Received: 19 July 2022 | Accepted: 24 August 2022

DOI: 10.1002/int.23087

RESEARCH ARTICLE

WILEY

Interpolation graph convolutional network for 3D point cloud analysis

Yao Liu[1] | **Lina Yao**[1,2] | **Binghao Li**[3] | **Claude Sammut**[1] | **Xiaojun Chang**[4]

[1]School of Computer Science and Engineering, University of New South Wales, Sydney, New South Wales, Australia

[2]Data 61, CSIRO & School of Computer Science and Engineering, University of New South Wales, Sydney, New South Wales, Australia

[3]School of Minerals and Energy Resources Engineering, University of New South Wales, Sydney, New South Wales, Australia

[4]Faculty of Engineering and Information Technology, University of Technology Sydney, Sydney, New South Wales, Australia

Correspondence
Yao Liu, School of Computer Science and Engineering, University of New South Wales, Sydney, New South Wales 2052, Australia.
Email: yao.liu3@unsw.edu.au

Abstract

The feature analysis of point clouds, a popular representation of three-dimensional (3D) objects, is rising as a hot research topic nowadays. Point cloud data bear a sparse and unordered nature, making many commonly used feature extraction methods, for example, Convolutional Neural Networks (CNNs) inapplicable, while previous models suitable for the task are usually complex. We aim to reduce model complexity by reducing the number of parameters while achieving better (or at least comparable) performance. We propose an Interpolation Graph Convolutional Network (IGCN) for extracting features of point clouds. IGCN uses the point cloud graph structure and a specially designed Interpolation Convolution Kernel to mimic the operations of CNN for feature extraction. On the basis of weight postfusion and multilevel-resolution aggregation, IGCN not only reduces the cost of calculating the interpolation operation but also improves the model's performance. We validate the performance of IGCN on both point cloud classification and segmentation tasks and explore the contribution of each module of our model through ablation experiments. Furthermore, we embed the

IGCN point cloud feature extraction module as a plug-and-play module into other frameworks and perform point cloud registration experiments.

KEYWORDS
classification and segmentation, graph convolutional network, interpolation operation, point clouds, registration

1 | INTRODUCTION

The prevalence of three-dimensional (3D) scanners, LiDARs, and RGB-D cameras makes it possible to acquire massive 3D data at low costs.[1] Compared with 2D images, 3D data can provide enriched geometric information and more accurate representation of environments and objects. Common representations of 3D data include voxels, meshes, and point clouds. Voxels represent 3D data as 3D grids and describe 3D objects by occupancy and nonoccupancy states[2]; they generally require huge memory for high-resolution objects.[3] Meshes are mainly used in computer applications to store and render 3D graphics via a composition of vertices, edges, and faces; mesh data often suffer from noise, missing data, and resolution problems.[4] In comparison, point cloud data have a simpler data structure and contain original spatial feature information; therefore, it becomes a popular representation form for 3D analysis. Besides, the acquisition of point cloud data does not depend on light sources and thus is more reliable in low-light environments. Currently, point clouds have broad applications in robotics[5] and autonomous driving.[6,7] Point clouds also have good prospects for applications at night[8] or in mine tunnels.[9,10]

Deep learning, especially Convolutional Neural Network (CNN), has been the main technique for image processing, due to its strong feature extraction capabilities.[11–15] However, many of these methods (e.g., standard CNN) are not directly applicable to point cloud data, which is sparse and unordered as opposed to the dense and regular image data. Some studies (e.g., multiview CNN[16] and group-view CNN[17]) transform point cloud data into a multiview projection image and then utilize 2D CNNs for feature extraction. These view-based methods face the challenge of losing 3D geometric information and sacrificing efficiency during aggregating multiview features. Other methods (e.g., VoxNet[18] and PointGrid[19]) transform point cloud data into volumetric occupancy and then utilize 3D CNNs for feature extraction; these methods, instead, may lose data accuracy in the voxelization process and the complex 3D CNN calculation. In comparison, extracting features directly from raw point cloud data without data transformation becomes a promising solution. PointNet[20] is a pioneering work towards overcoming the difficulty of extracting features from point cloud data via point-based methods. Since then, a range of methods have emerged on that basis, including point-based methods,[21,22] convolution-based methods,[23–25] and graph-based methods.[26–28] Point cloud classification and segmentation tasks are classical tasks derived from 2D images. However, due to the special nature of point cloud data, multiple scans and segmented scans are often required during collection. Therefore, point cloud registration is a very important front work in point cloud analysis. The point cloud registration task transforms two or more point clouds into the same coordinate system for merging, which is the basis of subsequent point cloud analysis. It is essential for reconstructing 3D models and evaluating the building structures of various industries,[29,30] especially in underground mining tunnels with dim light and complex structure.

In this paper, we propose an Interpolation Graph Convolutional Network (IGCN) for extracting features from point cloud data. IGCN can achieve 2D image-like feature extraction on point cloud data and finally obtain global features by extracting local features level by level to mimic operations of standard 2D CNN, as shown in Figure 1. Specifically, the interpolation operation makes Graph Convolutional Network suitable for processing sparse and unordered point cloud data, while the postfusion of weights and the aggregation of multilevel-resolution features together reduce the model complexity, ensuring the time and memory cost of interpolation is feasible for point clouds classification and segmentation. Overall, IGCN not only reduces model complexity with respect to the number of parameters but also outperforms multiple state-of-the-art methods in point cloud classification and segmentation. Our goal is to propose a model that can significantly reduce model parameters to reduce model complexity and computational cost, while maintaining a comparable result that will facilitate the future analysis of deep learning networks on point cloud. Finally, we apply the proposed IGCN as a general point cloud data feature extraction module to the field of point cloud registration for the alignment of unaligned point clouds. Point cloud registration contributes to industries such as underground mining through a case study in the fields of tunnels and underground excavation.

Our main contributions are summarized as follows:

- We propose an IGCN model for the classification and segmentation of point cloud data, which achieves better performance and reduces model complexity simultaneously. Meanwhile, IGCN can be used as a plug-and-play point cloud feature extraction module for the point cloud registration field.

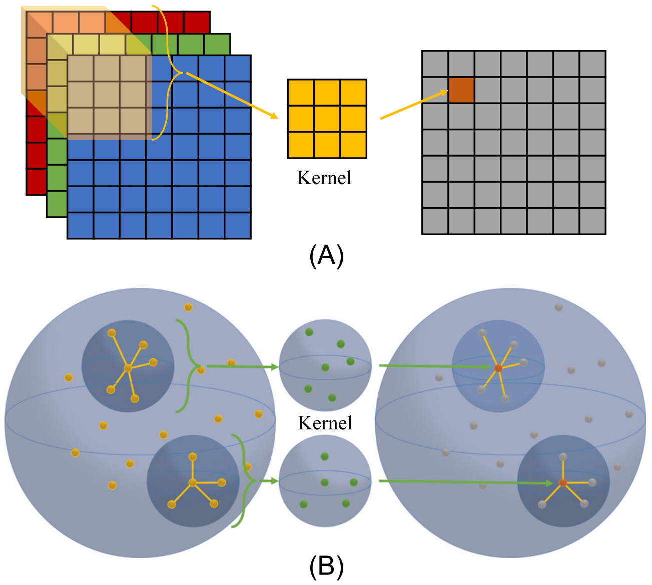

FIGURE 1 Examples of 2D CNN model and IGCN model. (A) Standard CNN process and (B) IGCN process. With the proposed convolution kernel, IGCN can mimic the CNN process on point cloud data. CNN, Convolutional Neural Network; IGCN, Interpolation Graph Convolutional Network. [Color figure can be viewed at wileyonlinelibrary.com]

- We introduce an interpolation algorithm into the Graph Convolutional Network, which makes it possible to apply CNN to sparse and unordered point cloud data while reducing the network parameters.
- We present weight postfusion and multilevel-resolution aggregation to reduce the time and memory cost resulted from the introduction of interpolation operations, leading to an overall improvement in the model performance.

2 | RELATED WORK

2.1 | Point-based methods

PointNet[20] is a pioneering method in 3D point cloud analysis. It directly uses raw point cloud data as input and takes into account the order independence. PointNet uses multiple shared fully connected layers (multilayer perceptrons [MLPs]) to deal with unordered 3D points and uses Max-pooling to extract global features for classification and segmentation tasks. PointNet extracts only global features by each point after independent learning but cannot extract local features between points, so its performance is limited. PointNet++[21] solves this problem by dividing the space into several spherical regions and using PointNet to extract features for each region. Specifically, PointNet++ proposes the Set Abstraction (SA) level, each containing the sampling layer, the grouping layer, and the PointNet layer. Then, it learns local features level by level through feature extraction of multiple SA levels. Finally, it uses Max-pooling to obtain global features for classification tasks or the proposed Feature Propagation layer (which is time-consuming) to obtain pointwise features for segmentation tasks. On the basis of PointNet++, PointWeb[22] uses Adaptive Feature Adjustment, which leverages local neighborhood information to enhance the features of points. Point-based methods are mostly based on PointNet methods with improvements, adding local information to improve performance and extracting information through fully connected layers.

2.2 | Convolution-based methods

Convolution can extract features better than fully connected layers, so convolution-based methods have been developed. The challenge with convolution operation lies in designing the suitable convolution kernels for sparse and unordered point cloud data. Existing convolution-based methods cover both continuous and discrete convolution methods. Relation-shape CNN (RS-CNN)[24] and DensePoint[25] are representatives of continuous convolution methods. RS-CNN[24] obtains local features through convolution of points and their surrounding points, and then obtains global features by conducting the above level by level. The convolution operation uses Euclidean distance and relative position information to obtain weights through shared MLPs. On the basis of RS-CNN, DensePoint[25] reduces the model parameters and calculation cost by aggregating multilayer features. It does not need to increase the number of out channels but output the same low number of channels each time. It reduces the model parameters and computational complexity through cross-level aggregation without reducing the accuracy. Pointwise-CNN[31] is a discrete convolution method, which divides the regular grid and allows the points in the same grid to share the same weights for convolution. The convolution-based approach focuses on designing suitable convolution kernels to conduct discrete or continuous convolution on the point cloud data.

2.3 | Graph-based methods

The raw point cloud data contains 3D coordinates and other information, such as reflectivity. Point cloud data have no topological structure, and the representation of point cloud data as a model with a topological structure can enrich the representation ability of point clouds. Therefore, graph-based methods are becoming popular for point cloud analysis. Edge-Conditioned Convolution (ECC)[26] is a pioneering work of point clouds processing using graph-based methods. ECC converts point cloud data into graph-based data by taking the points in the point clouds data as vertices and uses directed lines as edges to connect the points and their neighbors. ECC uses a filter-generating network to generate edge features through vertex coordinate features and transform those features into weight information. ECC constructs a new graph for the next feature extraction: it first updates the vertex features by the weight information in the neighborhood; then, it uses the voxel-based Max-pooling for downsampling. Dynamic Graph CNN (DGCNN)[27] is a graph-based method similar to the PointNet structure. Compared with the ECC, DGCNN has no voxel-based downsampling and uses feature information instead of coordinate information to construct the graph. The proposed Edge Convolution (EdgeConv) layer uses vertex features to learn edge features and uses aggregated edge features to update vertex features. In DGCNN, the number of vertices does not change after each EdgeConv layer, but the receptive field increases gradually as each vertex can learn a wider range of features. 3DGCN[28] performs convolution operations similar to 2D CNN by defining learning deformable kernels. After constructing the graph, in the sphere formed by each vertex and its neighbors, the neighbor points are normalized with the vertex as the ball center, the original coordinates are replaced by direction vectors, and the similarity function of the convolution is a cosine similarity function. The cosine similarity is irrelevant to absolute distances, thus improving the model performance. The graph-based methods rely on the graph structure to construct the point cloud data and extract information by updating the features of vertex and edges through the graph structure.

2.4 | Point cloud registration methods

Determining the transformation matrix of two unaligned point cloud data is a prerequisite for many high-level tasks. Early 3D registration used the Iterative Closest Point method, but the computational complexity of this explicit estimation method increases quadratically with the number of points.[32] In the feature descriptor-based method, PPFNet[33] and FoldNet[34] manually produce rotation-invariant features using point pairs of angles and distances. However, the descriptors are less discriminatory than when features are extracted directly from point cloud data. With the introduction of deep learning, significant improvements have been made in learning-based feature descriptors compared with hand-crafted descriptors. 3DSN[35] proposes a smoothed density voxel grid as input to the 3D CNN, and FCGF[36] is based on this to improve computational efficiency. The other method is the direct registration method, which in contrast to the descriptor-based method, does not extract feature descriptors but achieves alignment directly from the information of the two point clouds, and this method allows the pose estimation to be embedded in the learning pipeline. PointNetLK[37] is representative of this method, estimating relative transformations in an iterative process by combining the feature extraction of PointNet[20] with the Lucas/Kanadelike[38] optimization algorithm.

3 | METHODOLOGY

3.1 | Overview

We propose an IGCN for the classification and segmentation tasks of point clouds, as shown in Figure 2. This feature extraction module can be used as a plug-and-play point cloud feature extractor and applied to other tasks, such as point cloud registration. The model consists of Interpolation Convolution Kernel generation and point patch graph construction. The convolution operation is achieved by multilevel-resolution aggregation and postweight fusion, which preserves the superior performance of the original model while reducing the model's complexity.

First of all, we define the notations used in this paper. We denote a point cloud instance as a set P, which contains N points as follows.

$$P = \{p_n | n = 1, 2, 3, \ldots, N\}, \tag{1}$$

where p_n is the nth point in P. We only use the 3D coordinate information of the point cloud, denoted by $p_n = (x_n, y_n, z_n)$, where x_n, y_n, and z_n are the 3D coordinate information. $f(p_n)$ denotes the feature of p_n, and $d(p_n)$ denotes the direction vector of p_n with respect to the specified centroid. Therefore, each input point cloud instance can be represented as an $N \times 3$ matrix. Similarly, we define the convolution kernel as

$$K = \{k_i | i = A, B, C, \ldots\}, \tag{2}$$

where k_i is a weight parameter, $w(k_i)$ is the weight parameter value of k_i, and $d(k_i)$ is the direction vector of k_i with respect to the specified centroid, which is also the relative coordinate of k_i.

3.2 | Interpolation convolution kernel

Taking a 2D RGB image, for example, the kernel of a standard CNN is a grid containing weight parameters. Since the 2D image is regular and ordered, each weight parameter of the kernel will correspond to one pixel on the image, and convolution is achieved by sliding the position of the kernel to share the weight parameters. In particular, the part of the pixels corresponding to

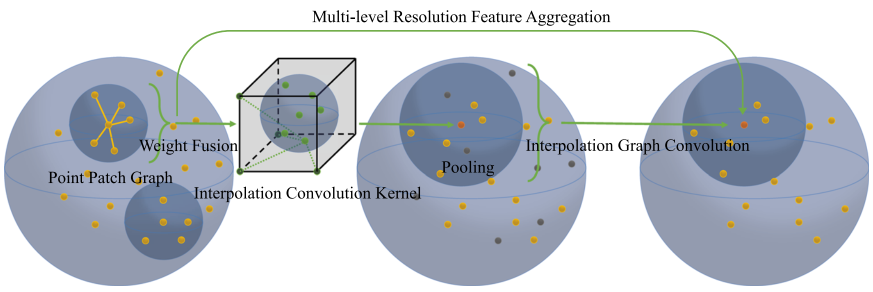
Multi-level Resolution Feature Aggregation
Weight Fusion
Point Patch Graph
Interpolation Convolution Kernel
Pooling
Interpolation Graph Convolution

FIGURE 2 Overview of interpolation graph convolutional network. The model consists of Interpolation Convolution Kernel generation and point patch graph construction, and the convolution operation is achieved by multilevel-resolution aggregation and postweight fusion. [Color figure can be viewed at wileyonlinelibrary.com]

the kernel is called the image patch. A straightforward way of introducing 2D CNNs into point clouds is to provide a kernel that corresponds to the circled point patches, which contain the same number and position of weight parameters as the points contained in point patches.

The core of convolution is the shared weight parameter. but for point clouds, each circled point patch holds a different number and position points, making it impossible to determine a fixed convolution kernel to implement the point cloud convolution operation. To solve this challenge, we propose an IGCN by defining the Interpolated Convolution Kernel. Motivated by the fact that any four points in space that are not co-planar can completely represent a space, we generate convolution kernels (interpolation kernels) with arbitrary numbers and positions by defining a kernel (original kernel) containing four weight parameters with a spatial linear interpolation algorithm.

Figure 3 illustrates the interpolation convolution kernel. The original kernel is defined as the four vertices of a cube with side length 2, namely, k_A, k_B, k_C, and k_D. The spatial coordinate system is established by taking the center of the cube as the coordinate origin, the $k_A \rightarrow k_B$ direction as the positive X-axis direction, the $k_A \rightarrow k_C$ direction as the positive Y-axis direction, and the $k_A \rightarrow k_D$ direction as the positive Z-axis direction. Thus, the original kernel is denoted as $K = \{k_A, k_B, k_C, k_D\}$, where $d(k_A) = (-1,-1,-1)$, $d(k_B) = (1,-1,-1)$, $d(k_C) = (-1,1,-1)$, and $d(k_D) = (-1,-1,1)$. For any point k'_m with certain coordinates in the space, we use the original kernel K to calculate its interpolation weight parameter value (the interpolation algorithm is shown in Equation 3 and obtain the convolution kernel $K' = \{k'_m | m = \text{i, ii, iii, ...}\}$).

$$\begin{aligned} w(k'_m) &= \frac{x_{k'_m} - x_{k_A}}{x_{k_B} - x_{k_A}} \times (w(k_B) - w(k_A)) \\ &+ \frac{y_{k'_m} - y_{k_A}}{y_{k_C} - y_{k_A}} \times (w(k_C) - w(k_A)) \\ &+ \frac{z_{k'_m} - z_{k_A}}{z_{k_D} - z_{k_A}} \times (w(k_D) - w(k_A)) + w(k_A) \\ &= \frac{d(k'_m) + 1}{2} \cdot \begin{bmatrix} w(k_B) - w(k_A) \\ w(k_C) - w(k_A) \\ w(k_D) - w(k_A) \end{bmatrix} + w(k_A). \end{aligned} \tag{3}$$

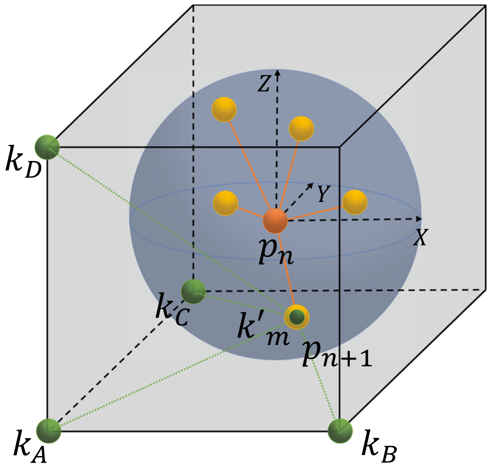

FIGURE 3 Interpolation convolution kernel. In the coordinate system space, the interpolation weight parameter for the position is calculated based on the p_{n+1} coordinates. [Color figure can be viewed at wileyonlinelibrary.com]

For any point patch, it can be normalized to a sphere with radius 1 and placed in the cube formed by the original kernel. In the coordinate system formed by the original kernel, any point in the point patch has a relative spatial position; so the interpolation weight parameter for that point position can be calculated from the weight parameter of the original kernel, generating convolution kernels with matching numbers and position for different point patches. The interpolation algorithm calculates the convolution kernel from the original kernel by assigning a weight parameter to each point in the point patch without caring about the sparsity and disorder of the points in the point patch. The original kernel used for neural network optimization contains a fixed number of four weight parameters. Therefore, more points in the point patch do not increase the complexity of the model.

3.3 | Interpolation graph convolution

In 2D CNN, feature patches can be obtained by sliding a fixed-size window over the feature map from left to right and from top to bottom. The division of point patches in point clouds can usually be improved using k-Nearest Neighbors (KNNs) or Radius Nearest Neighbors (Radius-NN) methods. In this study, we select the Query Ball Point method proposed by PointNet++[21] to divide point patches. The Query Ball Point method is essentially a Radius-NN method with a fixed number of neighbors, which combines the advantages of Radius-NN and KNN. Specifically, it takes a point in the point cloud instance as the center point and finds a fixed number of neighbors within a defined range. In the ball formed by the maximum search radius, more than the specified number of neighbors will be discarded; if neighbors are insufficient, the first neighbor point will be replicated to make up the gap. Each point in the point cloud instance is regarded as a center point to form a point patch.

We obtain the point patch P' as

$$P' = \mathcal{Q}(P) = \left\{p_n^l | l = 1, 2, 3, \ldots, L\right\}, \tag{4}$$

where $\mathcal{Q}()$ is the Query Ball Point method, the subscript n denotes the nth point in the point cloud with P as the point patch center, and L is the number of points contained in the point patch. In the point patch, the graph structure $G(P')$ is formed by connecting lines from the center point p_n^1 to the rest of the neighboring points and itself $\{p_n^l | l = 1, 2, 3, \ldots, L\}$.

$$G(P') = \{V(P'), E(P')\}, \tag{5}$$

$$V(P') = \left\{f\left(p_n^l\right) | l = 1, 2, 3, \ldots, L\right\}, \tag{6}$$

$$E(P') = \left\{d\left(p_n^l\right) | l = 1, 2, 3, \ldots, L\right\}, \tag{7}$$

where V is the vertex set and E is the edge set. The position and number of convolution kernel K' coincide with the point patch P', so we can use $E(P')$ to calculate $w(K')$. Therefore, we propose an Interpolated Graph Convolution (IGConv) as

$$\begin{aligned} IGConv(V(P'), \mathcal{I}(K, E(P'))) &= IGConv(f(P'), w(K')) \\ &= \mathcal{A}(\mathcal{F}(f(P'), w(K'))), \end{aligned} \tag{8}$$

where $\mathcal{I}()$ is the interpolation calculation of Equation (3), K is the original kernel, $\mathcal{A}()$ is the aggregation function (i.e., the maximum method in our model), and $\mathcal{F}()$ is the fusion function (i.e., inner product in our model).

Interpolation operation tends to cause huge time and memory cost of $\mathcal{F}()$. Therefore, we perform weights fusion at different stages to reduce this cost, as shown in Figure 4.

$$\mathcal{F}(f(P'), w(K')) = \mathcal{N}(f(P') \cdot w(K')), \tag{9}$$

$$\mathcal{F}(f(P'), w(K')) = \mathcal{N}(f(P')) \cdot w(K'). \tag{10}$$

Equation (9) is preweight fusion and Equation (10) is postweight fusion, where $\mathcal{N}()$ is a neural network method, for example, linear regression or 1D convolution. $\mathcal{N}()$ has the same input channel as $f(P')$ and the same output channel as $\mathcal{F}(f(P'), w(K'))$. A simple analysis shows that only $\frac{1}{input\_channel}$ of the original calculation cost is required by preweight fusion and only $\frac{output\_channel}{(input\_channel)^2}$ of the original calculation cost is required by postweight fusion. Here, we use linear regression and postweight fusion to enhance feature extraction while eliminating the computational cost caused by the interpolation.

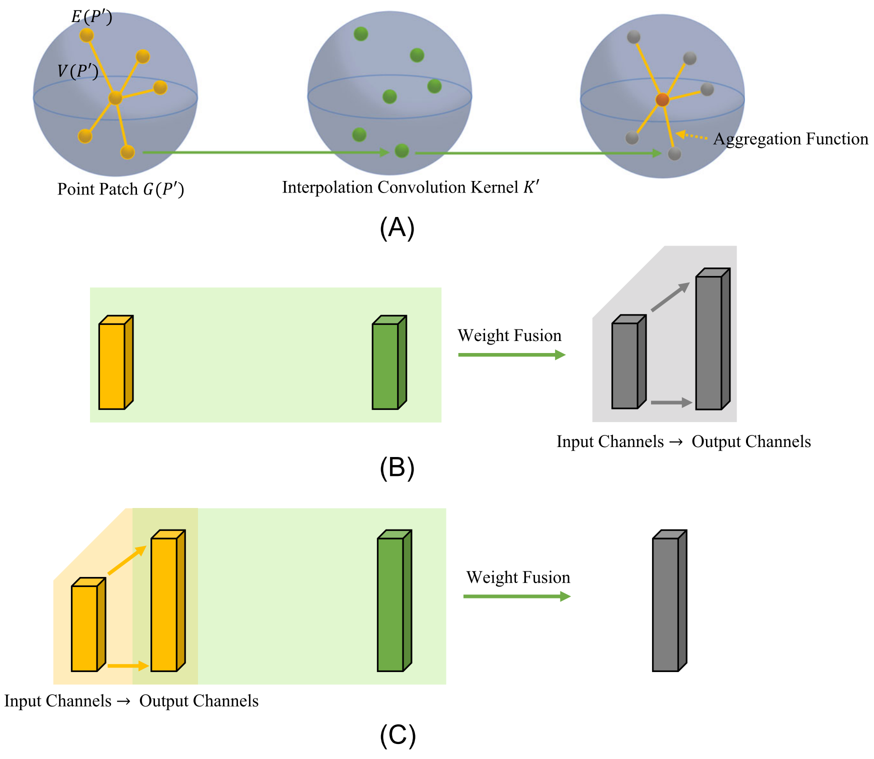
$E(P')$
$V(P')$
Point Patch $G(P')$
Interpolation Convolution Kernel K'
Aggregation Function
(A)
Weight Fusion
Input Channels → Output Channels
(B)
Weight Fusion
Input Channels → Output Channels
(C)

FIGURE 4 Interpolation graph convolution and weights fusion. (A) Feature aggregation process for interpolation graph convolution, (B) weight fusion first, followed by extension of feature dimensions, and (C) extension of feature dimensions first, followed by weight fusion. [Color figure can be viewed at wileyonlinelibrary.com]

3.4 | Network structure

On the basis of PointNet++[21] and DensePoint,[25] we further reduce model complexity by multilevel-resolution features aggregation. A standard 2D CNN extracts features level by level in increasing feature channels, so the features in the final fully connected layer have a higher dimension. In contrast, our model conducts feature extraction by aggregating multilevel-resolution features, avoiding the need to raise the channels to a high dimension. As shown in Figure 5, the input of each level is a concatenation of the outputs of all previous levels. By extracting features level by level, the input channels keep increasing while the output channels keep constant.

$$C_{\text{out}}^{n} = Conv\left(Concat\left(C_{\text{out}}^{n-1}, C_{\text{out}}^{n-2}, \ldots, C_{\text{out}}^{1}\right)\right), \tag{11}$$

$$G(P'^{n}) = Concat\left(C_{\text{out}}^{1}, C_{\text{out}}^{2}, \ldots, C_{\text{out}}^{n}\right), \tag{12}$$

where C_{out}^{n} denotes the output of the nth level of convolution, the output channel of each level is fixed at k, the input channel is $(n-1)\times k$ (except for the first level), and $G(P'^{n})$ is the feature of the nth level of the point patch graph with $n \times k$ channels. In our model, each Interpolation Graph Convolution takes all the outputs of the previous Interpolation Graph Convolution as input, but fixes the output at a lower dimension, which ensures the overall high efficiency of the model.

The pooling layer follows the 3DGCN[28] approach with a simple and efficient pooling method. We use the same graph structure as the IGConv process and then downsample it by Max-pooling.

$$P^{n} = Pooling(P^{n-1})|r = \frac{Num(P^{n-1})}{Num(P^{n})}, \tag{13}$$

where r is the pooling rate, meaning the $(n-1)$th layer has r times the number of points in the nth layer.

For the classification task, after extracting the features, we apply IGCN again to obtain the final feature with higher channels, aggregate them into global features, and finally conduct classification using a fully connected layer (MLP). For the segmentation task, we directly aggregate the extracted

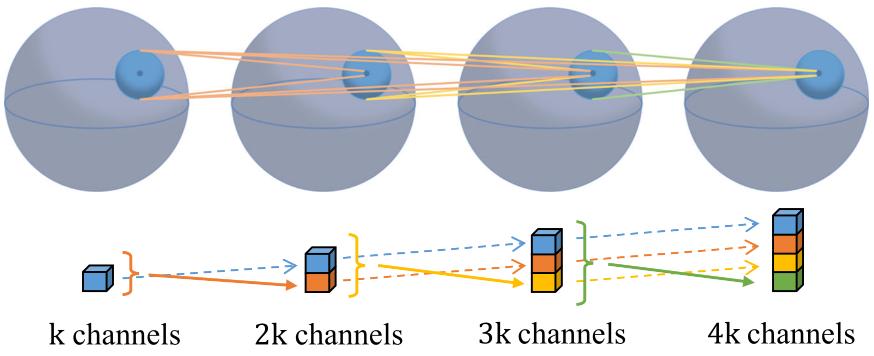

FIGURE 5 Multiresolution features aggregation. The input at each level is a concatenation of the outputs from all previous levels. By extracting features level by level, the input channels are continuously increased while the output channels remain the same. [Color figure can be viewed at wileyonlinelibrary.com]

features into global features with lower channels, concatenate the features of all levels and the One-hot vector from PointNet[20] into pointwise aggregated features, and finally conduct classification for each point by the shared MLP. Since the number of sampling points in each level decreases due to pooling layers in the segmentation task, we find the nearest neighboring points and making directly replicate them as upsampling points—this makes it possible to concatenate the features in all levels. By doing this, we make up for the points discarded during sampling and complete the concatenation of features across levels for the segmentation task. For the point cloud registration task, we adopt the structure of the PointNetLK[37] model and replace the point cloud feature extractor PointNet in the original model with the point cloud feature extractor composed of IGCN. The Lucas–Kanada (LK) algorithm[38] is originally applied to 2D images or 3D grids because 2D images or 3D grids are regular and dense representations that can easily define the convolution process. The PointNetLK method extracts point cloud features from PointNet and further optimizes them using the LK algorithm. Our IGCN is a point cloud feature extraction method that is fully mimicking a standard CNN so it can be easily embedded under the PointNetLK algorithm framework and improve alignment performance.

4 | EXPERIMENTS

4.1 | Parameter settings and model details

Our model takes the coordinate information of the 3D point cloud instance as input, as shown in Figure 6. The first IGCN has four input channels and contains the 3D relative coordinates and the 1D relative Euclidean distance of the point patch graph. In the feature extraction stage,

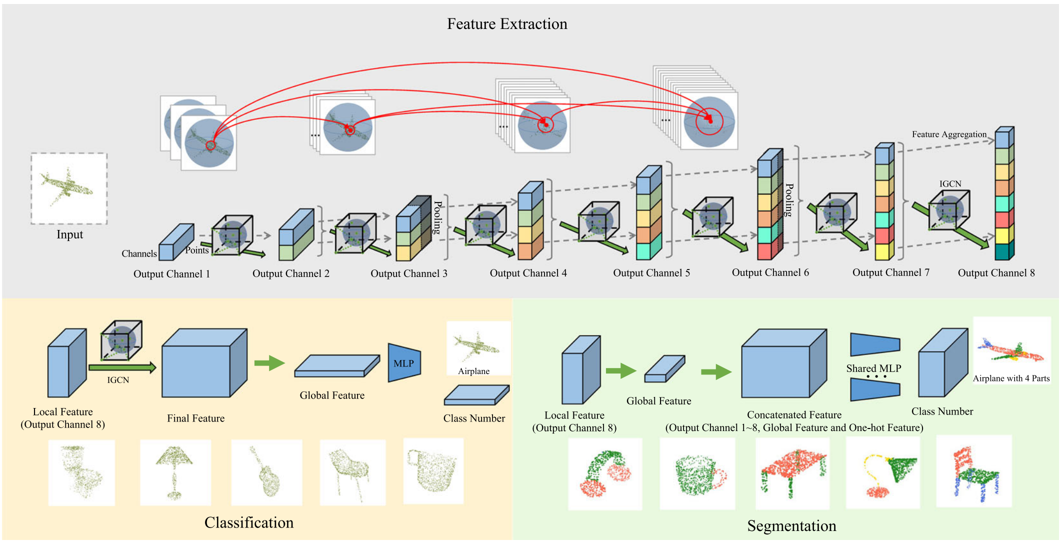

FIGURE 6 The model for classification and segmentation. The input to the feature extraction module is the three-dimensional (3D) coordinate information of the point cloud data and the module consists of eight layers of IGCN and two layers of Pooling. The classification module takes the output of the feature extraction module as input and proceeds through IGCN, Max-pooling, and MLP to obtain the final classification result. The segmentation module concatenates all IGCN outputs of the feature extraction module as well as Global Feature and One-hot Feature to obtain feature information, and obtains segmentation results by Shared MLP. IGCN, Interpolation Graph Convolutional Network; MLP, multilayer perceptron. [Color figure can be viewed at wileyonlinelibrary.com]

each IGCN has a fixed output channel of 32. According to the model we designed, each IGCN output channel will be concatenated to each previous output channel, so after eight IGCN feature extractions, the feature dimension will reach $8 \times 32 = 256$ dimensions. A pooling layer follows every three IGCNs with a pooling rate of 4. When constructing the point patch graph, the radius before the first input and after the two pooling layers are set to 0.25, 0.39, and 0.63, respectively. The number of points contained in the point patch graph is set to 32, as this number is twice the average number of points that can be covered by the point patch graph. The reason for this setting is that in point cloud instances, the distribution of points is uneven and a larger than average number of points collected in the point patch graph facilitates better feature extraction in dense regions of points. In our model, the initial learning rate is set to 0.001 and the learning rate is decayed by half every 10 epochs using the ADAM optimizer, with a total number of epochs of 100.

In the classification task, we use IGCN to raise the 256-dimensional features after feature extraction to 1024 dimensions to obtain the final features, and then make them into 1×1024 global features by feature aggregation, and finally classify them through the fully connected layer (MLP). In MLP the feature dimension is first reduced to 256, and then to the number of classes. In the segmentation task, we obtain the global feature directly by aggregating the 256-dimensional features obtained in the feature extraction stage. Then the concatenated feature is obtained by concatenating the output features in the feature extraction stage, global feature, and One-hot vector through up-sampling. The feature dimension of One-hot vector is the number of categories of instances. Finally we achieve the segmentation task by classifying each point through a shared fully connected network (shared MLP). In Shared MLP, the feature dimension first becomes 1024, then drops to 512, and finally to the number of categories. In the point cloud registration task, we embed the point cloud feature extraction module obtained from the training of the classification task into the PointNetLK framework, replacing the PointNet module in the original model to complete the point cloud alignment.

4.2 | Classification task and segmentation task

We perform the classification task on ModelNet40,[39] which contains 40 classes of 12,311 3D point cloud instances. Each instance is downsampled to 1024 points and normalized. We use 9843 for training and 2468 for testing. The classification results in terms of the overall accuracy of our model and several other models are shown in Table 1. The results show that our model substantially reduces the network parameters while achieving better performance in several models when the number of input points is 1k. Compared with nonpoint cloud models, such as the OctNet[40] and Subvolume[41] models, our model achieves better accuracy, especially when compared with the Subvolume model, where the number of parameters in our model is only 3.4% of the Subvolume model. In comparison with the point cloud model, our model outperforms the PointNet[20] model based on the point, the Pointwise-CNN[31] model based on CNN, the ECC[26] model based on the graph, as well as the Deep Set[42] model in terms of classification accuracy, while being only 16.4% of the PointNet model in terms of the number of model parameters. Our model is slightly less accurate in comparison to some advanced models, but our model maintains a comparable result while using a lower number of parameters. Especially, in comparison with the InterpCNN[43] model, which also uses an interpolation algorithm, although it is slightly higher than our model in terms of accuracy, the number of parameters used by our model is only 4.5% of those used by InterpCNN. At a time when deep

TABLE 1 Classification results on ModelNet40

Method	Input	#params (M)	#points	OA (%)
OctNet[40]	Hybrid grid octree	–	–	86.5
Subvolume[41]	Voxels	16.6	–	89.2
Deep Sets[42]	Points	–	1k	87.1
PointNet[20]	Points	3.48	1k	89.2
pointNet++[21]	Points	1.48	1k	90.7
Pointwise-CNN[31]	Points	–	1k	86.1
ECC[26]	Points	–	1k	87.4
DGCNN[27]	Points	1.84	1k	92.9
3DGCN[28]	Points	0.89	1k	92.1
InterpCNN[43]	Points	12.8	1k	93.0
IGCN (ours)	**Points**	**0.57**	**1k**	**90.1**

Note: "#params" represents the number of parameters of the model, "#points" represents the number of points contained in the input instance, and "OA" represents the overall accuracy.

learning models are increasing the complexity in pursuit of high accuracy, the goal of our model is to maintain a comparable result while reducing the complexity of the model by significantly reducing the number of model parameters.

In particular, we compare our model with DGCNN and 3DGCN with respect to the number of parameters. These two models also have the graph structure, and also conduct the classification task by feature extraction and multilayer fully connected layers (the results are shown in Figure 7). 3DGCN has a variable kernel structure and in the original paper attempts are made to set up different numbers of kernels, here we set up 1 and 4 kernels, respectively, to participate in the comparison. Figure 7 shows the accumulated number of parameters for each level. We consider a module with a convolutional layer or fully connected layer as a level. Then, our model has 11 levels, DGCNN has eight levels, and 3DGCN has seven levels. The numbers of parameters in DGCNN and 3DGCN increase dramatically after four levels due to channel expansion, while our model maintains a low number of parameters. The parameters in our model after accumulating to 11 levels are still smaller in size than those in either DGCNN with eight levels or 3DGCN with seven levels. Moreover, as can be seen from the line in the figure, the increase in the number of parameters in our model is mostly due to the final multilayer fully connected layers, which is a great advantage over other models as our model uses very few parameters for the feature extraction stage. By comparing the previous layers, our model uses a very low number of parameters to obtain the feature dimensions that can be classified in the fully connected layer, which shows that our IGCN layer can obtain comparable results with very few parameters through the interpolation algorithm. The results reveal our interpolation algorithm, postweight fusion, and multilevel-resolution aggregation can effectively reduce the number of model parameters while preserving the performance.

Figure 8 shows a comparison of the convergence speed of the classification task, here we compare the IGCN model with the DGCNN model and the PointNet model. In our experiments, the IGCN achieves the highest classification accuracy of 90.1% in the 22nd epoch. In our reproduced DGCNN, the highest classification accuracy of 90.1% is achieved in the 98th epoch, which is the same accuracy as that of our IGCN. In the reproduced PointNet, the best

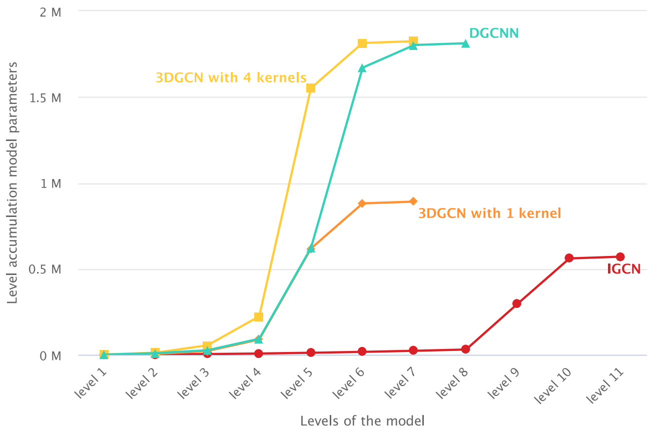

FIGURE 7 Comparison of the model parameters number. The convolutional layer or fully connected layer is regarded as a level. The first some layers of each model are the convolution-like layers for feature extraction, while the later layers are the fully connected layers for the classification task. 3DGCN, three-dimensional Graph Convolutional Network; DGCNN, Dynamic Graph Convolutional Neural Network; IGCN, Interpolation Graph Convolutional Network. [Color figure can be viewed at wileyonlinelibrary.com]

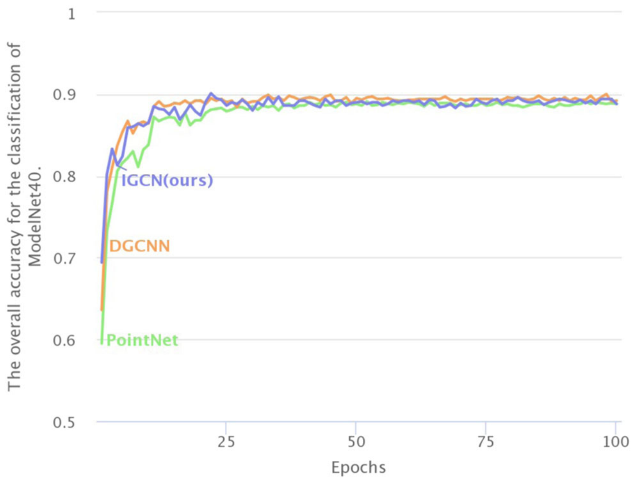

FIGURE 8 Comparison of the model convergence speed. The IGCN achieves an optimal 90.1% classification accuracy at the 22nd epoch, the reproduced DGCNN achieves the same optimal 90.1% classification accuracy at the 98th epoch, and the reproduced PointNet achieves an optimal 89.1% classification accuracy at the 52nd epoch. DGCNN, Dynamic Graph Convolutional Neural Network; IGCN, Interpolation Graph Convolutional Network. [Color figure can be viewed at wileyonlinelibrary.com]

classification accuracy of 89.1% is achieved in the 52nd epoch. Our model uses less epochs to achieve the highest classification accuracy, and the highest classification accuracy is the same as that of DGCNN. Therefore, it can be seen that our model converges faster than DGCNN and PointNet models. It can also be seen that in the first three epochs, especially the first epoch, our classification accuracy is much higher than the other models, reaching 69.4%, which further confirms the advantage of our model's convergence speed.

We evaluate the segmentation performance of our model on ShapeNetPart data set,[44] which contains 16,881 point cloud instances in 16 categories. Each category contains between 2 and 6 parts with a total of 50 parts. The evaluation standard is mean intersection over union (mIoU), where class mIoU is the average of 16 categories and instance mIoU is the average of all instances. The results of different models are shown in Table 2. As can be seen from the table, our model uses 1k points, while Kd-Net[45] uses 4k points, PointNet[20] uses 2k points, PointNet++[21] uses 2k points and the normal vector, as well as DGCNN[27] uses 2k points. The ShapeNetPart data set contains 16 classes, and different models have different segmentation performances for different classes. It is clear that our model outperforms the current optimal DGCNN model in two classes, ear phone and mug, and simultaneously obtains comparable results in class mIoU and instance mIoU. Our model achieves the best performance in the two classes while using a lower number of parameters, and the overall performance is comparable, indicating that our IGCN can significantly reduce model complexity and maintain adequate model performance through the interpolation algorithm. The results of the segmentation visualization are shown in Figure 9, where we compare the PointNet model and the 3DGCN model with the results of our model. The visualization results contain six categories from the ShapeNetPart data set, for example, Motorbike, Car, Earphone, Mug, Pistol, and Rocket.

4.3 | Ablation study

We compare the results of preweight fusion (Equation 9) and postweight fusion (Equation 10) for the classification task on ModelNet40 in Table 3. Postweight fusion can better extract features at a lower calculation cost thanks to the weight-independent dimensional transformation before fusion. Taking the eighth IGCN as an example, with 224 input channels and 32 output channels, the computational cost of using prefusion of features is $\frac{224(input\ \ channels)}{32(output\ \ channels)}$ times that of postfusion. In comparison with the classification results, our postfusion not only saves computational cost but also improves the classification accuracy by 1.9%.

We explore the impact of the aggregation function on IGCN by replacing the maximum method (Equation 8) with the summation method and the average method (the results for the classification task on ModelNet40 are shown in Table 3). The results show the superiority of the maximum method over the two alternatives. This is mainly due to the fact that the maximum method is not affected by the replication operation, because when we construct the point patch graph, we need to contain a fixed number of points within the specified range, and when there are not enough points, we use the replication operation. Since the summation method and the average method are affected by the redundancy value of replication, the maximum method has a better performance, 5.4% and 7.1% higher than the summation method and the average method, respectively.

Table 4 compares the effect of the number of neighbor points on classification accuracy, again with the ModelNet40 data set. The number of neighbor points here refers to the number of points contained in the point patch graph. Our point cloud instance consists of 1024 points and is scaled to a cube of side length 2. Each time a point patch graph is constructed with a radius of 0.25, 0.39, and 0.63, it can be calculated that each point patch graph contains an average of 16 points. Here we set the number of neighbor points to 16, 24, 32, 48, and 64 for comparison experiments. It can be seen that the highest

TABLE 2 Segmentation results on ShapeNetPart

Method	Input	Class mIoU	Instance mIoU	Air plane	Bag	Cap	Car	Chair	Ear phone	Guitar	Knife	Lamp	Laptop	Motor bike	Mug	Pistol	Rocket	Skate board	Table
Kd-Net[45]	4k	77.4	82.3	80.1	74.6	74.3	70.3	88.6	73.5	90.2	87.2	81.0	94.9	57.4	86.7	78.1	51.8	69.9	80.3
PointNet[20]	2k	80.4	83.7	83.4	78.7	82.5	74.9	89.6	73.0	91.5	85.9	80.8	95.3	65.2	93.0	81.2	57.9	72.8	80.6
PointNet++[21]	2k, nor	81.9	85.1	82.4	79.0	87.7	77.3	90.8	71.8	91.0	85.9	83.7	95.3	71.6	94.1	81.3	58.7	76.4	82.6
SCN[46]	1k	81.8	84.6	83.8	80.8	83.5	79.3	90.5	69.8	91.7	86.5	82.9	96.0	69.2	93.8	82.5	62.9	74.4	80.8
DGCNN[27]	2k	82.3	85.2	84.0	83.4	86.7	77.8	90.6	74.7	91.2	87.5	82.8	95.7	66.3	94.9	81.1	63.5	74.5	82.6
3DGCN[28]	1k	82.1	85.1	83.1	84.0	86.6	77.5	90.3	74.1	90.9	86.4	83.8	95.6	66.8	94.8	81.3	59.6	75.7	82.8
IGCN (ours)	**1k**	**81.1**	**84.3**	**81.1**	**79.9**	**82.9**	**76.8**	**89.6**	**77.7**	**89.8**	**85.8**	**82.0**	**95.1**	**64.8**	**95.1**	**81.3**	**60.1**	**73.2**	**82.6**

Note: "mIoU" represents the mean intersection over union, "4k" means the input is 4k points, and so forth, "nor" means the direction vector, and the accuracy in the table is a percentage (%).

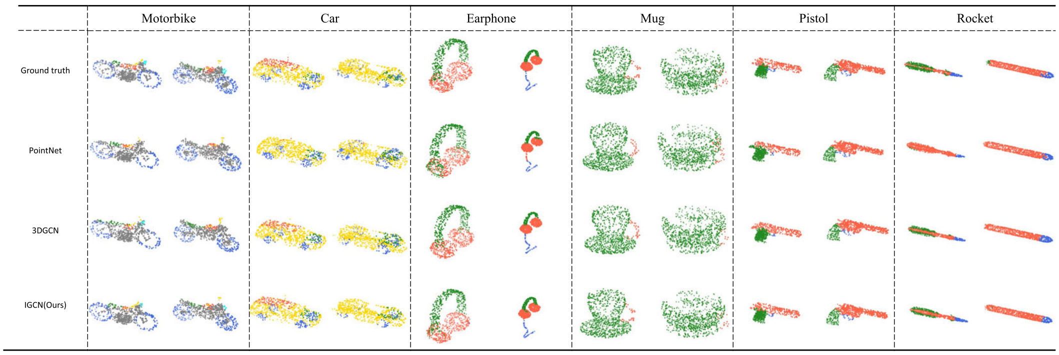

FIGURE 9 Visualization of segmentation results on ShapeNetPart data set. The segmentation experimental results of Motobike, Car, Earphone, Mug, Pistol, and Rocket are presented under Ground Truth, PointNet model, 3DGCN model, and IGCN model, respectively, where different colors represent different parts in instance, and each instance has 2–6 parts. 3DGCN, three-dimensional Graph Convolutional Network; IGCN, Interpolation Graph Convolutional Network. [Color figure can be viewed at wileyonlinelibrary.com]

TABLE 3 Ablation study

Module	Ablation experiments	OA (%)
Original model	PostWeight fusion and max method	90.1
Weight fusion	Preweight fusion	88.2
Aggregation function	Sum method	84.7
Aggregation function	Average method	83.0

Note: "OA" represents the overall accuracy for the classification of ModelNet40. Comparison of the impact of weight fusion and aggregation functions.

TABLE 4 Ablation study

Number of neighbors	OA (%)
16	87.8
24	88.8
32	90.1
48	89.6
64	88.4

Note: "OA" represents the overall accuracy for the classification of ModelNet40. Comparison of the impact of the number of neighbors.

classification accuracy is achieved when the number of neighboring points is 32. This is because the point cloud data are not evenly distributed and setting the number of neighboring points twice the average helps extract information in dense areas. Fewer neighbors will not extract features accurately, while a larger number of neighbors may contain redundant information, thus reducing accuracy.

Recall that we use a total of eight layers of IGCN for feature extraction and obtain $8 \times 32 = 256$ channels for classification and segmentation in our model. Here we use

TABLE 5 Ablation study

Number of layers	Feature extraction module structure	OA (%)
5	IGCN × 3 → Pooling → IGCN × 2	89.5
8	IGCN × 3 → Pooling → IGCN × 3 → Pooling → IGCN × 2	90.1
11	IGCN × 3 → Pooling → IGCN × 3 → Pooling → IGCN × 3 → Pooling → IGCN × 2	87.9
3	IGCN × 2 → Pooling → IGCN × 1	84.1
5	IGCN × 2 → Pooling → IGCN × 2 → Pooling → IGCN × 1	88.8
7	IGCN × 2 → Pooling → IGCN × 2 → Pooling → IGCN × 2 → Pooling → IGCN × 1	82.5

Note: "OA" represents the overall accuracy for the classification of ModelNet40. Comparison of the impact of the number of layers.

Abbreviation: IGCN, Interpolation Graph Convolutional Network.

different numbers of IGCN and pooling layers to evaluate our model. First we still use three layers of IGCN with one layer of Pooling as a group to build a five-layer structure and an 11-layer structure, respectively, as shown in Table 5. The classification accuracy of the ModelNet40 data set is also applied here for comparison and it can be seen that the eight-layer structure improves the accuracy by 0.6% compared with the five-layer structure and 2.2% compared with the 11-layer structure. This can be explained by the fact that a smaller number of layers leads to insufficient feature extraction capability, while a higher number of layers leads to over-fitting of the model, both of which affect the performance of the model. We then build three-, five-, and seven-layer structures with a group of two layers of IGCN and one layer of Pooling, and the results are shown in Table 5. It can be seen that the extraction ability of two-layer IGCN is significantly lower than that of three-layer IGCN, so the eight-layer feature extraction module constructed by three layers of IGCN with one layer of pooling is the optimal structure.

4.4 | Case study

Here we perform a case study, embedding our feature extraction module consisting of eight layers of IGCN into the PointNetLK framework for point cloud registration learning. In our case study, we are following a strategy of pretraining on the ModelNet40 data set and fine-tuning on the target data. First we embed the feature extraction module into the PointNetLK framework and then use the model parameters learned in the classification task as pretraining parameters for the point cloud registration task. Subsequently, the ModelNet data instances are split in two by means of settings in the PointNetLK framework and randomly rotated to be used for training the parameters of the registration task. Finally, we fine-tune the trained model on the WHU-TLS[29] data set, which is then used for the alignment task, and the visualization is shown in Figure 10. The WHU-TLS data set contains point cloud scans from 11 different environments, for example, subway station, high-speed railway platform, mountain, forest, park, campus, residence, riverbank, heritage building, underground excavation, and tunnel. Here we present a point cloud registration visualization of tunnels and underground excavations as an example. In Figure 10, the red color indicates the target data, the green

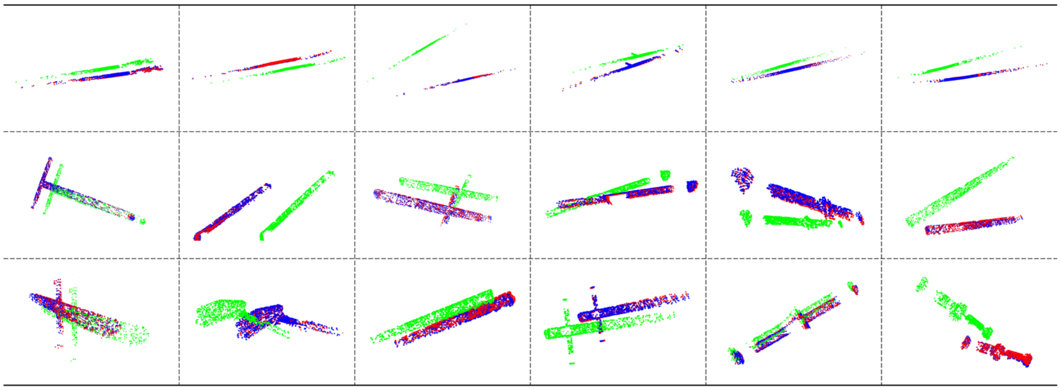

FIGURE 10 Visualization of point cloud registration results on the WHU-TLS data set. The red color indicates the target data, the green color indicates the source data, and the blue color indicates the alignment results. The first row contains six scenes from the tunnel, and the second and third rows contain 12 scenes from the underground excavation. [Color figure can be viewed at wileyonlinelibrary.com]

color indicates the source data, and the blue color indicates the alignment results. The first row contains six scenes from the tunnels as well as the second and third rows contain 12 scenes from the underground excavations. It can be seen that our IGCN feature extraction module can be used as a plug-and-play point cloud feature extraction module that can be easily embedded in other modules for other tasks as it fully mimics the operation of a standard CNN. The classification training results on ModelNet40 can also be used as pretraining parameters for other tasks. Our model converts the unordered and sparse point cloud feature extraction process into a process similar to a standard CNN, which will benefit the field of point cloud analysis and contribute to industries, such as underground mining.

5 | CONCLUSION

In this paper, we propose and validate the performance of an IGCN for point cloud classification and segmentation. Moreover, our IGCN feature extraction module acts as a plug-and-play point cloud feature extraction module that can be integrated with other frameworks for tasks, such as point cloud registration. Our model reduces the network parameters and model complexity via the interpolation method. It also uses specially designed postweight fusion and multilayer resolution aggregation to reduce the calculation cost of interpolation while achieving superior performance. Our proposed IGCN allows unordered and sparse point cloud feature extraction to be performed in a process similar to that of a standard CNN, contributing to the development of the field of point cloud analysis.

ACKNOWLEDGEMENT

This study was supported by the Cooperative Research Centres Projects (CRCP) Grants "DeepIoT" project. Open access publishing facilitated by University of New South Wales, as part of the Wiley - University of New South Wales agreement via the Council of Australian University Librarians.

DATA AVAILABILITY STATEMENT

(1) The data that support the findings of this study are available in ModelNet40 at 10.1109/CVPR.2015.7298801.[39] These data were derived from the following resources available in the public domain: https://modelnet.cs.princeton.edu/. (2) The data that support the findings of this study are available in ShapeNetPart at 10.1145/2980179.2980238.[44] These data were derived from the following resources available in the public domain: https://shapenet.org/. (3) The data that support the findings of this study are available in WHU-TLS at 10.1016/j.isprsjprs.2020.03.013.[29] These data were derived from the following resources available in the public domain: https://3s.whu.edu.cn/ybs/en/benchmark.htm.

ORCID

Yao Liu http://orcid.org/0000-0002-5271-0536

REFERENCES

1. Guo Y, Wang H, Hu Q, Liu H, Liu L, Bennamoun M. Deep learning for 3D point clouds: a survey. *IEEE Trans Pattern Anal Mach Intell*. 2021;43(12):4338-4364.
2. Xiang Y, Choi W, Lin Y, Savarese S. Data-driven 3D voxel patterns for object category recognition. In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7–12, 2015*. IEEE Computer Society; 2015:1903-1911.
3. Tatarchenko M, Dosovitskiy A, Brox T. Octree generating networks: efficient convolutional architectures for high-resolution 3D outputs. In: *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22–29, 2017*. IEEE Computer Society; 2017:2107-2115.
4. Cosmo L, Rodolà E, Bronstein MM, Torsello A, Cremers D, Sahillioglu Y. Partial matching of deformable shapes. In: *9th Eurographics Workshop on 3D Object Retrieval, 3DOR@Eurographics 2016, Lisbon, Portugal, May 8, 2016*. Eurographics Association; 2016.
5. Kuntz A, Fu M, Alterovitz R. Planning high-quality motions for concentric tube robots in point clouds via parallel sampling and optimization. In: *IEEE International Conference on Intelligent Robots and Systems*. IEEE; 2019:2205-2212.
6. Lang AH, Vora S, Caesar H, Zhou L, Yang J, Beijbom O. PointPillars: fast encoders for object detection from point clouds. In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16–20, 2019*. Computer Vision Foundation/IEEE; 2019:12697-12705.
7. Chen C, Huang T. Camdar-adv: generating adversarial patches on 3D object. *Int J Intell Syst*. 2021;36(3):1441-1453.
8. Rashed H, Ramzy M, Vaquero V, Sallab AE, Sistu G, Yogamani SK. FuseMODNet: real-time camera and LiDAR based moving object detection for robust low-light autonomous driving. In: *2019 IEEE/CVF International Conference on Computer Vision Workshops, ICCV Workshops 2019, Seoul, Korea (South), October 27–28, 2019*. IEEE; 2019:2393-2402.
9. Kot T, Novák P, Babjak J. Visualization of point clouds built from 3D scanning in coal mines. In: *Proceedings of the 2016 17th International Carpathian Control Conference, ICCC 2016*. IEEE; 2016:372-377.
10. Guo J, Jiang J, Wu L, Zhou W, Wei L. 3D modeling for mine roadway from laser scanning point cloud. In: *2016 IEEE International Geoscience and Remote Sensing Symposium, IGARSS 2016, Beijing, China, July 10–15, 2016*. IEEE; 2016:4452-4455.
11. Krizhevsky A, Sutskever I, Hinton GE. ImageNet classification with deep convolutional neural networks. *Commun ACM*. 2017;60(6):84-90.
12. Liu J, Zhu K, Lu W, Luo X, Zhao X. A lightweight 3D convolutional neural network for deepfake detection. *Int J Intell Syst*. 2021;36(9):4990-5004.
13. Zheng X, Yu X, Yin Y, Li T, Yan X. Three-dimensional feature maps and convolutional neural network-based emotion recognition. *Int J Intell Syst*. 2021;36(11):6312-6336.
14. Ghaderzadeh M, Aria M, Hosseini A, Asadi F, Bashash D, Abolghasemi H. A fast and efficient CNN model for B-ALL diagnosis and its subtypes classification using peripheral blood smear images. *Int J Intell Syst*. 2022;37(8):5113-5133.

15. Liu G, Zhang Q, Cao Y, Tian G, Ji Z. Online human action recognition with spatial and temporal skeleton features using a distributed camera network. *Int J Intell Syst*. 2021;36(12):7389-7411.
16. Su H, Maji S, Kalogerakis E, Learned-Miller EG. Multi-view convolutional neural networks for 3D shape recognition. In: *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7–13, 2015*. IEEE Computer Society; 2015:945-953.
17. Feng Y, Zhang Z, Zhao X, Ji R, Gao Y. GVCNN: group-view convolutional neural networks for 3D shape recognition. In: *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18–22, 2018*. Computer Vision Foundation/IEEE Computer Society; 2018:264-272.
18. Maturana D, Scherer SA. VoxNet: a 3D convolutional neural network for real-time object recognition. In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2015, Hamburg, Germany, September 28–October 2, 2015*. IEEE; 2015:922-928.
19. Le T, Duan Y. PointGrid: a deep network for 3D shape understanding. In: *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18–22, 2018*. Computer Vision Foundation/IEEE Computer Society; 2018:9204-9214.
20. Qi CR, Su H, Mo K, Guibas LJ. PointNet: deep learning on point sets for 3D classification and segmentation. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21–26, 2017*. IEEE Computer Society; 2017:77-85.
21. Qi CR, Yi L, Su H, Guibas LJ. PointNet++: deep hierarchical feature learning on point sets in a metric space. In: *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4–9, 2017, Long Beach, CA, USA*; 2017:5099-5108.
22. Zhao H, Jiang L, Fu C, Jia J. PointWeb: enhancing local neighborhood features for point cloud processing. In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16–20, 2019*. Computer Vision Foundation/IEEE; 2019:5565-5573.
23. Wu W, Qi Z, Li F. PointConv: deep convolutional networks on 3D point clouds. In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16–20, 2019*. Computer Vision Foundation/IEEE; 2019:9621-9630.
24. Liu Y, Fan B, Xiang S, Pan C. Relation-shape convolutional neural network for point cloud analysis. In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16–20, 2019*. Computer Vision Foundation/IEEE; 2019:8895-8904.
25. Liu Y, Fan B, Meng G, Lu J, Xiang S, Pan C. DensePoint: learning densely contextual representation for efficient point cloud processing. In: *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27–November 2, 2019*. IEEE; 2019:5238-5247.
26. Simonovsky M, Komodakis N. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21–26, 2017*. IEEE Computer Society; 2017:29-38.
27. Wang Y, Sun Y, Liu Z, Sarma SE, Bronstein MM, Solomon JM. Dynamic graph CNN for learning on point clouds. *ACM Trans Graph*. 2019;38(5):146:1-146:12.
28. Lin Z, Huang S, Wang YF. Convolution in the cloud: learning deformable kernels in 3D graph convolution networks for point cloud analysis. In: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13–19, 2020. Computer Vision Foundation/IEEE; 2020: 1797-1806.
29. Dong Z, Liang F, Yang B, et al. Registration of large-scale terrestrial laser scanner point clouds: a review and benchmark. *ISPRS J Photogramm Remote Sens*. 2020;163:327-342.
30. Hua S, Liu Q, Yin G, Guan X, Jiang N, Zhang Y. Research on 3D medical image surface reconstruction based on data mining and machine learning. *Int J Intell Syst*. 2022;37(8):4654-4669.
31. Hua B, Tran M, Yeung S. Pointwise convolutional neural networks. In: *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18–22, 2018*. Computer Vision Foundation/IEEE Computer Society; 2018:984-993.
32. Besl PJ, McKay ND. A method for registration of 3-D shapes. *IEEE Trans Pattern Anal Mach Intell*. 1992;14(2):239-256.
33. Deng H, Birdal T, Ilic S. PPFNet: global context aware local features for robust 3D point matching. In: *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18–22, 2018*. Computer Vision Foundation/IEEE Computer Society; 2018:195-205.

34. Deng H, Birdal T, Ilic S. PPF-FoldNet: unsupervised learning of rotation invariant 3D local descriptors. In: *Proceedings of the Computer Vision—ECCV 2018—15th European Conference, Munich, Germany, September 8–14, 2018, Part V*. Lecture Notes in Computer Science. Vol 11209. Springer; 2018:620-638.
35. Gojcic Z, Zhou C, Wegner JD, Wieser A. The perfect match: 3D point cloud matching with smoothed densities. In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16–20, 2019*. Computer Vision Foundation/IEEE; 2019:5545-5554.
36. Choy C, Park J, Koltun V. Fully convolutional geometric features. In: *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27–November 2, 2019*. IEEE; 2019: 8957-8965.
37. Aoki Y, Goforth H, Srivatsan RA, Lucey S. PointNetLK: robust & efficient point cloud registration using PointNet. In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16–20, 2019*. Computer Vision Foundation/IEEE; 2019:7163-7172.
38. Lucas BD, Kanade T. An iterative image registration technique with an application to stereo vision. In: *Proceedings of the 7th International Joint Conference on Artificial Intelligence, IJCAI '81, Vancouver, BC, Canada, August 24–28, 1981*. William Kaufmann; 1981:674-679.
39. Wu Z, Song S, Khosla A, et al. 3D ShapeNets: a deep representation for volumetric shapes. In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7–12, 2015*. IEEE Computer Society; 2015:1912-1920.
40. Riegler G, Ulusoy AO, Geiger A. OctNet: learning deep 3D representations at high resolutions. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21–26, 2017*. IEEE Computer Society; 2017:6620-6629.
41. Qi CR, Su H, Nießner M, Dai A, Yan M, Guibas LJ. Volumetric and multi-view CNNs for object classification on 3D data. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27–30, 2016*. IEEE Computer Society; 2016:5648-5656.
42. Zaheer M, Kottur S, Ravanbakhsh S, Póczos B, Salakhutdinov R, Smola AJ. Deep sets. In: Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4–9, 2017, Long Beach, CA, USA; 2017:3391-3401.
43. Mao J, Wang X, Li H. Interpolated convolutional networks for 3D point cloud understanding. In: *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27–November 2, 2019*. IEEE; 2019:1578-1587.
44. Yi L, Kim VG, Ceylan D, et al. A scalable active framework for region annotation in 3D shape collections. *ACM Trans Graph*. 2016;35(6):210:1-210:12.
45. Klokov R, Lempitsky VS. Escape from cells: deep Kd-networks for the recognition of 3d point cloud models. In: *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22–29, 2017*. IEEE Computer Society; 2017:863-872.
46. Xie S, Liu S, Chen Z, Tu Z. Attentional ShapeContextNet for point cloud recognition. In: *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18–22, 2018*. Computer Vision Foundation/IEEE Computer Society; 2018:4606-4615.

How to cite this article: Liu Y, Yao L, Li B, Sammut C, Chang X. Interpolation graph convolutional network for 3D point cloud analysis. *Int J Intell Syst*. 2022;37:12283-12304. doi:10.1002/int.23087