# You Should Learn to Stop Denoising on Point Clouds in Advance

**Chuchen Guo**[1], **Weijie Zhou**[1], **Zheng Liu**[1*], **Ying He**[2]

[1] School of Computer Science, China University of Geosciences (Wuhan)
[2]School of Computer Science and Engineering, Nanyang Technological University
guocconly1@foxmail.com, zhouzz0722@cug.edu.cn, liu.zheng.jojo@gmail.com, yhe@ntu.edu.sg

## Abstract

Point clouds have become the preferred data format for a variety of tasks in 3D vision and graphics. However, raw point clouds often contain significant noise. This paper introduces the Adaptive Stop Denoising Network (ASDN), a novel approach aimed at restoring high-quality point clouds from noisy data. Our method is built upon a pivotal observation: during the denoising phase, high-noise points draw more focus from the network, which may suppress the points that have already been effectively denoised. This observation has led us to develop an adaptive strategy that ceases denoising already cleaned points to prevent over-denoising, while continuing to refine points that remain noisy. We employ a U-Net architecture complemented by an adaptive classifier, which utilizes a recoverability factor to assess the completion of denoising and make dynamic decisions about when to halt the process. Our method not only demonstrates superior noise removal efficiency but also preserves geometric details more effectively, reducing over- or under-denoising artifacts. Extensive experiments and evaluations demonstrate that our method outperforms the state-of-the-art both qualitatively and quantitatively.

**Code** — https://github.com/git-guocc/ASDN

## Introduction

Point clouds are the predominant data for describing the shapes and structures of real-world objects. With the rapid advancement of 3D acquisition equipment such as LiDAR and depth cameras, more and more point clouds are routinely obtained and wildly employed in various industries, such as autonomous driving, robotic navigation, and architectural modeling. However, the raw point clouds are inevitably contaminated by noise, resulting in declining point cloud quality. Therefore, point cloud denoising, which aims to restore point cloud quality, is a fundamental issue in 3D vision. Despite the significant progress, traditional denoising methods are still limited by their reliance on local point cloud information and their struggle with complex noise patterns (Liu et al. 2023). Furthermore, in order to produce promising results, traditional methods typically require numerous parameters and laborious parameter tuning.

*Corresponding Author.


The success of neural networks on point clouds, such as PointNet (Charles et al. 2017a), PointNet++ (Charles et al. 2017b), and DGCNN (Wang et al. 2019), has recently led to the adoption of deep learning approaches for point cloud denoising. For example, PointCleanNet (Rakotosaona et al. 2020) employs a two-stage learning network that first classifies and removes outliers and then estimates correction vectors to project noisy points onto the underlying surface for noise removal. Pointfilter (Zhang et al. 2021) shares a similar network structure to PointCleanNet, but it elaborately designs a bilateral loss function that considers both point and normal information to preserve sharp edges. DMRDenoise (Luo and Hu 2020) performs manifold reconstruction during the downsampling process to reduce the impact of noisy points. Score (Luo and Hu 2021) estimates the noise-convolved distribution score from noisy input and uses gradient ascent to iteratively relocate noisy points to the underlying surface for denoising. IterativePFN (de Silva Edirimuni et al. 2023b) simulates progressive smoothing using an adaptive loss function and several iterative modules.

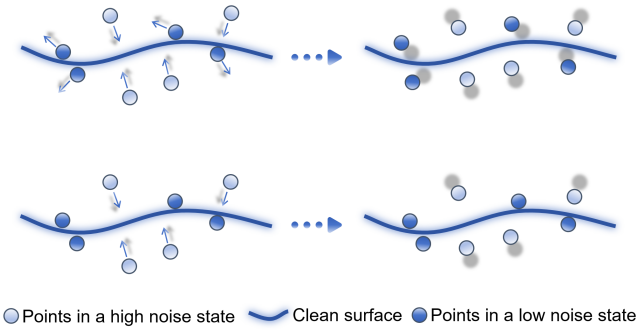


Figure 1: Our main observation and motivation. Top: at a certain denoising phase, points in a high-noise state may lead to the network focusing more on them, potentially causing points in a low-noise state to be suppressed and negatively impacted in the subsequent phase. Bottom: this insight inspires us to stop denoising the well-denoised points and continue to denoise those points in the high-noise state.

It is crucial to realize that during the denoising process, the noise and geometry information are intricately inter-

twined and inherently complicated. Most existing learning-based methods consider denoising as a holistic and homogeneous smoothing process, making them unsuitable for dynamic and heterogeneous information recovery in latent space. Thus, the existing methods cannot adapt well to complex noise and varying surface regions, resulting in over- or under-denoising artifacts of the denoising results. The complexity of noise and varying surface regions presents significant challenges in the denoising task.

To tackle the challenges mentioned, we introduce entropy to quantify the quality of point clouds. The entropy value reflects the data randomness and uncertainty, making it a crucial indicator to evaluate the point cloud quality. Therefore, we reformulate denoising as an entropy restoration process, aiming to restore the entropy of the denoised point cloud as close as possible to its noise-free counterpart.

To achieve this, we propose a novel metric, the recoverability factor $\rho$, to measure the challenge of entropy restoration. $\rho$ is calculated as the entropy ratio in the noisy point cloud to that in the clean one. A lower $\rho$ indicates a more challenging restoration. Then, we develop the Adaptive Stop Denoising Network (ASDN), a U-Net-based model that integrates an adaptive classifier with the recoverability factor. Our method leverages the adaptive classifier to dynamically assess the denoising process, allowing for early termination for those already well-denoised points.

As shown in Figure 1, our method decides to stop denoising low-noise points with a high recoverability factor $\rho$, indicating these points are already well-denoised, while continuing to denoise those high-noise points with a lower $\rho$. In other words, our method can stop denoising at well-denoised points to prevent over-smoothing artifacts, while continuing to denoise high-noise points to remove noise completely. In contrast, when the previous methods encounter relatively high- and low-noise points at a given denoising moment. Due to the supervision of the loss function, points in a high-noise state have a greater potential for loss decreasing, which may drive the network to become aggressive and prioritize fitting these points. Unfortunately, this aggressive behavior suppresses and negatively impacts the well-denoised points in the subsequent phase, as shown in Figure 1.

Our main contributions are summarized as follows:

- We find that stopping well-denoised points early will potentially prevent over-denoising artifacts, providing new cues for point cloud denoising.
- We introduce entropy as a measure for characterizing point cloud complexity. Furthermore, we define the recoverability factor as the entropy ratio between noisy and clean point clouds to evaluate the difficulty of entropy restoration. This factor guides our adaptive denoising procedure.
- We design a U-Net-based network, integrating the adaptive classifier that utilizes the recoverability factor to dynamically evaluate the denoising process and determine whether to halt it early for already well-denoised points.
- We qualitatively and quantitatively demonstrate that our method outperforms state-of-the-art approaches on synthetic and real-scanning benchmarks.

# Related Work

Given the multitude of point cloud denoising techniques, our review is confined to deep learning approaches that are directly relevant to our method.

## Estimating Dispalcement Vectors for Denoising

Several methods directly infer point displacements towards the underlying surface. PointCleanNet (Rakotosaona et al. 2020) infers outliers to remove them, then predicts displacements for the remaining noisy points for denoising. Pointfilter (Zhang et al. 2021), leveraging a similar network architecture, enhances this process with a bilateral loss function that integrates both point and normal data to preserve sharp features. Chen et al. (2022) proposed a recurrent network adept at capturing multiscale feature representations. Li and Sheng (2023) introduced a unified framework that concurrently eliminates outliers and reduces noise within a single-stage model. IterativePFN (de Silva Edirimuni et al. 2023b) emulates a progressive smoothing process, employing a progressive loss to incrementally refine denoised points toward the clean surface.

## Two-phase Point Cloud Denoising

Two-phase approaches first smooth noisy normals before updating point positions. Lu et al. (2020) divided the noisy point cloud into two point sets, only inferring multiple normals on the feature point set to preserve geometry features. Wei et al. (2021) employed a dual neural network to filter noisy normals, leveraging geometric cues. Zhou et al. (2022) and Zhang et al. (2024) devised a coarse-to-fine technique that treats normal filtering and refinement as two separate tasks, thereby enhancing the quality of the resulting normals. Recently, there have been several methods of exploring jointly denoising and normal filtering. de Silva Edirimuni et al. (2023a) utilized a contrastive learning approach to train a decoder capable of outputting vectors encompassing both denoised points and normals. In contrast, PCDNF (Liu et al. 2023) and PN-Internet (Yi et al. 2024) adopt a multi-task learning framework, utilizing two interlinked network branches that concurrently restore noisy points and normals.

## Inferring Underlying Surfaces for Denoising

Certain denoising techniques aim to infer clean underlying surfaces from noisy point clouds for noise removal. TotalDenoising (Hermosilla, Ritschel, and Ropinski 2019), for instance, employs an unsupervised method to project noisy points onto a clean surface, utilizing spatial priors. DMRDenoise (Luo and Hu 2020) first identifies an underlying downsampled surface, which serves as a basis for reconstructing a clean surface. Following this, it upsamples points on the reconstructed surface to achieve denoising. Score (Luo and Hu 2021), on the other hand, commences by estimating the noise-convolved distribution score from the noisy input and subsequently employs gradient ascent optimization to restore point clouds. Zhao et al. (2022) introduced an approach that perturbs embedding features in the latent space, capturing the inherent commonalities to reconstruct the latent clean surface. Mao et al. (2022) developed PD-Flow, a

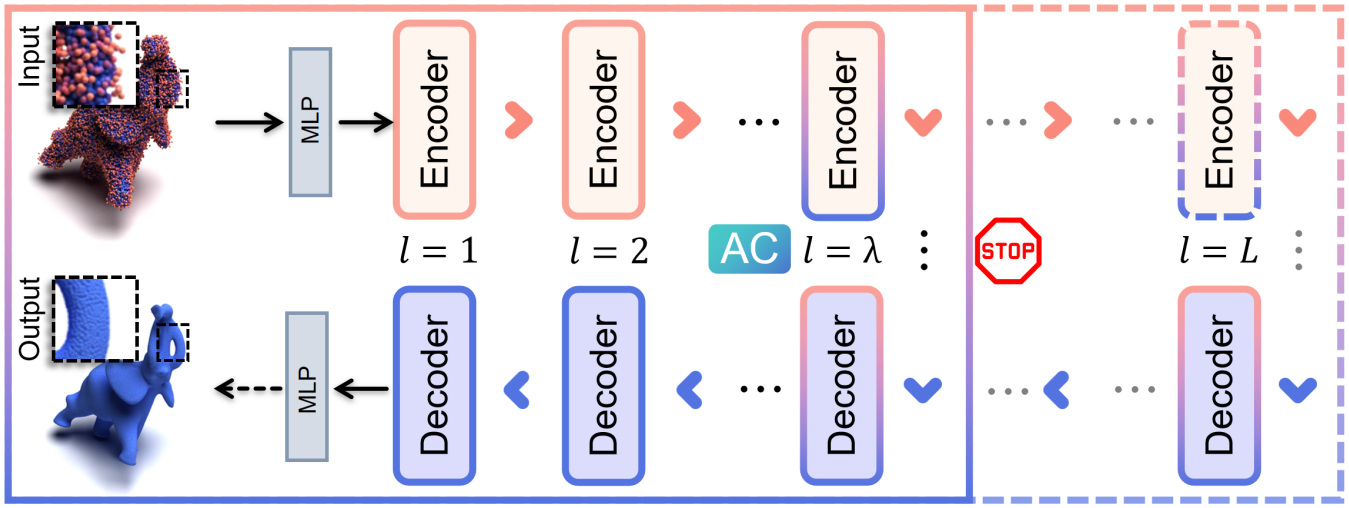


Figure 2: Architecture of the Adaptive Stop Denoising Network. The network consists of $L$ layers, with the encoder acting as the denoising unit. During this phase, our Adaptive Classifier predicts the recoverability factor $\rho$ of the current noisy point set, assesses the completion level of denoising, and dynamically chooses $\lambda$ to terminate the denoising process. This timely transition to the decoder accelerates the point cloud recovery process.

framework utilizing normalizing flows for the Euclidean-to-latent space mapping, aimed at restoring noisy point clouds. Chen et al. (2023) leveraged gradient fields to project noisy points towards the underlying clean surface, facilitating convergence.

## Method

### Main Backbone

Our Adaptive Stop Denoising Network (ASDN), grounded in the U-Net architecture, comprises multiple encoder and decoder modules.

To obtain the inputs of ASDN, we divide a given noisy point cloud $P$ into overlapped point cloud patches $\{\mathcal{P}\}$, where $\mathcal{P}$ is defined as: $\mathcal{P} = \{x_i | x_i \in \text{KNN}(x_{\text{center}}, P, n)\}$, where $\text{KNN}(x_{\text{center}}, P, k)$ represents finding the nearest $k$ points in $P$ to a reference point $x_{\text{center}}$. We set $n = 1000$ in our ASDN. As the denoised patches $\{\widehat{\mathcal{P}}\}$ have overlapping regions, we employ the stitching strategy outlined in (de Silva Edirimuni et al. 2023b) to generate the final denoised point cloud $\widehat{P}$.

The ASDN framework is depicted in Figure 2. Initially, the point cloud patch $\mathcal{P}$ undergoes MLP to capture high-dimensional information $f_0 \in \mathbb{R}^{n \times d}$. Then, our encoder, denoted $E_l$, takes $f_{l-1}$ as input and generates encoder feature $f_l$:

$$f_l = E_l(f_{l-1}) \in \mathbb{R}^{n_l \times d_l}, l = 1, 2, ..., \lambda, \quad (1)$$

where $n_l$ and $_l$ represent the number of points and the feature dimension at layer $l$ respectively, and $\lambda$ ranges from 2 to $L$, with $L$ being the total number of layers in ASDN, set to 4 in our configuration. After that, our decoder, denoted $D_l$, takes the decoder feature $h_l$ ($h_\lambda = f_\lambda$) and the encoder feature $f_{l-1}$ to produce the decoder feature $h_{l-1}$ as follows:

$$h_{l-1} = D_l(h_l, f_{l-1}), \quad l = 2, \ldots, \lambda. \quad (2)$$

Finally, we infer the displacement vectors $\Delta\mathcal{P}$ of the noisy patch $\mathcal{P}$ from the uppermost decoder feature by a MLP as

$$\Delta\mathcal{P} = \text{MLP}(h_1). \quad (3)$$

Thus, for $\mathcal{P}$, we can obtain the restored point cloud as

$$\widehat{\mathcal{P}} = \mathcal{P} + \Delta\mathcal{P}. \quad (4)$$

### Point Cloud Entropy and Recoverability Factor

Entropy, as a fundamental concept in information theory, quantifies the intrinsic unpredictability within data, reflecting data randomness and disorder (Shannon 1948). In order to evaluate the point cloud quality, we introduce the metric of point cloud entropy. This quantifies the spatial distribution and density of a given point cloud, thereby revealing structural complexity and inherent variability of the point distribution. Given a point set $\mathcal{P}$, the procedure of calculating point cloud entropy is outlined in Algorithm 1.

However, point cloud entropy cannot directly measure the denoising difficulty. To leverage point cloud entropy in denoising, we propose the **recoverability factor**, which is the ratio of noisy and clean point cloud entropies:

$$\rho = \frac{H(\mathcal{P})}{H(\overline{\mathcal{P}})}. \quad (5)$$

This factor, with its adaptable nature, inherits entropy characteristics and is tailored for denoising and addressing issues arising from diverse resolutions and point distributions. A lower $\rho$ signifies a more challenging recovery process, implying higher noise levels or greater complexity within the point cloud, and vice versa. The recoverability factor guides our classifier in deciding when to persist with denoising or to halt early, thus maintaining an optimal equilibrium between noise removal and preservation of geometry features.

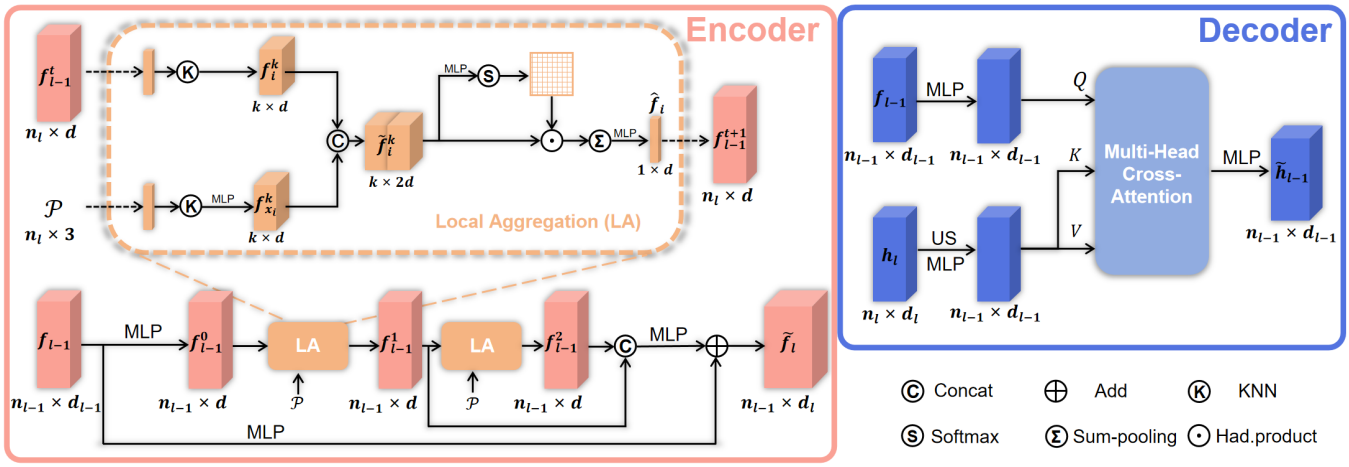


Figure 3: In our ASDN network, the encoder and decoder modules are specialized for point cloud denoising and recovery, respectively. The encoder's core is a residual expansion that integrates local feature aggregation (LA), while the decoder is equipped with cross-attention.

**Algorithm 1:** Point Cloud Entropy Calculation

**Input:** Point set $\mathcal{P} \in \mathbb{R}^{n\times 3}$, voxel size $v$
**Output:** Point cloud Entropy $H(\mathcal{P})$

**1. Voxelization:** Divide $\mathcal{P}$ into voxels $\{V_o\}$ with voxel size $v$.

**2. Point Counting:** For each voxel $V_o$ count its points:

$$\delta_o = \sum_i \mathbb{I}(x_i \in V_o),$$

where $\mathbb{I}$ is the indicator function that returns 1 if the point $x_i$ is within voxel $V_o$, or 0 otherwise.

**3. Probability Calculation:** Normalize the point counts:

$$q_o = \frac{\delta_o}{n}.$$

**4. Entropy Calculation:** Compute the Shannon entropy:

$$H(\mathcal{P}) = -\sum_{\{q_o | q_o > 0\}} q_o \log q_o.$$

To effectively map the recoverability factor $\rho$ to the stopping layer $\lambda$ within our U-Net architecture, we employ a non-linear transformation as follows:

$$\lambda = \lceil L - (L-1)\cdot \log(\gamma \cdot \rho + 1) \rceil, \lambda \in \{2, \ldots, L\}, \quad (6)$$

where $\gamma$ is a parameter to control the degree of non-linearity mapping. The ceiling function $\lceil \cdot \rceil$ is utilized to enforce $\lambda$ as an integer value, thereby ensuring that the stopping layer $\lambda$ adaptatively selection based on $\rho$.

## Adaptive Classifier

Our adaptive classifier network, constructed with EdgeConv layers from DGCNN (Wang et al. 2019), can infer the recoverability factor from the current point set in the denoising process. Then, it is straightforward to derive the stopping layer in our U-Net architecture from (6).

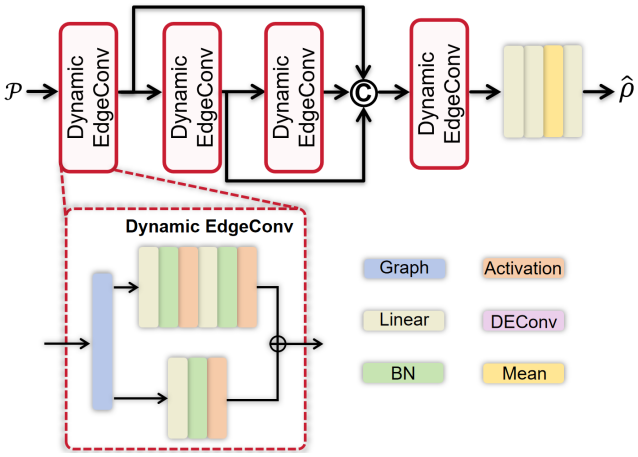


Figure 4: Our Adaptive Classifier, utilized to predict the recoverability factor $\hat{\rho}$ of the current point set.

Given a point set $\mathcal{P} \in \mathbb{R}^{n\times 3}$, a $k$-nearest neighbor graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}_0)$ is constructed based on the 3D Euclidean distance, with $\mathcal{V}$ as vertices and $\mathcal{E}_0$ as edges. The raw feature for each point $x_i \in \mathcal{P}$ is computed as follows:

$$g_i^0 = \sum_{j:(i,j)\in\mathcal{E}_0} \text{MLP}(x_i \parallel (x_i - x_j)), \quad (7)$$

where $(i, j)$ denotes an edge and $\parallel$ represents concatenation. The subsequent feature vectors are similarly defined as:

$$g_i^m = \sum_{j:(i,j)\in\mathcal{E}_m} \text{MLP}\left(g_i^{m-1} \parallel \left(g_i^{m-1} - g_j^{m-1}\right)\right), \quad (8)$$

with $m \in \{1, 2\}$ and $\mathcal{E}_m$ is the set of edges updated based on the features $\{g_i^{m-1}\}$ from the previous layer. These in-

termediate features are concatenated to form $g_i$ as

$$g_i = g_i^0 \parallel g_i^1 \parallel g_i^2. \tag{9}$$

A dynamic graph convolution operation is then applied to the concatenated features:

$$\mathbf{g} = \{ \sum_{j:(i,j)\in\mathcal{E}_3} \text{MLP}\left(g_i \parallel (g_i - g_j)\right)\}, \tag{10}$$

where $\mathcal{E}_3$ is the set of edges updated based on the features $\{g_i\}$. The notation $\{\cdot\}$ encapsulates the feature set for the point set $\mathcal{P}$. Subsequently, consecutive fully connected layers are applied to $\mathbf{g}$ to predict the recoverability factor, which is a scalar value:

$$\widehat{\rho} = \text{MLP}\left(\frac{1}{n}\sum_{i=1}^{n}\text{MLP}\left(\text{MLP}(\mathbf{g})\right)\right). \tag{11}$$

## Encoder

Our encoder is structured around residual expansion, utilizing local aggregation (LA) blocks as depicted in Figure 3. The $l$-th layer of our encoder, denoted as $E_l$, takes $f_{l-1}$ and a point set $\mathcal{P}_{l-1}$ as inputs and applies two sequential LA blocks to extract features $\{f_{l-1}^1, f_{l-1}^2\}$. This process is formulated as follows:

$$f_{l-1}^{t+1} = \text{LA}(f_{l-1}^t, \mathcal{P}), \quad f_{l-1}^0 = \text{MLP}(f_{l-1}), \quad t \in \{0, 1\}. \tag{12}$$

Then, we employ residual expansion to derive encoder feature $f_l$ of the $l$-th layer as follows:

$$\widetilde{f}_{l-1} = \text{MLP}(f_{l-1}^1 \parallel f_{l-1}^2) + \text{MLP}(f_{l-1}). \tag{13}$$

Subsequently, we performed the Farthest Point Sampling (FPS) algorithm sampling to obtain $f_l$ and $\mathcal{P}_l$.

**Local aggregation**. The LA architecture is detailed in the top panel of Figure 3. For each point $x_i \in \mathcal{P}_{l-1}$, we collect its KNN neighbors represented as $f_{x_i}^k \in \mathbb{R}^{k\times 3}$. Meanwhile, we extract the feature representation $f_i$ for each point and obtain its neighboring features $f_i^k \in \mathbb{R}^{k\times d}$. These features are concatenated to form $\widetilde{f}_i^k \in \mathbb{R}^{k\times 2d}$. To refine the feature $\widetilde{f}_i^k$, an MLP followed by a softmax function is applied to emphasize significant elements, expressed as:

$$\widehat{f}_i = \text{SumPooling}(\text{Softmax}(\text{MLP}(\widetilde{f}_i^k)) \odot \widetilde{f}_i^k), \tag{14}$$

where $\odot$ denotes the Hadamard product, SumPooling$(\cdot)$ squeezes the input features into a single feature by summation. After this operation for each point in the previous layer, the locally aggregated feature is obtained as $f_{l-1}^{t+1} = \{\widehat{f}_i\}_{i=1}^{n_l} \in \mathbb{R}^{n_l\times d_l}$.

## Decoder

Our decoder, as illustrated in Figure 3, is designed with a cross-attention mechanism. The $l$-th layer of the decoder, denoted by $D_l$, processes the input features $f_{l-1}$ and $h_l$ to produce the output decoder feature $h_{l-1}$.

The refinement of the decoder feature $h_l$ involves establishing feature correlations with $f_{l-1}$. We designate $f_{l-1}$ as the query and $h_l$ as the key-value pair and apply a linear transformation $\varphi(\cdot)$ to map them into distinct feature spaces:

$$\{Q, K, V\} = \{\varphi\left(f_{l-1}\right), \varphi\left(\text{US}\left(h_l\right)\right), \varphi\left(\text{US}\left(h_l\right)\right)\}, \tag{15}$$

where US$(\cdot)$ denotes an up-sampling operation. In particular, when $l = \lambda$, $h_l = f_l$.

Then, we employ multi-head cross attention (MHCA), as described in (Vaswani et al. 2017), to generate a feature that captures the correlation between the encoder and decoder features as follows:

$$\widetilde{h}_{l-1} = \text{MLP}(\text{MHCA}(Q, K, V)). \tag{16}$$

Finally, we use the skip connection to produce the decoder feature of the previous layer as

$$h_{l-1} = \text{MLP}\left(\widetilde{h}_{l-1} \parallel f_{l-1}\right) \in \mathbb{R}^{n_{l-1}\times d_{l-1}}. \tag{17}$$

## Loss Function

We utilize the InfoCD loss, as introduced in (Lin et al. 2023), to quantify the discrepancy between the denoised and ground-truth point clouds. The InfoCD loss is formulated as follows:

$$\mathcal{L}_{\text{InfoCD}} \propto \text{CD}(\widehat{\mathcal{P}}, \overline{\mathcal{P}}) + \frac{1}{\tau}\text{Reg}(\widehat{\mathcal{P}}, \overline{\mathcal{P}}), \tag{18}$$

where CD denotes the Chamfer distance, a metric that assesses the fidelity of denoised point cloud $\widehat{\mathcal{P}}$ to ground truth $\overline{\mathcal{P}}$. The second term serves as a regularization to promote the alignment of point cloud distributions, and the parameter $\frac{1}{\tau}$ is a scaling factor that adjusts the influence of the regularization term.

To train our classifier network, we define the adaptive classifying loss as follows:

$$\mathcal{L}_{\text{AE}}(\widehat{\rho}, \overline{\rho}) = |\widehat{\rho} - \overline{\rho}|^2, \tag{19}$$

where $\overline{\rho}$ is the ground-truth recoverability factor corresponding to $\widehat{\rho}$.

# Experiments

## Datasets and Settings

**Training dataset**. Our framework's training is conducted utilizing the PUNet dataset (Yu et al. 2018). The training dataset comprises 40 meshes, from which point clouds are derived at 10K, 30K, and 50K resolutions, totaling 120 training point clouds.Noisy data is generated by introducing Gaussian noise with a standard deviation that varies from 0.05 to 0.2 times the radius of the bounding sphere.

**Testing datasets**. We compare our method with competing approaches on the PUNet dataset (Yu et al. 2018) at 10K and 50K resolutions, yielding 40 point clouds. We also employ the real-scanned Kinect dataset (Wang, Liu, and Tong 2016) to assess the generalization of our method.The Paris-rue-Madame dataset (Serna et al. 2014), featuring real Paris street scenes scanned with a 3D mobile laser scanner, was evaluated for its real-world noise and serves as a solid benchmark for assessing our method's performance on actual data.

**Implementation**. Our method, developed in PyTorch, is trained on a NVIDIA GeForce RTX 3090 GPU, employing

| Method | 10K points | | | | | | 50K points | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 1% noise | | 2% noise | | 3% noise | | 1% noise | | 2% noise | | 3% noise | |
| | CD | P2M | CD | P2M | CD | P2M | CD | P2M | CD | P2M | CD | P2M |
| PCN (2020) | 3.515 | 1.148 | 7.467 | 3.965 | 13.067 | 8.737 | 1.049 | 0.346 | 1.447 | 0.608 | 2.289 | 1.285 |
| GPDNet (2020) | 3.780 | 1.337 | 8.007 | 4.426 | 13.482 | 9.114 | 1.913 | 1.037 | 5.021 | 3.736 | 9.705 | 7.998 |
| DMR (2020) | 4.482 | 1.722 | 4.982 | 2.115 | 5.892 | 2.846 | 1.162 | 0.469 | 1.566 | 0.800 | 2.432 | 1.528 |
| PointFilter (2021) | 2.461 | 0.443 | 3.534 | 0.862 | 5.089 | 1.849 | 0.758 | 0.182 | 0.907 | 0.251 | 1.599 | 0.710 |
| Score (2021) | 2.521 | 0.463 | 3.686 | 1.074 | 4.708 | 1.942 | 0.716 | 0.150 | 1.288 | 0.566 | 1.928 | 1.041 |
| PDFlow (2022) | 2.126 | 0.381 | 3.246 | 1.010 | 4.447 | 1.999 | 0.651 | 0.164 | 1.173 | 0.581 | 1.914 | 1.210 |
| DeepPSR (2023) | 2.353 | 0.306 | 3.350 | 0.734 | 4.075 | 1.242 | 0.649 | 0.076 | 0.997 | 0.296 | 1.344 | **0.531** |
| MAG (2023) | 2.498 | 0.459 | 3.629 | 1.054 | 4.686 | 1.923 | 0.706 | 0.146 | 1.287 | 0.557 | 1.931 | 1.045 |
| IterativePFN (2023b) | 2.056 | **0.218** | 3.043 | 0.555 | 4.241 | 1.376 | 0.605 | **0.059** | 0.803 | **0.182** | 1.971 | 1.012 |
| **Ours** | **1.880** | 0.219 | **2.581** | **0.515** | **3.079** | **0.946** | **0.515** | 0.069 | **0.676** | 0.188 | **1.304** | 0.655 |

Table 1: Quantitative evaluation on PUNet (Yu et al. 2018) with CD and P2M metrics ($\times 10^4$). The best and second-best results are highlighted in bold and underlined, respectively.

the Adam optimizer with a learning rate of $1 \times 10^{-4}$. Prior to training, point clouds are normalized to a unit sphere. Then, we employ the FPS and KNN algorithms to sample 1K-sized patches. We derive the entropy values for the adaptive classifier's training for the patches. The training process commenced with a pre-training phase for the adaptive classifier, followed by a joint training phase with the ASDN network, leveraging the pre-trained classifier.

| Method | Kinect | |
|---|---|---|
| | CD | P2M |
| PCN (2020) | 22.48 | 13.29 |
| GPDNet (2020) | 23.09 | 11.78 |
| PointFilter (2021) | 18.85 | **10.29** |
| Score (2021) | 19.66 | 11.08 |
| PDFlow (2022) | 19.87 | 11.69 |
| IterativePFN (2023b) | 18.69 | 10.92 |
| **Ours** | **18.22** | 10.38 |

Table 2: Quantitative evaluation on Kinect v2.

## Quantitative Results

We evaluate our method and the competing approaches on synthetic data (Yu et al. 2018), as shown in Table 1. The results demonstrate that our method achieves excellent performance on both sparse 10K and dense 50K point clouds. We found that at low resolution and low noise levels, PDFlow and IterativePFN achieve good results, with IterativePFN maintaining strong performance even on 50K point clouds under low noise levels. However, our method consistently outperforms the tested methods, especially when the noise level reaches higher levels, such as 3%. Under high noise levels, the results of other methods are less ideal, while our method significantly surpasses them, effectively demonstrating the efficacy of our approach. Next, we further evaluate our performance on the real-world scanned data (Wang, Liu, and Tong 2016) and list the results in Table 2. As we can see, our method outperforms the other methods in comparison in the sense that the CD values are significantly smaller.

## Qualitative Results

In Figure 5, we demonstrate the denoised results on PUNet (Yu et al. 2018), utilizing 50K points with a noise level of 2%. The color map illustrates the P2M distances, indicating that our denoised results closely approximate the ground truth. Furthermore, our method delivers visually pleasing results and more accurately recovers geometric features compared to all other competing methods. Our results are free from artifacts in smooth areas, a notable advancement over existing approaches.In Figure 6, we present results on a street scene (Serna et al. 2014). As we can see, our method not only effectively restores the details of the car but also significantly reduces noise and irregular shapes in the point cloud.

| Ablation | 1% noise | | 2% noise | | 3% noise | |
|---|---|---|---|---|---|---|
| | CD | P2M | CD | P2M | CD | P2M |
| w/o AC | 1.973 | 0.230 | 2.794 | 0.555 | 3.580 | 1.108 |
| $d = 64$ | 1.870 | 0.222 | 2.575 | 0.526 | 2.976 | 1.023 |
| $L = 3$ | 2.016 | 0.245 | 2.785 | 0.556 | 3.829 | 1.317 |
| $L = 5$ | **1.865** | 0.247 | **2.532** | 0.549 | **2.936** | 1.134 |
| **Ours** | 1.880 | **0.219** | 2.581 | **0.515** | 3.079 | **0.946** |

Table 3: Ablation study results on 10K points with different noise levels.

## Ablation Studies

To confirm the effectiveness of our early stopping denoising strategy, we conduct an experiment that removed a key component—the adaptive classifier (AC)—from our Adaptive Stopping Denoising Network (ASDN) and compared its performance to the full ASDN setup. This comparison helps us

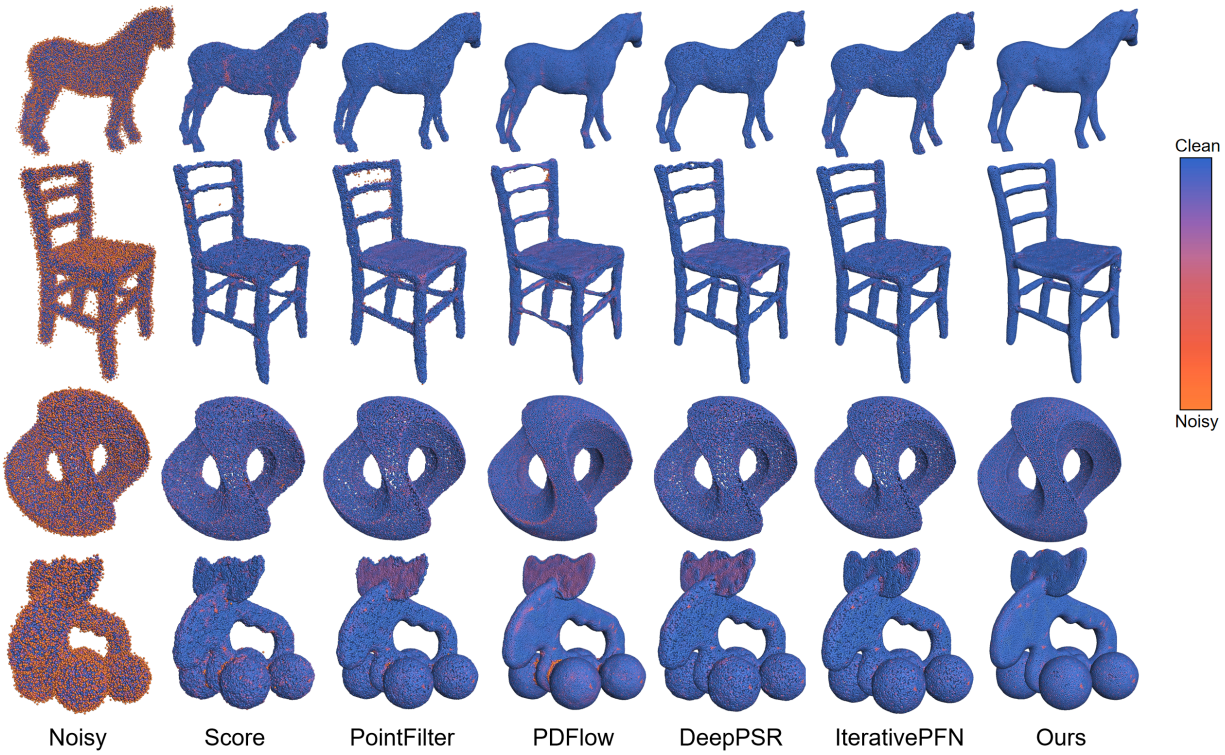


Figure 5: Visual results of point-wise P2M distance for shapes at 50K resolution with 2% Gaussian noise.

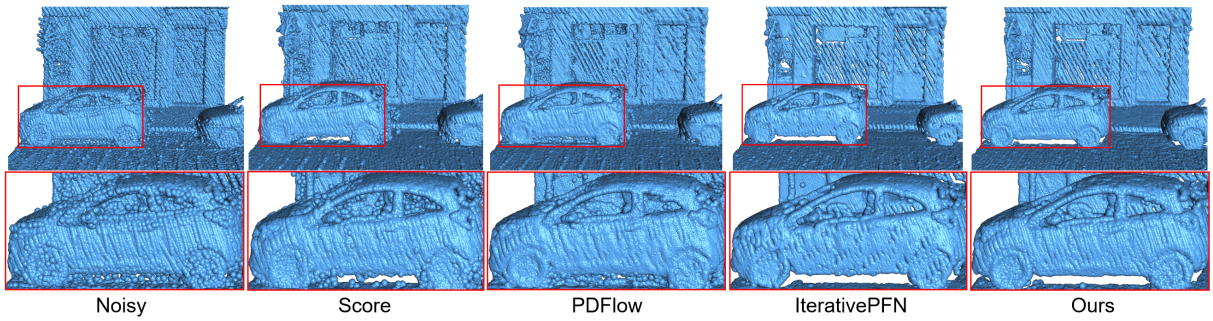


Figure 6: Visual results for a real-world point cloud scene.

understand how crucial the AC is for achieving better results during the denoising process. The comparison shows that our proposed early-stopping strategy can help our method produce better denoised results, as shown in Table 3. Subsequently, we expand the feature dimensions in each layer of our network to create a larger network variant, establishing a baseline with the initial feature dimension $f_0$ set at 32, then escalated to 64. In addition, we vary the number of layers $L$ in our ASDN network to explore its impact on performance, with $L$ originally set to 4. This series of ablation studies provides a comprehensive understanding of the contribution of each component to the overall network efficacy.

## Conclusion

In this paper, we have proposed a U-Net architecture incorporating an adaptive classifier for point cloud denoising. Our classifier employs a newly introduced recoverability factor to evaluate the denoising completion, enabling the adaptive stopping of the denoising process. In contrast to previous approaches, our method brings a novel perspective to address the potential issue of over-denoising on those already clean points during the denoising process. Experimental results demonstrate that our method outperforms state-of-the-art techniques. Our adaptive stop-denoising strategy shows promise in various point cloud tasks, such as completion and normal estimation. Future work will focus on analyzing the theoretical foundations of point cloud entropy and the recoverability factor.

## Acknowledgments

This work was supported by National Key Research and Development Program of China (2022YFB3904100).

## References


Charles, R. Q.; Su, H.; Kaichun, M.; and Guibas, L. J. 2017a. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 77–85.

Charles, R. Q.; Yi, L.; Su, H.; and Guibas, L. J. 2017b. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. In *Adv. Neural Inf. Process. Syst. (NeurIPS)*, volume 30, 5105–5114.

Chen, H.; Du, B.; Luo, S.; and Hu, W. 2023. Deep Point Set Resampling via Gradient Fields. *IEEE Trans. Pattern Anal. Mach. Intell.*, 45(3): 2913–2930.

Chen, H.; Wei, Z.; Li, X.; Xu, Y.; Wei, M.; and Wang, J. 2022. RePCD-Net: Feature-Aware Recurrent Point Cloud Denoising Network. *Int. J. Comput. Vision*, 130(3): 615–629.

de Silva Edirimuni, D.; Lu, X.; Li, G.; and Robles-Kelly, A. 2023a. Contrastive Learning for Joint Normal Estimation and Point Cloud Filtering. *IEEE Trans. Vis. Comput. Graph.*

de Silva Edirimuni, D.; Lu, X.; Shao, Z.; Li, G.; Robles-Kelly, A.; and He, Y. 2023b. IterativePFN: True Iterative Point Cloud Filtering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 13530–13539.

Hermosilla, P.; Ritschel, T.; and Ropinski, T. 2019. Total Denoising: Unsupervised learning of 3d point cloud cleaning. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 52–60.

Li, Y.; and Sheng, H. 2023. A single-stage point cloud cleaning network for outlier removal and denoising. *Pattern Recognit.*, 138: 109366.

Lin, F.; Yue, Y.; Zhang, Z.; Hou, S.; Yamada, K.; Kolachalama, V. B.; and Saligrama, V. 2023. InfoCD: A Contrastive Chamfer Distance Loss for Point Cloud Completion. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Liu, Z.; Zhao, Y.; Zhan, S.; Liu, Y.; Chen, R.; and He, Y. 2023. PCDNF: Revisiting Learning-based Point Cloud Denoising via Joint Normal Filtering. *IEEE Trans. Vis. Comput. Graph.*, 30(8): 5419–5436.

Lu, D.; Lu, X.; Sun, Y.; and Wang, J. 2020. Deep feature-preserving normal estimation for point cloud filtering. *Comput-Aided Des.*, 125: 102860.

Luo, S.; and Hu, W. 2020. Differentiable manifold reconstruction for point cloud denoising. In *Proceedings of the ACM International Conference on Multimedia (ACM MM)*, 1330–1338.

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

Mao, A.; Du, Z.; Wen, Y.-H.; Xuan, J.; and Liu, Y.-J. 2022. PD-Flow: A Point Cloud Denoising Framework with Normalizing Flows. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 13663.

Pistilli, F.; Fracastoro, G.; Valsesia, D.; and Magli, E. 2020. Learning graph-convolutional representations for point cloud denoising. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 103–118.

Rakotosaona, M.-J.; La Barbera, V.; Guerrero, P.; Mitra, N. J.; and Ovsjanikov, M. 2020. PointCleanNet: Learning to denoise and remove outliers from dense point clouds. *Comput. Graph. Forum*, 39(1): 185–203.

Serna, A.; Marcotegui, B.; Goulette, F.; and Deschaud, J.-E. 2014. Paris-rue-Madame database: a 3D mobile laser scanner dataset for benchmarking urban detection, segmentation and classification methods. In *International Conference on Pattern Recognition Applications and Methods (ICPRAM)*, 819–24.

Shannon, C. E. 1948. A mathematical theory of communication. *The Bell system technical journal*, 27(3): 379–423.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *Advances in neural information processing systems (NeurIPS)*, volume 30, 6000–6010.

Wang, P.-S.; Liu, Y.; and Tong, X. 2016. Mesh denoising via cascaded normal regression. *ACM Trans. Graph.*, 35(6): 232–1.

Wang, Y.; Sun, Y.; Liu, Z.; Sarma, S. E.; Bronstein, M. M.; and Solomon, J. M. 2019. Dynamic Graph CNN for Learning on Point Clouds. *ACM Trans. Graph.*, 38(5): 1–12.

Wei, M.; Chen, H.; Zhang, Y.; Xie, H.; Guo, Y.; and Wang, J. 2021. GeoDualCNN: Geometry-supporting Dual Convolutional Neural Network for Noisy Point Clouds. *IEEE Trans. Vis. Comput. Graph.*

Yi, C.; Wei, Z.; Qiu, J.; Chen, H.; Wang, J.; and Wei, M. 2024. PN-Internet: Point-and-Normal Interactive Network for Noisy Point Clouds. *IEEE Trans. Geosci. Remote Sens.*, 1–1.

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 (CVPR)*, 2790–2799.

Zhang, D.; Lu, X.; Qin, H.; and He, Y. 2021. Pointfilter: Point cloud filtering via encoder-decoder modeling. *IEEE Trans. Vis. Comput. Graph.*, 27(3): 2015–2027.

Zhang, Y.; Wei, M.; Zhu, L.; Shen, G.; Wang, F. L.; Qin, J.; and Wang, Q. 2024. Norest-Net: Normal Estimation Neural Network for 3-D Noisy Point Clouds. *IEEE Trans. Neural Netw. Learn. Syst.*

Zhao, T.; Gao, P.; Tian, T.; Ma, J.; and Tian, J. 2022. From Noise Addition to Denoising: A Self-Variation Capture Network for Point Cloud Optimization. *IEEE Trans. Vis. Comput. Graph.*

Zhao, Y.; Zheng, H.; Wang, Z.; Luo, J.; and Lam, E. Y. 2023. Point Cloud Denoising Via Momentum Ascent in Gradient Fields. In *2023 IEEE International Conference on Image Processing (ICIP)*, 161–165.

Zhou, H.; Chen, H.; Zhang, Y.; Wei, M.; Xie, H.; Wang, J.; Lu, T.; Qin, J.; and Zhang, X.-P. 2022. Refine-Net: Normal Refinement Neural Network for Noisy Point Clouds. *IEEE Trans. Pattern Anal. Mach. Intell.*