ITC 4/53

Information Technology and Control
Vol. 53 / No. 4/ 2024
pp. 1119-1138
DOI 10.5755/j01.itc.53.4.37310

Point Cloud Upsampling Network Incorporating Dynamic Graph Convolution and Multi-Head Attention

Received 2024/05/16 | Accepted after revision 2024/08/19

HOW TO CITE: Yang, X., Chen, F., Li, Z., Liu, G. (2024). Point Cloud Upsampling Network Incorporating Dynamic Graph Convolution and Multi-Head Attention. *Information Technology and Control*, 53(4), 1119-1138. https://doi.org/10.5755/j01.itc.53.4.37310

Point Cloud Upsampling Network Incorporating Dynamic Graph Convolution and Multi-Head Attention

Xiaoping Yang

Department of Information Physics and Engineering, School of Physics, Nanjing University of Science and Technology, Nanjing 210094, China;
College of Physics and Electronic Information Engineering, Guilin University of Technology, Guilin, Guangxi, 541006, China;
Guangxi Key Laboratory of Embedded Technology and Intelligent System, Guilin University of Technology, Guilin 541004, Guangxi, China

Fei Chen

College of Physics and Electronic Information Engineering, Guilin University of Technology, Guilin, Guangxi, 541006, China;
Guangxi Key Laboratory of Embedded Technology and Intelligent System, Guilin University of Technology, Guilin 541004, Guangxi, China

Zhenhua Li

Department of Information Physics and Engineering, School of Physics, Nanjing University of Science and Technology, Nanjing 210094, China

Guanghui Liu

Guilin Saipu Electronic Technology Limited Company, Guilin, Guangxi 541004, China

Corresponding authors: gutyxp@126.com (Yang); lizhenhua@njust.edu.cn (Li)

To address the problems that graph convolution uses a fixed graph structure, fails to capture dynamic or changing graph structure information, and is prone to bias by employing the same attention. A point-cloud upsampling network (DGCMSA-PU) incorporating Dynamic Graph Convolutional (DGCNN) and Multi-head Self-Attention (MHSA) is designed. DGCNN is utilised for up-sampling and a MHSA mechanism is incorporated to simultaneously fuse information from different attention heads. The edge relationships between nodes in the graph data are captured by edge convolution (EdgeConv), and the graph structure is dynamically constructed based on the relationships between nodes. Then the features of the point cloud are extracted by the three attention heads with different weights and different foci. Finally, an up-down-up structure is used to extend the features and reconstruct the 3D coordinates of the output point cloud. The superiority of DGCMSA-PU in the up-sampling task is verified through experiments comparing it with existing up-sampling networks, and the robustness of the network to noise and varying number of input point clouds, as well as the important role of the Multi Headed Attention module in the performance improvement of the network, are analysed through robustness and ablation experiments.

KEYWORDS: Dynamic graph convolution; Multi headed self attention mechanism; Point cloud up-sampling.

1. Introduction

Due to hardware and computational limitations in current 3D measurement technologies, directly acquired raw point clouds are often sparse, unevenly distributed, and may contain noise, leading to insufficient precision in the measured data and affecting subsequent work. To obtain dense and clean point cloud data, point cloud upsampling algorithms designed specifically to address this issue have become one of the hot topics in the field of point cloud research.

Traditional point cloud upsampling algorithms are optimization-based [18, 24, 9], relying on fitting local geometric information such as normal estimation and grid generation. However, these methods are often constrained by shape priors, thereby impacting the overall structure.

In recent years, the introduction of PointNet [2] and PointNet++ [3] has demonstrated the effectiveness and feasibility of using deep neural networks to process point clouds. Consequently, with the rapid development of deep learning technologies, there has been active exploration of various deep learning-based point cloud upsampling methods to address this challenge.

Yu et al. [14] introduced PU-Net, the first data-driven network for point cloud upsampling. PU-Net employs a multi-branch Multilayer Perceptron (MLP) to learn and expand multi-scale features for each point in the input point set, which are then used to reconstruct the upsampled point set. However, this approach extracts features from different downsampling levels separately for each point, resulting in reduced resolution and overlooking local details and neighbor information. Zhang et al. [27] combine single-point, local, and global features to process point clouds, thereby improving task accuracy. Additionally, Yu et al. [15] proposed EC-Net, the first edge-aware upsampling network, which learns features for each point in the input point set by regressing point coordinates and distances to edges. Consequently, EC-Net can handle sharp features detected by edge detection, enabling precise point set expansion and 3D reconstruction. Nevertheless, to annotate sharp edges, manual drawing of lines on each 3D grid is required during data preprocessing, which is a cumbersome and costly process in terms of both manual effort and time. Li et al. [21] presented PU-GAN, which learns a diverse distribution of upsampled points and extends point features based on GANs. However, by upscaling the point set through duplicating point features, it significantly restricts the variation of the final output point cloud. Additionally, the discriminator structure is complex and unstable. Inspired by adversarial networks, Zeng et al. [26] progressively extract low-dimensional latent vectors of features from point clouds in an incremental manner by cascading generative adversarial networks, completing point cloud upsampling through coordinate reconstruction. Similarly, Kulikajevas et al. [12] use a hybrid neural network composed of a single classifier network and multiple reconstruction networks to achieve point cloud upsampling in the form of 3D reconstruction. In their design, the reconstruction nodes in the multi-branch reconstruction network focus on the feature learning of specific objects or similar objects, making it easier to train new object types without retraining the entire network.

However, the aforementioned point upsampling networks treat different upsampling rates as independent tasks, requiring a one-to-one correspondence between the model and the upsampling rate during the network training phase. In practical applications, this directly results in inefficient storage and computational efficiency. To overcome this issue, Luo et al. [17] proposed a novel design for flexible-scale point cloud upsampling based on edge vector approximation, termed PU-EVA. PU-EVA encodes the connectivity of adjacent edges through affine combinations based on edge vectors and constrains the approximation error within the second-order term of the Taylor expansion. Furthermore, PU-EVA decouples the upsampling scale using a network architecture, enabling arbitrary upsampling rates in a single training session, albeit subject to limitations in network size and operational memory.

Aggregating point information is an indispensable step in point cloud deep learning today, and clustering algorithms are one of the common methods. For example, Ryselis et al. [22] use a scalable bounding box to aggregate points to reduce the inefficiency of independent domain searches. However, this method relies on the expansion step size, which may not be suitable for point clouds with different densities. Graph convolution [10] can process non-Euclidean data by constructing graph structures and aggregating graph information. In recent years, Graph Convolutional Networks (GCN) have been increasingly applied to point clouds, offering flexibility in learning features of nodes, edges, or subgraphs [19]. To better capture local multi-scale point information and aggregate neighbor information for each point, Qian et al. [7] proposed PU-GCN. They leverage the powerful capabilities of graphs and design two GCN-based modules in the upsampling module, namely Inception Dense GCN for feature extraction and NodeShuffle for feature expansion. This approach performs well in encoding local features and generating new points without the need for any additional tools (such as edges or normals). However, it may lose some global point cloud structural information to a certain extent. Nevertheless, Pierdicca et al. [20] use an improved KNN in the input layer to select neighboring points by utilizing raw coordinates, normalized coordinates, color features, and normal vector features, thereby enhancing task accuracy. This method combines geometric and radiometric properties, which may compensate for this drawback. With the development of attention mechanisms [5], Wang et al. [13] employed Graph Attention Convolution (GAC) for feature learning to address the issues of standard convolutional methods easily neglecting global structures and attention mechanisms overlooking local connections in point clouds. Similarly, Jing et al. [11] construct a topology using KNN to extract information, and then use an attention mechanism to select the most important features within the topology, thereby better representing different point cloud features. Hu [8] combined generative adversarial strategies with graph convolution in brain point cloud reconstruction, achieving spontaneous transformation from images to point clouds and recovering various details of the brain through hierarchical perception. Xiao [23] et al. designed a parallel multi-scale feature extraction module (PMA) and utilized edge convolution for feature expansion. Gao et al. [6] calculate attention coefficients based on edge convolution by considering local neighborhood correlations and local projection depth. Li et al. [16] employed a Transformer-based multi-stage learning framework for point cloud upsampling, utilizing a point-wise optimization network to adjust the spatial positions of each point after dense point generation. The application of graph convolution provides new insights into point cloud upsampling tasks. Graph convolution offers greater flexibility and can effectively handle non-Euclidean structured data. For sparse 3D point cloud data, graph convolution operations can be used to aggregate information from neighboring nodes, effectively utilizing relationships between nodes for feature extraction and expansion, ultimately reconstructing dense 3D point clouds.

GCN and DGCNN [25] are both deep learning models used for graph data processing. However, GCN utilize fixed graph structures, failing to capture dynamic or changing graph structural information. In contrast, DGCNN employs dynamic graph structures, reconstructing the graph structure in each convolutional layer based on the relationships between nodes, thereby better capturing both local and global information in graph data. This paper proposes a Point Cloud Upsampling Network named DGCMSA-PU, which combines DGCNN and MHSA. By integrating DGCNN with MHSA, the network captures features representations at different scales and conducts feature fusion, enhancing the richness and diversity of feature representations. Additionally, a top-down-bottom-up structure is employed in the feature expansion module to improve the granularity of generated points.

2. Methodology

2.1. DGCNN

2.1.1. Edge Convolution

let $X=\{x_1,\dots,x_n\}$ denote the point cloud consisting of n points, where each point contains coordinate information $x_i=(x_i,y_i,z_i)$. The local point cloud structure is represented by a directed graph $G=(\nu,\varepsilon)$, where $\nu=\{1,\dots,n\}$ denotes the vertices and $\varepsilon\subseteq\{\nu\times\nu\}$ denotes the edges.

Considering each point as a central point, we construct the neighborhood graph of the central points using K-nearest neighbors (KNN). Based on this graph structure, we calculate the features of adjacent points x_j for a point x_i using an MLP to obtain the edge feature e_{ij} as the graph feature. These edge features are aggregated to characterize the new feature x_i' of the central point x_i.

$$e_{ij}=h_\Theta\left(x_i,x_j\right). \tag{1}$$

Here, h_Θ represents a feature extraction function with a set of learnable parameters (e.g., MLP).

On the edge features, a channel-wise aggregation function operation denoted by * is used to define the edge convolution operation. Thus, the output of the edge convolution for the $i-th$ point is as follows:

$$x_i'=\underset{j:(i,j)\in\varepsilon}{*}h_\Theta\left(x_i,x_j\right). \tag{2}$$

To ensure permutation invariance of the point cloud, * requires to be independent of the input order. In edge convolution, symmetric functions such as summation (sum) or maximum (max) can be used for aggregation operations. There are four choices for the edge function:

1 When $x_1,\dots,x_n$ represent pixels in a two-dimensional image, and G represents connected regions of fixed size around each pixel, h_Θ selects weight multiplication, and * selects addition. Therefore, x_{im}' is the weighted sum of edge features:

$$x_{im}'=\sum_{j:(i,j)\in\varepsilon}\theta_m\cdot x_j, \tag{3}$$

where each θ_m has the same dimension as x, and $\cdot$ denotes the Euclidean inner product.

2 When $x_1,\dots,x_n$ represent scattered points in three-dimensional space, PointNet utilizes the edge function h_Θ:

$$h_\Theta\left(x_i,x_j\right)=\overline{h_\Theta}\left(x_i\right). \tag{4}$$

The above formulas encode the information of each point x_i in the global shape while ignoring the local neighborhood structure formed by x_i and neighboring points x_j. To capture local information, PointNet++ utilizes the multi-scale grouping (MSG) or multi-resolution grouping (MRG) method to group points within a certain radius neighborhood around the central point. Thus, we have:

$$h_\Theta\left(x_i,x_j\right)=\overline{h_\Theta}\left(x_j\right) \tag{5}$$

$$x_{im}'=\sum_{j\in V}\left(h_\theta\left(x_j\right)g\left(u\left(x_i,x_j\right)\right)\right). \tag{6}$$

where the function g is the Gaussian kernel, and the function u calculates the pairwise distances in Euclidean space within the MSG or MRG grouping neighborhood.

3 When h_Θ adopts the mean-centered subtraction of neighboring points from the central point, only the local neighborhood information is encoded, while the global information of the central point is lost. Thus, we have:

$$h_\Theta\left(x_i,x_j\right)=\overline{h_\Theta}\left(x_j-x_i\right). \tag{7}$$

Due to the limitations of the aforementioned three edge functions h_Θ, EdgeConv adopts a mean-centered subtraction x_j-x_i to capture local neighborhood information while preserving the coordinates of the region center x_i to capture global shape information. Thus, we have:

$$h_{\Theta}\left(x_i, x_j\right) = \overline{h_{\Theta}}\left(x_i, x_j - x_i\right). \tag{8}$$

2.1.2. Dynamic Update

To gradually acquire high-level feature information, convolutional neural networks typically consist of multiple convolutional layers. However, as convolution is performed layer by layer, the point cloud graph structure input to each layer may differ, resulting in different feature spaces for the output. Because of variations in feature space across dimensions, it is not reasonable to use the same GCN structure at each layer. Therefore, DGCNN adopts a different strategy, utilizing EdgeConv at each layer to construct local neighborhoods, whether in coordinate space or feature space. EdgeConv treats each point as a central point, computing edge features between it and its neighboring points, then aggregates these features to generate a new representation for the point. Feature extraction at each layer first involves computing pairwise distance matrices in either coordinate or feature space using EdgeConv, then constructing new local neighborhoods based on the principle of nearest neighbors, thereby forming different graph structures:

$$G^{(l)} = \left(\nu^{(l)}, \varepsilon^{(l)}\right), \tag{9}$$

where l denotes the number of layers in the network. When $l=1$, $G^{(l)}$ is represented by points $\nu^{(l)}$ in a 64-dimensional feature space and edges $\varepsilon^{(1)}$.

2.2. Multi-head Self-attention Mechanism

In the Transformer attention mechanism, each layer of the encoder performs two operations: self-attention and feed-forward. Each layer of the decoder performs three operations: self-attention, encoder-decoder attention, and feed-forward. Both self-attention and encoder-decoder attention utilize the MHSA [28] mechanism.

The MHSA mechanism is an improved technique based on the self-attention mechanism, primarily applied in sequence modeling and natural language processing tasks. The self-attention mechanism calculates the relative importance of each position in the input sequence with respect to other positions, thereby determining the degree of attention paid to different positions in the sequence. The multi-head attention mechanism further extends the capabilities of self-attention by allowing the model to perform multiple self-attention computations in different “heads” or subspaces, enabling it to capture more information and relationships at different levels.

The MHSA mechanism can be viewed as an extension of the single-head attention mechanism and is a widely used technique in natural language processing. It allows neural networks to focus on multiple aspects simultaneously when processing inputs. The MHSA mechanism utilizes multiple sets of Q, K, and V to obtain multiple sets of feature representations. This enables the network to fully leverage various information present in the input data to identify and extract features of different importance levels.

Specifically, the MHSA mechanism can be detailed into five steps:

1. Head Creation: Partition the input data into multiple parts and construct a separate attention head for each part.
2. Linear Transformation: Perform multiple linear transformations on the input to map it to different subspaces. Each subspace corresponds to a “head,” each with its own weight matrix and bias vector.
3. Attention Computation: Within each head, calculate attention weights between the query Q, key K, and value V, generating a weighted representation for each position. This process is similar to self-attention and can utilize dot-product attention or variants of other attention mechanisms.
4. Head Fusion: Aggregate and concatenate or average the attention-weighted outputs from each head to obtain the final multi-head attention representation. This captures the diversity and richness among different heads, providing comprehensive information.
5. Linear Transformation and Output: Combine the multi-head attention representation with another linear transformation to obtain the final output representation. This linear transformation can be a simple fully connected layer used to map the multi-head attention representation to the desired dimensionality.

The advantage of the multi-head attention mechanism lies in its ability to simultaneously focus on different levels and aspects of information and combine the diversity among different heads. It allows the model to learn and capture various relationships in different representation subspaces, enhancing the model's expressiveness and generalization performance. Multi-head self-attention can handle different types and levels of input data, improving model performance and accuracy by focusing on key information. Additionally, MHSA enables parallel computation, ensuring a considerably large receptive field without sacrificing computational efficiency.

2.3. DGCMSA-PU

To enhance point cloud upsampling, a point cloud upsampling network named DGCMSA-PU is designed, which integrates DGCNN and MHSA. The overall framework is illustrated in Figure 1.

The network primarily consists of three modules: the feature extraction module, the feature expansion module, and the coordinate reconstruction module. For the original input of an $N \times 3$ point cloud, given the massive number of points, it is partitioned into multiple Patch blocks. These Patch blocks serve as input to the MHSA-DGCNN module for feature extraction.

DGCNN, employing EdgeConv, captures the edge relationships between nodes in graph data. It updates the feature representation of central nodes by aggregating the features of neighboring nodes. Additionally, the inclusion of the MHSA enables the network to focus on the correlations between different nodes, weighting the features to enhance the network's feature extraction capabilities.

After feature extraction, resulting in $N \times C$ point cloud features, they are input into the feature expansion module. Utilizing the top-down-top feature expansion approach, GCN is employed to upsample point features, followed by downsampling to regress to the original features. The difference in features before and after upsampling is computed, and the difference tensor is upsampled and aggregated with the previous upsampling results. This process yields expanded features of $rN \times C'$, where r is the upsampling rate, C' is the feature channel dimension, and N is the number of training points.Finally, the dense $rN \times 3$ point cloud data output is obtained through the coordinate reconstruction module.

2.3.1. Feature Extraction Module

DGCNN utilizes EdgeConv to extract edge features, constructing a per-point k nearest neighbor graph as illustrated in Figure 2, where each edge node points towards the central node. EdgeConv is employed to extract edge features between the central node and its neighboring nodes. Then, an aggregation function is applied to update the central node using the edge features and information from the original $k+1$ points. The decentralized method is utilized to capture the global shape structure and the global features of nodes captured by the difference between edge nodes and central nodes in the local neighborhood information.

In the feature extraction module, the multi-head self-attention mechanism is incorporated to weight the features, summing the features from multiple sets

Figure 1
DGCMSA-PU

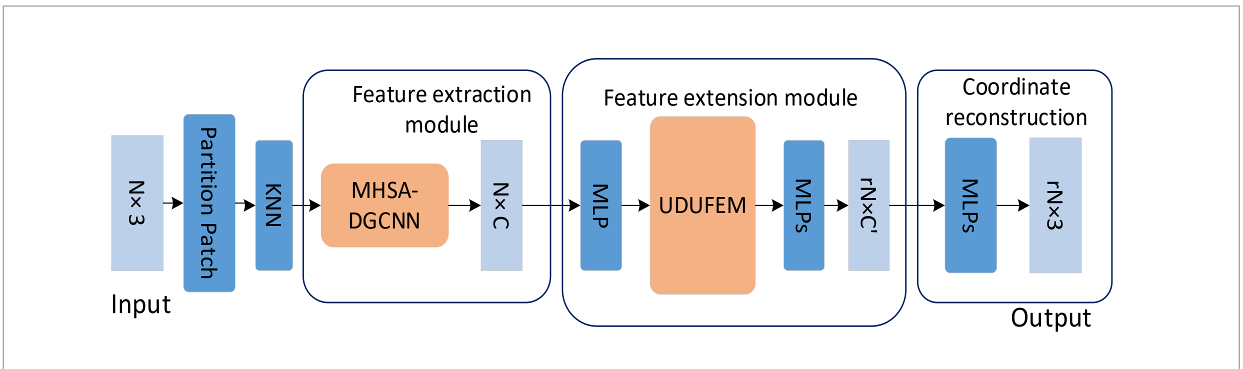
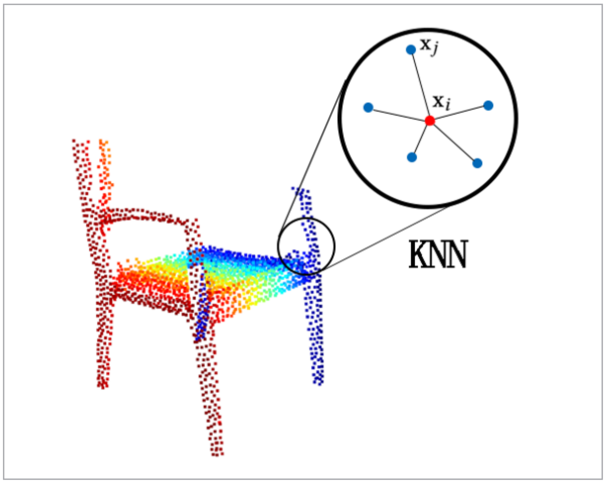


Figure 2
An undirected graph constructed by KNN



of self-attention networks. Finally, the output features of the attention module are obtained, as depicted in Figure 3.

The output features of a single self-attention mechanism are represented as $\mathrm{Attention}(Q,K,V)$:

$$\mathrm{Attention}(Q,K,V)=WV = f_{soft\max}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{10}$$

A single attention head has only one learned space, while multiple attention heads have multiple learned spaces:

$$head = Attention(QW_i^Q, KW_i^K, VW_i^V) \tag{11}$$

$$MultiHead(Q,K,V) = Concat(head_1,...,head_h)W^O \tag{12}$$

The attention mappings are divided into multiple attention mapping modules for Q, K, and V, using different weight matrices W_i^Q, W_i^K, and W_i^V. Each attention head has its own attention region. Finally, the attention mappings obtained from each attention head are merged. The overall weight matrix W^O determines the degree of attention for each attention head. By mapping Q, K, and V to different spaces and optimizing different parts of the features, different attention heads learn features. This operation balances the potential bias of using the same attention, making the feature representation more diverse.

The MHSA-DGCNN module, depicted in Figure 4, combines DGCNN as the foundation with MHSA. Unlike GCN, which utilizes a fixed graph structure, DGCNN employs a dynamic graph convolutional neural network for feature extraction. It not only utilizes the coordinate features of individual points but also fully leverages the local structural information of the point cloud and the geometric correlations between points. The feature extraction network consists of four EdgeConv layers, with an MHSA module added after each EdgeConv. The multi-head attention mechanism adaptsively weights features at both local and global scales, thereby capturing contextual information more effectively. Edge convolution enhances the feature representation of central nodes by propagating features from neighboring nodes. By combining edge convolution with the multi-head attention mechanism, the model leverages the ability of context awareness and feature integration to enhance its

Figure 3
MHSA

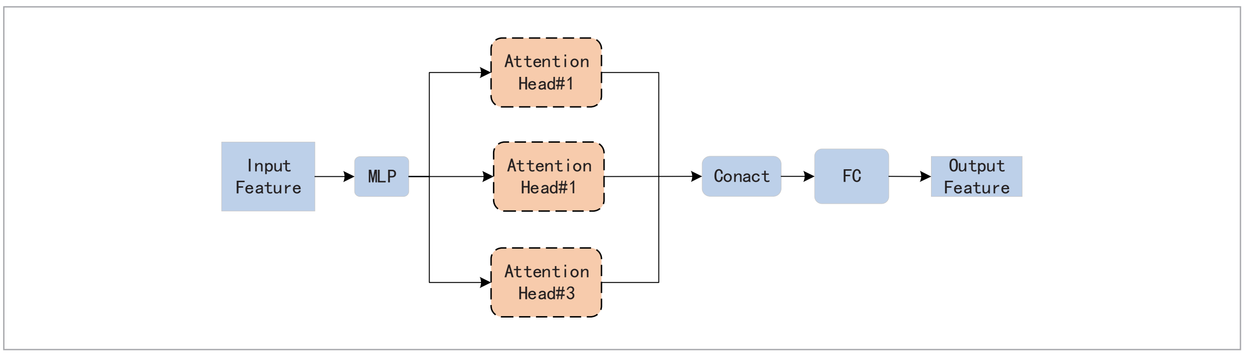
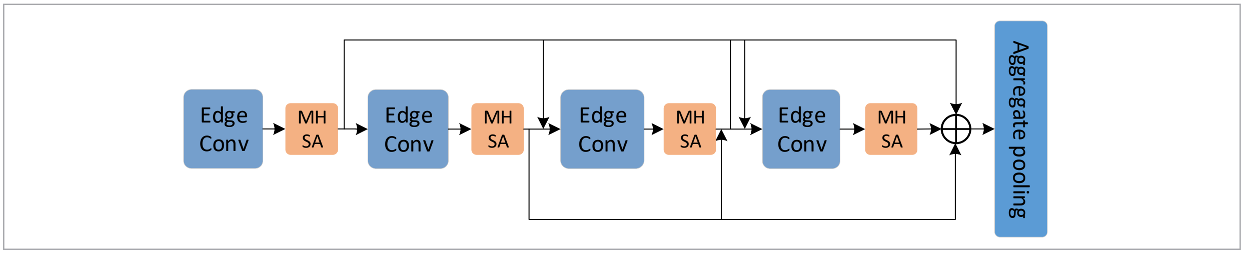


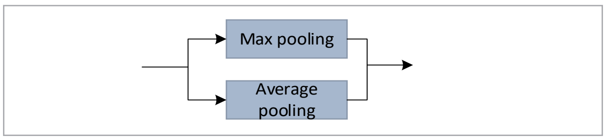
Figure 4
MHSA-DGCNN



ability to represent node features in graph data. Additionally, the residual network concept is introduced during feature extraction, incorporating residual connections to improve network performance, making the network easier to optimize, and alleviating to some extent the problem of gradient vanishing associated with increasing depth in deep neural networks. Subsequently, the point cloud undergoes symmetric pooling to generate global feature vectors.

An aggregation pooling layer, as illustrated in Figure 5, aggregates the global features. The aggregation pooling layer combines the features produced by the two channels using a parallel combination of max-pooling and average-pooling functions, thereby reducing feature loss.

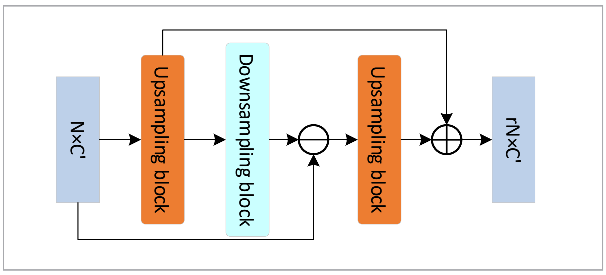
Figure 5
Aggregate pooling



5.3.2. Feature Extension Module

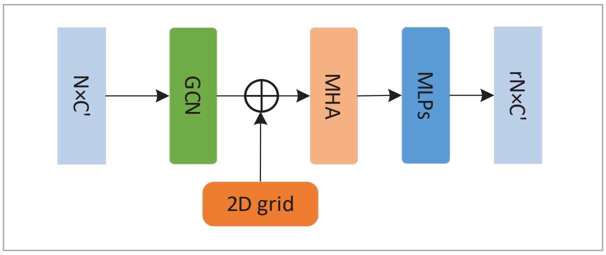
Inspired by PU-GAN and Transformer, a novel up-down-up feature expansion module is introduced in the feature expansion stage, as depicted in Figure 6. It consists of two parts: the upsampling block and the downsampling block. The upsampling block incorporates GCN, enabling it to encode spatial information from point neighborhoods and learn new features from latent space, instead of simply using convolutional neural networks. Moreover, a multi-head self-attention mechanism is applied to rapidly aggregate global spatial information and fine-tune the coordinates based on spatial information, thereby enhancing the feature expansion capability. Initially, point features undergo upsampling (after the MLP), generating upsampled features, followed by downsampling to regress to the original features. Instead of directly constructing the original point cloud, residual learning is applied to fine-tune the expanded features by computing the difference between the features before and after upsampling. This difference tensor is then inputted into the upsampling block and the MLP layer for upsampling. The resulting features are summed with the previously upsampled features. This step adopts a feature offset strategy to fine-tune the expanded features, avoiding cumbersome multi-step training while ensuring that the generated points do not deviate from the geometric surface of the patch block, thereby enhancing the granularity of the generated points.

Figure 6
Up-down-up feature extension module



To increase variation among repeated features, the direct replication of point features as employed in PU-Net is not utilized in the upsampling process. Instead, a grid mechanism inspired by FoldingNet [1] is employed, as depicted in Figure 7. After replicating the input point cloud features r times, local neighborhood information is captured using graph convolution,

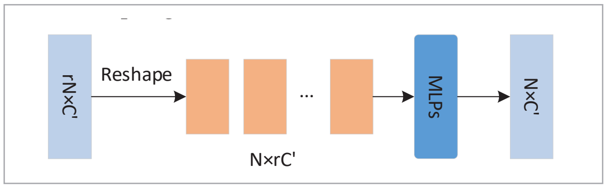
Figure 7
Upsampling block



leveraging learnable parameters. Subsequently, the two-dimensional grid mechanism from FoldingNet is applied, where a unique 2D vector is added after each replicated feature to augment its shape characteristics. This vector is appended to each feature vector corresponding to the respective point cloud, dispersing and distributing the replicated point cloud more uniformly. Moreover, the multi-head self-attention mechanism is employed to introduce context dependencies, optimizing different parts of the features, enhancing the integration of connected features, and facilitating self-correction, thus better incorporating the correlation between point features into the model. Through three attention heads with different weights and focuses, features of the point cloud are extracted, balancing the biases that may exist in using the same attention and making the representation of features more diverse. Finally, an MLP layer is used to regress point features to generate the output upsampled features.

The structure of the downsampling block is illustrated in Figure 8. To reduce the sampling of expanded features, the upsampled features are reshaped through downsampling operations, and then input into a set of MLP layers for regressing the original features.

Figure 8
Downsampling block



3. Results and Discussion

3.1. Datasets and Processing

The PU1K dataset is a novel large-scale dataset specifically created for point cloud upsampling tasks, as depicted in Figure 9. PU1K comprises 1147 3D models, divided into 1020 training samples and 127 test-

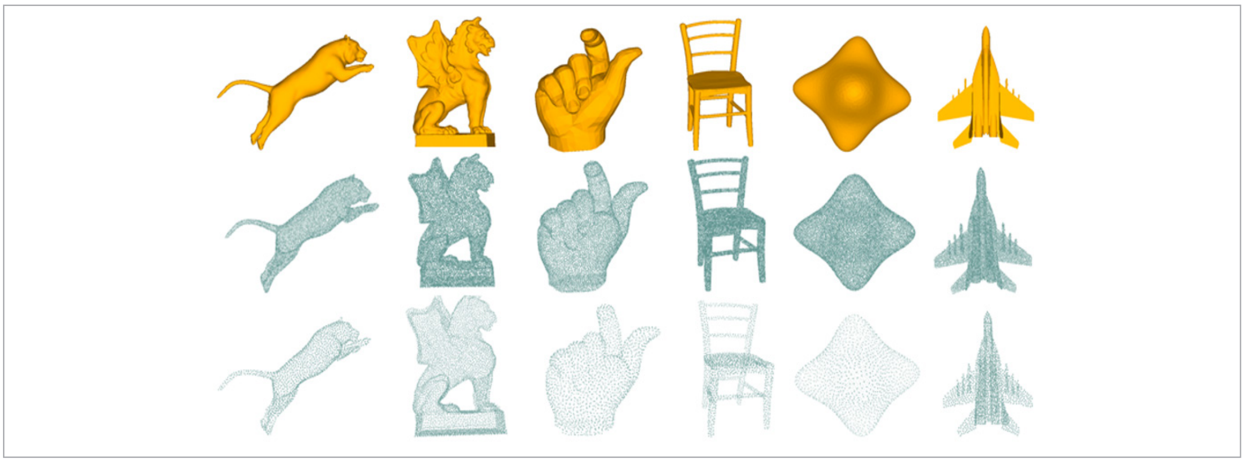
Figure 9
PU1K Dataset

Figure 10
Sydney Urban Objects Dataset

ing samples. The training set includes 120 3D models compiled from the PU-GAN dataset and 900 different models collected from ShapeNetCore. The testing set consists of 27 models from PU-GAN and over 100 models from ShapeNetCore. The models from ShapeNetCore are selected from 50 different categories. By randomly selecting 200 models from each category, a total of 1000 models with varying levels of shape complexity are obtained to encourage diversity. The Sydney Urban Objects Dataset [29] includes various common urban road objects scanned using Velodyne HDL-64E LIDAR, as shown in Figure 10.

Using Meshlab for point cloud processing and visualization is a common practice in the field. To prepare data for training and testing, surface patch block generation is the first step in data preprocessing. Intuitively, the point cloud should be partitioned into patch blocks, treating each patch as a single input when there are a large number of points within an object. Subsequently, Poisson disk sampling is applied to each patch to ensure coverage of the entire point cloud. This process generates pairs of original mesh grids and sampled point clouds (Input) along with ground truth point clouds. As illustrated in Figure 11, the first row represents the original mesh grid, the second row displays the Ground Truth point cloud (8192 points), and the third row shows the Input (2048 points).

Figure 11
Sampling results

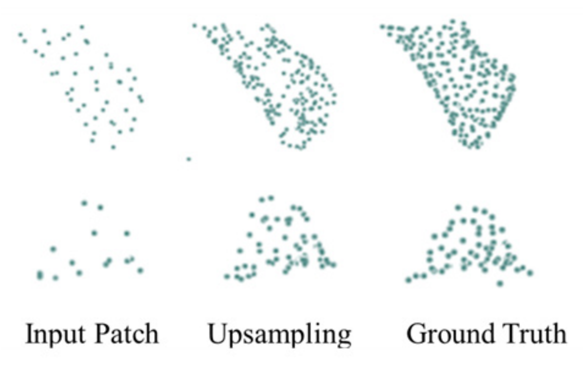


For the training data, 50 patch blocks are cropped from each 3D model as inputs to the network. In PU1K, a total of 51,000 training patch blocks are obtained. Each patch consists of 256 points as the low-resolution input and 1024 points as the ground truth.

For the testing data, each object is represented by 2048 points as the input point cloud, while the ground truth point cloud comprises 8192 points using an upsampling rate of $r = 4$.

During testing, the same processing approach as MPU and PU-GAN is employed, namely patch-by-patch. Firstly, M central points are selected using the farthest point sampling (FPS) method, and a fixed number of points are selected around each central point using the k-nearest neighbor algorithm, forming M clusters of point clouds. The upsample model is applied to each point cloud cluster separately to obtain the upsampled results, i.e., dense point clouds. Then, the overlapping patch outputs are merged according to the total number of points needed for upsampling (e.g., if the input point cloud contains 2048 points and requires a 4x upsampling, the resulting point cloud will contain 8192 points). Subsequently, the farthest point sampling algorithm is used again to resample the merged point cloud, resulting in the final point cloud output. The process is illustrated in Figure 12.

Figure 12
Sampling process



3.2. Loss Function

To ensure that the generated points are evenly distributed on the object surface, a combined loss is employed as the loss function. This loss function encompasses reconstruction loss (L_{rec}), repulsion loss (L_{rep}), and uniformity loss (L_{uni}). The loss function L_G is expressed as follows, where λ_{rec}, λ_{rep}, and λ_{uni} denote the weights:

$$L_G = \lambda_{rec} L_{rec} + \lambda_{rep} L_{rep} + \lambda_{uni} L_{uni} . \tag{13}$$

Reconstruction loss L_{rec}: Chamfer Distance (CD) can better capture the shape to encourage the output points to be located close to the underlying object surface. Therefore, CD is used as the reconstruction loss to assess the similarity between the output point set Output and the Ground Truth, represented as:

$$L_{rec} = D_{cd}(S_1, S_2) = d(S_1, S_2) + d(S_2, S_1) \tag{14}$$

$d(S_1, S_2)$ and $d(S_2, S_1)$ respectively denote the sum of minimum distances from any point in one point set to the other point set. A smaller value of $D_{cd}(S_1, S_2)$ indicates a better final reconstruction result.

Repulsion loss L_{rep}: Utilizing repulsion force loss to distribute the upsampled output points more uniformly rather than clustering around the original input points, expressed as:

$$L_{rep} = \sum_{i=0}^{rN} \sum_{i' \in K(i)} \eta\left(\| p_{i'} - p_i \|\right) w\left(\| p_{i'} - p_i \|\right), \tag{15}$$

where r represents the upsampling rate and N denotes the number of input points. For each point p_i, K nearest neighbor points $p_{i'}$ are selected, and their distances are computed as $d = \| p_{i'} - p_i \|$. The repulsion term is defined as $\eta(d) = -d$, and $\omega(d) = e^{-\frac{d^2}{h^2}}$ is a rapidly decaying weight function. When multiplied together, L_{rep} becomes large when the distance is too close or too far, so the generated points should maintain an appropriate distance to reduce the repulsion loss L_{rep}.

Uniformity loss L_{uni}: The repulsion loss ensures that the upsampled points are as separated as possible globally, but it does not guarantee uniform distribution of points locally. To ensure local point uniformity, a uniformity function is employed, expressed as:

$$L_{uni} = \sum_{j=1}^{M} U_{imbalance}(S_j) \cdot U_{clutter}(S_j) \tag{16}$$

$U_{imbalance}$ ensures the uniform distribution of upsampled points globally, while $U_{clutter}$ ensures uniform distribution within local neighborhoods.

3.3. Environment Configuration and Parameter Setting

To ensure fairness and accuracy in comparisons, and to minimize the impact of environmental factors, all methods in this study were trained and tested on the same computer hardware. Details are provided in Table 1.

Table 1
Experimental environment

System/Platform	Configuration/Version
OS	Ubuntu18.04
GPU	QuadroRTX5000(16GB)
CPU	AMD EPYC 7302
Memory	64GB
CUDA	11.3
Deep Learning Framework	PyTorch 1.10.2
Programming Language	Python 3.8

During training, the weights for each loss function are set as follows: $\lambda_{rec}=100$, $\lambda_{rep}=2$, and $\lambda_{uni}=10$. The batch size is set to 32, and the initial learning rate is 0.001. At the 60th epoch, the learning rate is reduced to 0.0001, and at the 100th epoch, it is further reduced to 0.00001. The total number of epochs for training is 120. Both training and testing processes employ an upsampling rate of 4.

3.4. Analysis of Experimental Results

3.4.1. Comparison with Existing Upsampling Algorithms

The up-sampling networks were trained and tested using the PU1k dataset. From the obtained up-sampling results, three objects of different shape levels were randomly selected, including simple and smooth objects as well as complex and highly detailed objects, to evaluate the performance of each network's upsampling results. Figure 13 shows the upsampling results.

From the up-sampled point clouds and their magnified details, it is evident that our method produces fewer outliers while preserving details closer to the real structure. Specifically, examining the details of the handle of the handbag (first row) indicates that the method successfully up-samples relatively smooth input point clouds, resulting in significantly fewer outliers compared to other methods, and achieving a smooth and evenly distributed surface on the object. Additionally, the armrests of the chair (second row) demonstrate the superiority of our method in maintaining geometric shapes, as the up-sampling results from PU-Net and PU-GAN introduce excessive noise. Although PU-GCN can roughly restore the overall contours, the effects are not sufficiently smooth, irregular, and exhibit poor edge restoration. Our method automatically updates the graph structure at each EdgeConv operation, capturing more correlations among the data, thereby enhancing the model's expressive power. Furthermore, through the MHSA module, it integrates different relationships and fea-

Figure 13
Upsampling results

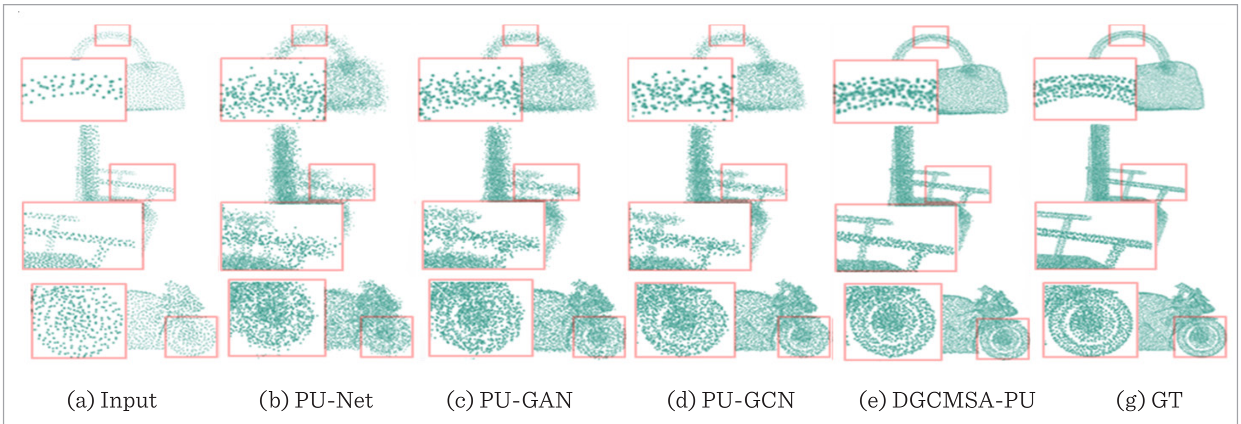


Table 2
Quantitative Evaluation of Upsampled Networks

NetWork	CD (10^{-3})	HD (10^{-3})	P2F (10^{-3})	Uni (10^{-3})	Time (ms)
PU-Net	3.135	16.634	6.392	22.136	10.081
PU-GAN	1.986	14.320	3.531	18.092	17.325
PU-GCN	0.815	13.682	4.894	15.342	12.612
DGCMSA-PU	0.706	10.629	3.870	11.378	10.331

ture representations during the feature extraction process, enabling the restoration of more shapes and edge details.

Observations of the motorcycle's wheels (third row) reveal that other methods tend to overlook the object's own geometric shapes during the up-sampling process. In contrast, our method can accurately restore the object's geometric shape, resulting in clearer descriptions of wheel contour features, fewer outliers, better up-sampling effects on fine structures such as brake discs, and a point cloud structure after up-sampling that is closer to the real structure.

In the quantitative comparison, the proposed model was evaluated against three up-sampling networks, namely PU-Net, PU-GAN, and PU-GCN, using the same evaluation metrics. The results are summarized in Table 2.

These evaluation metrics, in addition to CD mentioned in Section 3.2, include Hausdorff Distance (HD), Point to Surface (P2F), and Uniformity (Uni). Their computation methods are as follows:

$$HD(A,B) = \max(\sup_{a\in A} \inf_{b\in B} \|a-b\|_2 , \sup_{b\in B} \inf_{a\in A} \|a-b\|_2) \tag{17}$$

$$P2F(A,B) = \frac{1}{|A|}\sum_{a\in A, b\in B}\|a-b\|_2 \tag{18}$$

$$\mathrm{Uni}(A) = \frac{1}{|A|}\sum_{a\in A}\mathrm{Var}(a) \tag{19}$$

According to the objective evaluation metrics, compared to PU-GCN, CD, HD, and P2F decreased by 0.109×10^{-3}, 3.053×10^{-3}, and 1.024×10^{-3}, respectively. The inclusion of uniform loss in the joint loss function improved the uniformity of the generated points, resulting in a decrease of 3.964×10^{-3} in the Uni metric. These experimental results validate the effectiveness of the feature extraction module, which integrates DGCNN and the MHSA, as well as the up-down-up feature expansion module, in feature extraction and capturing spatial structures.

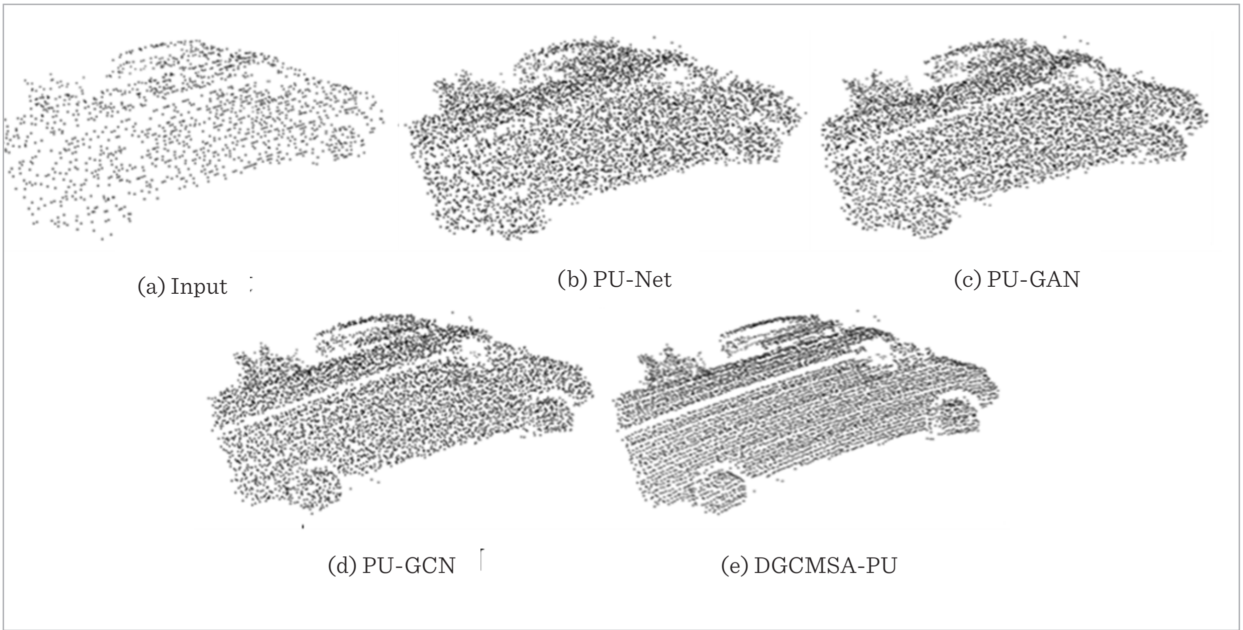
In the quantitative comparison, the proposed model was evaluated against three up-sampling networks, namely PU-Net, PU-GAN, and PU-GCN, using the same evaluation metrics. The results are summarized in Table 2. According to the objective evaluation metrics, compared to PU-GCN, CD, HD, and P2F decreased by 0.109×10^{-3}, 3.053×10^{-3}, and 1.024×10^{-3}, respectively. The inclusion of uniform loss in the joint loss function improved the uniformity of the generated points, resulting in a decrease of 3.964×10^{-3} in the Uni metric. These experimental results validate the effectiveness of the feature extraction module, which integrates DGCNN and the MHSA, as well as the up-down-up feature expansion module, in feature extraction and capturing spatial structures.

3.4.2. Upsampling Results of Real Scanning Data from On-board Lidar

Using the models trained on the PU1K dataset, we upsampled the point clouds from the Sydney Urban Objects dataset and compared the upsampling results with those of other networks. This evaluation was performed on both individual objects scanned by the vehicle-mounted LiDAR and complete 360-degree scan point clouds.

From Figure 14, it is evident that the point cloud data obtained by vehicle-mounted LiDAR scanning of a car in a real-world scenario is sparse, blurry, and subject to occlusion. Clearly, our proposed point cloud upsampling method significantly improves the resolution of the radar-scanned point cloud and outperforms other networks in detail representation. Spe-

Figure 14
The on-board lidar scans the upsampling results of a single object



cifically, our method can effectively reconstruct the outline of the vehicle and generate an adequate number of feature points.

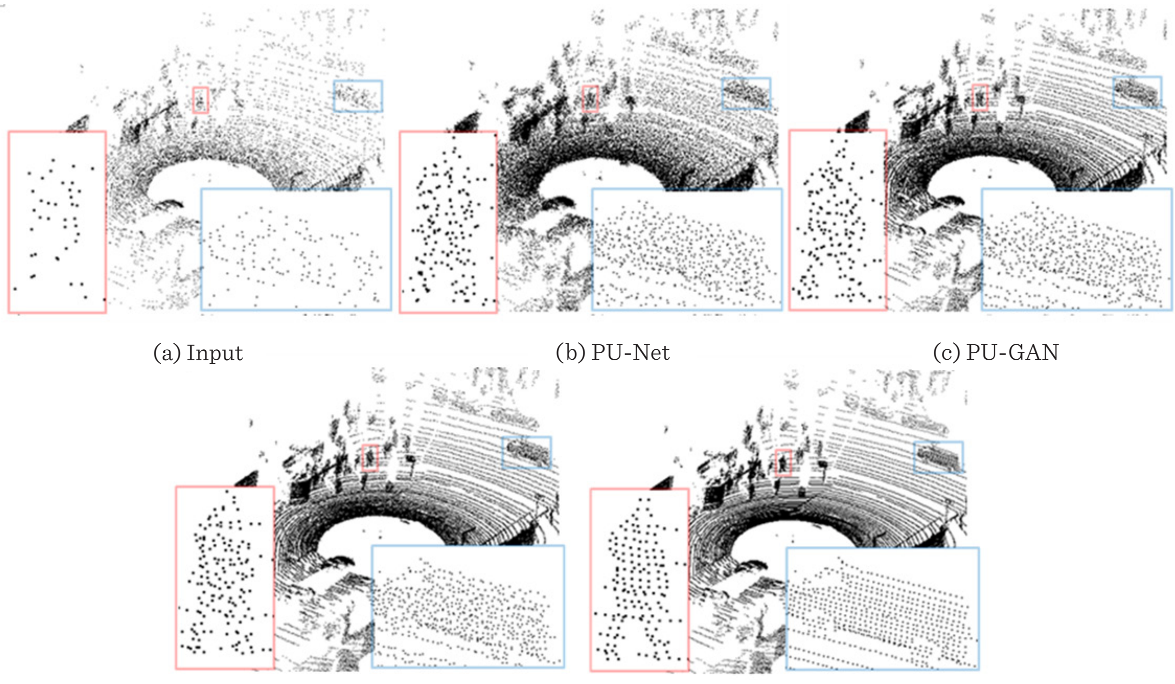
When the input point cloud contains few points, PU-Net and PU-GAN can only expand the number of points without effectively reconstructing the geometric surface information expected from the original point cloud model. Consequently, they exhibit poor performance in handling details such as the vehicle's wheels, the windows of the cabin, and the roof rack, with the shapes of windows and wheels being almost indistinguishable. While PU-GCN can recover some details, such as the outline of the windows and the circular rear wheels of the vehicle, they still fall short in geometric detail and exhibit uneven point distribution.

Our method utilizes DGCNN for feature extraction, dynamically constructing a graph structure based on the input data's features to better capture inter-data relationships. Furthermore, by integrating multi-head attention after each EdgeConv layer, each attention head can focus on different relationships and feature representations, thereby enhancing the model's expressiveness and generalization performance. Even for sparse point cloud inputs, our method successfully preserves local fine-grained details, including the retention of holes between the wheels and the body of the car.

The complete point cloud image scanned by the on-board LiDAR during driving is used as the input point cloud for upsampling. Figure 15 shows the upsampling results, indicating a significant improvement in the resolution of LiDAR-scanned point clouds achieved by the proposed method. From the enlarged details of the output point clouds, it can be observed that the input point cloud data for vehicles and pedestrians are represented by sparse and irregular points, making it difficult to discern contours and details. The output point clouds from PU-Net and PU-GAN remain scattered, failing to capture the contours of objects. While the outputs from PU-GCN reveal some outlines, they still struggle to distinguish the shapes of objects. In comparison, the upsampling effect of our network is superior, distinctly outlining both human and vehicle profiles. The experiments demonstrate that our network achieves good information recovery for sparse input point clouds from real LiDAR scans, successfully reconstructing the shapes of vehicles and pedestrians on the road. The restoration of such scene information is crucial for applications based on LiDAR-scanned point clouds.

Figure 15

Upsampling results of point clouds scanned by on-board lidar 360



(a) Input (b) PU-Net (c) PU-GAN (d) PU-GCN (e) DGCMSA-PU

3.4.3. Study of Robustness

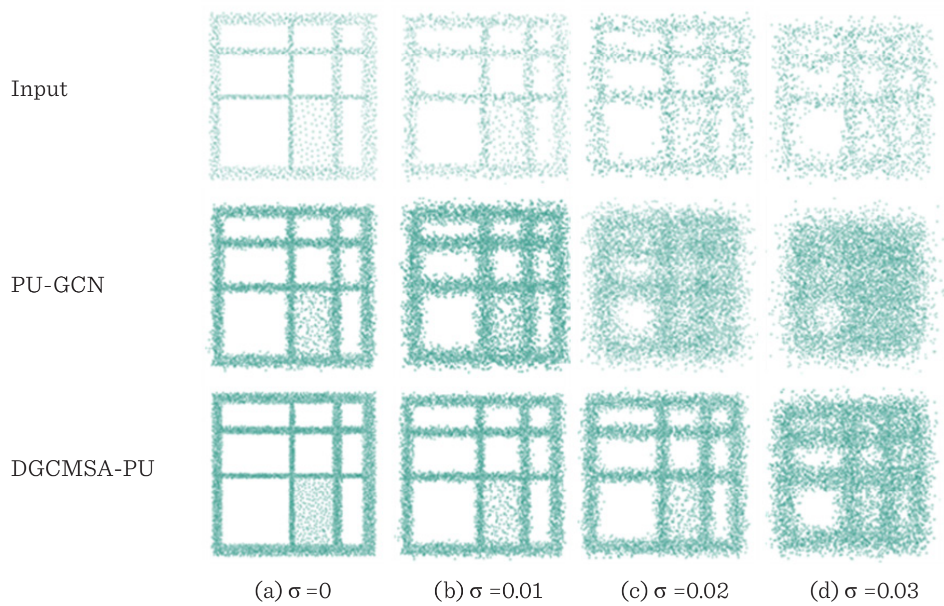
To verify the robustness of our proposed method against noise interference, we perturbed the input point clouds with additive Gaussian noise at different proportions and then performed upsampling on them. As shown in Figure 16, the first row represents the input point clouds, while the second and third rows display the upsampled point clouds generated by different networks. From left to right, the noise proportions added to the input data are: no noise, Gaussian noise σ with 0.01, 0.02, and 0.03.

The MHSA module in the proposed DGCMSA-PU network can perform multiple attention operations in parallel. Even if one attention head fails to capture effective features, other attention heads can still provide useful information, thus alleviating the limitations of a single attention mechanism and enhancing the network's robustness. The results demonstrate that our proposed network outperforms other upsampling methods under the influence of noise at different proportions, producing fewer outliers and preserving finer details. As the noise level increases, the differences become more pronounced. For instance, in the window frame depicted in the figure, the upsampling result of PU-GCN exhibits blurred contours, more outliers, and lacks uniformity and smoothness. Moreover, with increasing noise proportions, the boundaries between window panes become increasingly blurred, making it difficult to discern their specific shapes. In contrast, the upsampling results of our proposed DGCMSA-PU network, although somewhat blurred due to noise, manage to preserve the basic shape and contour, demonstrating satisfactory visualization effects. The experimental results indicate that the upsampling network proposed in this paper exhibits good robustness to noise, mitigating the impact of various noise sources in real scanning scenarios.

In point cloud processing tasks, the number of input points may vary due to factors such as sampling density, scene complexity, or data collection methods. To

Figure 16

Effect of different jitter coefficients on the upsampled network



ensure the robustness of the model in practical applications, its performance needs to be evaluated under different numbers of input point clouds. By varying the number of input points, different densities of point cloud data can be simulated to assess the model's performance under conditions of fewer or more points, thereby verifying its ability to handle point clouds of different densities. Specifically, input point clouds of 256 points, 512 points, and 1024 points were used for upsampling by the network, and the resulting upsampling results were compared and analyzed. The experimental results are as follows.

From the quantitative evaluation results in Tables 3, 4, and 5, it is evident that even with a smaller number of input points, the upsampling performance of the proposed network in this paper remains superior to that of other networks. Compared to PU-Net, PU-GAN, and PU-GCN, almost all evaluation metrics show better results. As the number of input points increases, the performance of the upsampling network improves. When only 256 points are input, compared to PU-GCN, the proposed network in this paper exhibits a decrease of 0.457×10^{-3} in CD, 3.581×10^{-3} in HD, and 0.905×10^{-3} in P2F. Additionally, the uniformity of the upsampling results is also superior,

Table 3

Upsampling result with Input=256

NetWork	CD (10^{-3})	HD (10^{-3})	P2F (10^{-3})	Uni (10^{-3})	Time (ms)
PU-Net	4.528	45.432	15.634	50.668	1.599
PU-GAN	3.836	38.162	8.631	33.206	3.036
PU-GCN	3.271	31.264	7.324	36.786	2.324
DGCM-SA-PU	2.712	26.354	6.063	24.327	1.712

Table 4

Upsampling result with Input=512

NetWork	CD (10^{-3})	HD (10^{-3})	P2F (10^{-3})	Uni (10^{-3})	Time (ms)
PU-Net	2.998	35.241	11.189	40.131	2.387
PU-GAN	2.734	29.564	8.136	21.135	6.331
PU-GCN	2.096	21.862	6.746	24.413	5.301
DGCM-SA-PU	1.837	20.136	5.638	16.324	3.135

Table 5

Upsampling result with Input=1024

NetWork	CD (10^{-3})	HD (10^{-3})	P2F (10^{-3})	Uni (10^{-3})	Time (ms)
PU-Net	1.884	25.320	8.305	37.429	6.324
PU-GAN	1.602	20.631	5.364	11.841	12.574
PU-GCN	1.424	15.932	4.364	13.362	10.058
DGCM-SA-PU	0.967	12.351	3.459	9.226	7.669

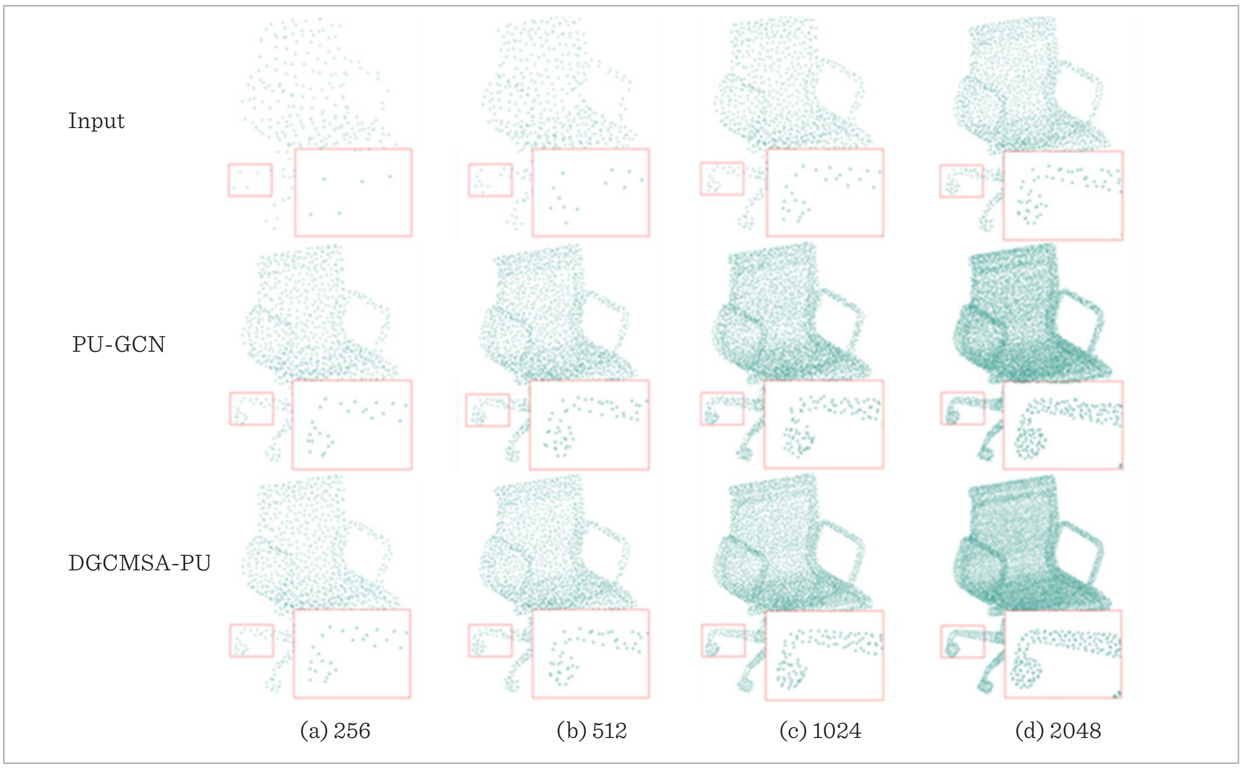
with Uni decreasing by 4.136×10^{-3}. The experiments demonstrate that the network exhibits better robustness to different densities of input point clouds, even achieving higher-quality upsampling point clouds when the input point cloud density is low.

From the quantitative evaluation results in Tables 3, 4, and 5, it is evident that even with a smaller number of input points, the upsampling performance of the proposed network in this paper remains superior to that of other networks. Compared to PU-Net, PU-GAN, and PU-GCN, almost all evaluation metrics show better results. As the number of input points increases, the performance of the upsampling network improves. When only 256 points are input, compared to PU-GCN, the proposed network in this paper exhibits a decrease of 0.457×10^{-3} in CD, 3.581×10^{-3} in HD, and 0.905×10^{-3} in P2F. Additionally, the uniformity of the upsampling results is also superior, with Uni decreasing by 4.136×10^{-3}. The experiments demonstrate that the network exhibits better robustness to different densities of input point clouds, even achieving higher-quality upsampling point clouds when the input point cloud density is low.

The visual experimental results of the network's robustness to different input point cloud densities are depicted in Figure 17. Even with a minimal number of input points, the network proposed in this paper is capable of generating higher-quality upsampling point clouds, with minimal occurrence of outliers and retention of details closer to the real structure. As the input point cloud density increases, the sampling results approach the Ground Truth more closely. From

Figure 17

Upsampling results of input points with different densities



the enlarged details of the chair, it can be observed that the network successfully reconstructs detailed surface features of the chair's wheels, with uniformly distributed generated points on the surface and minimal scattered points, enabling a clear depiction of the wheel's specific shape. The experiments demonstrate the network's good robustness to point clouds with different input densities, producing sampling results with fewer outliers and restoring the original geometric shape, closely resembling the Ground Truth. This network can be effectively applied to the upsampling task of sparse point clouds obtained from real vehicle-mounted LiDAR scans.

3.4.4. Ablation Experiment

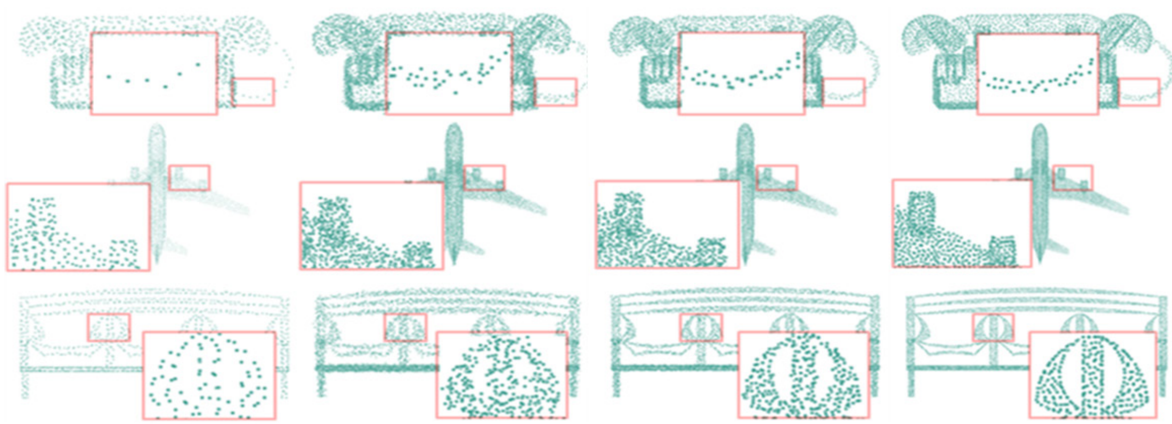
To validate the contribution of MHSA in the network model, ablation experiments were conducted where the multi-head self-attention (MHSA) module was removed from the feature extraction module. The network was retrained without MHSA, and the same dataset was used for testing. The upsampling results were then compared with the previous results. The visual results are shown in Figure 18, and the quantitative results are presented in Table 6.

From the close-up regions of the telephone (first row), airplane (second row), and chair (third row), it is evident that the network model proposed in this paper exhibits fewer outliers and more specific contour information in the point cloud models. When the multi-head self-attention module is removed, the upsampling results for the telephone lines (first row) exhibit blurred and scattered contours. However, with the inclusion of the multi-head self-attention module (MHSA), the specific shape of the telephone lines is better restored. For the details of the airplane engine, the addition of the MHSA module optimizes feature representation, as multiple attention heads can learn different features. This reduces the impact of outliers on contour information, resulting in clearer contour feature descriptions in the point cloud model. Similarly, for the details of the chair, better restoration is achieved, with more specific detail feature representations and upsampling results closer to the Ground Truth (GT). Quantitative evaluation results also indicate that when the multi-head self-attention (MHSA) module is removed, the performance of HD and P2F metrics significantly decreases, resulting in poorer upsampling results. With the inclusion of the multi-head self-attention (MHSA) module in the network, the performance decreases by 0.053×10^{-3} in CD,

Table 6
Results of ablation experiments

MHSA	CD(10^{-3})	HD(10^{-3})	P2F(10^{-3})	Uni(10^{-3})	Time(ms)
	0.675	9.961	2.634	8.456	10.246
√	0.622	8.705	2.475	7.712	11.327

Figure 18
Visualization of ablation experiment results



(a) Input (b) NO MHSA (c) Complete network (d) Ground Truth

1.256×10^{-3} in HD, and 0.159×10^{-3} in P2F. Although there is some increase in processing time, the uniformity is improved. These results demonstrate that the multi-head self-attention (MHSA) module diversifies feature representation, leading to a significant enhancement in upsampling performance.

4. Conclusion

This paper proposes a point cloud upsampling network called DGCMSA-PU, which integrates dynamic graph convolution and multi-head self-attention. Firstly, the overall structure and implementation process of the network are analyzed, followed by a detailed explanation of the feature extraction module and the up-down-up feature expansion module that combines dynamic graph convolution and multi-head self-attention. DGCNN enhances feature representation by capturing edge relationships between nodes through edge convolutions and propagating feature information from neighboring nodes to the central node. The multi-head attention mechanism integrates information from different heads simultaneously, enabling comprehensive information exchange and integration. The up-down-up feature expansion structure captures both global semantic information and local details, thereby enriching and diversifying feature representation and improving the granularity of generated points.

Experimental comparisons with existing upsampling networks demonstrate that DGCMSA-PU outperforms other networks in almost all evaluation metrics. Subsequently, upsampling experiments are conducted on real-world vehicle-mounted LiDAR scan data to further validate the generalization performance of the proposed method in real scenes. Robustness studies indicate that DGCMSA-PU exhibits good robustness to noise and different point inputs.

Finally, ablation experiments are conducted to verify the importance of each module in the entire upsampling process. All experimental results confirm the practicality and effectiveness of the proposed network, DGCMSA-PU, laying the foundation for its practical application.

For example, in a typical SLAM system, the raw point cloud data acquired by sensors needs to undergo preprocessing and feature extraction before being used for pose estimation and map updating. Our upsampling technique can enhance the raw point cloud data, providing higher resolution and more detailed data, which will help improve the accuracy of feature extraction, leading to more reliable pose estimation and map construction. In the future, this method can be implemented on a Field-Programmable Gate Array (FPGA) and integrated with sensors. By leveraging its powerful parallel processing and computation capabilities, it can achieve efficient data processing and analysis in real-time applications.

Acknowledgement

The authors acknowledge the National Natural Foundation of China (NSFC62221004); Guangxi Key Research and Development Program (AB23026149, AB24010073); Guilin Scientific Research Project (202201131-1); Guilin Scientific Research Project (20220107-1); Guilin Scientific Research Project (20210101-5)

References

1. Aoqing, Y., Feng, C., Yiru, S., Dong T. Folding Net: Point cloud auto-encoder via deep grid defor-mation. In IEEE Conf. on Computer Vision and Pat-tern Recognition(CVPR), 2018: 206-215. https://doi.org/10.1109/CVPR.2018.00029
2. Charles, R, Q., Hao, S., Kaichun, M., Leonidas, J. G. PointNet: Deep learning on point sets for 3D classification and segmentation. In 2017 IEEE Con-ference on Computer Vision and Pattern Recogni-tion (CVPR), 2017: 77-85. https://doi.org/10.1109/CVPR.2017.16
3. Charles, R, Q., Yi, L., Hao ,S., Leonidas, J. G. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. arXiv preprint arXiv, 2017: 652-660.
4. Cigneni, P., Callieri, M., Corsini, M., Dellepiane, M., Ganovelli, F., Ranzuglia, G. Meshlab:an open-source mesh processing tool. Eurographics Italian chapter conference, 2008: 129-136. DOI:10.2312/LocalChapterEvents/ItalChap/ItalianChapConf2008/129-136.
5. Dzmitry, B,. Kyunghyun, C., Yoshua, B. Neural machine translation by jointly learning to align and translate, Proceedings of International Conference on Learning Representations (ICLR-15), 2015: 1-15.
6. Gao, J., Lan, J., Wang, B., Li, F. SDANet: spatial deep attention-based for point cloud classification and segmentation. Machine Learning, 2022, 111(4): 1327-1348. https://doi.org/10.1007/s10994-022-06148-1

7. Guocheng, Q., Abdulellah, A., Guohao, L., Ali, T., Bernard, G. Pu-gcn: point cloud upsampling using graph convolutional networks, In: 2021 IEEE Con-ference on Computer Vision and Pattern Recogni-tion (CVPR), 2021: 11683-11692. https://doi.org/10.1109/CVPR46437.2021.01151
8. Hu, B. Research on brain point cloud reconstruction model based on generative adversarial strategy and graph convolutional neural network. Chinese Academy of Sciences (Shenzhen Institute of Advanced Technology), 2022.
9. Hui, H., Dan, L., Hao, Z., Uri, A., Daniel, C. Con-solidation of unorganized point clouds for surface reconstruction. ACM Trans. on Graphics (SIG-GRAPH Asia), 2009, 28(5):176:1-7.https://doi.org/10.1145/1618452.1618522
10. Jie, Z., Ganqu, C., Shengding, H., Zhengyan, Z., Cheng, Y., Zhiyuan, L., Lifeng, W., Changcheng, L., Maosong, S. Graph neural networks: a review of methods and applications. AI Open, 2020, 1: 57-81. https://doi.org/10.1016/j.aiopen.2021.01.001
11. Jing, W., Zhang, W., Li, L., Di, D., Chen, G., Wang, J. AG-Net: An attention-based graph network for point cloud classification and segmentation. Remote Sensing, 2022, 14(4): 1036. https://doi.org/10.3390/rs14041036
12. Kulikajevas, A., Maskeliūnas, R., Damaševičius, R., Misra, S. Reconstruction of 3D object shape using hybrid modular neural network architecture trained on 3D models from ShapeNetCore dataset. Sensors, 2019, 19(7): 1553. https://doi.org/10.3390/s19071553
13. Lei, W., Yuchun, H., Yaolin, H., Shenman, Z., Jie, S. Graph attention convolution for point cloud se-mantic segmentation, Proceedings of IEEE Confer-ence on Computer Vision and Pattern Recognition (CVPR-19) , 2019: 10288-10297. https://doi.org/10.1109/CVPR.2019.01054
14. Lequan, Y., Xianzhi, L., Chi-Wing, F., Daniel, C., Pheng-Ann, H. PU-Net: Point cloud upsampling network. In: 2018 IEEE Conference on Computer Vision and Pattern Recognition(CVPR), 2018: 2790-2799. https://doi.org/10.1109/CVPR.2018.00295
15. Lequan, Y., Xianzhi, L., Chi-Wing, F., Daniel, C., Pheng-Ann, H. EC-Net: An edge-aware point set consolidation network. In European Conference on Computer Vision (ECCV), 2018: 386-402. https://doi.org/10.1007/978-3-030-01234-2_24
16. Li, Z., Bai, Z., Xiao, X., Zhang, Y., You, Y. Point cloud upsampling network fusing transformer and multi-stage learning framework. Computer Science: 2023: 1-14.
17. Luqing, L., Lulu, T., Wanyi, Z., Shizheng, W., Zhi-Xin, Y. Pu-eva: an edge-vector based approxima-tion solution for flexible-scale point cloud upsam-pling, Proceedings of IEEE International Confer-ence on Computer Vision (ICCV-21) , 2021: 16188-16197. https://doi.org/10.1109/ICCV48922.2021.01590
18. Marc, A., Johannes, B., Daniel, C., Shachar, F., Da-vid, L., Claudio, T, Silva. Computing and rendering point set surfaces. IEEE Trans. Vis. & Comp.Graphics, 2003, 9(1): 3-15. https://doi.org/10.1109/TVCG.2003.1175093
19. Peng, C., Xiao, W., Jian, P., Wenwu, Z. A survey on network embedding. IEEE Trans, 2019, 31 (5): 833-852. https://doi.org/10.1109/TKDE.2018.2849727
20. Pierdicca, R., Paolanti, M., Matrone, F., Martini, M., Morbidoni, C., Malinverni, E., Frontoni, E., Lingua, A. Point cloud semantic segmentation using a deep learning framework for cultural heritage. Remote Sensing, 2020, 12(6): 1005. https://doi.org/10.3390/rs12061005
21. Ruihui, L., Xianzhi, L., Chi-Wing, F., Daniel, C., Pheng, A. PU-GAN: A point cloud upsampling ad-versarial network. In: 2019 IEEE International Con-ference on Computer Vision (ICCV), 2019: 7202-7211. https://doi.org/10.1109/ICCV.2019.00730
22. Ryselis, K., Blažauskas, T., Damaševičius, R., Maskeliūnas, R. Computer-aided depth video stream masking framework for human body seg-mentation in depth sensor images. Sensors, 2022, 22(9): 3531. https://doi.org/10.3390/s22093531
23. Xiao, X., Bai, Z., Li, Z., Liu, X., Du, J. Parallel mul-ti-scale point Cloud Upsampling Method with At-tention mechanism. Computer Science, 2024, 51(8): 183-191. DOI: 10.11896/jsjkx.230500094.
24. Yaron, L., Daniel, C., David, L., Hillel, T. Parame-terization-free projection for geometry reconstruc-tion. ACM Trans. on Graphics (SIGGRAPH), 2007, 26(3): 22: 1-5. https://doi.org/10.1145/1276377.1276405
25. Yue, W., Yongbin, S., Ziwei, L., Sanjay, E., Mi-chael, M., Justin, M. Dynamic Graph CNN for Learning on Point Clouds. Association for Compu-ting Machinery (ACM), 2019, 38(5): 1-12. https://doi.org/10.1145/3326362
26. Zeng, J. Research on 3d point cloud quality im-provement method based on deep learning. Xiamen University, 2020.
27. Zhang, X., Fu, C., Zhao, Y., Xu, X. Hybrid feature CNN model for point cloud classification and seg-mentation. IET Image Processing, 2020, 14(16): 4086-4091. https://doi.org/10.1049/iet-ipr.2020.0658
28. Zhengwei, W., Qi, S., Tomas, E. Generative Adver-sarial Networks in Computer Vision: A Survey and Taxonomy. ACM Comput. 2021, 54(2): 1-38. https://doi.org/10.1145/3439723
29. Zhipeng, L., Jonathan, L., Zhenlong, X., Z. Geroge, M., Xiaojie, C. Learning high-level features by fus-ing multi-view representation of MLS point clouds for 3D object recognition in road environments. IS-PRS Journal of Photogrammetry and Remote Sens-ing, 2019, 150: 44-58. https://doi.org/10.1016/j.isprsjprs.2019.01.024