PUNO: A Neural Operator Framework for Point Cloud Upsampling

Zijian Xiao[1], Yining Xu[1], Yingjie Huang[1], Li Yao[1,2*]

[1]School of Computer Science and Engineering, Southeast University, Nanjing 210096, China
[2]Key Laboratory of New Generation Artificial Intelligence Technology and Its Interdisciplinary Applications (Southeast University), Ministry of Education, China

Abstract

We propose PUNO, a novel deep operator-based framework for point cloud upsampling, addressing the challenge of reconstructing high-resolution geometries from sparse point clouds. PUNO generalizes the neural operators proven effective in image super-resolution to 3D point cloud upsampling. Moreover, it first designs a network for point cloud tasks to achieve vertex displacement and manifold parameterization, thereby forming a coarse geometric representation that is compatible with super-resolution neural operators. This is followed by iterative kernel integral approximations in the function space and backprojection to generate the target coordinates, fully utilizing the high-frequency information in the function space. Unlike prior work, PUNO performs transformations in both the data domain and the function domain, with the solution space containing richer basis functions, yielding finer results that mitigate the ill-posed nature of sparse data. It also benefits global continuity. Extensive experiments demonstrate its superior accuracy, robustness, and generalization ability.

Introduction

As the most common representation of 3D data, point cloud has been widely applied in various practical scenario. However, real-world point clouds are often noisy and sparse, limiting their use in downstream tasks like classification, reconstruction, and segmentation. To address this, various point cloud upsampling methods have been developed (Yu et al. 2018; Qian et al. 2020; Feng et al. 2022; Rong et al. 2024), aiming to generate clean, dense point clouds from sparse inputs while faithfully restoring geometric structures. Therefore, upsampling models can be integrated into other methods for related tasks, such as point cloud completion (Zhou et al. 2022), surface reconstruction(Ren et al. 2023), etc. Recently, neural implicit functions have been proposed to represent 3D shapes at arbitrary resolutions(Mescheder et al. 2019; Chen and Zhang 2019; Park et al. 2019; Peng et al. 2020), providing a foundation for continuous upsampling. Unlike traditional discrete representations, neural network-based representations have the signal of continuous function evaluation at specified coordinates, where the function is typically parameterized by a multilayer perceptron (MLP), thereby extending finite-resolution point clouds into continuous representations.

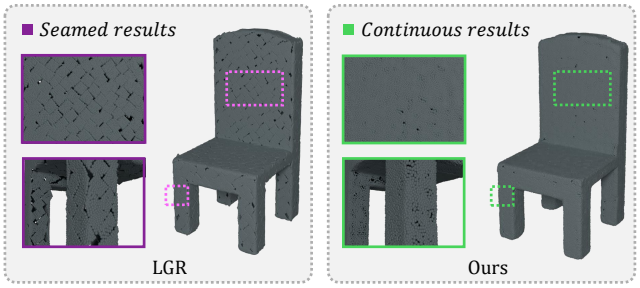

Figure 1: A comparison between our method and LGR (Ren et al. 2023). LGR is a typical method based on local parameterization. When the upsampling rate is very high (for example, the $64\times$ sampling effect shown in the figure), LGR exhibits significant patching artifacts, whereas our method maintains good global continuity.

There are mainly two types of neural network-based upsampling methods: generation-based(Qian et al. 2020; Feng et al. 2022; Yu et al. 2021) and refinement-based(He et al. 2023; Li et al. 2021; Luo and Hu 2021; Mao et al. 2022; Wang, Ang Jr, and Lee 2020). These methods can be seen as fitting the data space, essentially approximating $y = f(x)$ given (x, y). Although previous methods have carefully designed a variety of representations for f, they are all affected by the difficulty of 3D coordinate prediction, resulting in a series of issues that affect point cloud quality, such as outliers and shrinkage artifacts. Some may also lack global continuity, as shown in Figure 1. We considered whether it would be possible to perform a fitting in the functional domain as well, that is, to use a neural operator to transform functions, allowing an f_0 that is already close to the truth value to further improve its accuracy. To address this issue, we propose the PUNO upsampling framework, which employs a 2-stage shape approximation. In the first stage, the framework performs a neural approximation of the data domain using a hybrid generative-refinement approach. This approach adjusts the input distribution and learns a point-wise parametric mapping to approximate the underlying surface geometry. The second stage leverages neural operators

*Corresponding author

to approximate transformations in the function space, which is the primary distinction of PUNO from existing methods.

Recently proposed neural operator architectures (NO) (Kovachki et al. 2023; Li et al. 2020; Lu, Jin, and Karniadakis 2019) are novel in computational physics for numerically efficient PDE solvers. Based on operator theory, neural operators learn mappings between infinite-dimensional function spaces, inherently enabling continuous function evaluation and showing promising potential in many applications (Guibas et al. 2021; Hwang et al. 2022). In particular, this has also achieved success in the field of image super-resolution (Wei and Zhang 2023; Han and Zhang 2024). Typically, the neural operator consists of three components: (1) lifting, (2) iterative kernel integration, and (3) projection. Following this insight, our neural operator employs a lifting function that combines position encoding and linear layers to encode Euclidean coordinates into high-dimensional embeddings. Then it learns an attention-based neural operator in the dual space of the lifting function, which is equivalent to a kernel integral from a continuous perspective. Extensive experiments demonstrate that our method outperforms previous continuous point cloud upsampling methods in reconstruction accuracy, robustness, and generalization. In summary, our main contributions are as follows.

1. We first propose PUNO, a novel framework that combines a parameterization mapping network for robust data-space fitting, and adapts super-resolution operators from the image domain to the shape surface, enabling the point cloud upsampling task to benefit from the capabilities of neural operators. It performs transformations in the high-frequency function space, which can approximate more accurate details at arbitrary scales.
2. Our method employs a Fourier-style lifting function and adopts Galerkin-type attention to approximate kernel integrals. By leveraging mappings between high-frequency function spaces with rich basis functions, we refine the upsampling process, ensuring that the generated point clouds achieve high-fidelity results with enhanced geometric accuracy and continuity.
3. We conducted extensive experiments that demonstrate the numerical superiority of our method over state-of-the-art algorithms in restoring faithful geometric structures, achieving high-fidelity upsampling, and exhibiting robustness and generalization ability.

Related Work

Learning-based Point Cloud Upsampling

PU-Net(Yu et al. 2018) was the first point upsampling network. Subsequently, the progressive upsampling method MPU(Yifan et al. 2019) was introduced, achieving excellent performance for large upsampling factors. MetaPU(Ye et al. 2021) first upsamples to a high resolution and then downsamples to the target resolution. PU-GCN(Qian et al. 2021a) improved performance by proposing a new NodeShuffle module. Dis-PU(Li et al. 2021) employs a global refinement module at the end of the pipeline. PU-GAN(Li et al. 2019) was the first structure based on GAN(Goodfellow et al. 2014) to synthesize uniformly distributed points. The first geometry-centric method is PUGeo-Net(Qian et al. 2020), while MAFU(Qian et al. 2021b) also utilizes local differential geometric constraints. Subsequently, NePs(Feng et al. 2022) proposed using implicit neural fields to expand discrete point clouds, achieving upsampling at arbitrary scales. Grad-PU(He et al. 2023), on the other hand, models the point cloud upsampling problem as midpoint interpolation and location refinement to achieve infinite resolution. Recently, RepKPU(Rong et al. 2024) introduced a kernel point-based representation and designed a Kernel-to-Displacement paradigm for upsampling, while PUDM (Qu et al. 2024) proposes to use diffusion models for upsampling, and SPU-IMR (Nie et al. 2025) employs masks to reconstruct dense point clouds.

Neural Operator

In recent years, a new neural network architecture known as the Neural Operator has been proposed to learn invariant solutions of partial differential equations (PDEs) through infinite-dimensional operator learning. Representative works in this area include (Gupta, Xiao, and Bogdan 2021; Kovachki et al. 2023; Li et al. 2020; Lu, Jin, and Karniadakis 2019). Other works have focused more on efficiency (Rahman, Ross, and Azizzadenesheli 2022). In the field of 3D computer vision, neural operators are typically designed to extend traditional numerical methods to solve PDEs derived from geometry. Notable works include (Wang et al. 2019a), which learns non-linear operators from data for downstream applications; (Li et al. 2024), which focuses on large-scale scenes; and the recently proposed neural Laplacian operator specifically for 3D point clouds (Pang et al. 2024), which successfully solves problems such as heat diffusion, mechanical deformation, and spectral filtering on manifolds. In the field of image super-resolution, which shares many similarities with the task of upsampling, neural operators have also been proven to enrich the basis in functional space, thereby producing exciting effects(Wei and Zhang 2023). Due to their powerful mapping capabilities in the function space, neural operators have been widely applied, yet they have not been utilized for point cloud upsampling problems so far.

Methodology

Foundation. A valuable advantage of NO is that it does not require knowledge of the underlying PDE, allowing us to introduce it in the following abstract form:

$$\begin{aligned}(Lf)(x) &= y(x), \quad x \in \Omega \\ f(x) &= y_0(x), \quad x \in \partial\Omega\end{aligned}$$

Here, $f : \Omega \to \mathbb{R}^{d_f}$ is the solution function residing in the Banach space $\mathcal{F}$, and L is an operator-valued functional that maps the coefficient function of the PDE to $f^* \in \mathcal{F}^*$, the dual space of $\mathcal{F}$. Since the inverse operator L^{-1} often does not exist, NO aims to directly map the coefficients to the solution within a tolerable error. This idea of NO inspires us to consider transformations in the function space, using

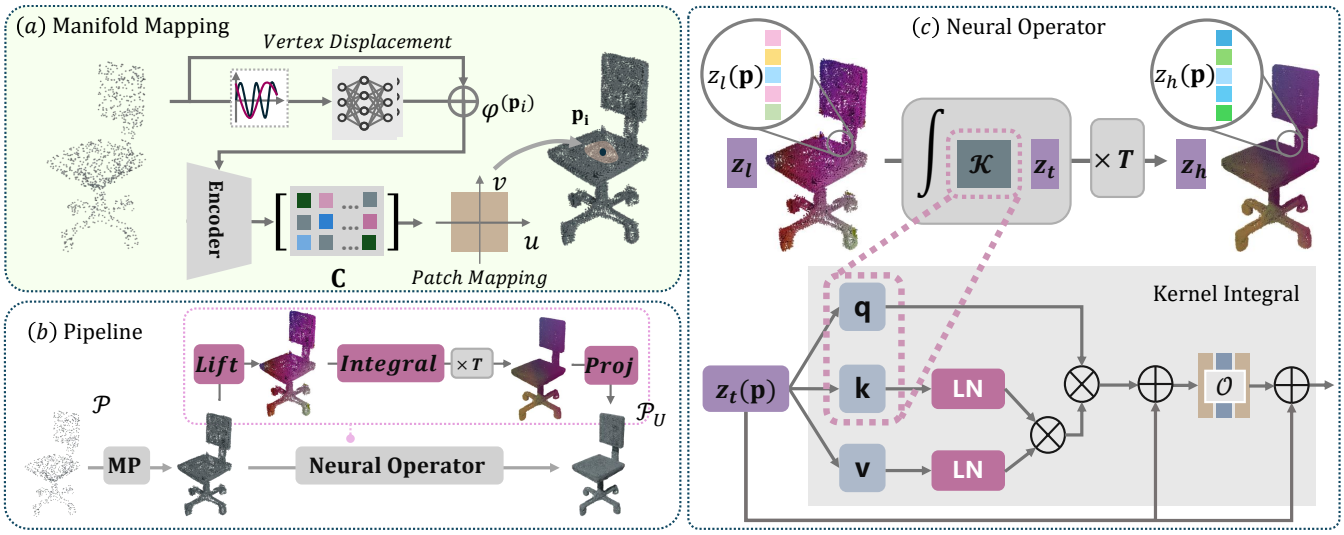

Figure 2: Illustration of the proposed PUNO framework. (a) illustrates the process of Manifold Mapping (**MP**). The vertex displacement predicted by the network is added to the original coordinates to adjust the initial distribution. Then, the parameter matrix $\mathbf{C}$ encoded by the network is optimized to generate multiple local parameterized mappings, forming a rough upsampling result that provides an initial region for the neural operator. (b) shows the architecture of the entire pipeline. After **MP**, the neural operator further optimizes the upsampling result to finally obtain a high-precision point cloud. This process includes a $Lift$ function that converts coordinate space to latent representation, kernel integral iteration, and back-projection to 3D space via the $Proj$ function. (c) details the internal operation of the neural operator, which transforms the function space through discrete kernel integration approximated by Galerkin-type Attention.

the data space fitting result $f(x) = y_0$ as the initial value for iteration and further obtaining the true value $Lf = y$ through the neural operator L.

Revisiting Super-Resolution Neural Operator in Image Processing. Super-Resolution Neural Operator (SRNO)(Wei and Zhang 2023; Han and Zhang 2024) has achieved great success in image Processing. The goal of SRNO is to learn a mapping between two infinite dimensional spaces by using a finite collection of observations of input-output image pairs. It follows a common paradigm of neural operators: lifting, kernel integration, and then back-projection onto the image grid. They use a simple convolutional encoder to replace the lifting function, and after completing the integral transformation in the image region, project back to the RGB space.

The key factors contributing to the success of SRNO and the feasibility of applying neural operators to the point cloud super-resolution task are as follows: (1) Performing transformations in function spaces that contain more high-frequency information offers a higher degree of freedom. (2) It explores the common latent basis for the entire training set and directly maps to the solution, enabling better approximation of some difficult samples. However, 3D point cloud data is much more complex and does not have a regular grid like image data. To address these issues, We adopt a more flexible Fourier-style encoding to lift 3D data to the high-frequency space and utilize manifold parameterization mappings to convert kernel integrals over regular 2D regions into integrals over the shape surface, allowing the 2D super-resolution neural operator to be extended to the more challenging 3D point cloud super-resolution task, thus achieving performance that surpasses previous point cloud upsampling methods.

Problem Statement. Given a sparse point cloud $\mathcal{P} = \{\mathbf{p}_i | \mathbf{p}_i \in \mathbb{R}^3\}_{i=1}^{N}$ sampled from the surface of a 3D model, which contains N points, and a specified upsampling ratio U, our goal is to generate a dense and uniformly distributed point cloud $\mathcal{P}_U = \{\mathbf{p}_{ir} | \mathbf{p}_{ir} \in \mathbb{R}^3\}_{i,r=1}^{N,U}$ that contains more geometric details, and these geometric structures are as close as possible to the original 3D model.

Framework Overview. Based on the above discussion, we design two modules, MP and NO, which represent the function f and the neural operator L mentioned above, respectively. **MP** is a manifold parameterization mapping on the input point cloud to obtain a set of local regions, denoted as the coarse regions $\mathcal{S}_C$:

$$\mathcal{S}_C = \mathbf{MP}(\mathcal{P}) \tag{1}$$

Next, we use the neural operator **NO** to map the coarse regions to refined regions, and the points on the coarse regions are correspondingly mapped to fine-tuned points. This process can be formally described as follows:

$$\mathcal{S}_U = \mathbf{NO}(\mathcal{S}_C) \tag{2}$$

By taking a dense sampling $\mathcal{P}_C$ on $\mathcal{S}_C$, we can derive $\mathcal{P}_U$:

$$\mathcal{P}_U = \mathbf{NO}(\mathcal{P}_C) \tag{3}$$

MP and **NO** constitute our entire PUNO framework, which we will introduce in detail below.

Manifold Parameterization Mapping

Vertex Displacement Function. The distribution of the input point cloud is typically random and noisy. Therefore, before discussing manifold parameterization, we first introduce a displacement network to transform the sampling of the input data into an approximate uniform sampling. Intuitively, the displacement value of each input point $\mathbf{p}$ should be related to its relative position with nearby points $\mathbf{p}_k$ and the overall shape $\Omega_k(\mathbf{p})$ they form. Hence, we first compute a k-nearest neighbor graph from the input point cloud and embed $\mathbf{p}$ and $\mathbf{p}_k$ through an MLP $m_1(\cdot)$ to obtain the per-point features $\mathbf{F}_k(\mathbf{p})$ of all neighbors. We then extract the mean and maximum features to obtain the features $\mathbf{F}(\mathbf{p})$ of the entire region $\Omega_k(\mathbf{p})$:

$$\mathbf{F}_k(\mathbf{p}) = m_1(\mathbf{p} - \mathbf{p}_k) \tag{4}$$

$$\mathbf{F}(\mathbf{p}) = \max\{\mathbf{F}_k(\mathbf{p})\}_{k=1}^{K} \oplus \operatorname{avg}\{\mathbf{F}_k(\mathbf{p})\}_{k=1}^{K} \tag{5}$$

where $\oplus$ stands for the concatenation operator. To enhance the neural network's ability to capture high-frequency information, we also compute the position encoding of the point cloud derived from positional encoding(Mildenhall et al. 2020):

$$\mathbf{Enc}(\mathbf{p}) = (\sin 2^0\mathbf{p}, \cos 2^0\mathbf{p}, ..., \sin 2^L\mathbf{p}, \cos 2^L\mathbf{p}) \tag{6}$$

Then, we concatenate the position encoding with the global features of the shape $\Omega(\mathbf{p})$ and feed them into another MLP $m_2(\cdot)$ to decode the per-point displacement vectors. The final optimized input is:

$$\sigma(\mathbf{p}) = \mathbf{p} + m_2(\mathbf{F}(\mathbf{p}) \oplus \mathbf{Enc}(\mathbf{p})) \tag{7}$$

In this stage, we use the uniform point cloud generated by Poisson Disk sampling as the ground truth to train the simple displacement network.

Manifold Parameter Mapping. $\mathbf{MP}$ accomplishes two things: (1) providing a good initial value for $\mathbf{NO}$; (2) transforming the 2D integral region relied on by the image NO to the shape surface. Based on the fact that 3D manifold surfaces are homeomorphic to a 2D plane, we can fix a 2D planar region $\mathcal{D} \in \mathbb{R}^2$ and construct different parameterization mappings to represent various 3D local patches. In classical geometric analysis, this technique is commonly called the Monge patch, which describes a surface patch $\Gamma : \mathcal{D} \in \mathbb{R}^2 \to \mathbb{R}^3$ as a polynomial $h(u, v)$ over the tangent plane at $\mathbf{p}$ that is given by $[u, v, h(u, v)]$ in the local coordinate system. When transformed into the global coordinate system, this local patch can generally be expressed in the form of $[x(u, v), y(u, v), z(u, v)]$, where $x(u, v)$, $y(u, v)$, and $z(u, v)$ are all bivariate polynomials in terms of u and v, thus parameterizing the 3D coordinates using u and v. Specifically, we use quadratic polynomial surfaces as the basis elements for local approximation.

$$\varphi^{(\mathbf{p})}(\mathbf{u}) = \mathbf{p} + \mathbf{C}\Phi(\mathbf{u}) \tag{8}$$

Here, $\mathbf{C}\Phi(\mathbf{u})$ represents the local parametric surface patch and $\mathbf{p}$ is the offset. $\mathbf{C} \in \mathbb{R}^{3\times 6}$ is the coefficient matrix. $\mathbf{u} = [u, v]^\top \in \mathcal{D}$ represents the 2D coordinates on the parameter domain. $\Phi(\mathbf{u}) = \left[1, u, v, u^2, uv, v^2\right]^\top$ is the polynomial basis. Therefore, the final coarse surface region is constructed as:

$$\mathcal{S}_C = \bigcup_{\mathbf{p}\prime \in \sigma(\mathcal{P})} \bigcup_{\mathbf{u}\in\mathcal{D}} \varphi^{(\mathbf{p}\prime)}(\mathbf{u}) \tag{9}$$

In this representation, the coarse approximation of the local patch is transformed into an optimization problem for the coefficient matrix $\mathbf{C}$. We directly extract shape features from the point cloud and then use an MLP to predict the per-point matrix $\mathbf{C}$ (see Figure 2.).

Neural Functional Operator

Lifting Function. Neural operators derived from partial differential equations typically use simple pointwise functions to extend input channels. For the point cloud upsampling problem, due to the complexity of spatial shapes, we need to consider integrating information from different frequencies and implicitly embedding it into a high-dimensional space. Therefore, we first utilizes the position encoding in Equation 6 to introduce Fourier style bases, and then extends the dimensions through a linear layer, thereby mapping the position information from the point cloud to a high-dimensional space containing different frequencies. We denote the lifting mapping as $z(\cdot) : \mathbb{R}^3 \to \mathbb{R}^d$, where d is the extended dimension.

Kernel Integral. The neural operator will directly act on the function space as a kernel integral. We aim to map the lifting function z_l in the low-resolution coarse region $\mathcal{S}_C$ to the function z_h in the high-resolution region, and then project it back into the high-resolution region $\mathcal{S}_U$. The kernel integral operator can be approximated using the Monte-Carlo method (omitting the constant multiple of the area scale):

$$K \circ z_l(\mathbf{p}) = \int_{\mathcal{S}_C} \mathcal{K}(\mathbf{p}, \mathbf{p}')z_l(\mathbf{p}')\mathrm{d}\mathbf{p}' \approx \sum_{i=1}^{NU} \mathcal{K}(\mathbf{p}, \mathbf{p}_i)z_l(\mathbf{p}_i) \tag{10}$$

This process can be iteratively performed, equivalent to decomposing the operator K into a composition of multiple operators $K = K_m \circ \cdots \circ K_1$, where $z_{t+1} = K_t \circ z_t$. We can design each integral operator as a block in the network. A global approach is to mimic traditional attention mechanisms by designing the matrix $\mathbf{Ker}$ (where the element in the i-th row and the j-th column $\mathbf{Ker}_{ij} = \mathcal{K}(\mathbf{p}_i, \mathbf{p}_j)$) in the form of $\mathbf{Ker}(\mathbf{p}_i, \mathbf{p}_j) = softmax(\mathbf{q}(\mathbf{p}_i)\mathbf{k}(\mathbf{p}_j))$. Here, $\mathbf{q} = \mathbf{W}_q z$, $\mathbf{k} = \mathbf{W}_k z$, $\mathbf{v} = \mathbf{W}_z z$, where $\mathbf{W}_q, \mathbf{W}_k, \mathbf{W}_v \in \mathbb{R}^d$ correspond to the query, key, and value functions, respectively. However, this implies that for each iteration, we need to learn a matrix $NU \times NU$, resulting in a complexity of $O((NU)^2 d)$. That is unacceptable for point cloud upsampling because the number of points NU in a densely sampled shape is typically in the tens of thousands. An intuitive modification to reduce complexity is to design the operator with local support, which means that only a few points near the corresponding query point have weights, thus compressing the matrix $\mathbf{Ker}$ into a sparse format. However, this would cause the operator to lose its global nature and reduce accuracy (see Table 5).

To balance this issue, we follow SRNO(Wei and Zhang 2023) to adopt a Galerkin-type attention operator to design

dataset		ShapeNet				PU-GAN				PU1K			
metrics	Post	CD↓	HD↓	P2F avg↓	P2F std↓	CD↓	HD↓	P2F avg↓	P2F std↓	CD↓	HD↓	P2F avg↓	P2F std↓
PU-Net(Yu et al. 2018)	✗	2.837	27.53	10.39	17.93	1.513	18.6	31.33	51.47	1.14	15.15	22.44	47.19
PuGeo(Qian et al. 2020)	✗	0.585	6.522	4.454	3.628	0.663	6.742	5.404	4.407	0.84	10.488	4.533	3.817
NePs(Feng et al. 2022)	✔	0.586	6.84	4.334	4.176	<u>0.531</u>	<u>6.034</u>	4.492	3.933	0.734	10.384	4.519	4.592
Grad-PU(He et al. 2023)	✔	0.764	12.634	6.607	7.573	0.784	10.017	6.801	8.784	0.933	14.478	6.36	9.371
LGR(Ren et al. 2023)	✗	<u>0.576</u>	6.469	3.142	3.208	0.596	6.638	<u>3.293</u>	<u>3.791</u>	0.815	10.539	3.544	**3.441**
RepKPU(Rong et al. 2024)	✗	0.592	<u>5.976</u>	**1.275**	**2.634**	0.544	7.116	3.473	4.233	<u>0.724</u>	<u>9.546</u>	**2.241**	3.698
SPU-IMR(Nie et al. 2025)	✔	0.701	8.109	3.607	3.952	0.572	8.769	4.537	4.666	0.827	11.761	4.311	4.844
Ours	✗	**0.524**	**5.446**	<u>2.04</u>	<u>2.789</u>	**0.437**	**5.009**	**2.758**	**3.204**	**0.693**	**8.896**	<u>2.664</u>	<u>3.622</u>

Table 1: Quantitative comparisons of different methods. CD is scaled by a factor of 10^{-4}, while HD and P2F are scaled by a factor of 10^{-3}. '✗' indicates no post-processing is required, while '✔' means post-processing is needed. The best and Second-best results are highlighted in bold and underlined, respectively.

our kernel integral. As shown in Figure 2, the Galerkin-type attention is linearized, removing the $softmax$ layer, thus allowing the inner product order to be transformed and simplifying the complexity to $O(NUd^2)$. We define $\mathbf{z} = (z(\mathbf{p}_1), ..., z(\mathbf{p}_{NU}))^\top \in \mathbb{R}^{NU\times d}$ and let $\mathbf{Q} = \mathbf{z}\mathbf{W}_q$, $\mathbf{K} = \mathbf{z}\mathbf{W}_k$,and $\mathbf{V} = \mathbf{z}\mathbf{W}_v$. The columns of $\mathbf{Q}/\mathbf{K}/\mathbf{V}$ contain the vector representations of the learned basis functions, which span certain subspaces of the latent representation Hilbert space.

$$\mathbf{z}' = \mathbf{Q}(\tilde{\mathbf{K}}^\top\tilde{\mathbf{V}})/NU \tag{11}$$

Here, $\tilde{\mathbf{K}} = \mathbf{LN}(\mathbf{K})$,$\tilde{\mathbf{V}} = \mathbf{LN}(\mathbf{V})$ with $\mathbf{LN}(\cdot)$ denoting the normalization layer. We use the subscript to denote the column of the matrix, and it can be seen that the component-wise inner product form is

$$\mathbf{z}'_j = \frac{1}{NU}\sum_{l=1}^{d}\langle\tilde{\mathbf{K}}_l^\top, \tilde{\mathbf{V}}_j\rangle\mathbf{Q}_l \tag{12}$$

From Equation 10 and 12, it can be seen that our NO iteration process on the point cloud is the update of the function basis. The l-th row of $\tilde{\mathbf{K}}^\top\tilde{\mathbf{V}}$ contains the coefficients of the linear combination of the basis vectors $\{\mathbf{Q}_l\}_{l=1}^d$ to form the output $\mathbf{z}'$. As a result, the global correlation is reflected in the components of $\tilde{\mathbf{K}}^\top\tilde{\mathbf{V}}$. The linear Galerkin-type attention can achieve a quasi-optimal approximation within the current approximation space spanned by the columns of $\mathbf{Q}$ (Cao 2021) and provide global aggregation for the output z at each sampling point, which further ensures continuity.

However, $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$ primarily depend on the input point cloud $\mathcal{P}_C$ and the lifting functions z_l defined on it. We also wish to introduce a basis that includes additional useful information that is not contained in $z(\mathbf{p})$, to enrich the high-resolution point cloud. Therefore, we introduce an FFN $\mathcal{O}$ with a non-linear layer at the positions concatenated K_t (as shown in Figure 2), which continuously enriches the basis during the iteration process. Thus, our final iteration of the neural operator (NO) is described by:

$$z_{t+1}(\mathbf{p}) = z_t(\mathbf{p}) + \mathcal{O}(K_t \circ z_t(\mathbf{p}) + z_t(\mathbf{p})) \tag{13}$$

Finally, $\mathbf{z}_h$ is projected back into Euclidean space through a linear MLP. Our experimental results have validated that, despite the significant reduction in complexity achieved by implementing kernel integration using a Galerkin-type attention mechanism, it is still able to converge to desirable results.

metrics	CD ↓		HD ↓		P2F ↓	
method	mean	media	mean	media	mean	std
PU-Net	2.714	1.869	40.2	29.6	12.97	18.33
PuGeo	0.684	0.460	8.08	6.50	7.517	6.447
NePs	<u>0.609</u>	0.489	7.55	6.04	<u>7.491</u>	6.226
Grad-Pu	0.871	0.665	14.1	11.1	8.634	9.251
LGR	0.895	0.696	8.80	7.33	7.858	5.826
RepKpu	0.651	<u>0.426</u>	<u>7.28</u>	<u>5.60</u>	7.539	6.061
SPU-IMR	0.838	0.638	8.81	7.28	<u>6.199</u>	<u>5.332</u>
Ours	**0.597**	**0.369**	**7.1**	**5.44**	**4.004**	**4.111**

Table 2: Quantitative comparisons of different methods with Gaussian noise (mean 0, std 0.01). CD is scaled by a factor of 10^{-4}, while HD and P2F are scaled by a factor of 10^{-3}. The best and Second-best results are highlighted in bold and underlined, respectively.

Consequently, PUNO can effectively leverage all available information without relying on local support, thereby significantly maintaining good global continuity. Meanwhile, the rich basis in the functional space enables PUNO to surpass existing upsampling methods in terms of accuracy.

Experiments

Implementation Details

All experiments for PUNO are conducted using PyTorch. We train the model on a 40GB Nvidia A100 GPU using the Adam optimizer. The batch size is set to 32, and the upsampling rate U is set to $16\times$. The initial learning rate is set to the maximum value of $lr_{max} = 10^{-4}$, then decreased gradually to a minimum of $lr_{min} = 5 \times 10^{-6}$ using a cosine decay schedule with a period of 10 epochs. The dimension of the vertex displacement MLP is 64, the exponent for position encoding is $L = 4$, and the knn size is 8. For parameter mapping, we use the DGCNN(Wang et al. 2019b) backbone. The hidden layer dimension is 128, and the knn size is also 8. the 2D parameter domain is chosen as a fixed square region $\mathcal{D} = [-0.1, 0.1] \times [-0.1, 0.1]$. The Galerkin

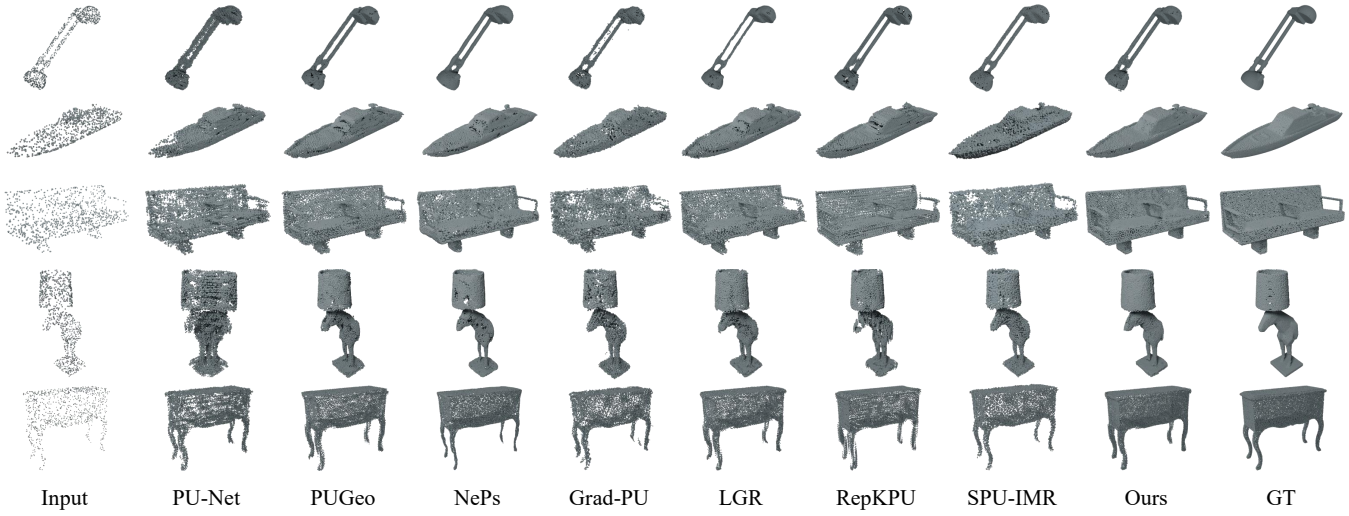

Figure 3: 16× point cloud upsampling results with input size of 1024. We recommend zooming in to view more details.

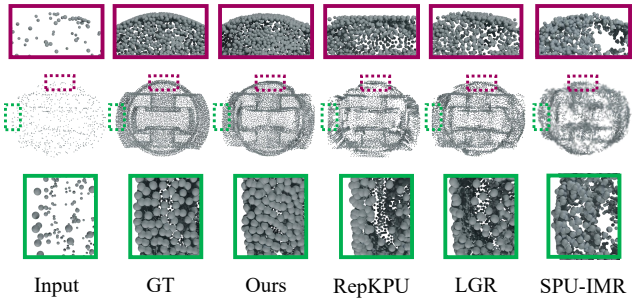

Figure 4: Visualization 16× results with unseen data. The color boxes correspond to enlarged local details.

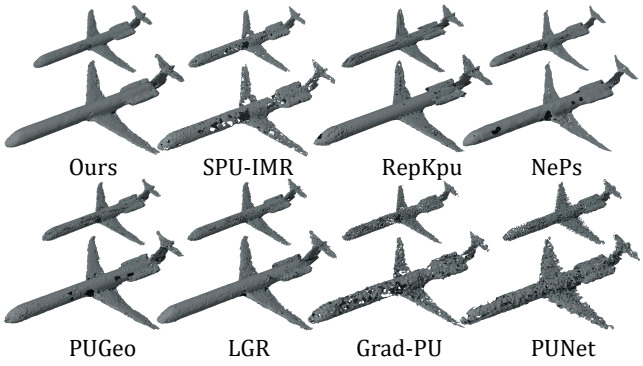

Figure 5: Visual evaluation of impacts on surface reconstruction.

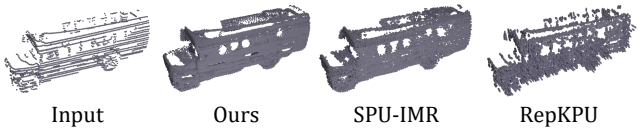

Figure 6: Result on LiDAR data from KITTI dataset.

attention mechanism has 8 heads, and the hidden layer dimension is 128. We use the Chamfer Distance as the loss function, which also serves as one of the evaluation metrics. In the main training procedure, we adopt a 3-stage training process. First, we train the vertex displacement model with sparse point clouds for 50 epochs, then train the manifold parameter mapping for 300 epochs, and finally train the neural Galerkin attention network for 300 epochs. Following convention, we adopt three evaluation metrics: Chamfer Distance (CD)(Fan, Su, and Guibas 2017), Hausdorff Distance (HD)(Berger et al. 2013), and Point-to-Surface Distance (P2F)(Erler et al. 2020).

Comparison Result

We compared our PUNO with state-of- the-art methods, including PU-Net(Yu et al. 2018), PUGeo(Qian et al. 2020), NePs(Feng et al. 2022), Grad-PU(He et al. 2023), LGR(Ren et al. 2023), RepKPU(Rong et al. 2024) and SPU-IMR(Nie et al. 2025). We mainly follow (Nie et al. 2025; Ren et al. 2023) for training and testing on the ShapeNet dataset, and we also follow (Rong et al. 2024; He et al. 2023; Feng et al. 2022; Qian et al. 2020) for evaluation on PU-GAN and PU1K. As listed in Table 1, our method achieves optimal metrics, while we do not need post-processing techniques such as oversampling followed by the FPS algorithm. Figure 3 shows the visualization results in the ShapeNet dataset. Visually, the results of PU-Net are generally disordered, while Grad-PU exhibits regional clustering in sampling. SPU-IMR, NePs, PUGeo, and LGR all show varying degrees of shape shrinkage. Although RepKPU generates results with high precision, the distribution is rather flat and exhibits a more noticeable arrangement pattern. In contrast, the shapes upsampled from sparse point clouds using our method achieve the highest fidelity, with a uniform distribution and strong global continuity. Other methods, by comparison, produce surfaces that are less continuous and may

ratio U	$8\times$	$16\times$
Ours+Softmax Attention	27.6 GiB	$\gg$ 40 GiB
Ours+Galerkin	**2.0 GiB**	**2.1 GiB**

Table 3: Comparison of memory usage among different approximation methods of the integral kernel during inference on a single example.

metrics	CD↓	HD↓	P2F↓
lin.	0.547	5.591	2.498
cub.	0.541	5.635	2.732
Ours	**0.524**	**5.446**	**2.04**

Table 4: Comparison results of different modes of **MP**. "lin." denotes linear mapping, and "cub." denotes cubic polynomial mapping. CD is scaled by a factor of 10^{-4}, while HD and P2F are scaled by a factor of 10^{-3}.

suffer from shape shrinkage or dispersion.

We also compared the reconstruction results to illustrate the importance of good continuity and uniform point cloud distribution for downstream tasks. We employ the same Ball-pivoting algorithm (Bernardini et al. 2002) as in (Rong et al. 2024) for surface reconstruction. As shown in Figure 5, our point cloud achieves the best reconstruction results.

Robustness

We add Gaussian noise with a standard deviation of $\tau = 0.01$ to the input point clouds and retrain our network, as well as all comparison methods. As listed in Table 2, our upsampling framework still achieves the best performance. We also conducted tests on the real-scan dataset KITTI(Geiger et al. 2013) (see Figure 6), and our method also demonstrated good robustness.

Generalization

We further evaluate the generalizability of our neural operator framework by testing on unseen data. Our geometry-aware framework generalizes well to unseen categories. The visualization results are shown in Tables 4. For CAD parts (Figure 4), our method generates point clouds with better continuity, higher fidelity, and uniform distribution.

Ablation Study

We performed an ablation study on ShapeNet to analyze our framework. Table 5 shows the results without the vertex displacement function and neural operator, as well as with different types of implementation of the neural operator.

Ablation study of the vertex displacement function. Removing the vertex displacement network significantly reduces upsampling accuracy. That is because the vertex displacement function adjusts random and noisy inputs to an approximate clean Poisson disk sampling distribution, reducing non-uniformity in the generated parametric surface.

Ablation study of the neural kernel integral operator. The neural kernel integral operator is essential for functional domain transformations in our framework. Removing

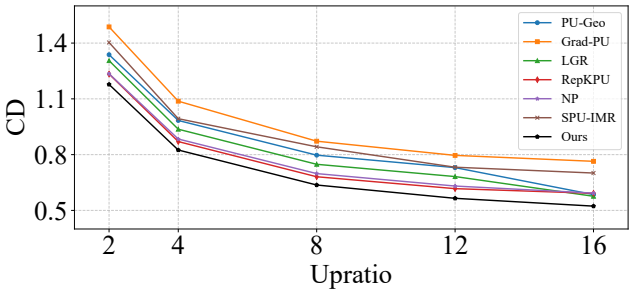

Figure 7: Comparison results of different upsampling rates on the ShapeNet dataset. CD is scaled by a factor of 10^{-4}

metrics	CD↓	HD↓	P2F↓
w/o σ	0.615	6.906	7.892
w/o NO	0.555	5.544	3.632
Ours+Local	0.538	6.192	3.315
Ours+Galerkin	**0.524**	**5.446**	**2.04**

Table 5: Ablation comparisons on the ShapeNet dataset. CD is scaled by a factor of 10^{-4}, while HD and P2F are scaled by a factor of 10^{-3}. Here, σ is the vertex displacement network. **NO** is the neural operator module. **Local** refers to the implementation of local perception, and **Galerkin** represents the implementation in this paper.

it degrades our method to data-space fitting and reduces upsampling accuracy, underscoring its importance. Replacing Galerkin-type attention with a k-nearest neighbors ($k = 16$) local-based approach results in lower accuracy. Table 3 also presents two globally continuous approximations. Compared with softmax attention, the Galerkin-type approximation saves significantly more memory.

Ablation study of different upsampling rates. We conducted experiments using upsampling rates ranging from $2\times$ to $16\times$, as illustrated in Figure 7. Our method achieves superior performance at various sampling rates.

Ablation studies of manifold parameterization with different polynomial orders. As shown in Table 4, the second-order polynomial achieves the best performance. We believe that the first-order polynomial lacks precision, while the third-order polynomial tends to overfit.

Conclusion

We introduce PUNO, a continuous point cloud upsampling framework using neural operators. PUNO generalizes across discretization levels by learning mappings between finite-dimensional function spaces. It approximates coarse regions from sparse inputs using a lightweight neural network with vertex displacement and manifold parameter mappings. These regions are then lifted to higher-dimensional spaces for iterative kernel integration via a linear attention operator, approximating the target function similarly to the Petrov-Galerkin projection. Experiments demonstrate that PUNO outperforms existing methods in accuracy, robustness, generalization, and global shape continuity.

Acknowledgments

This research work was supported in part by the Big Data Computing Center of Southeast University and by the Key Laboratory of New Generation Artificial Intelligence Technology and Its Interdisciplinary Applications (Southeast University), Ministry of Education, China.

References

Berger, M.; Levine, J. A.; Nonato, L. G.; Taubin, G.; and Silva, C. T. 2013. A benchmark for surface reconstruction. *ACM Transactions on Graphics (TOG)*, 32(2): 1–17.

Bernardini, F.; Mittleman, J.; Rushmeier, H.; Silva, C.; and Taubin, G. 2002. The ball-pivoting algorithm for surface reconstruction. *IEEE transactions on visualization and computer graphics*, 5(4): 349–359.

Cao, S. 2021. Choose a transformer: Fourier or galerkin. *Advances in neural information processing systems*, 34: 24924–24940.

Chen, Z.; and Zhang, H. 2019. Learning implicit fields for generative shape modeling. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 5939–5948.

Erler, P.; Guerrero, P.; Ohrhallinger, S.; Mitra, N. J.; and Wimmer, M. 2020. Points2surf learning implicit surfaces from point clouds. In *European Conference on Computer Vision*, 108–124. Springer.

Fan, H.; Su, H.; and Guibas, L. J. 2017. A point set generation network for 3d object reconstruction from a single image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 605–613.

Feng, W.; li, J.; Cai, H.; Luo, X.; and Zhang, J. 2022. Neural Points: Point Cloud Representation with Neural Fields for Arbitrary Upsampling. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Geiger, A.; Lenz, P.; Stiller, C.; and Urtasun, R. 2013. Vision meets robotics: The kitti dataset. *The international journal of robotics research*, 32(11): 1231–1237.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative adversarial nets. *Advances in neural information processing systems*, 27.

Guibas, J.; Mardani, M.; Li, Z.; Tao, A.; Anandkumar, A.; and Catanzaro, B. 2021. Efficient token mixing for transformers via adaptive fourier neural operators. In *International Conference on Learning Representations*.

Gupta, G.; Xiao, X.; and Bogdan, P. 2021. Multiwavelet-based operator learning for differential equations. *Advances in neural information processing systems*, 34: 24048–24062.

Han, L.; and Zhang, X. 2024. Scalable Super-Resolution Neural Operator. In *Proceedings of the 32nd ACM International Conference on Multimedia*, 10036–10045.

He, Y.; Tang, D.; Zhang, Y.; Xue, X.; and Fu, Y. 2023. Gradpu: Arbitrary-scale point cloud upsampling via gradient descent with learned distance functions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 5354–5363.

Hwang, R.; Lee, J. Y.; Shin, J. Y.; and Hwang, H. J. 2022. Solving pde-constrained control problems using operator learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, 4504–4512.

Kovachki, N.; Li, Z.; Liu, B.; Azizzadenesheli, K.; Bhattacharya, K.; Stuart, A.; and Anandkumar, A. 2023. Neural operator: Learning maps between function spaces with applications to pdes. *Journal of Machine Learning Research*, 24(89): 1–97.

Li, R.; Li, X.; Fu, C.-W.; Cohen-Or, D.; and Heng, P.-A. 2019. Pu-gan: a point cloud upsampling adversarial network. In *Proceedings of the IEEE/CVF international conference on computer vision*, 7203–7212.

Li, R.; Li, X.; Heng, P.-A.; and Fu, C.-W. 2021. Point cloud upsampling via disentangled refinement. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 344–353.

Li, Z.; Kovachki, N.; Azizzadenesheli, K.; Liu, B.; Bhattacharya, K.; Stuart, A.; and Anandkumar, A. 2020. Fourier neural operator for parametric partial differential equations, arXiv. *arXiv preprint arXiv:2010.08895*.

Li, Z.; Kovachki, N.; Choy, C.; Li, B.; Kossaifi, J.; Otta, S.; Nabian, M. A.; Stadler, M.; Hundt, C.; Azizzadenesheli, K.; et al. 2024. Geometry-informed neural operator for large-scale 3d pdes. *Advances in Neural Information Processing Systems*, 36.

Lu, L.; Jin, P.; and Karniadakis, G. E. 2019. Deeponet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. *arXiv preprint arXiv:1910.03193*.

Luo, S.; and Hu, W. 2021. Score-based point cloud denoising. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 4583–4592.

Mao, A.; Du, Z.; Hou, J.; Duan, Y.; Liu, Y.-j.; and He, Y. 2022. Pu-flow: A point cloud upsampling network with normalizing flows. *IEEE Transactions on Visualization and Computer Graphics*, 29(12): 4964–4977.

Mescheder, L.; Oechsle, M.; Niemeyer, M.; Nowozin, S.; and Geiger, A. 2019. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 4460–4470.

Mildenhall, B.; Srinivasan, P. P.; Tancik, M.; Barron, J. T.; Ramamoorthi, R.; and Ng, R. 2020. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *ECCV*.

Nie, Z.; Wu, Q.; Lv, C.; Quan, S.; Qi, Z.; Wang, M.; and Yang, J. 2025. SPU-IMR: Self-supervised Arbitrary-scale Point Cloud Upsampling via Iterative Mask-recovery Network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, 6236–6244.

Pang, B.; Zheng, Z.; Li, Y.; Wang, G.; and Wang, P.-S. 2024. Neural Laplacian Operator for 3D Point Clouds. *ACM Transactions on Graphics (TOG)*, 43(6): 1–14.

Park, J. J.; Florence, P.; Straub, J.; Newcombe, R.; and Lovegrove, S. 2019. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of*

the IEEE/CVF conference on computer vision and pattern recognition, 165–174.

Peng, S.; Niemeyer, M.; Mescheder, L.; Pollefeys, M.; and Geiger, A. 2020. Convolutional occupancy networks. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16*, 523–540. Springer.

Qian, G.; Abualshour, A.; Li, G.; Thabet, A.; and Ghanem, B. 2021a. Pu-gcn: Point cloud upsampling using graph convolutional networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 11683–11692.

Qian, Y.; Hou, J.; Kwong, S.; and He, Y. 2020. PUGeo-Net: A Geometry-Centric Network for 3D Point Cloud Upsampling. In *European Conference on Computer Vision*, 752–769. Springer.

Qian, Y.; Hou, J.; Kwong, S.; and He, Y. 2021b. Deep magnification-flexible upsampling over 3d point clouds. *IEEE Transactions on Image Processing*, 30: 8354–8367.

Qu, W.; Shao, Y.; Meng, L.; Huang, X.; and Xiao, L. 2024. A conditional denoising diffusion probabilistic model for point cloud upsampling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 20786–20795.

Rahman, M. A.; Ross, Z. E.; and Azizzadenesheli, K. 2022. U-no: U-shaped neural operators. *arXiv preprint arXiv:2204.11127*.

Ren, S.; Hou, J.; Chen, X.; He, Y.; and Wang, W. 2023. Geoudf: Surface Reconstruction from 3D Point Clouds via Geometry-guided Distance Representation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 14214–14224.

Rong, Y.; Zhou, H.; Xia, K.; Mei, C.; Wang, J.; and Lu, T. 2024. RepKPU: Point Cloud Upsampling with Kernel Point Representation and Deformation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 21050–21060.

Wang, X.; Ang Jr, M. H.; and Lee, G. H. 2020. Cascaded refinement network for point cloud completion. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 790–799.

Wang, Y.; Kim, V.; Bronstein, M.; and Solomon, J. 2019a. Learning geometric operators on meshes. In *Representation Learning on Graphs and Manifolds 2019 (ICLR workshop)*.

Wang, Y.; Sun, Y.; Liu, Z.; Sarma, S. E.; Bronstein, M. M.; and Solomon, J. M. 2019b. Dynamic Graph CNN for Learning on Point Clouds. *ACM Trans. Graph.*, 38(5).

Wei, M.; and Zhang, X. 2023. Super-Resolution Neural Operator. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Ye, S.; Chen, D.; Han, S.; Wan, Z.; and Liao, J. 2021. Meta-PU: An arbitrary-scale upsampling network for point cloud. *IEEE transactions on visualization and computer graphics*, 28(9): 3206–3218.

Yifan, W.; Wu, S.; Huang, H.; Cohen-Or, D.; and Sorkine-Hornung, O. 2019. Patch-based progressive 3d point set upsampling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 5958–5967.

Yu, L.; Li, X.; Fu, C.-W.; Cohen-Or, D.; and Heng, P.-A. 2018. Pu-net: Point cloud upsampling network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2790–2799.

Yu, X.; Rao, Y.; Wang, Z.; Liu, Z.; Lu, J.; and Zhou, J. 2021. Pointr: Diverse point cloud completion with geometry-aware transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, 12498–12507.

Zhou, H.; Cao, Y.; Chu, W.; Zhu, J.; Lu, T.; Tai, Y.; and Wang, C. 2022. Seedformer: Patch seeds based point cloud completion with upsample transformer. In *European conference on computer vision*, 416–432. Springer.