Point Cloud Upsampling Using Conditional Diffusion Module with Adaptive Noise Suppression

Boqian Zhang*, Shen Yang*, Hao Chen, Chao Yang, Jing Jia, Guang Jiang †
Xidian University
{24011211294, 20009101459, haochen_cv, 24011211295}@xidian.stu.edu.cn,
jjia@mail.xidian.edu.cn, gjiang@mail.xidian.edu.cn

Abstract

Point cloud upsampling can improve the quality of the initial point cloud, significantly enhancing the performance of downstream tasks such as classification and segmentation. Existing methods mostly focus on generating the geometric details of point clouds, neglecting noise suppression. To address this, we propose a novel network based on a conditional diffusion model, incorporating the Adaptive Noise Suppression (ANS) module, which we refer to as PDANS. The ANS module assigns weights to each point and determines the removal strategy based on these weights, reducing the impact of noisy points on the sampling process. The module first selects the neighborhood set for each point in the point cloud and performs a weighted sum between the point and its neighbors. It then adjusts the removal points based on the weighted sum, effectively mitigating the bias caused by outliers. We introduce the TreeTrans (TT) module to capture more correlated feature information. This module learns the interaction between high-level and low-level features, resulting in a more comprehensive and refined feature representation. Our results on several widely used benchmark datasets demonstrate that PDANS exhibits exceptional robustness in noisy point cloud processing and outperforms current state-of-the-art(SOTA) methods in terms of performance. Code is available at https://github.com/Baty2023/PDANS.

1. Introduction

Point clouds, as one of the key representations for 3D reconstruction [17] [22], are widely applied in autonomous driving [5] [47], embodied intelligence [18] [42], and AR/VR. However, real-world scanned point clouds are often sparse, noisy, and uneven, negatively impacting their performance in downstream tasks such as classification and segmentation. Point cloud upsampling can transform sparse and noisy point clouds into dense, artifact-free representations. In recent years, significant progress has been made in this task [15] [27] [45] [40].

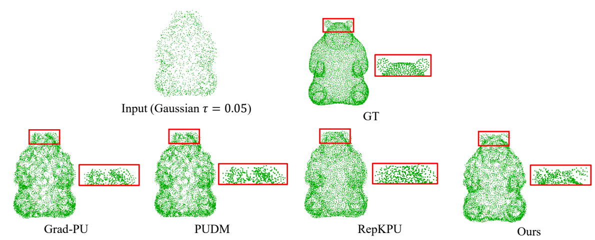

Figure 1. Most existing methods achieve satisfactory results when input without noise, but they often struggle to effectively recover the geometric structure (e.g., panda ears) when dealing with input with noise.

Inspired by deep learning, several methods have adopted deep networks to perform this task. Existing upsampling methods can be broadly classified into two categories: those based on generative models and those based on refinement methods. Generative models [15] [27] [28] first extract features from sparse point clouds into a latent space and then design a generator based on the extracted features to generate the coordinates of dense point clouds. The limitation of such methods lies in the difficulty of accurately predicting 3D coordinates, leading to potential outliers or artifacts in the generated dense point clouds [8] [45]. Refinement-based methods [4] [39] [30] first use interpolation to generate a coarse dense point cloud, and then employ a neural network to extract geometric features from the coarse point cloud for refinement. Combined with reconstruction loss functions, this process refines the point cloud to match the ground truth.

These methods share two common issues: (1) The features lack significant dominance or distinguishability, leading to generating point clouds without clear structure, which tend to be sparse and scattered. (2) Most of these methods

*These authors contributed equally to this work.
†Corresponding authors.

rely on loss functions such as CD loss, EMD loss, and other similar metrics, which are highly sensitive to outliers. This sensitivity makes the model ineffective at handling noisy point clouds [12] [38], as shown in Figure1.

To address the above issues, we propose a point cloud upsampling method with adaptive noise suppression, PDANS. The network is designed based on a conditional diffusion model, leveraging the inherent denoising properties of the diffusion model's generative process to enhance its adaptability to noisy point clouds. The network also integrates an Adaptive Noise Suppression (ANS) module, which we designed to compute downsampling weights for each point by utilizing the relationships between the point and its neighboring points. This approach enables the extraction of more accurate downsampled points and features. It effectively reduces the impact of noisy points during feature extraction and minimizes the bias caused by outliers.

To enhance feature representation, we propose a structure based on graph convolutional networks [14], called the TreeTrans (TT) module. This module incorporates ancestral and recurrent terms[32] to fully utilize the features outputted by each encoder layer. Additionally, we apply both local and global attention mechanisms to the output features, further improving the efficiency and expressiveness of the feature representation.

Additionally, we designed a conditional point cloud network and a diffusion point cloud network. The conditional point cloud network provides relevant features of sparse point clouds to the diffusion point cloud network through a mutual attention module, guiding the diffusion network to generate the required noise for point cloud densification. Similar to PUDM[29], we incorporate the diffusion model, thereby eliminating the reliance on CD loss. Combined with the ANS module we designed, PDANS demonstrates exceptional robustness in noise experiments. Our main contributions are as follows:

- We designed a noise suppression module named ANS, which adaptively adjusts removal points and their neighborhood features to improve robustness against noise and outliers.
- The proposed TT module fully leverages features across different layers of the encoder, incorporating ancestral terms, recurrent terms, and local and global attention mechanisms to achieve more comprehensive and refined feature representations.
- We designed an end-to-end network architecture named PDANS, which utilizes a mutual attention mechanism and a diffusion model to guide the densification of sparse point clouds. The ANS module further enhances the network's ability to handle noise.
- Extensive experiments demonstrate the outstanding performance of PDANS in point cloud upsampling benchmarks, particularly in generating geometric details and handling noisy point clouds effectively.

2. Related Work

2.1. Learning-based point cloud upsampling

Several methods [25] [26] [36] [11] shave been developed to extract features from 3D data directly with the strong data-driven and representational capabilities of deep learning networks. PU-Net [45] was the first learning-based method for point cloud and upsampling. Following its proposed architecture, many methods have emerged: MPU [44], PU-GAN [15], Dis-PU [16], PUGCN [27], and RepKPU [30]. The NePs [7] method enhances neural radiance fields by combining 2D information with 3D features, enabling point cloud upsampling. Some networks attempt to perform point cloud upsampling based on denoising paradigms. They generate a rough upsampled point cloud through interpolation, and then refine it using neural networks, such as Grad-PU [8] and PU-Flow [21].

Most methods predict the coordinates of dense point clouds from sparse point cloud features and then rely on upsampling modules to expand the point cloud, or further refine the coordinates from the coarsely densified point clouds. Since these methods typically depend on CD (Chamfer Distance) and EMD (Earth Mover's Distance) as loss functions, they exhibit high sensitivity to noise. In this paper, we design an additional denoising module and utilize conditional DDPM to reduce the dependence on CD.

2.2. DDPM for 3D Generation

Inspired by the success of image generation tasks [10] [34] [9], direct point cloud generation using DDPM has gained increasing attention. Luo *et al*. [19] is the first to apply DDPM to unconditional point cloud generation. The DAM [35] network explores the interpretability of diffusion models for point cloud generation tasks. Nunes *et al*. [23] apply DDPM to the 3D-LiDAR point cloud completion task in real-world scenarios. The PDR [20] network takes the initial point cloud as input, but it introduces a coarse and refinement network, which increases time overhead. PUDM [29] densifies point clouds through conditional diffusion, but its performance in noise suppression is suboptimal.

This paper uses a conditional diffusion model to perform point cloud upsampling. To enhance the model's denoising capability, we design an additional ANS module.

3. Background of Conditional Diffusion Module

We assume that the dense point cloud x is sampled from a meaningful distribution P_{data}, and that the latent variable $x_T \sim \mathcal{N}(\mathbf{0}_{3N}, \boldsymbol{I}_{3N\times 3N})$ is sampled from a noise distribution P_{latent} which is easier to handle, where $\mathcal{N}$ follows a

Gaussian distribution. DDPM[10] establishes a transfer relationship between x and x_T through two Markov chains. This is an autoregressive process: the forward process progressively adds noise to x until x_T, and x_T is denoised back to x through the reverse process. We use the timestep $t \sim U(T)$ to control the speed of transformation.

Forward Process The process gradually adds noise to x_0 until x_T through the probability $q(x_t|x_{t-1})$. The entire process can be defined as:

$$q(x_1, \cdots, x_T|x_0) = \prod_{t=1}^{T} q(x_t|x_{t-1}) \tag{1}$$

Where $q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I)$, where the hyperparameter β_t is a positive constant. Based on the derivation by Ho *et al*. [10], x_T can be computed directly from x_0. Define $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$, so that x_T can be sampled from the following formula:

$$x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon} \tag{2}$$

Where $\boldsymbol{\epsilon}$ follows a standard normal distribution.

Reverse Process The reverse process aims to predict and eliminate the noise added at each step t. It is conditioned on a set of conditions, which may include incomplete point clouds c, upsampling factor r, etc. We denote the set of conditions as $\boldsymbol{C} = \{c_i \mid i = 1, \ldots, N_c\}$, where N_c represents the number of conditions. We can define the reverse process from denoising x_T to x_0 as follows:

$$p_{\boldsymbol{\theta}}(x_0, \cdots, x_{T-1}|x_T, \boldsymbol{C}) = \prod_{t=1}^{T} p_{\boldsymbol{\theta}}(x_{t-1}|x_t, \boldsymbol{C}) \tag{3}$$

Where $p_{\boldsymbol{\theta}}(x_{t-1}|x_t, \boldsymbol{C}) = \mathcal{N}(x_{t-1}; \mu_{\boldsymbol{\theta}}(\boldsymbol{x}_t, \boldsymbol{C}, t), \sigma_t^2 \boldsymbol{I})$, where the mean $\mu_{\boldsymbol{\theta}}(x_t, \boldsymbol{C}, t)$ is the result predicted by a neural network, and σ_t^2 is a time-dependent constant. Based on the derivation by DDIM [33], the reverse process can be written in the following form:

$$\begin{aligned} x_{t-1} = \sqrt{\bar{\alpha}_{t-1}}(&\frac{x_t - \sqrt{1-\bar{\alpha}_t}\epsilon_\theta(x_t, \boldsymbol{C}, t)}{\sqrt{\bar{\alpha}_t}}) + \\ &\sqrt{1 - \bar{\alpha}_{t-1} - \sigma_t^2}\epsilon_\theta(x_t, \boldsymbol{C}, t) + \sigma_t \epsilon_t \end{aligned} \tag{4}$$

Where $\epsilon_t \sim \mathcal{N}(\mathbf{0}_{3N}, \boldsymbol{I}_{3N\times 3N})$ is standard Gaussian noise independent of x_T. The noisy point cloud x_T is progressively denoised according to the above equation until it recovers x_0. At each time step of the reverse process, conditions are added, allowing control over the generation of dense point clouds, as shown in the process diagram in Figure 2.

Training process under given conditions The goal is to optimize the denoising model's prediction of the noise ϵ_θ at step t. Given a set of conditions $\boldsymbol{C}$ and under the condition that the time step $t \sim U(T)$, the training objective of DDPM is:

$$\mathcal{L}(\theta) = \mathbb{E}_{\boldsymbol{t}\sim U(T), \epsilon\sim\mathcal{N}(0,I)} \|\epsilon - \epsilon_\theta(\sqrt{1-\bar{\alpha}_t}\epsilon + \sqrt{\bar{\alpha}_t}x_0, \boldsymbol{C}, t)\|^2 \tag{5}$$

In the above equation, there are no CD loss or EMD loss terms commonly used in traditional point cloud upsampling networks, because DDPM naturally defines a one-to-one pointwise correspondence between two consecutive point clouds, as shown in equation1.

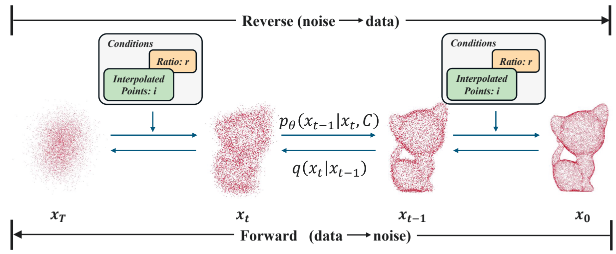

Figure 2. Visualization of the forward and reverse processes of PDANS.

4. Method

This section introduces each module's design and the network's overall structure. In PDANS, we propose two modules: the Adaptive Noise Suppression (ANS) module and the TreeTrans (TT) module. These modules are described in Sections 4.1 and 4.2, respectively. The design of the overall PDANS network is presented in Section 4.3.

4.1. Adaptive Noise Suppression(ANS) Module

In the encoding stage, many frameworks perform downsampling using Farthest Point Sampling (FPS), followed by selecting neighboring points within a certain radius [20] or via KNN [1] to enhance the features of the sampled points. However, two issues arise when the input point cloud contains significant noise: (1) FPS is sensitive to outliers[43][41], making it challenging to process noisy point clouds effectively. (2) FPS does not filter noisy points, leading to the inclusion of noise features in the enhanced features, which affects the accuracy and reliability of the sampling.

We propose the Adaptive Noise Suppression (ANS) module to address the issues abovementioned, as shown in Figure 3. Unlike FPS, we first select the neighborhood point cloud $\mathcal{P}_K \in \mathbb{R}^{N\times 3\times K}$ for all points in the initial point cloud $\mathcal{P} \in \mathbb{R}^{N\times 3}$, where K denotes the number of neighboring points for each point. These neighborhood point clouds contain both correct and noisy points. We then input the neighborhood point cloud, the initial point cloud, and their corresponding features into the ANS module to perform the removal operation.

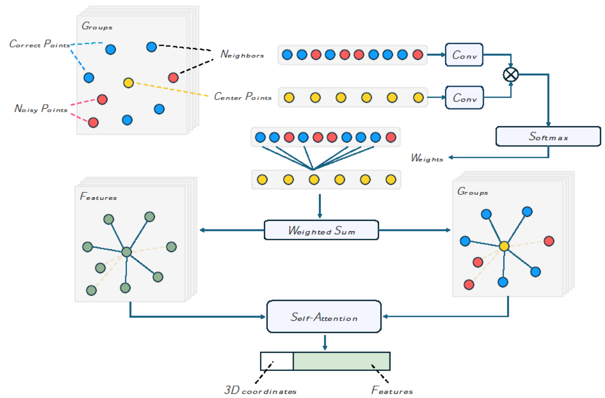

Figure 3. Visualization of the Adaptive Noise Suppression (ANS) module

For each point x_i selected from $\mathcal{P}$, we select its corresponding feature f_i from $\mathcal{F}s \in \mathbb{R}^{N \times D_l}$, where $\mathcal{F}s$ is the feature set obtained by reordering the initial point cloud. We use KNN to select its K nearest neighbors $x_{i,1}, \ldots, x_{i,K} \in \mathcal{N}(x_i)$ and their corresponding features $f_{i,1}, \ldots, f_{i,K} \in \mathcal{N}(f_i)$, where $\mathcal{N}(x_i)$ represents the K nearest neighbors of point x_i. Next, we calculate the point and feature weights using an attention mechanism, with both points and features updated similarly. $w(f_i)$ calculation can be expressed as:

$$w(f_i) = \sum_{j=1}^{K} \theta(x_i, x_{i,j}) \cdot \gamma(f_{i,j}) \tag{6}$$

In the equation, the function θ is used to compute the relationship between a neighbor point and the center point. The function $\gamma(f_{i,j})$ represents the linear transformation of the neighborhood feature, expressed as $\gamma(f_{i,j}) = W_\gamma f_{i,j} + b_\gamma$, where W_γ and b_γ are learnable parameters.

The function θ is computed using the dot product similarity in the attention mechanism:

$$\theta(x_i, x_{i,j}) = \mathrm{softmax}(q(x_i) \cdot k(x_{i,j})^T / \sqrt{C_q}) \tag{7}$$

Where $q(x_i)$ and $k(x_{i,j})$ represent the linear transformations of the center point x_i, and the neighborhood points $\mathcal{N}(x_i)$, which can be implemented using a 1×1 convolution kernel. The input to $q(x_i)$ is the real coordinates of the center point, while the input to $k(x_{i,j})$ is the relative coordinates between the neighborhood point and the center point. The convolution kernel has an output channel of C_q and an input channel of 3. Using real and relative coordinates, the model effectively combines global and local spatial information, enhancing its ability to perceive the 3D spatial geometry.

We then use the computed $w(f_i)$ to refine the initial points and their features. This operation selects points with more surrounding noise points as removing points, resulting in more accurate features. In other words, we can find the suitable refined point x_i^* and its corresponding feature f_i^*, expressed as:

$$x_i^* = \delta_p(w(f_i))x_i, \tag{8}$$

$$f_i^* = \delta_f(w(f_i))f_i. \tag{9}$$

Where δ_p and δ_f is activation function. Finally, we fuse the points x_i^* and their corresponding features f_i^* through an attention module to obtain the module's final output features f_{out}.

4.2. TreeTrans(TT) Module

In the decoding process of encoded point clouds, previous methods often begin with three rounds of interpolation, followed by feature fusion using neighboring point clouds [26], and some methods adopt point-adaptive deconvolution techniques, such as PDR [20]. These methods are typically built on UNet architecture [31], which leads to several issues: (1) The model becomes overly dependent on global features, thereby neglecting local features, and (2) The utilization of multi-scale features is not comprehensive enough, leading to suboptimal performance when handling complex point clouds.

We propose the TreeTrans (TT) module as our decoding network to address the above issues. We first employ two operations similar to TreeGCN[32]: the cyclic and the ancestor operations, which propagate features from the root node to the leaf nodes. After obtaining the intermediate features f_{tree}, we combine f_{tree} with the corresponding features from the encoder using local and global attention mechanisms to obtain the output f_{out}.

In traditional graph convolutional networks, the cyclic operation uses a separate weight to transform the current node feature f_i^l to the next layer. However, a single weight often fails to capture all the relevant information in point clouds. Thus, we employ a K-support-based cyclic operation to refine the feature representation progressively:

$$S_i^{l+1} = F_K^l(f_i^l) \tag{10}$$

Where F_K^l is a fully connected layer containing K nodes, and S_i^{l+1} represents the cyclic operation at layer l. The cyclic operation is then combined with the ancestor operation to update the node features, ensuring that each layer incorporates information from previous layers during computation. The updated node features can be written as:

$$f_{tree} = \sigma\left(S_i^{l+1} + \sum_{m_j \in A(f_i^l)} U_j^l m_j\right) \tag{11}$$

Where σ is the activation function, with ReLU used in this paper. $A(f_i^l)$ denotes the set of all ancestor nodes, and U_j^l is

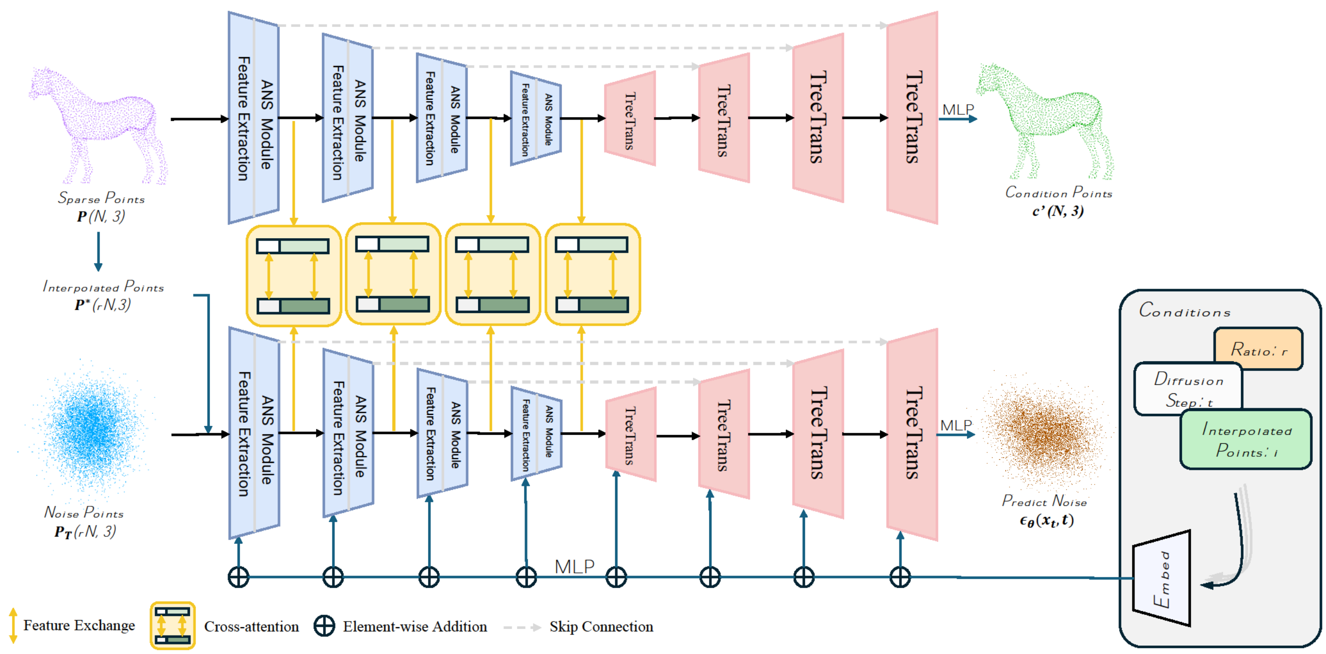

Figure 4. **Overview of PDANS.** The CP network (upper branch) and DP network (lower branch) both consist of an encoder made up of feature extraction and the ANS module, with the TT module serving as the decoder. They exchange information through a mutual attention module. the upsampled ratio and interpolated points are used as conditions to provide prior knowledge of the DP network.

the linear transformation weight from ancestor nodes to the current node.

After obtaining f_{tree}, we input it into the local attention and global attention modules. In the local attention mechanism, the corresponding encoder output features f^{l+1} are combined with f_{tree} using local attention, as described by the following formula:

$$f_{local} = \text{softmax}\left(\frac{Q(f^{l+1}) \cdot K(f_{tree})^T}{\sqrt{d}}\right) \cdot V(f_{tree}) \tag{12}$$

$Q(f^{l+1})$ and $K(f_{tree})$ represent two convolution operations that map f^{l+1} and f_{tree} from their respective dimensions to d. Then, the attention weights are obtained by calculating the dot product similarity followed by a softmax operation. Afterward, the weighted sum with $V(f_{tree})$ gives the local attention feature f_{local}. Once the local attention is computed, the resulting f_{local} undergoes a global attention mechanism to produce the final output f_{out} of the TT module. The specific implementation is as follows:

$$f_{out} = \text{softmax}\left(\frac{q(f_{local}) \cdot k(f_{local})^T}{\sqrt{d}}\right) \cdot v(f_{local}) \tag{13}$$

4.3. Network Architecture

In Sections 4.1 and 4.2, we introduced the network's encoding module, ANS, and decoding module, TT, respectively. In this section, we present the overall framework of PDANS, which consists of two main components: the Conditional Point Cloud (CP) network and the Diffusion Point Cloud (DP) network. Figure 4 provides a detailed illustration of this process.

CP-Net Our encoder consists of a feature extraction network and the ANS module. The decoder comprises the TT module. Both downsampling and upsampling are performed four times. Before each sampling output, we embed the time step t through an MLP: t is encoded into a C_t-dimensional vector space. The output feature $f_{out} \in \mathbb{R}^{N \times D_l}$ is then transformed through MLPs into the dimension $f'_{out} \in \mathbb{R}^{N \times C_t}$. We then add the two to integrate the time step information into the feature, resulting in the output $f_{l+1} \in \mathbb{R}^{N \times D_{l+1}}$. Additionally, we only input the sparse point cloud into the CP-Net to ensure the extraction of effective features.

DP-Net The DP-Net shares the same architecture as the CP-Net but requires additional conditional information to guide the diffusion process. First, we convert the sparse point cloud $\mathcal{P} \in \mathbb{R}^{N \times 3}$ into an interpolated point cloud $\mathcal{P}^* \in \mathbb{R}^{rN \times 3}$ using upsampling by a factor of r. Then, we sum $\mathcal{P}^*$ with the noisy point cloud $\mathcal{P}_T \in \mathbb{R}^{rN \times 3}$ and input them together into the DP-Net. Meanwhile, we extract global features from $\mathcal{P}^*$ and encode both the time step t and the upsampling factor r. These pieces of information are incorporated as conditions at each stage of the encoder and decoder.

The information provided by the CP-Net is further guided to the DP-Net through a cross-attention (CA) module. Given the outputs of the encoder at each layer for the CP-Net and DP-Net: $F_c \in \mathbb{R}^{N_c \times D_c}$ and $F_d \in \mathbb{R}^{N_d \times D_d}$, the CA module first reshapes the two feature sets using MLPs: $F_c \rightarrow (Q) \in \mathbb{R}^{N_c \times C_i}$ and $F_d \rightarrow (K, V) \in \mathbb{R}^{N_d \times C_i}$. Then, the features are fused according to the following formula:

$$F_f = MLP(softmax(\frac{QK^T}{\sqrt{C_i}})V) + F_c \tag{14}$$

Subsequently, F_f is input into the feedforward network to produce the final features. The same method can also be applied reversely, allowing data to flow between the two networks.

4.4. Loss

For the DP-Net, its loss can be referred to by equation 5. For the CP-Net, we aim for it to learn more features of the sparse point cloud. Therefore, we add MLPs at the end of the decoder to output a sparse point cloud $\mathcal{P}'$, which is then supervised by MSE as follows:

$$\mathcal{L}_C(\mathcal{P}, \mathcal{P}') = \mathbf{MSE}\|\mathcal{P} - \mathcal{P}'\|^2 \tag{15}$$

Combining these loss functions, The training objective is:

$$\mathcal{L}_{total} = \mathcal{L}(\theta) + \alpha \mathcal{L}_C \tag{16}$$

where α means a weighting factor (α = 0.5 in this paper).

5. Experiments

5.1. Implementation Details

Experiment Setup. All PDANS experiments are run in a PyTorch[24] environment. The PDANS model is trained on a single Nvidia 4080 GPU using the Adam optimizer[13], with a batch size 12 for 350 epochs. The channel sizes for the CP-Net and DP-Net are set to (64, 128, 256, 512) and (128, 256, 256, 512), respectively.

Dataset. In our experiments, we utilize two public benchmarks (PUGAN[15], PU1K[27]) for evaluation. We adhere to the official training/testing partitioning protocols for these datasets. Poisson disk sampling [46] is used to generate 24,000 patches on PUGAN and 69,000 patches on PU1K for training. Each patch contains 256 points, with the corresponding ground truth containing 1,024 points. Additionally, 27 and 127 point cloud datasets are used for testing, respectively. The sparse point clouds contain 2,048 points, and the number of points in ground truth is set to 2,048$\times r$ according to the upsampling factor r.

Evaluation Metrics Following previous methods, we select three metrics to evaluate our model: Chamfer Distance(CD $\times 10^{-3}$)[6], Hausdorff distance(HD $\times 10^{-3}$)[2] and Point to Surface Distance(P2F $\times 10^{-3}$)[45].

5.2. Evaluation on PUGAN Dataset

We perform sampling on PUGAN at low (4$\times$) and high (16$\times$) upsampling rates. The reductions in the three metrics shown in Table 1 demonstrate that our model has a significant advantage in describing geometric details. The 16$\times$ upsampled point cloud shown in Figure 5 further supports our claim, as the generated point clouds have fewer outliers and are closer to the ground truth geometric features.

Table 1. The results of 4× and 16× on PUGAN. Our method significantly surpasses other methods in terms of CD and HD.

Methods	4×			16×		
	CD↓	HD↓	P2F↓	CD↓	HD↓	P2F↓
PU-Net[45]	0.529	6.805	4.460	0.510	8.206	6.041
MPU[44]	0.292	6.672	2.822	0.219	7.054	3.085
PU-GAN[15]	0.282	5.577	2.016	0.207	6.963	2.556
PU-GCN[27]	0.268	3.201	2.489	0.161	4.283	2.632
Grad-PU[8]	0.245	2.369	1.893	0.108	2.352	2.127
RepKpU[30]	0.254	2.698	**1.882**	0.107	3.293	**1.957**
PUDM[29]	0.131	1.220	1.912	0.082	1.120	2.114
Ours	**0.119**	**1.2065**	1.897	**0.078**	**1.046**	2.028

5.3. Evaluation on PU1K Dataset

Table 2 shows the results of 4x upsampling on the PU1K dataset. Compared to other methods, we demonstrate superior performance in both CD and HD metrics. The PU1K dataset is more complex and diverse than others, and our designed TT module exhibits tremendous advantages in handling such complex data. It helps approximate the valid geometric details more effectively, achieving higher accuracy.

Methods	CD↓	HD↓	P2F↓
PU-Net[45]	1.155	15.170	4.834
MPU[44]	0.935	13.327	3.511
PU-GAN[15]	0.735	9.939	4.234
PU-GCN[27]	0.585	7.577	2.499
Grad-PU[8]	0.404	3.732	1.474
RepKpU[30]	0.331	2.700	**1.261**
PUDM[29]	0.217	2.164	1.477
Ours	**0.155**	**1.588**	1.354

Table 2. The results of 4× on PU1K. Our method outperforms other methods across nearly all metrics.

5.4. Robustness Test

Experiment setup. Based on the testing meshes provided by PUGAN, we construct a 4× upsampling dataset to conduct a comprehensive robustness test. Ground-truth point clouds are sampled from testing meshes via Poisson disk sampling. We tested three different types of noise: Gaussian, random, and Laplacian. These three types of noise are the most common in real-world point clouds, making

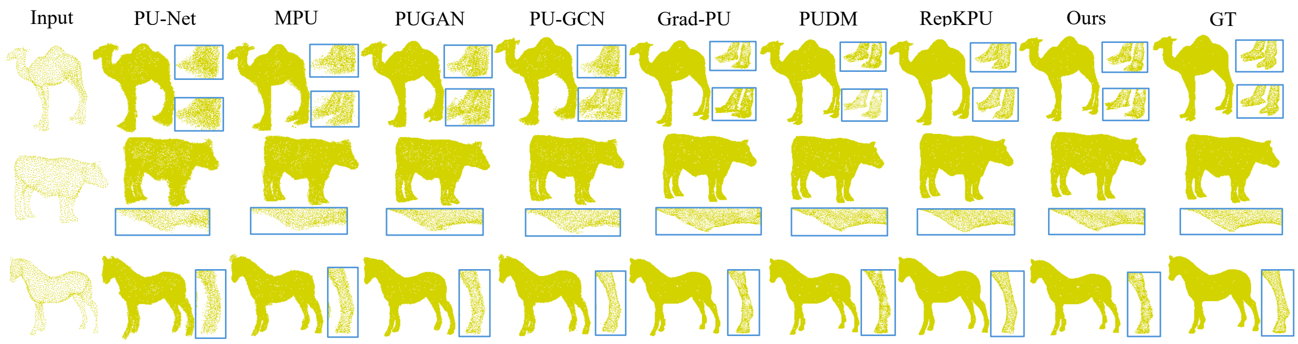

Figure 5. Comparison (16×) on PUGAN dataset.

the results more practical. We set four different noise levels for Gaussian noise: $\tau = 0.01$, $\tau = 0.02$, $\tau = 0.05$, and $\tau = 0.1$. We used two noise levels for Laplacian and random noise: $\tau = 0.05$ and $\tau = 0.1$. For the input with 2048 points, we tested the results under each noise type and noise level.

The results of Gaussian noise Tables 3 and 4 present the quantitative results of Gaussian noise testing under different noise levels. As shown in the tables, our method significantly outperforms other models in terms of resistance to Gaussian noise. Thanks to the design of the ANS module in our model, which further enhances noise resistance. Therefore, our approach excels at noise suppression and surpasses other methods in both CD and HD metrics.

Table 3. The results of 4× at low-level($\tau = 0.01$ and $\tau = 0.02$) Gaussian noise on PUGAN. Our method significantly outperforms other methods in terms of Gaussian noise.

Methods	$\tau = 0.01$			$\tau = 0.02$		
	CD↓	HD↓	P2F↓	CD↓	HD↓	P2F↓
PU-Net[45]	0.628	8.068	9.816	1.078	10.867	16.401
MPU[44]	0.506	6.978	9.059	0.929	10.820	15.621
PU-GAN[15]	0.464	6.070	7.498	0.887	10.602	15.088
PU-GCN[27]	0.448	5.586	6.989	0.816	8.604	13.798
Grad-PU[8]	0.414	4.145	6.400	0.766	7.336	11.534
RepKpU[30]	0.384	4.937	6.775	0.800	8.415	14.124
PUDM[29]	0.210	2.430	6.070	0.529	5.471	9.742
Ours	**0.185**	**1.918**	**5.964**	**0.309**	**3.020**	**9.513**

The results of random noise and Laplacian noise Tables 5 and 6 present the quantitative test results under noise levels $\tau = 0.05$ and $\tau = 0.1$. The results clearly show that the PDANS network we designed exhibits superior adaptability in high-noise environments. Figure 6 illustrates that, in noisy point clouds, our results show more complete structures with fewer outliers. Our method captures more geometric details, leading to higher-quality point clouds. In contrast, other methods tend to upsample blindly, resulting in point clouds with disorganized structures. While our method still maintains SOTA performance, it shows slightly lower results than Gaussian noise. The experimental results validate the effectiveness of the ANS module we designed. By removing noisy point clouds, we can better learn the objects' structural information, thereby generating high-quality point clouds. As a result, PDANS demonstrates good robustness under different types and levels of noise, proving its stability in complex noisy environments.

Table 4. The results of 4× at high-level($\tau = 0.05$ and $\tau = 0.1$) Gaussian noise on PUGAN. Our method significantly outperforms other methods in terms of Gaussian noise.

Methods	$\tau = 0.05$			$\tau = 0.1$		
	CD↓	HD↓	P2F↓	CD↓	HD↓	P2F↓
PU-Net[45]	1.370	13.729	23.249	1.498	14.193	23.846
MPU[44]	1.247	11.645	22.189	1.321	12.415	23.841
PU-GAN[15]	1.124	9.091	21.252	1.271	10.911	23.174
PU-GCN[27]	1.263	9.869	22.835	1.456	11.063	25.213
Grad-PU[8]	0.978	8.057	16.927	1.118	8.946	18.845
RepKpU[30]	1.237	12.335	20.518	1.360	12.902	22.473
PUDM[29]	0.618	5.386	14.751	0.853	6.239	16.845
Ours	**0.476**	**4.038**	**14.607**	**0.543**	**4.327**	**16.476**

Table 5. The results of 4× at high-level($\tau = 0.05$ and $\tau = 0.1$) random noise on PUGAN.

Methods	$\tau = 0.05$			$\tau = 0.1$		
	CD↓	HD↓	P2F↓	CD↓	HD↓	P2F↓
PU-Net[45]	1.490	14.473	23.223	1.725	15.442	25.251
MPU[44]	1.224	10.842	20.456	1.545	11.645	23.512
PU-GAN[15]	1.034	7.757	18.617	1.327	9.700	21.321
PU-GCN[27]	1.045	9.643	18.899	1.325	10.877	21.633
Grad-PU[8]	1.067	6.634	17.734	1.399	7.215	21.028
RepKpU[30]	0.789	4.782	17.813	0.837	3.902	20.953
PUDM[29]	0.998	6.110	**17.558**	1.310	6.732	20.564
Ours	**0.722**	**3.161**	17.634	**0.682**	**4.598**	**19.401**

5.5. Result on Real Dataset

We conducted experiments on the ScanObjectNN [37] dataset, where point clouds are real-scanned, non-uniform and noisy. Since there is no ground truth point cloud, we performed a visual comparison in Fig. 7 . The experiment

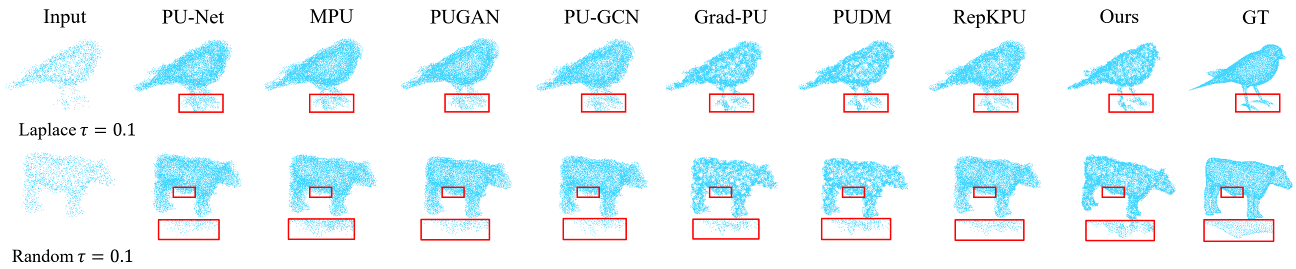

Figure 6. Visualization results at 4× on PUGAN with random noise and Laplace noise.

Table 6. The results of 4× at high-level($\tau = 0.05$ and $\tau = 0.1$) Laplace noise on PUGAN.

Methods	$\tau = 0.1$			$\tau = 0.05$		
	CD↓	HD↓	P2F↓	CD↓	HD↓	P2F↓
PU-Net[45]	1.943	32.646	26.965	1.861	27.563	24.737
MPU[44]	1.569	20.329	25.164	1.421	19.049	22.783
PU-GAN[15]	1.486	14.919	24.148	1.334	14.243	21.879
PU-GCN[27]	1.587	15.711	25.134	1.425	14.915	22.876
Grad-PU[8]	1.233	10.348	18.576	1.065	9.349	16.591
RepKpU[30]	1.345	13.104	22.295	1.194	12.377	20.218
PUDM[29]	0.704	6.284	18.283	0.643	7.924	16.871
Ours	**0.563**	**4.051**	**16.287**	**0.533**	**4.433**	**16.195**

used an input with 2048 points, followed by a 16× upsampling. The Ball-pivoting algorithm [3] was then applied to reconstruct the surface from the upsampled point cloud. Based on the reconstruction results, our generated point clouds exhibit fewer gaps and smoother surfaces, which benefits surface reconstruction.

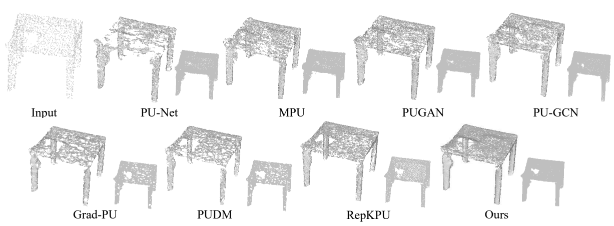

Figure 7. Visual evaluation of impacts on surface construction on real-scanned ScanObjectNN dataset.

5.6. Ablation Study

With/Without ANS Module To demonstrate the effectiveness of the ANS module, we compared it with other network architectures. Model A replaces the ANS module with an encoder composed of Dense Blocks [8], Model B uses an encoder based on Set Abstraction [29], and Model C employs an encoder from PointNet++ [26]. Our network is referred to as Model D. We tested these models on the PU-GAN dataset with Gaussian noise level $\tau = 0.1$, with results presented in Table7.

Table 7. The results of 4× at high-level($\tau = 0.1$) Gaussian noise on PUGAN with different encoders.

Module	CD↓	HD↓	P2F↓
A	1.248	9.631	19.657
B	0.853	6.348	16.849
C	1.168	14.498	23.189
D	**0.543**	**4.327**	**16.476**

With/Without TT Module We conducted 16× upsampling tests on the PUGAN dataset, as shown in Table 8. The experiments indicate a significant performance drop when the TT module is removed, underscoring the importance of the TT module in enhancing the quality of the model generation.

Table 8. TT Module Ablation Study

Methods	CD↓	HD↓	P2F↓
Without TT Module	0.094	1.124	2.158
With TT Module	**0.078**	**1.046**	**2.028**

6. Conclusion

We propose a novel conditional diffusion model-based network, PDANS, designed for point cloud upsampling. Suppressing noise and ensuring the geometric authenticity of generated results have long been primary research foci in point cloud upsampling. To address these issues, we have designed the ANS and TT modules. The ANS module integrates point cloud features with neighborhood information, enabling adaptive noise removal from the point cloud. Meanwhile, the TT module effectively captures local and global geometric details, allowing for a more accurate reconstruction of geometries close to real values. Extensive experiments validate that the proposed PDANS outperforms existing state-of-the-art methods in performance.

References

[1] Tenindra Abeywickrama, Muhammad Aamir Cheema, and David Taniar. K-nearest neighbors on road networks: a

journey in experimentation and in-memory implementation. *arXiv preprint arXiv:1601.01549*, 2016. 3

[2] Matthew Berger, Joshua A Levine, Luis Gustavo Nonato, Gabriel Taubin, and Claudio T Silva. A benchmark for surface reconstruction. *ACM Transactions on Graphics (TOG)*, 32(2):1–17, 2013. 6

[3] Fausto Bernardini, Joshua Mittleman, Holly Rushmeier, Cláudio Silva, and Gabriel Taubin. The ball-pivoting algorithm for surface reconstruction. *IEEE transactions on visualization and computer graphics*, 5(4):349–359, 1999. 8

[4] Haolan Chen, Shitong Luo, Wei Hu, et al. Deep point set resampling via gradient fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(3):2913–2930, 2022. 1

[5] Yaodong Cui, Ren Chen, Wenbo Chu, Long Chen, Daxin Tian, Ying Li, and Dongpu Cao. Deep learning for image and point cloud fusion in autonomous driving: A review. *IEEE Transactions on Intelligent Transportation Systems*, 23(2): 722–739, 2021. 1

[6] 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. 6

[7] Wanquan Feng, Jin Li, Hongrui Cai, Xiaonan Luo, and Juyong Zhang. Neural points: Point cloud representation with neural fields for arbitrary upsampling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18633–18642, 2022. 2

[8] Yun He, Danhang Tang, Yinda Zhang, Xiangyang Xue, and Yanwei Fu. Grad-pu: 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*, pages 5354–5363, 2023. 1, 2, 6, 7, 8

[9] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022. 2

[10] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020. 2, 3

[11] F. Y. Hu, C. L. Li, T. Zhou, H. F. Cheng, and M. M. Gu. A survey on point cloud completion algorithms for deep learning. *Journal of Image and Graphics*, 30(02):0309–0333, 2025. 2

[12] Sheng Yu Huang, Hao-Yu Hsu, and Frank Wang. Spovt: Semantic-prototype variational transformer for dense point cloud semantic completion. *Advances in Neural Information Processing Systems*, 35:33934–33946, 2022. 2

[13] Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 6

[14] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016. 2

[15] 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. 1, 2, 6, 7, 8

[16] 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. 2

[17] Chen-Hsuan Lin, Chen Kong, and Simon Lucey. Learning efficient point cloud generation for dense 3d object reconstruction. In *proceedings of the AAAI Conference on Artificial Intelligence*, 2018. 1

[18] Kangcheng Liu, Yuzhi Zhao, Zhi Gao, and Ben M Chen. Weaklabel3d-net: A complete framework for real-scene lidar point clouds weakly supervised multi-tasks understanding. In *2022 international conference on robotics and automation (ICRA)*, pages 5108–5115. IEEE, 2022. 1

[19] Shitong Luo and Wei Hu. Diffusion probabilistic models for 3d point cloud generation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2837–2845, 2021. 2

[20] Zhaoyang Lyu, Zhifeng Kong, Xudong Xu, Liang Pan, and Dahua Lin. A conditional point diffusion-refinement paradigm for 3d point cloud completion. *arXiv preprint arXiv:2112.03530*, 2021. 2, 3, 4

[21] 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*, 29(12):4964–4977, 2022. 2

[22] Luke Melas-Kyriazi, Christian Rupprecht, and Andrea Vedaldi. Pc2: Projection-conditioned point cloud diffusion for single-image 3d reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12923–12932, 2023. 1

[23] Lucas Nunes, Rodrigo Marcuzzi, Benedikt Mersch, Jens Behley, and Cyrill Stachniss. Scaling diffusion models to real-world 3d lidar scene completion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14770–14780, 2024. 2

[24] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019. 6

[25] 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. 2

[26] 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. 2, 4, 8

[27] Guocheng Qian, Abdulellah Abualshour, Guohao Li, Ali Thabet, and Bernard Ghanem. Pu-gcn: Point cloud upsampling using graph convolutional networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11683–11692, 2021. 1, 2, 6, 7, 8

[28] Shi Qiu, Saeed Anwar, and Nick Barnes. Pu-transformer: Point cloud upsampling transformer. In *Proceedings of the*

Asian conference on computer vision, pages 2475–2493, 2022. 1

[29] Wentao Qu, Yuantian Shao, Lingwu Meng, Xiaoshui Huang, and Liang Xiao. A conditional denoising diffusion probabilistic model for point cloud upsampling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20786–20795, 2024. 2, 6, 7, 8

[30] Yi Rong, Haoran Zhou, Kang Xia, Cheng Mei, Jiahao Wang, and Tong Lu. Repkpu: Point cloud upsampling with kernel point representation and deformation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21050–21060, 2024. 1, 2, 6, 7, 8

[31] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pages 234–241. Springer, 2015. 4

[32] Dong Wook Shu, Sung Woo Park, and Junseok Kwon. 3d point cloud generative adversarial network based on tree structured graph convolutions. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 3859–3868, 2019. 2, 4

[33] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020. 3

[34] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020. 2

[35] Hanxiao Tan. Dam: Diffusion activation maximization for 3d global explanations. *arXiv preprint arXiv:2401.14938*, 2024. 2

[36] 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. 2

[37] Mikaela Angelina Uy, Quang-Hieu Pham, Binh-Son Hua, Thanh Nguyen, and Sai-Kit Yeung. Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1588–1597, 2019. 7

[38] Tong Wu, Liang Pan, Junzhe Zhang, Tai Wang, Ziwei Liu, and Dahua Lin. Balanced chamfer distance as a comprehensive metric for point cloud completion. *Advances in Neural Information Processing Systems*, 34:29088–29100, 2021. 2

[39] Peng Xiang, Xin Wen, Yu-Shen Liu, Yan-Pei Cao, Pengfei Wan, Wen Zheng, and Zhizhong Han. Snowflakenet: Point cloud completion by snowflake point deconvolution with skip-transformer. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5499–5509, 2021. 1

[40] Xu Yan, Chaoda Zheng, Zhen Li, Sheng Wang, and Shuguang Cui. Pointasnl: Robust point clouds processing using nonlocal neural networks with adaptive sampling. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5589–5598, 2020. 1

[41] Jiancheng Yang, Qiang Zhang, Bingbing Ni, Linguo Li, Jinxian Liu, Mengdie Zhou, and Qi Tian. Modeling point clouds with self-attention and gumbel subset sampling. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3323–3332, 2019. 3

[42] Lei Yang, Yanhong Liu, Jinzhu Peng, and Zize Liang. A novel system for off-line 3d seam extraction and path planning based on point cloud segmentation for arc welding robot. *Robotics and Computer-Integrated Manufacturing*, 64:101929, 2020. 1

[43] Zetong Yang, Yanan Sun, Shu Liu, and Jiaya Jia. 3dssd: Point-based 3d single stage object detector. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11040–11048, 2020. 3

[44] 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. 2, 6, 7, 8

[45] 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. 1, 2, 6, 7, 8

[46] Cem Yuksel. Sample elimination for generating poisson disk sample sets. In *Computer Graphics Forum*, pages 25–32. Wiley Online Library, 2015. 6

[47] Yuchao Zheng, Yujie Li, Shuo Yang, and Huimin Lu. Global-pbnet: A novel point cloud registration for autonomous driving. *IEEE Transactions on Intelligent Transportation Systems*, 23(11):22312–22319, 2022. 1