Invertible Residual Neural Networks with Conditional Injector and Interpolator for Point Cloud Upsampling

Aihua Mao[1†], **Yaqi Duan**[1†*], **Yu-Hui Wen**[2*], **Zihui Du**[1], **Hongmin Cai**[1] and **Yong-Jin Liu**[3]
[1]South China University of Technology
[2]Beijing Jiaotong University
[3]Tsinghua University
{ahmao, hmcai}@scut.edu.cn, {csduanyaqi, csusami}@mail.scut.edu.cn, yhwen1@bjtu.edu.cn, liuyongjin@tsinghua.edu.cn

Abstract

Point clouds obtained by LiDAR and other sensors are usually sparse and irregular. Low-quality point clouds have serious influence on the final performance of downstream tasks. Recently, a point cloud upsampling network with normalizing flows has been proposed to address this problem. However, the network heavily relies on designing specialized architectures to achieve invertibility. In this paper, we propose a novel invertible residual neural network for point cloud upsampling, called PU-INN, which allows unconstrained architectures to learn more expressive feature transformations. Then, we propose a conditional injector to improve nonlinear transformation ability of the neural network while guaranteeing invertibility. Furthermore, a lightweight interpolator is proposed based on semantic similarity distance in the latent space, which can intuitively reflect the interpolation changes in Euclidean space. Qualitative and quantitative results show that our method outperforms the state-of-the-art works in terms of distribution uniformity, proximity-to-surface accuracy, 3D reconstruction quality, and computation efficiency.

1 Introduction

In recent years, with the development of depth data sensors, point clouds have become one of the most popular data types representing the 3D world. However, the collected point cloud data is usually sparse and irregular due to the complexity of real-world illumination and materials. Low-quality point clouds have a direct impact on the performance of downstream tasks, e.g., self-driving cars, robotics, surface reconstruction, and medical analysis, etc. Therefore, upsampling sparse and irregular point clouds into dense and uniform data has attracted increasing attention in computer vision and computer graphics.

Currently, existing upsampling methods, e.g., PU-Net [Yu *et al.*, 2018], PU-GAN [Li *et al.*, 2019], PU-GCN [Qian *et al.*, 2021a], DIS-PU [Li *et al.*, 2021] and PC2-PU [Long *et al.*, 2022], usually perform feature extraction to encode the feature of the point cloud into a latent code. Then, these methods duplicate the latent code with some copies to increase the number of points, and finally reconstruct the 3D coordinates of points in Euclidean space by decoding from the duplicated latent code. Specifically, the coordinate reconstruction is constrained by evaluation metrics, such as uniformity and smoothness. The feature extraction and the coordinate reconstruction need to learn the parameters separately, and guarantee the relevance between the encoding and decoding processes. Different from these methods, a normalizing flow network architecture called PU-Flow [Mao *et al.*, 2022] has been proposed for point cloud upsampling recently. The normalizing flow network parameterizes a bijective mapping between a complex distribution and a simple distribution. Thanks to the invertibility, PU-Flow achieves no information loss in the encoding and decoding processes. However, this method relies on designing structured Jacobians and specialized architectures, which have a significant impact on the model performance [Dinh *et al.*, 2016; Miyato *et al.*, 2018].

To achieve unconstrained Jacobians and architectures, in this paper we construct an invertible network with residual blocks by introducing lipschitz constraints [Chen *et al.*, 2019]. To further improve the expressiveness of the network, we design a conditional injector, which includes a feature extractor module and a fusion module. The feature extractor module performs adaptive encoding of global and local features of sparse point clouds. The feature fusion module uses the attention mechanism [Vaswani *et al.*, 2017] to aggregate global features with local features and outputs enhanced features. The enhanced features are injected into the main architecture, to enhance the nonlinear variability.

For upsampling, we also design a lightweight interpolator to scale the points, based on semantic similarity distance in the latent space modeled by the invertible network. The interpolation in the latent space can be reflected in the Euclidean space through the reverse mapping of the invertible network. Furthermore, we selected 25 complex models from Sketchfab, called the PU25 dataset, to quantitatively evaluate the upsampling point set. We compare our results with state-of-the-art (SOTA) optimization and learning-based methods. Benefiting from our proposed invertible residual network with conditional injector and interpolator, our upsampling results have better uniformity and are closer to the underlying surface, and can better preserve the fine-grained information of the object.

The main contributions of this paper can be summarized as follows:

- We propose an easily embedded point cloud upsampling algorithm, called PU-INN, based on an invertible residual neural network, which has an unconstrained and expressive architecture by enforcing a Lipschitz constraint.
- We propose a conditional injector that includes a feature extractor and a feature fusion module, to enhance the feature transformation capability of the invertible neural network. The feature extractor module extracts fine local and coarse-grained global features, and the fusion module is designed to control the flow of information from local semantic features to global features based on an efficient channel attention.
- We propose a lightweight interpolator based on the semantic similarity distance. Considering that PU-INN learns a bidirectional mapping between the point cloud and the latent space, the interpolation in the latent space can immediately reflect the changes in Euclidean space.
- Experimental results show that PU-INN achieves better performance in qualitative and quantitative evaluations compared with SOTA methods.

2 Related Works

Optimization-Based Upsampling. Traditional methods have tried various optimization strategies to generate upsampled point clouds. Early point upsampling methods usually adopted shape priors [Alexa *et al.*, 2003; Lipman *et al.*, 2007], and depended on the hand-crafted prior quality. Moreover, some methods relied on local geometric fitting, such as normal estimation [Huang *et al.*, 2009] and smooth surface assumptions [Huang *et al.*, 2013; Wu *et al.*, 2015]. In general, the above metioned methods are not data-driven, thus leading to insufficient assumptions or requiring geometric attributes.

Learning-Based Upsampling. Deep learning methods introduce promising advances to optimization-based algorithms, owing to their data-driven characteristics and neural network learning capabilities. Deep neural networks [Qi *et al.*, 2017a; Qi *et al.*, 2017b; Wang *et al.*, 2019; Thomas *et al.*, 2019; Li *et al.*, 2018] can learn features directly from point clouds. PU-Net [Yu *et al.*, 2018], as a pioneer, proposed the first deep neural network for point cloud upsampling. MPU [Yifan *et al.*, 2019] and PU-GAN [Li *et al.*, 2019] proposed a patch-based progressive upsampling method and a GAN-based framework separately. More recently, PU-GEO [Qian *et al.*, 2020] considered the local geometric metric to upsample the point cloud. PU-GCN [Qian *et al.*, 2021a] improved the performance of point cloud upsampling by using a graph-based module. DIS-PU [Li *et al.*, 2021] applied a disentangled refinement framework to adjust the coarse upsampled points into fine scale points. Flexible-PU [Qian *et al.*, 2021b] formulated the point cloud upsampling problem in an explicit manner by using the linear approximation theorem. PC2-PU [Long *et al.*, 2022] paid more attention to patch correlation and position correction.

The above mentioned methods extract latent features by an encoding process and reconstruct coordinates by a decoding process. On the contrary, in this paper we proposed a method based on normalizing flows to unify the encoding and decoding processes.

Normalizing Flow in Point Cloud. Some inverible neural networks based on normalizing flows have been proposed to achieve efficient inverse computation by adopting the dimensional decomposition architecture, such as NICE [Dinh *et al.*, 2014], Real-NVP [Dinh *et al.*, 2016], i-RevNet [Jacobsen *et al.*, 2018] and Glow [Kingma and Dhariwal, 2018]. However, these methods suffered from restricted transformations with sparse or structured Jacobians, thus leading to weak nonlinear transformation capability. Transformations that scale to high-dimensional data relied on specialized architectures. Recently, an invertible residual network [Behrmann *et al.*, 2019] has been proposed based on invertible residual blocks, which have forward and backward Lipschitz boundaries, to avoid strict architectural constraints. To reduce the complexity and boost the adaptability, the invertible residual network [Behrmann *et al.*, 2019] has been further improved by generalizing the Lipschitz constraint to induced mixed norms [Chen *et al.*, 2019].

PointFlow [Yang *et al.*, 2019] applied continuous normalizing flows [Chen *et al.*, 2018] to 3D point clouds, but continuous flows caused slow convergence. Both DPF-Net [Klokov *et al.*, 2020] and PU-Flow [Mao *et al.*, 2022] used affine coupling layers [Dinh *et al.*, 2016] to model discrete normalizing flow. These networks had a strong inductive bias that may hinder their application in supervised tasks. To improve the transformation capability, our PU-INN is inspired by an invertible residual network [Chen *et al.*, 2019] and proposed with free-form Jacobian. Moreover, we impose a flexible conditional injector and a uniformity loss to further enhance the performance of the invertible residual network. Furthermore, the upsampling is performed with a point interpolator based on semantic similarity distance.

3 Method

Given a 3D point cloud $\mathcal{X} = \{x_i \in \mathbb{R}^3\}_{i=1}^N$ with N points and a user-specified upsampling factor r, we aim to generate a dense and homogeneous point cloud set $\mathcal{P} = \{p_i \in \mathbb{R}^3\}_{i=1}^{r\times N}$ with $r \times N$ points.

In this study, we put $\mathcal{X}$ into the injector, which fuses the local and global features extracted from the input, to get the conditional information $\mathcal{C}$. Then, we feed $\mathcal{X}$ and $\mathcal{C}$ together into an invertible residual neural network, to get the high-dimensional output in the latent Riemannian space. Finally, we interpolate in this latent space to get $\tilde{Z}$, which is transformed to the point cloud $\mathcal{P}$ in the Euclidean space by the inverse mapping of the invertible neural network.

3.1 Invertible Residual Neural Network

Our invertible neural network is constructed with some invertible residual layers. An invertible residual layer maps the original point cloud distribution D_x to the specified simple distribution D_z through a transformation function F, which consists of M invertible residual blocks,

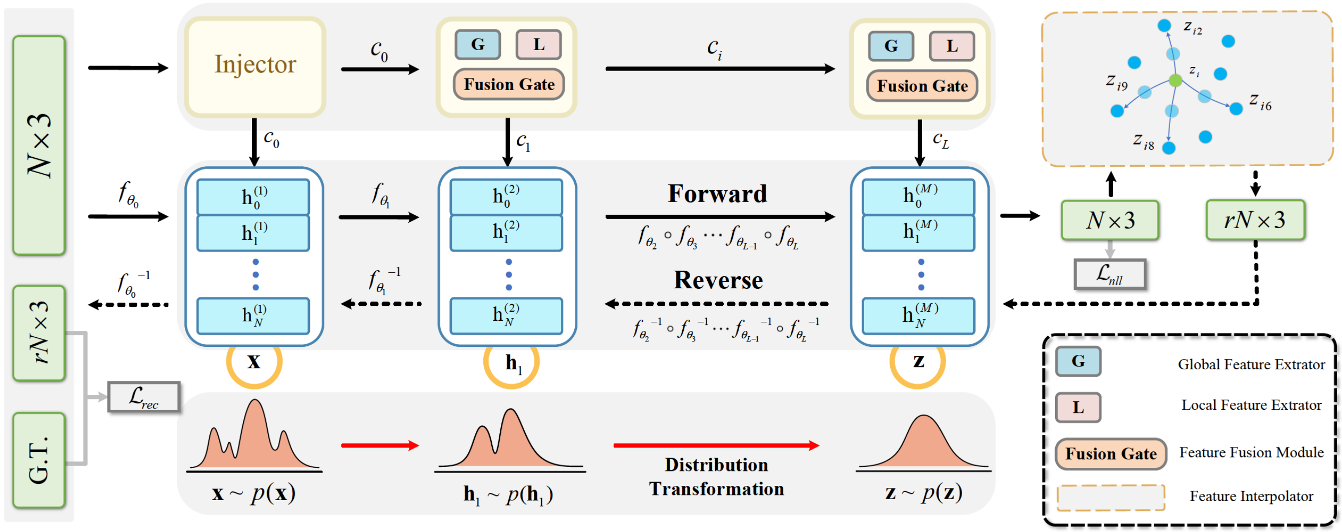

Figure 1: **The Architecture of PU-INN**. The upper shows the overall architecture of the PU-INN model containing three main parts: the INN stream in middle row (Section 3.1), Conditional Injector in first row (Section 3.2), and Point Interpolator (Section 3.3). The bottom row indicates the transformation of the distribution. The INN stream consists of some residual blocks.

$$
\begin{aligned}
h^{l+1} &= f_{\theta_l}(h^l) \quad l = 1, ..., M \\
h^1 &= x, \quad h^M = z \sim \mathcal{N}(0, \mathrm{I})
\end{aligned} \tag{1}
$$

where $f_{\theta_l}(h^l) = h^l + g_{\theta_l}(h^l)$, and $g_{\theta_l}(h^l)$ computes the transformation of the input features h^l at the l-th invertible residual block f_{θ_l}. x is the original point cloud coordinates, and z obeys the specified simple distribution (like Gaussian distribution).

To ensure that each residual block is invertible, we enforce a Lipschitz constraint [Behrmann *et al.*, 2019] as follows:

$$
\mathrm{Lip}(g_{\theta_l}) < 1 \tag{2}
$$

where the $\mathrm{Lip}(g_{\theta_l})$ is the Lipschitz norm of the function g_{θ_l}. According to Banach's fixed-point theorem, the fixed point of the inverse function $h^l = h^{l+1} - g_{\theta_l}(h^l)$ is calculated by iterating a certain number of steps to converge to h^l, which satisfies the accuracy requirement.

The training objective of the invertible neural network is as follows:

$$
F^* = \arg\min_{\theta}(-\log p_x(\mathbf{x})) \tag{3}
$$

where $\log p_x(\mathbf{x}) = \log p_z(\mathbf{z}) + \log|\det J_F)|$. $p_x(\mathbf{x})$ and $p_z(\mathbf{z})$ are the probability distribution of D_x and D_z, respectively. $\log|\det J_F|$ is the Jacobian matrix log-determinant of the transformation function.

3.2 Conditional Injector

We note that the constraint — the dimensionality of the output must be the same as the input in each invertible residual block (in Section 3.1) — limits the capacity of nonlinear transformation. Thus, we propose a conditional injector for injecting the necessary feature information into the original network to enhance its transformation capability. Then Equation (1) becomes:

$$
\mathcal{C}_0 = \mathrm{INJ}(h^0), \quad h^1 = f_{\theta_0}(h^0; \mathcal{C}_0) \tag{4}
$$

where $\mathcal{C}_0$ is the output of the injector $\mathrm{INJ}(h^0)$, which encodes h^0 as a conditional information, $\mathcal{C}_{v+1} = \mathrm{INJ}(\mathcal{C}_v)(v > 0)$. The injector consists of a feature extractor and a feature fusion module.

Point Feature Extractor

As shown in Figure 2, we propose a local and a global feature extractor to extract the fine and coarse-grained features of the point cloud, respectively.

Local Feature. To avoid the overall transformation from affecting the local information, we obtain relative point positions by subtracting the center point from the the k-nearest neighbor points. Then, we explicitly encode the relative point positions to enhance the neighboring point feature. We extract local features as follows:

$$
\mathcal{H}_L = \mathrm{MLP}\left(T(x_i) \oplus x_i^k \oplus \|x_i - x_i^k\|\right) \tag{5}
$$

where the T represents the T-Net [Qi *et al.*, 2017a] which solves the rotation invariance problem, and $\oplus$ is the concatenation operation. $||x_i - x_i^k||$ computes the euclidean distance between the neighboring points x_i^k and the centroid point x_i, where x_i^k is the k-th nearst neighbor of x_i. The detailed architecture of MLP (Multi Layer Perceptron) can be found in Appendix A. Finally, we use the attentive pooling [Hu *et al.*, 2020] instead of the traditional maximum pooling to preserve details during upsampling.

Global Feature. We propose the global feature extractor based on dense connections [Liu *et al.*, 2019a]. Specifically, each dense connection learns from its previous layer and the raw input (on the same local region) simultaneously. Without explicit local information encoding, this global extractor can focus on the shape of point clouds, contour and other

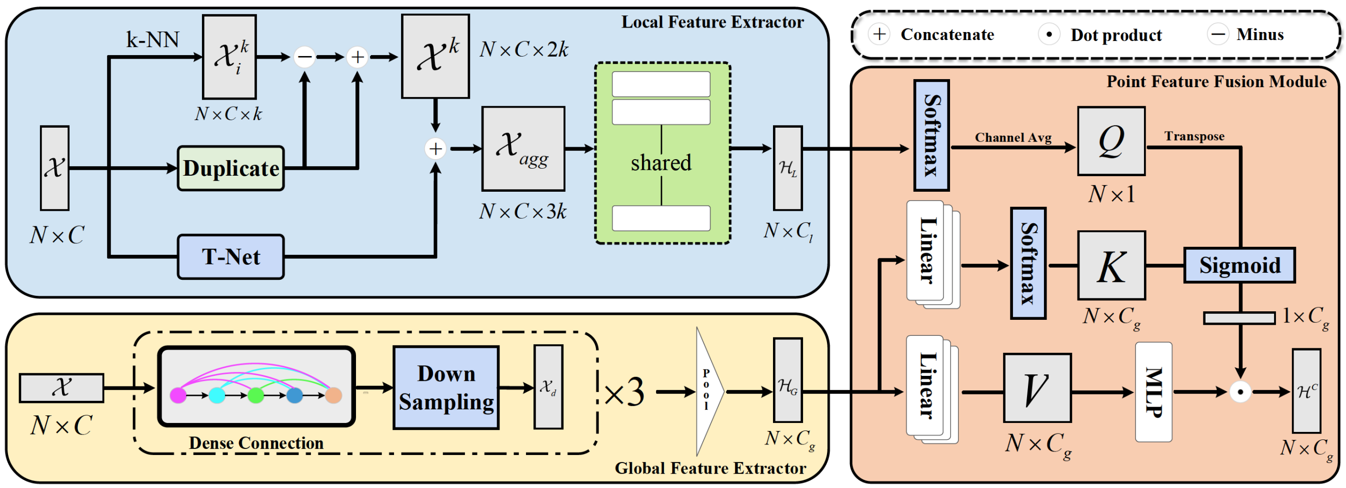

Figure 2: **Conditional Injector.** The conditional injector consists of three modules: Local Feature Extractor (Section 3.2), Global Feature Extractor (Section 3.2), and Feature Fusion Module (Section 3.2).

global information. We extract densely contextual representation $\mathcal{H}_G$ as:

$$\mathcal{H}_G = \mathrm{MLP}(\mathrm{MLP}(\mathrm{MLP}(x_i) \oplus x_i) \oplus x_i) \tag{6}$$

This extractor can use the dense connectivity pattern to repeatedly aggregate multi-level and multi-scale features. Instead of using the attentive pooling in the local feature extractor, a normal max pooling is used to extract global features.

Point Feature Fusion

We develop the point feature fusion module to incorporate local information $\mathcal{H}_L$ with global features $\mathcal{H}_G$ for aggregation. The fusion module also controls how much information from the global features should be merged with the local features based on the efficient channel attention [Vaswani *et al.*, 2017]. The fused feature map is computed as follows:

$$\mathcal{H}^C = \mathrm{Norm}\left(\frac{Q^T K}{\sqrt{N}}\right) \cdot \mathrm{MLP}(V) \tag{7}$$

where $\mathrm{Norm}(\cdot)$ denotes a sigmoid function. $Q = \mathrm{op}(\sigma(H_L)) \in \mathbb{R}^{N\times 1}$ denotes queries of local features, and $\mathrm{op}(\cdot)$ is a channel average operation. $K = W_K\sigma(H_G) \in \mathbb{R}^{N\times C_g}$ denotes keys of global features, and σ denotes a Softmax operation. $V = W_V H_G \in \mathbb{R}^{N\times C_g}$ denotes values of global features. $W_K, W_V \in \mathbb{R}^{N\times N}$ are linear transformation parameters. $\mathcal{H}^C$ denotes output fusion features.

3.3 Point Interpolator

We design an interpolator for points expansion based on the semantic similarity distance. Unlike images with fixed topological relations, it is impossible to apply the continuous weight function [Liu *et al.*, 2019b; Wu *et al.*, 2019] to directly interpolate the feature map.

For each point, we perform interpolation in its k-nearest neighbors to generate new points, and the distance is calculated by similarity $\mathcal{S}_{i,j}$. Specifically, we set the cosine similarity score between the features of centroid and candidate points as a measure of the semantic similarity in the latent space. Given the fusion latent code $\boldsymbol{z} \in \mathbb{R}^{N_C=3}$, which is the last residual layer's N_C-dimensional output. The similarity $\mathcal{S}_{i,j}$ of points $\boldsymbol{z}_i$ and $\boldsymbol{z}_j$ is defined as follows:

$$\mathcal{S}_{i,j} = \frac{{\boldsymbol{z}_i}^T \boldsymbol{z}_j}{\|\boldsymbol{z}_i\|_2 \|\boldsymbol{z}_j\|_2} \tag{8}$$

where $\|\cdot\|_2$ is the L2 Normalization. Cosine distance can force the network to consider the spatial relationships between points, as two neighboring points naturally produce similar features and generate high cosine similarity.

By obtaining the neighborhood of each point, we can interpolate in it and learn the weight vector using a learning-based approach with the following formula:

$$\tilde{\boldsymbol{z}}_{i,l} = \frac{\sum_{u=1}^{k} \psi_l\left(\boldsymbol{z}_i, \boldsymbol{z}_{i,u}\right) \boldsymbol{z}_{i,u}}{\sum_{u=1}^{k} \psi_l\left(\boldsymbol{z}_i, \boldsymbol{z}_{i,u}\right)} \tag{9}$$

where $\boldsymbol{z}_{i,u}$ is the u-th nearest neighbor to z_i, and $\psi(\boldsymbol{z}_i, \boldsymbol{z}_{i,u}) \in \mathbb{R}^{N_C}$ is the embedding in the feature space, which encodes the local structure and global information. $\tilde{\boldsymbol{z}}_{i,l}$ is the l-th interpolated feature from the original feature $\boldsymbol{z}_i$, $l = 1, 2, \cdots, R$ (upsampling ratio). k is the number of nearest neighbors. The interpolation in the latent space can be reflected in the Euclidean space through the reverse mapping of the invertible neural network.

3.4 Loss Function

Prior Loss. Equation (3) allows us to train the flow layer by minimizing the negative log-likelihood by entering $\mathcal{X}$:

$$\mathcal{L}_{\text{nll}}(\mathcal{X}) = \mathcal{L}(\mathcal{X}, \mathcal{C}; \theta) = -\log p(\mathcal{X} \mid \mathcal{C}, \theta) \tag{10}$$

where $\mathcal{C} = \mathrm{INJ}(\mathcal{X})$. Optimizing the prior likelihood of $\mathcal{X}$ encourages encoding the shape representation to obtain high probability under a predefined prior $p_z(\mathbf{z})$, which is modeled by the flow module F_θ. In our experiments, the prior $p_z(\mathbf{z})$ is simply set to the standard Gaussian distribution $\mathcal{N}(0, \mathbf{I})$.

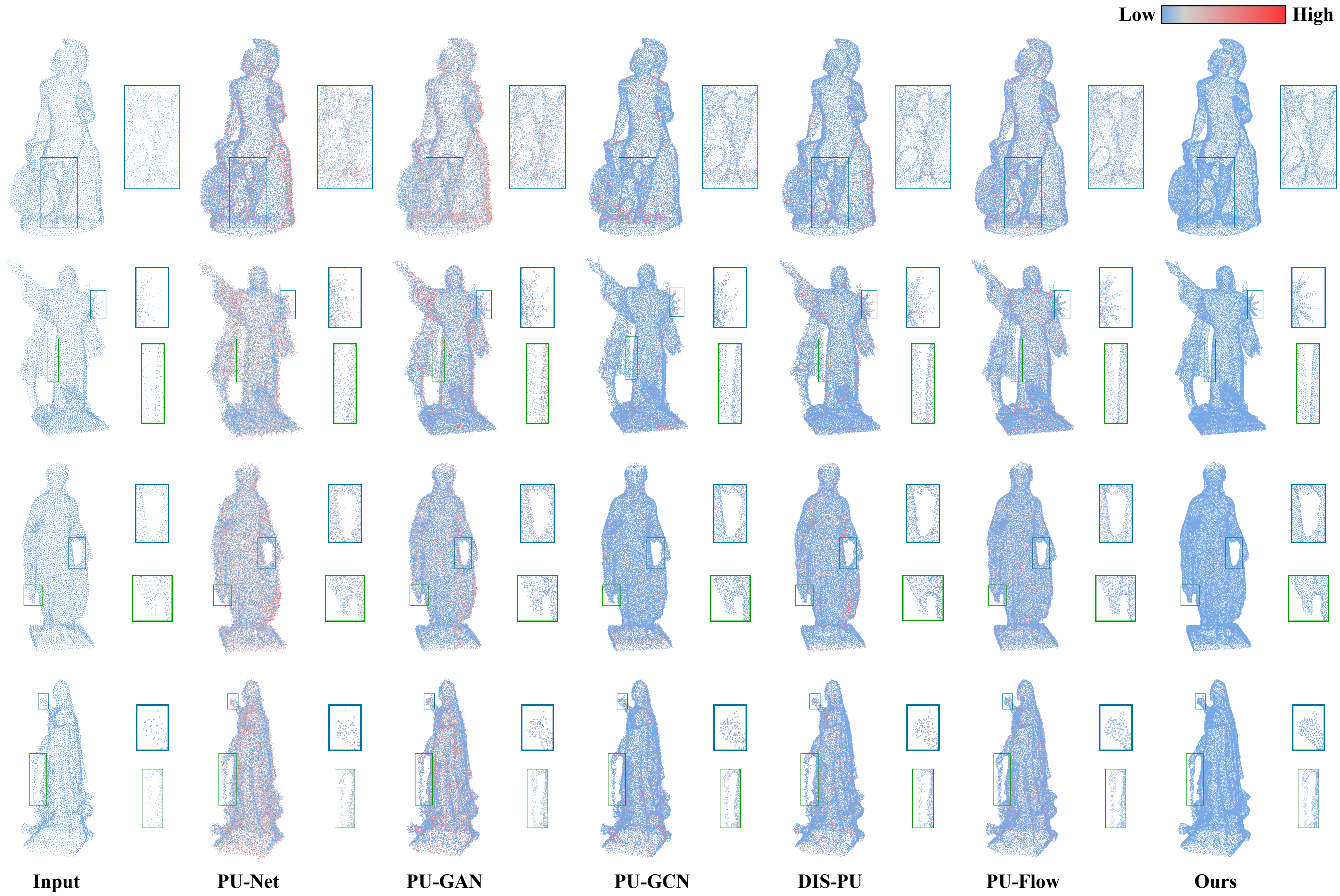

Figure 3: **Visual Comparisons.** Comparisons to state-of-the-art methods, PU-Net [Yu *et al.*, 2018], PU-GAN [Li *et al.*, 2019], PU-GCN [Qian *et al.*, 2021a], Dis-PU [Li *et al.*, 2021], PU-Flow [Mao *et al.*, 2022] in 4× upsampling experiments using 5000 input points from PU1K dataset [Qian *et al.*, 2021a]. We visualized the P2F errors by colors for each point.

Similarity Loss. We use the Earth Mover's distance (EMD)to measure the dissimilarity between two multidimensional distributions in a given feature space, and EMD is calculated as follows:

$$\mathcal{L}_{\text{EMD}}(P, \mathcal{X}) = \min_{\phi:\mathcal{P}\to\mathcal{X}} \sum_{p_i\in\mathcal{P}} \|p_i - \phi(p_i)\|_2 \tag{11}$$

where $\phi : P \to \mathcal{X}$ is the bijective mapping. EMD is fast, differential and robust to outliers[Fan *et al.*, 2017].

Uniform Loss. We also take the uniformity of the point cloud into consideration. To evaluate the uniformity, we make each point to be as far as possible from the rest of the points in the same point set as follows:

$$\mathcal{L}_{\text{Uni}}(\mathcal{P}) = \sum_{p_i\in\mathcal{P}} \sum_{p_j\in\mathcal{P}, j\neq i} \exp\left(-\frac{\|p_i - p_j\|^2}{(2k^2)}\right) \tag{12}$$

where k denotes the size of the considered neighborhood.

4 Experiments

Dataset. The data we use for training and testing comes from two datasets, PU1K [Qian *et al.*, 2021a] and Sketchfab [Sketchfab, 2011]. Both datasets include point clouds of different resolutions and their surface models for evaluating the reconstruction metrics. Some mesh models have complex geometry and high-frequency details. To fairly compare different methods, we perform the same enhancement operations on the point cloud data, including random scaling, rotation and point perturbation.

Implementation Details. For each training object, we randomly crop it into $M = 100$ overlayable patches, resulting in 102000 training surface patches. On each surface patch, we extract rN points as the groundtruth $\mathcal{Q}$ by Poisson-disk sampling. Then, we randomly sample $N = 256$ points from $\mathcal{Q}$ as training input $\mathcal{X}$.

We implement our network using the PyTorch platform and train 200 epochs using the Adam algorithm with an initial learning rate $\phi = 1e^{-4}$ (decayed by 0.7 every 40 epochs) by NVIDIA 3,090 GPU. The batch size is 16, and we empirically set the values of α, β, and γ in the compound loss function to 1000, 0.01, and 0.01, respectively. PU-INN has 10 residual layers, and each residual layer consists of 27 invertible residual blocks.

Figure 4: **Surface Comparisons.** Comparisons of reconstructed surfaces from upsampled points of various methods, (EAR [Huang *et al.*, 2013], PU-Geo [Qian *et al.*, 2020], PU-GCN [Qian *et al.*, 2021a], Dis-PU [Li *et al.*, 2021] in 4× upsampling experiments using 5000 input points from PU25 dataset.

4.1 Comparisons with SOTA Methods

Quantitative Comparison. To quantitatively evaluate the quality of the output point cloud, we use the Earth Mover's distance (EMD), Chamfer distance (CD) [Butt, 1998], Hausdorff distance (HD) [Huttenlocher, 1993], Uniform Metric(UNI), and Point-to-surface (P2F) distance. A lower evaluation metric value indicates better performance. Table 1 shows the comparison results under the dataset PU1K [Qian *et al.*, 2021a]. We observe that our method outperforms the previous SOTA methods on all the metrics. Specifically, the improvement of our method on EMD shows that it tends to encourage points to remain the underlying surface. Moreover, our method achieves the best results on CD and P2F metrics. The results show that PU-INN is more robust to noise compared with the other methods. Benefiting from the uniform loss constraint, our method outperforms the other methods especially on the uniformity metric, even for the challenging $16\times$ upsampling.

Qualitative Comparison. The qualitative results of different point cloud upsampling models are presented in Figure 3. It shows that baseline methods tend to generate more noise (e.g., PU-Net), or destroy tiny-part structures (e.g., PU-GAN). In high-curvature regions, PU-GCN cannot produce high-quality results. Dis-PU is able to well preserve tiny local structures by disentangled refinement module while losing the global information for guiding. Our output can better display the points' origin shape contour while preserving more structural details. As shown in Figure 4, we also reconstruct meshes from the generated upsampling points by SAP Algorithm [Peng *et al.*, 2021], to show that our results help to reconstruct complex meshes with high-frequency details and fewer artifacts.

We have also conducted experiments to verify our model's robustness with respect to different input sizes and noise, and PU-INN achieves the best performance in all the tested cases. We give more details in Appendix B, which also shows more qualitative comparisons (real-scanned data, non-uniform data and other datasets), and quantitative comparisons.

4.2 Ablation Study

To evaluate the effectiveness of the major modules in the conditional injector of our proposed model, we conduct an ablation study in four cases: (1) removing the global feature extractor (w/o Injector(Global)); (2) removing the local feature extractor (w/o Injector(Local)); (3) removing the feature fusion module (w/o Injector(Fusion)), and the global and local features are concatenated after a transformation with MLP; and (4) removing the injector (w/o Injector). The time metric is the cost of PU-INN's inference with only one object, which has 5000 points, and the upsampling ratio is 4x. Table 2 shows that our conditional injector can effectively enhance the performance of our model on the EMD, CD, HD, UNI and P2F metrics. Although the computational cost and reference time are higher by introducing the injector, the computaion efficiency of our full method still outperforms the SOTA methods.

4.3 Lightweight Network Design

The existing point cloud upsampling methods generally construct networks with a large number of parameters, resulting in a complicated training procedure and high computation cost. The amounts of parameters typically over 100K, and some methods even contain over 1000K parameters. As discussed in Section 1, we design a lightweight interpolator

Methods	4× Upsampling Without Noise					16× Upsampling Without Noise				
	EMD ↓	CD ↓	HD ↓	UNI ↓	P2F ↓	EMD ↓	CD ↓	HD ↓	UNI ↓	P2F ↓
EAR [Huang *et al.*, 2013]	7.915	4.919	7.784	4.141	6.479	8.515	5.064	8.255	4.594	6.964
PU-Net [Yu *et al.*, 2018]	4.015	3.143	4.495	1.919	2.315	4.295	3.426	4.649	2.061	2.651
PU-GAN [Li *et al.*, 2019]	3.514	1.158	4.251	1.630	2.225	3.647	1.461	4.525	1.726	2.519
PU-Geo [Qian *et al.*, 2020]	3.251	1.002	3.915	1.237	1.525	3.394	1.326	4.051	1.316	1.648
PU-GCN [Qian *et al.*, 2021a]	2.929	0.585	3.648	1.181	1.899	3.105	0.619	3.841	1.354	2.096
Dis-PU [Li *et al.*, 2021]	2.897	0.494	3.422	0.964	1.526	2.996	0.508	3.600	1.281	1.848
Flexible-PU [Qian *et al.*, 2021b]	2.692	0.426	3.588	1.231	1.425	2.859	0.495	3.693	1.261	1.658
PU-Flow [Mao *et al.*, 2022]	2.572	0.394	3.484	1.198	1.336	2.921	0.458	4.618	1.318	1.515
PC2-PU [Long *et al.*, 2022]	2.423	0.381	3.594	0.865	1.308	2.662	0.426	3.961	1.260	1.355
PU-INN	**2.042**	**0.375**	**2.815**	**0.699**	**1.181**	**2.166**	**0.401**	**3.052**	**0.712**	**1.301**
Methods	4× Upsampling With Noise					16× Upsampling With Noise				
	EMD ↓	CD ↓	HD ↓	UNI ↓	P2F ↓	EMD ↓	CD ↓	HD ↓	UNI ↓	P2F ↓
EAR [Huang *et al.*, 2013]	8.015	5.132	7.923	4.403	6.696	8.711	5.234	8.534	4.859	7.213
PU-Net [Yu *et al.*, 2018]	4.150	3.415	4.737	2.122	2.476	4.398	3.544	4.822	2.190	2.784
PU-GAN [Li *et al.*, 2019]	3.812	1.347	4.375	1.731	2.327	3.823	1.667	4.739	1.946	2.740
PU-Geo [Qian *et al.*, 2020]	3.384	1.235	4.105	1.407	1.636	3.616	1.583	4.312	1.520	1.808
PU-GCN [Qian *et al.*, 2021a]	3.204	0.830	3.939	1.466	2.107	3.233	0.811	3.988	1.626	2.238
Dis-PU [Li *et al.*, 2021]	3.153	0.762	3.721	1.264	1.748	3.174	0.661	3.759	1.549	1.953
Flexible-PU [Qian *et al.*, 2021b]	2.726	0.616	3.648	1.315	1.613	3.049	0.762	4.118	1.918	1.915
PU-Flow [Mao *et al.*, 2022]	2.747	0.620	3.719	1.309	1.438	3.205	0.613	4.773	1.536	1.753
PC2-PU [Long *et al.*, 2022]	2.594	0.612	3.625	1.261	1.435	2.816	0.716	3.998	1.416	1.617
PU-INN	**2.110**	**0.512**	**3.012**	**0.840**	**1.420**	**2.260**	**0.592**	**3.342**	**0.821**	**1.523**

Table 1: **Quantitative comparisons.** Quantitative comparisons to state-of-the-art methods on the PU1K dataset [Qian *et al.*, 2021a]. All metric units are 10^{-3}. The best results are denoted in bold.

Ablation	EMD ↓	CD ↓	HD ↓	UNI ↓
	P2F ↓	Time ↓	Params ↓	FLOPs ↓
w/o Injector(Global)	2.294	0.403	3.192	0.983
	1.492	3.17 ms	51.6 K	31.5 G
w/o Injector(Local)	2.364	0.761	3.067	0.991
	1.873	3.24 ms	55.6 K	35.6 G
w/o Injector(Fusion)	2.406	0.734	3.269	1.075
	1.894	4.89 ms	67.9 K	39.8 G
w/o Injector	2.576	0.923	3.562	1.297
	2.062	**3.09 ms**	**38.9 K**	**28.4 G**
PU-INN	**2.042**	**0.375**	**2.815**	**0.699**
	1.181	5.10 ms	68.1 K	45.2 G

Table 2: **Ablation study**. The components of the PU-INN are tested on PU1K dataset [Qian *et al.*, 2021a]. All metric units are 10^{-3} except the Time, Params and FLOPs. The best results are denoted in bold.

to reduce the model complexity for the development of point cloud upsampling methods.

Benefiting from the architecture of invertible residual neural network, the resblock can share weights in the forward and reverse phases, and the elimination of gradient calculation in the reverse phase saves the calculations in the training phase, and thus reducing the model's parameters and saving the inference time. To reduce the computation cost, we only set the first residual block of each residual learning layer as a conditional residual block. As shown in Table 3, our model has 6.8x fewer parameters and 2.8x fewer FLOPs than PU-Flow. PU-INN's inference speed performs up to 2.3x faster than PU-Flow.

Methods	#Params	FLOPs	Time
PU-NET [Yu *et al.*, 2018]	916.1K	186.2G	10.58ms
MPU [Yifan *et al.*, 2019]	1005.2K	162.9G	9.91ms
PU-GAN [Li *et al.*, 2019]	648.2K	294.5G	13.81ms
PU-Geo [Qian *et al.*, 2020]	1463.2K	148.3G	15.52ms
PU-GCN [Qian *et al.*, 2021a]	219.2K	90.5G	10.94ms
DIS-PU [Li *et al.*, 2021]	131.6K	85.4G	8.83ms
Flexible-PU [Qian *et al.*, 2021b]	324.5K	82.8G	8.92ms
PU-Flow [Mao *et al.*, 2022]	463.4K	125.4G	11.72ms
PC2-PU [Long *et al.*, 2022]	105.4K	68.3G	9.64ms
PU-INN	**68.1K**	**45.2G**	**5.10ms**

Table 3: Computation cost on the state-of-the-art methods in 4x upsampling ratio using 5000 uniform input points. Time: Inferencet Speed.

5 Conclusions

In this paper, we propose a point cloud upsampling method based on invertible residual neural networks. Considering that bi-directional mapping limits the dimensionality of the input and output, we propose a conditional injector, which can inject the necessary feature information into the original network without affecting the invertibility and improve its nonlinear transformation capability. Furthermore, we design an interpolator based on semantic similarity distance, by which interpolating in latent space immediately reflects changes in Euclidean space. Extensive qualitative and quantitative experiments show that our network outperforms SOTA methods. With the good performance of PU-INN in solving upsampling problems, we could extend its adaptability to advanced 3D vision tasks, such as semantic segmentation in future work.

Acknowledgments

This work was supported by the NSF of Guangdong Province under 2022A1515011573, the Key-Area Research and Development Program of Guangzhou City (202206030009), the National Natural Science Foundation of China (U21A20520 and No.62202257), Beijing Natural Science Foundation (L222008).

Contribution Statement

Aihua Mao:Conceptualization, Funding acquisition, Investigation, Project administration, Resources, Supervision, Validation, Validation of analysis, Writing-review & editing. **Yaqi Duan**: Conceptualization, Formal analysis, Investigation, Methodology, Software, Validation, Writing - original draft, Writing - review & editing. **Yu-Hui Wen**: Writing-review & editing. **Zihui Du**: Conceptualization, Investigation, Validation. **Hongmin Cai**: Writing-review & editing. **Yong-Jin Liu**: Funding acquisition, Writing-review & editing. * Corresponding author. † Equal first authorship.

References

[Alexa *et al.*, 2003] Marc Alexa, Johannes Behr, Daniel Cohen-Or, Shachar Fleishman, David Levin, and Claudio T. Silva. Computing and rendering point set surfaces. *IEEE Transactions on visualization and computer graphics*, 9(1):3–15, 2003.

[Behrmann *et al.*, 2019] Jens Behrmann, Will Grathwohl, Ricky TQ Chen, David Duvenaud, and Jörn-Henrik Jacobsen. Invertible residual networks. In *International Conference on Machine Learning*, pages 573–582. PMLR, 2019.

[Butt, 1998] Butt. Optimum design of chamfer distance transforms. *IEEE Transactions on Image Processing*, 7(10):1477–1484, 1998.

[Chen *et al.*, 2018] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

[Chen *et al.*, 2019] Tian Qi Chen, Jens Behrmann, David Duvenaud, and Jörn-Henrik Jacobsen. Residual flows for invertible generative modeling. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 9913–9923, 2019.

[Dinh *et al.*, 2014] Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

[Dinh *et al.*, 2016] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

[Fan *et al.*, 2017] Haoqiang Fan, Hao Su, and Leonidas J Guibas. 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*, pages 605–613, 2017.

[Geiger *et al.*, 2013] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[Hu *et al.*, 2020] Qingyong Hu, Bo Yang, Linhai Xie, Stefano Rosa, Yulan Guo, Zhihua Wang, Niki Trigoni, and Andrew Markham. Randla-net: Efficient semantic segmentation of large-scale point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11108–11117, 2020.

[Huang *et al.*, 2009] Hui Huang, Dan Li, Hao Zhang, Uri Ascher, and Daniel Cohen-Or. Consolidation of unorganized point clouds for surface reconstruction. *ACM transactions on graphics (TOG)*, 28(5):1–7, 2009.

[Huang *et al.*, 2013] Hui Huang, Shihao Wu, Minglun Gong, Daniel Cohen-Or, Uri Ascher, and Hao Zhang. Edge-aware point set resampling. *ACM transactions on graphics (TOG)*, 32(1):1–12, 2013.

[Huttenlocher, 1993] Huttenlocher. Comparing images using the hausdorff distance. *IEEE Transactions on pattern analysis and machine intelligence*, 15(9):850–863, 1993.

[Jacobsen *et al.*, 2018] Jörn-Henrik Jacobsen, Arnold Smeulders, and Edouard Oyallon. i-revnet: Deep invertible networks. *arXiv preprint arXiv:1802.07088*, 2018.

[Kingma and Dhariwal, 2018] Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. *Advances in neural information processing systems*, 31, 2018.

[Klokov *et al.*, 2020] Roman Klokov, Edmond Boyer, and Jakob Verbeek. Discrete point flow networks for efficient point cloud generation. In *European Conference on Computer Vision*. Springer, 2020.

[Li *et al.*, 2018] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on x-transformed points. *Advances in neural information processing systems*, 31:820–830, 2018.

[Li *et al.*, 2019] Ruihui Li, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. Pu-gan: a point cloud upsampling adversarial network. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7203–7212, 2019.

[Li *et al.*, 2021] Ruihui Li, Xianzhi Li, Pheng-Ann Heng, and Chi-Wing Fu. Point cloud upsampling via disentangled refinement. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 344–353, 2021.

[Lipman *et al.*, 2007] Yaron Lipman, Daniel Cohen-Or, David Levin, and Hillel Tal-Ezer. Parameterization-free projection for geometry reconstruction. *ACM Transactions on Graphics (TOG)*, 26(3):22–es, 2007.

[Liu *et al.*, 2019a] Yongcheng Liu, Bin Fan, Gaofeng Meng, Jiwen Lu, Shiming Xiang, and Chunhong Pan. Densepoint: Learning densely contextual representation for efficient point cloud processing. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5239–5248, 2019.

[Liu *et al.*, 2019b] Yongcheng Liu, Bin Fan, Shiming Xiang, and Chunhong Pan. Relation-shape convolutional neural network for point cloud analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8895–8904, 2019.

[Long *et al.*, 2022] Chen Long, WenXiao Zhang, Ruihui Li, Hao Wang, Zhen Dong, and Bisheng Yang. Pc2-pu: Patch correlation and point correlation for effective point cloud upsampling. In *Proceedings of the 30th ACM International Conference on Multimedia*, pages 2191–2201, 2022.

[Mao *et al.*, 2022] Aihua Mao, Zihui Du, Junhui Hou, Yaqi Duan, Yong-jin Liu, and Ying He. Pu-flow: a point cloud upsampling network with normalizing flows. *IEEE Transactions on Visualization and Computer Graphics, DOI:10.1109/TVCG.2022.3196334*, 2022.

[Miyato *et al.*, 2018] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

[Peng *et al.*, 2021] Songyou Peng, Chiyu "Max" Jiang, Yiyi Liao, Michael Niemeyer, Marc Pollefeys, and Andreas Geiger. Shape as points: A differentiable poisson solver. *CoRR*, abs/2106.03452, 2021.

[Qi *et al.*, 2017a] 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.

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

[Qian *et al.*, 2020] Yue Qian, Junhui Hou, Sam Kwong, and Ying He. Pugeo-net: A geometry-centric network for 3d point cloud upsampling. In *European Conference on Computer Vision*, pages 752–769. Springer, 2020.

[Qian *et al.*, 2021a] Guocheng Qian, Abdulellah Abualshour, Guohao Li, Ali Thabet, and Bernard Ghanem. Pugcn: Point cloud upsampling using graph convolutional networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11683–11692, 2021.

[Qian *et al.*, 2021b] Yue Qian, Junhui Hou, Sam Kwong, and Ying He. Deep magnification-flexible upsampling over 3d point clouds. *IEEE Trans. Image Process.*, 30:8354–8367, 2021.

[Sketchfab, 2011] Inc. Sketchfab. Sketchfab - the best 3d viewer on the web. https://sketchfab.com/, 2011. Accessed: 2021-11-25.

[Thomas *et al.*, 2019] 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/CVF International Conference on Computer Vision*, pages 6411–6420, 2019.

[Vaswani *et al.*, 2017] 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.

[Wang *et al.*, 2019] 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.

[Wu *et al.*, 2015] Shihao Wu, Hui Huang, Minglun Gong, Matthias Zwicker, and Daniel Cohen-Or. Deep points consolidation. *ACM Transactions on Graphics (ToG)*, 34(6):1–13, 2015.

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

[Yang *et al.*, 2019] Guandao Yang, Xun Huang, Zekun Hao, Ming-Yu Liu, Serge Belongie, and Bharath Hariharan. Pointflow: 3d point cloud generation with continuous normalizing flows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4541–4550, 2019.

[Yifan *et al.*, 2019] Wang Yifan, Shihao Wu, Hui Huang, Daniel Cohen-Or, and Olga Sorkine-Hornung. Patch-based progressive 3d point set upsampling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5958–5967, 2019.

[Yu *et al.*, 2018] Lequan Yu, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. Pu-net: Point cloud upsampling network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2790–2799, 2018.