Visual Media*, 7(2):187–199, 2021. (Cited on 3)

[18] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017. (Cited on 8)

[19] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751*, 2019. (Cited on 7)

[20] Chiyu Jiang, Avneesh Sud, Ameesh Makadia, Jingwei Huang, Matthias Nießner, Thomas Funkhouser, et al. Local implicit grid representations for 3d scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6001–6010, 2020. (Cited on 2)

[21] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In Yoshua Bengio and Yann LeCun, editors, *International Conference on Learning Representations (ICLR)*, 2014. (Cited on 3)

[22] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. *ACM siggraph computer graphics*, 21(4):163–169, 1987. (Cited on 3)

[23] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4460–4470, 2019. (Cited on 1, 2, 3, 4, 6, 7)

[24] Mateusz Michalkiewicz, Jhony K Pontes, Dominic Jack, Mahsa Baktashmotlagh, and Anders Eriksson. Deep level sets: Implicit surface representations for 3d shape inference. *arXiv preprint arXiv:1901.06802*, 2019. (Cited on 3)

[25] 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 *European conference on computer vision*, pages 405–421. Springer, 2020. (Cited on 4)

[26] Paritosh Mittal, Yen-Chi Cheng, Maneesh Singh, and Shubham Tulsiani. Autosdf: Shape priors for 3d completion, reconstruction and generation. *arXiv preprint arXiv:2203.09516*, 2022. (Cited on 3, 4)

[27] 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. (Cited on 3, 6)

[28] Yatian Pang, Wenxiao Wang, Francis EH Tay, Wei Liu, Yonghong Tian, and Li Yuan. Masked autoencoders for point cloud self-supervised learning. *arXiv preprint arXiv:2203.06604*, 2022. (Cited on 3)

[29] Wamiq Para, Paul Guerrero, Tom Kelly, Leonidas J Guibas, and Peter Wonka. Generative layout modeling using constraint graphs. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6690–6700, 2021. (Cited on 3)

[30] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 165–174, 2019. (Cited on 1, 3)

[31] Gaurav Parmar, Richard Zhang, and Jun-Yan Zhu. On aliased resizing and surprising subtleties in gan evaluation. In *CVPR*, 2022. (Cited on 8)

[32] Despoina Paschalidou, Amlan Kar, Maria Shugrina, Karsten Kreis, Andreas Geiger, and Sanja Fidler. Atiss: Autoregressive transformers for indoor scene synthesis. *Advances in Neural Information Processing Systems*, 34, 2021. (Cited on 3, 6)

[33] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. In *European Conference on Computer Vision*, pages 523–540. Springer, 2020. (Cited on 2, 3, 4, 6)

[34] 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. (Cited on 2, 3, 4)

[35] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in neural information processing systems*, pages 5099–5108, 2017. (Cited on 2, 3)

[36] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International Conference on Machine Learning*, pages 1530–1538, 2015. (Cited on 3)

[37] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. Octnet: Learning deep 3d representations at high resolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3577–3586, 2017. (Cited on 2)

[38] Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P Kingma. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. *arXiv preprint arXiv:1701.05517*, 2017. (Cited on 3)

[39] Yongbin Sun, Yue Wang, Ziwei Liu, Joshua Siegel, and Sanjay Sarma. Pointgrow: Autoregressively learned point cloud generation with self-attention. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 61–70, 2020. (Cited on 3, 6)

[40] Matthew Tancik, Ben Mildenhall, Terrance Wang, Divi Schmidt, Pratul P. Srinivasan, Jonathan T. Barron, and Ren Ng. Learned initializations for optimizing coordinate-based neural representations. In *CVPR*, 2021. (Cited on 3)

[41] Maxim Tatarchenko, Stephan R Richter, René Ranftl, Zhuwen Li, Vladlen Koltun, and Thomas Brox. What do single-view 3d reconstruction networks learn? In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3405–3414, 2019. (Cited on 6)

[42] Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6411–6420, 2019. (Cited on 2)

[43] Aaron Van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al. Conditional image generation with pixelcnn decoders. *Advances in Neural Information Processing Systems*, 29:4790–4798, 2016. (Cited on 3)

[44] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017. (Cited on 2, 3, 5)

[45] Aaron Van Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *International Conference on Machine Learning*, pages 1747–1756, 2016. (Cited on 3)

[46] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. (Cited on 2, 3)

[47] Xinpeng Wang, Chandan Yeshwanth, and Matthias Nießner. Sceneformer: Indoor scene generation with transformers. In *2021 International Conference on 3D Vision (3DV)*, pages 106–115. IEEE, 2021. (Cited on 3)

[48] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics (tog)*, 38(5):1–12, 2019. (Cited on 2, 3)

[49] Wenxuan Wu, Zhongang Qi, and Li Fuxin. Pointconv: Deep convolutional networks on 3d point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9621–9630, 2019. (Cited on 2)

[50] Yiheng Xie, Towaki Takikawa, Shunsuke Saito, Or Litany, Shiqin Yan, Numair Khan, Federico Tombari, James Tompkin, Vincent Sitzmann, and Srinath Sridhar. Neural fields in visual computing and beyond. *Computer Graphics Forum*, 2022. (Cited on 2, 3)

[51] Xingguang Yan, Liqiang Lin, Niloy J Mitra, Dani Lischinski, Danny Cohen-Or, and Hui Huang. Shapeformer: Transformer-based shape completion via sparse representation. *arXiv preprint arXiv:2201.10326*, 2022. (Cited on 3, 4)

[52] Xumin Yu, Lulu Tang, Yongming Rao, Tiejun Huang, Jie Zhou, and Jiwen Lu. Point-bert: Pre-training 3d point cloud transformers with masked point modeling. *arXiv preprint arXiv:2111.14819*, 2021. (Cited on 3)

[53] Biao Zhang and Peter Wonka. Training data generating networks: Shape reconstruction via bi-level optimization. In *International Conference on Learning Representations*, 2022. (Cited on 1, 3, 6)

[54] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. Point transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16259–16268, 2021. (Cited on 3)

# Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Appendix.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Appendix.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See Appendix.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendix.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A] We specify the random seed. The results are reproducible.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [Yes] See Appendix.

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [Yes]