frontiers | Frontiers in Computational Neuroscience

TYPE Original Research
PUBLISHED 30 August 2023
DOI 10.3389/fncom.2023.1204445

OPEN ACCESS

EDITED BY
Si Wu,
Peking University, China

REVIEWED BY
Laith Abualigah,
Amman Arab University, Jordan
Saghir Alfasly,
Mayo Clinic, United States

*CORRESPONDENCE
Chibiao Liu
✉ lcbsmc@163.com

RECEIVED 12 April 2023
ACCEPTED 03 August 2023
PUBLISHED 30 August 2023

CITATION
Gezawa AS, Liu C, Jia H, Nanehkaran YA, Almutairi MS and Chiroma H (2023) An improved fused feature residual network for 3D point cloud data.
Front. Comput. Neurosci. 17:1204445.
doi: 10.3389/fncom.2023.1204445

An improved fused feature residual network for 3D point cloud data

Abubakar Sulaiman Gezawa[1], Chibiao Liu[1]*, Heming Jia[1], Y. A. Nanehkaran[2], Mubarak S. Almutairi[3] and Haruna Chiroma[4]

[1]College of Information Engineering, Fujian Key Lab of Agriculture IOT Application, Sanming University, Sanming, Fujian, China, [2]Department of Software Engineering, School of Information Engineering, Yancheng Teachers University, Yancheng, Jiangsu, China, [3]College of Computer Science and Engineering, University of Hafr Al-Batin, Hafar Al Batin, Saudi Arabia, [4]College of Computer Science and Engineering Technology, Applied College, University of Hafr Al-Batin, Hafar Al Batin, Saudi Arabia

Point clouds have evolved into one of the most important data formats for 3D representation. It is becoming more popular as a result of the increasing affordability of acquisition equipment and growing usage in a variety of fields. Volumetric grid-based approaches are among the most successful models for processing point clouds because they fully preserve data granularity while additionally making use of point dependency. However, using lower order local estimate functions to close 3D objects, such as the piece-wise constant function, necessitated the use of a high-resolution grid in order to capture detailed features that demanded vast computational resources. This study proposes an improved fused feature network as well as a comprehensive framework for solving shape classification and segmentation tasks using a two-branch technique and feature learning. We begin by designing a feature encoding network with two distinct building blocks: layer skips within, batch normalization (BN), and rectified linear units (ReLU) in between. The purpose of using layer skips is to have fewer layers to propagate across, which will speed up the learning process and lower the effect of gradients vanishing. Furthermore, we develop a robust grid feature extraction module that consists of multiple convolution blocks accompanied by max-pooling to represent a hierarchical representation and extract features from an input grid. We overcome the grid size constraints by sampling a constant number of points in each grid using a simple K-points nearest neighbor (KNN) search, which aids in learning approximation functions in higher order. The proposed method outperforms or is comparable to state-of-the-art approaches in point cloud segmentation and classification tasks. In addition, a study of ablation is presented to show the effectiveness of the proposed method.

KEYWORDS

point clouds, part segmentation, classification, shape features, 3D objects recognition

1. Introduction

Three-dimensional (3D) data are a great asset in the computer vision field since it contains detailed information on the whole geometry of detected objects and scenes. With the availability of massive 3D datasets and processing power, it is now possible to apply deep learning to learn specific tasks on 3D data such as segmentation with classification (Varga et al., 2020; Ergün and Sahillioglu, 2023; Qi et al., 2023), recognition, and correspondence

(Long et al., 2021). There are several categories of 3D data representations including point cloud, voxel, mesh, multi views, octree, and many others. A comprehensive overview of point clouds and other 3D data representations may be found in the study by Bello et al. (2020) and Gezawa et al. (2020). Point cloud data processing employs a variety of approaches. Following dispatching a point cloud to a voxel grid that is quantized spatially in the grid space, volumetric models use a volumetric convolution to compute (Maturana and Scherer, 2015; Choy et al., 2016). Volumetric approaches correlate points with grid positions by using grids as data structuring technique and convolutional kernels in 3D to get data from nearby voxels. Although grid data structures are efficient, to maintain the granularity of the data position, a high voxel resolution is essential. The amount of processing and memory used grows in a cubical relationship with the voxel resolution since large point clouds are expensive to process. Furthermore, most point clouds contain ∼90% empty voxels (Zhou and Tuzel, 2018), processing no data could use a lot of computing power. Point-based models are another type of point cloud data processing paradigm. Unlike volumetric models, point-based models offer effective computation but have poor data organization. For instance, PointNet (Charles et al., 2017) aggregates the data in the network's final stage using the point cloud without quantization, as a result the precise locations of the data are preserved. However, the cost of computation rises in lockstep with the point number. Subsequent studies (Qi et al., 2017; Wang et al., 2018; Yifan et al., 2018; Qiangeng et al., 2019; Wang Y. et al., 2019) aggregate information using a downsampling approach at each layer. Graph convolutional networks (GCN) have been used in the network layer to generate a local graph for each point cluster (Simonovsky and Komodakis, 2017; Kuangen et al., 2019; Wang L. et al., 2019; Li et al., 2023) that can be regarded as a variant of the PointNet++ design (Qi et al., 2017). This architecture, however, is costly in terms of data structuring [e.g., Random Point Sampling (RPS)]. As reported by Zhijian et al. (2019), data structuring costs account for up to 88% of the entire computational cost in three common point-based models (Li Y. et al., 2018; Yifan et al., 2018; Wang Y. et al., 2019). Furthermore, SO-Net (Li J. et al., 2018) employs the self-organizing map (SOM; Kohonen, 1998) to create a set of points used to model a point cloud's spatial pattern. Even though SO-Net considers a point cloud's regional correlation, SOM is trained independently. As a result, SOM's spatial modeling and a specific point cloud task are no longer coupled. DGCB-Net (Tian et al., 2020) uses cutting-edge convolutional layers built by weight-shared multiple-layer perceptrons (MLPs), to automatically extract local features from the point cloud graph structure. A feature aggregation is formed by concatenating the features received from all edge convolutional layers. Rather than stacking multiple layers deep, the DGCB-Net adopts a strategy to flatly extend point cloud feature aggregation.

In this study, we utilize deep learning to develop an approach that manage enormous 3D object datasets without compromising shape resolution. The majority of handcrafted 3D features are limited to low 3D resolutions. For example, Chiotellis et al. (2016) and Zhou and Tuzel (2018) require each 3D model in the datasets to be down-sampled to 20,000 faces with Meshlab before they can be fed into the system. Additionally, a method is provided that can handle structural variations in 3D objects without the need for data pre-processing. Many machine learning algorithms, such as the support vector machine (SVM), are effective when the datasets are small and well-curated, which implies that the data have been carefully pre-processed and requires human intervention. To address these challenges, this study offers an improved fused feature network, an end-to-end framework that solves shape classification and segmentation tasks using a two-branch technique with feature representation learning. To efficiently simplify the network, we start by developing a feature encoding network with two independent building blocks and layer skips with batch normalization and ReLU in between. Because there are few layers through which to propagate, using the layer skips speeds up learning and lessens the effect of gradients vanishing. Figure 1 presents the entire network structure of the approach. In addition, we create a detail grid feature extraction module, which comprises various convolution blocks accompanied by a max-pooling to represent a hierarchical representation of several feature representations and extracts features from the input grid. Max-pooling is used in each of the pooling layers, resulting in each spatial dimension having a smaller grid and helps to manage overfitting by gradually lowering the representation's spatial dimension, the parameters in the network, and the amount of processing. This module includes a regular-structured enclosing volumetric grid that helps capture details and features hierarchically. To extract features of high-resolution inputs, this module is utilized in conjunction with the feature encoding network. To pull through the limitation of the grid size, the local region in every grid sampled a constant number of points using a simple KNN search which aids in learning approximation functions in higher order to better characterize the details of the features.

Our major contributions are as follows:

- We design an effective module named detail grid feature extraction (DGFE) module. This module aids 3D convolutions to hierarchically capture global information and reduces the grid size in each spatial dimension as well as managing overfitting by gradually lowering the spatial dimension of the representation making it viable for high-resolution 3D objects.
- We design a feature encoding network that uses two different building blocks with layer skips containing batch normalization and ReLU in between, resulting in fewer layers in the early training phase which helps speed learning and reduces the effect of gradients vanishing since there are few layers through which to propagate.
- We built a network using the modules that have been proposed, which achieves a notable balance of accuracy and speed.

2. Related work

2.1. 3D learning using voxel-based methods

To build on the advance of CNN models on images (He et al., 2016a; Huang et al., 2017), Voxnet and its revisions (Maturana and Scherer, 2015; Wang and Posner, 2015; Wu et al., 2015; Brock et al., 2016) start by converting a point cloud to a grid occupancy and then used convolution in a volumetric form. To overcome the problem

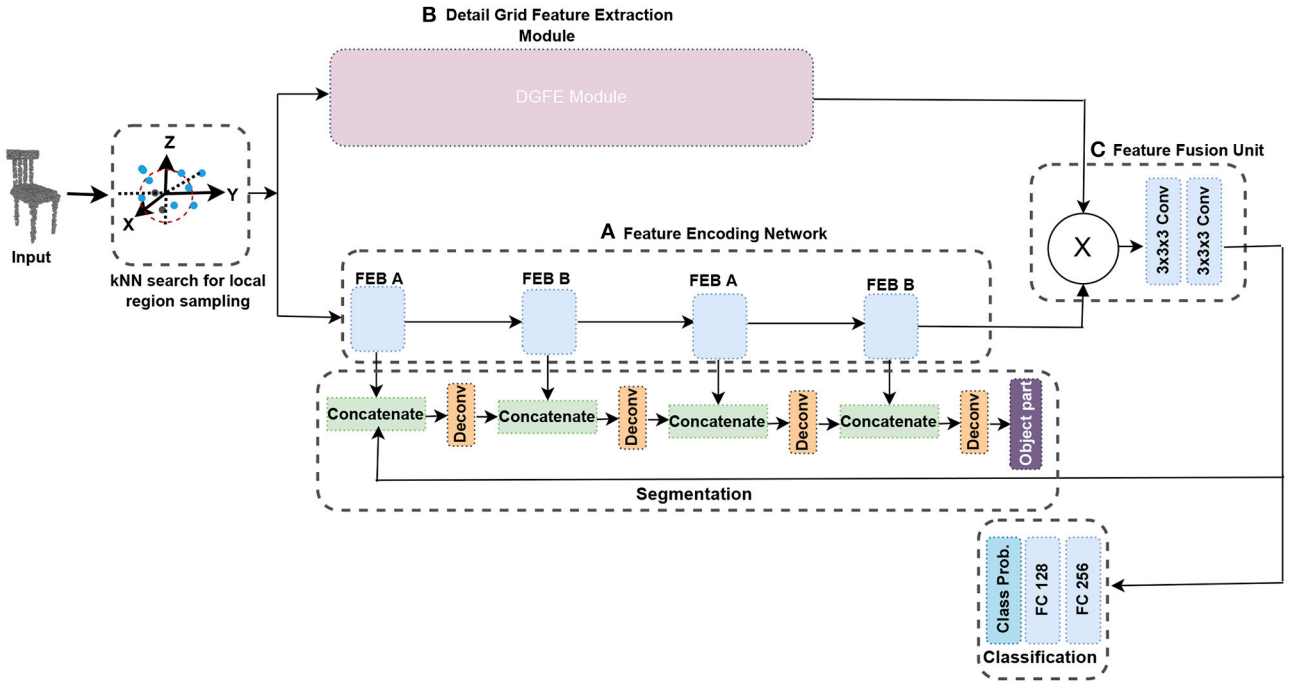

FIGURE 1
The complete architecture of the proposed method. The network is divided into three branches. The feature encoding network extract features from the input grid in **(A)**. The DGFE module exploits the detailed shape characteristics in **(B)**. The feature fusion unit which has two consecutive convolutional layers, fuses the features from the two branches to produce a feature with improved contextual representation by exploiting both local and global shape structures in **(C)**. See also Section 3.5.

of rising memory usage due to cubical expansion, OctNet creates structures like a tree for non-empty voxels to avoid computing in space. While the volumetric approach is effective at structuring data, it suffers from poor computational effectiveness and data granularity loss. Transformers have lately been incorporated into the model designs of many 3D vision approaches in response to the success of transformer-based designs in the two-dimensional (2D) domain. The transformer has improved previous 3D learning techniques because of its ability to read remote input and provide task-specific inductive biases. The point-voxel transformer for single-stage 3D detection (PVT-SSD) proposed by Yang et al. (2023) uses input-dependent query initialization and voxel-based sparse convolutions for strong feature encoding. The PVT-SSD overcame the drawbacks of both point clouds and voxels by combining their advantages. To reduce farthest point sampling (FPS) runtime, they used sparse convolutions to transform points into a limited number of voxels rather than directly sampling them. They also sampled non-empty voxels. The voxel features were adaptively blended with the point features to make up for the difficulty of quantization.

2.2. 3D learning using point cloud-based methods

Charles et al. (2017); Qi et al. (2017) pioneered the use of point-based models which used pooling to aggregate the point features to achieve the permutation invariant. To better capture local characteristics, methods such as kernel correlation (Atzmon et al., 2018; Wu et al., 2019) and extended convolutions (Thomas et al., 2019) are proposed. To resolve the ambiguity, the local point order is predicted by PointCNN (Li Y. et al., 2018) while RSNet (Huang et al., 2018) sequentially consumes points from various directions. In methods based on points, the cost of computation grows linearly with the points input. The cost of structuring data, nevertheless, turned out to be a performance bottleneck for large inputs. Recently, a dynamic sparse voxel transformer (DSVT) was presented by Wang et al. (2023) in an effort to widen the uses of transformers so that they may serve as a solid foundation for outdoor 3D perception just as they do for 2D vision. A number of local regions are split up into smaller ones in each window using DSVT based on sparsity, and each window's attributes are then computed fully in parallel. Another recent point cloud classification framework named point content-based transformer (PointConT) was introduced by Liu et al. (2023), and it employs local self-attention in the space of features rather than the 3D space. One of the main advantages of PointConT is that it takes advantage of the locality of points in the feature space by clustering sampled points with similar features into the same class and computing self-attention within each class, allowing for an efficient trade-off between collecting long-range dependencies and computational complexity.

2.3. Strategies for point data structuring

The majority of point-based methods (Qi et al., 2017; Li Y. et al., 2018; Bello et al., 2021; Gezawa et al., 2021) employ FPS (Eldar et al., 1997) to sample uniformly distributed group centers. However, it

does not account for the subsequent processing of the sampled points which may result in suboptimal performance. Random point sampling (RPS) has the advantage of having a minimal downtime. It is indeed, nevertheless, sensitive to variation in density. The KNN search we used for sampling the local region in each grid cell combines sampling and neighbor querying in a single step, making it faster than RPS.

SO-Net (Li J. et al., 2018), on the other hand, creates a self-organizing map. To split the spaces, KDNet (Klokov and Lempitsky, 2017) employs kd-tree. Gumble subset sampling is used instead of FPS by Yang et al. (2019). To create super points, Landrieu and Simonovsky (2018) employs a clustering algorithm. The majority of these approaches are either too slow or necessitate structure preprocessing. VoxelNet (Le and Duan, 2018; Zhou and Tuzel, 2018), for example, blends point-based and volumetric approaches by performing voxel convolution and employing the study by Charles et al. (2017) inside each voxel. Similar concepts are used by the fast model (Zhijian et al., 2019), whereas Lu et al. (2022) made use of ball query with graph convolution layers. However, the number of points is not steadily decreased over all layers. Our DGFE module, however, utilized max-pooling in each of the pooling layers, resulting in each spatial dimension having a smaller grid allowing it to be used for high-resolution 3D objects. Apart from those features, the local region in every grid sampled a constant number of points using a simple KNN search which aids in learning approximation functions in higher order to better characterize the detailed features.

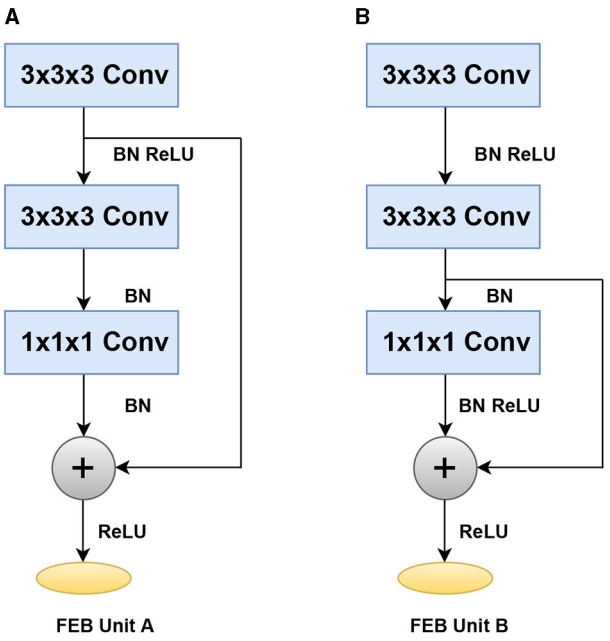

FIGURE 2
Building blocks of the feature encoding network with two different layer skips. **(A)** Feature encoding block (FEB unit A) **(B)** Feature encoding block (FEB unit B).

3. The proposed method

In this section, the KNN search for local region sampling is first introduced. Following that, we propose the feature encoding network that serves as the basis of the enhanced fused feature network. The split-transform-merge paradigm, which is based on the residual learning framework, is one of the primary building block we employ to design our feature encoding network (Figure 1A). One of the primary benefits of employing the residual network is its simplicity in training networks with many layers without raising the training error percentage. It also aids in solving the vanishing gradient problem by applying identity mapping. To compensate for structural changes in 3D objects, our feature encoding network employs two different building blocks [feature encoding block (FEB) unit A and feature encoding block (FEB) unit B], with layer skips in between. We begin with 3x3x3 convolutions twice, followed by 1x1x1 convolutions with a stride in each convolution to accommodate both small and large datasets without possible overfitting and to lower the spatial dimension of the representation. Then, we introduce the detail grid feature extraction module and finally the feature fusion unit. The complete framework is presented Figure 1.

3.1. KNN search for local region sampling

Point clouds are typically represented as raw coordinates of points in 3D space. Here, we will go over how our model extracts features from 3D objects when given a point cloud of number of points (N) as input. When provided with an input of N $\times$ 3 set of point clouds, the object is then subdivided into equal-sized 3D voxels, such as 64 $\times$ 64 $\times$ 64, 16 $\times$ 16 $\times$ 16 or 8 $\times$ 8 $\times$ 8. Using KNN, K points will be sampled from each grid cell. To avoid extra computation, those with empty points will be padded with zeros. In contrast to standard KNN, in which the search area consists of all points, it just needs to search among non-empty voxels in our situation, making the query much faster. Unlike VoxNet (Maturana and Scherer, 2015) which represents the 3D structure using an occupancy grid, we build a grid from point clouds and designate the grid's key feature to the points that are inside each grid. Some grids, on the other hand, may contain a different point number. This implies that we need a grid that will share kernels in 3D convolution. Moreover, for addressing this constraint, we utilized a sampling strategy that ensures each grid has an equal point number. In particular, if there are beyond K points in the grid, we use the KNN sampling strategy to choose K points from the total points. K points are sampled with substitution when the points inside a grid are below K. Consequently, each grid will have the same number of points, allowing us to encode the grid feature so that each grid feature has the same feature size vector which enables us to extract hierarchical features of the object using 3D convolutional kernels.

3.2. Feature encoding network

We concentrate on developing a robust network for shape classification and segmentation that achieves a notable balance of accuracy and speed. The feature encoding network is one of the key blocks that we create by making use of the split-transform-merge

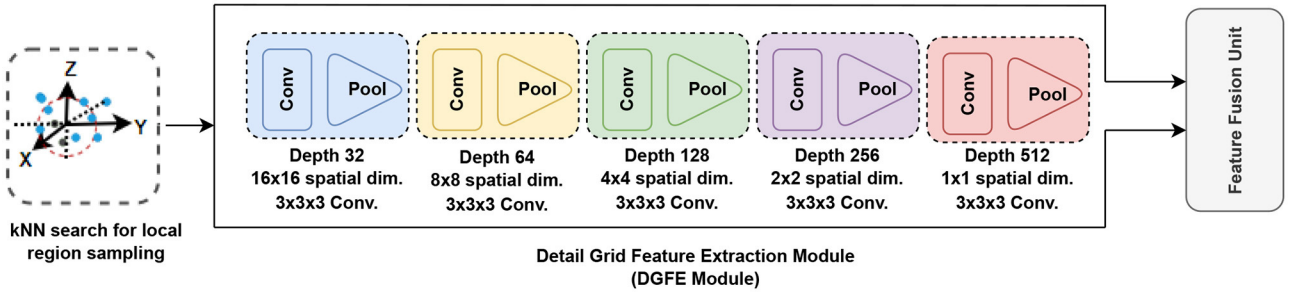

FIGURE 3
Detail grid feature extraction module (DGFE Module). This module extracts features from the input grid using many convolution blocks. Max-pooling is used in each of the pooling layers, resulting in each spatial dimension having a smaller grid and helps to manage overfitting by gradually lowering the representation's spatial dimension, the parameters in the network, and the amount of processing.

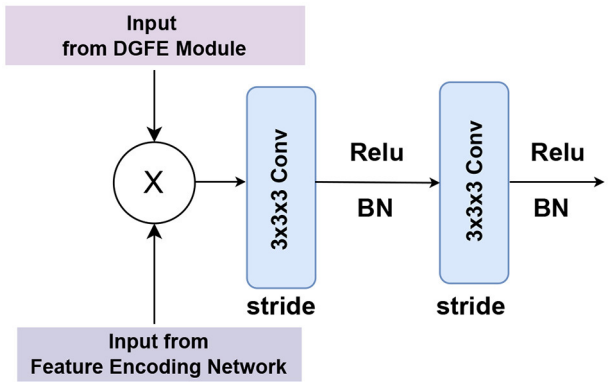

FIGURE 4
Illustration of the detailed design of the feature fusion unit, which consists of two consecutive 3x3x3 convolutions with BN and ReLU in between, as well as a stride in each convolution to help manage overfitting.

paradigm, inspired by the residual learning framework design in the study by Szegedy et al. (2015), He et al. (2016a,b), and Elhassan et al. (2021) and leveraging its powerful representational ability. These networks are scalable structures that bundle building units with the same linked shape which are referred to as residual units or blocks. The original blocks in the study by He et al. (2016b) compute as follows:

$$O_i = h(I_i) + f\left(I_i, Weights_i\right), \tag{1}$$

$$I_{i+1} = f(O_i). \tag{2}$$

In this case, I_i represents the i-th block's input feature. $Weights_i = \left\{Weights_i, k \mid 1 \le k \le K\right\}$ contains biases and weights connected to block i-th. K stands for total layers in a block. f signifies the block function, such as a pile of convolutional layers of two 3x3 in Equation 1. The operation following element-wise addition is represented by the function f, which is ReLU in Equation 1. The h function is designated as an identity mapping: $h(I_i) = I_i$. Similarly, if function f is identity mapping, $I_{i+1} \equiv O_i$. Putting Equation 2 into Equation 1 yields:

$$I_{i+1} = I_i + f\left(I_i, Weights_i\right). \tag{3}$$

To efficiently accelerate training and reduce the number of parameters, the feature encoding network uses two separate construction blocks, such as Feature encoding block (FEB unit A) and feature encoding block (FEB unit B), with layer skips containing batch normalization (BN) and ReLU in between. The BN and ReLU are regarded as the weight layers' pre-activation, according to He et al. (2016b). We make some minor changes here by using the ReLu with BN and Conv before the addition of operation. We start with 3x3x3 convolutions twice, followed by 1x1x1 convolutions, and then we apply the BN and ReLu before the addition. We use a stride in each convolution to help manage overfitting by gradually reducing the spatial dimension of the representation. The feature encoding network's design is shown in Figure 2.

3.3. Detail grid feature extraction module

To represent numerous hierarchical feature representations, the detail grid feature extraction module employs several convolution blocks and max-pooling and extracts features from the input grid, as shown in Figure 3. Max-pooling is used in each of the pooling layers, resulting in each spatial dimension having a smaller grid and helps to manage overfitting by gradually lowering the representation's spatial dimension, the parameters in the network, and the amount of processing. BN (Ioffe and Szegedy, 2015) can be done to any set of network activations using:

$$y = g(Hu + p) \tag{4}$$

where H and p are model parameters that have been learned, and $g(.)$ denotes a non-linearity being ReLU or sigmoid. By normalizing $z = Hu + p$, the BN transform can be introduced right before the non-linearity. Since z is normalized, $y = g(Hu + p)$ can be replaced with

$$y = g(BN(Hu)) \tag{5}$$

where the BN (Ioffe and Szegedy, 2015) is used separately for each dimension of $z = Hu$, with a distinct set of learned parameters for each dimension. We utilized a $3 \times 3 \times 3$ kernel with stride 1 convolution and a ReLU (Nair and Hinton, 2010) in each convolution layer. The initial block employs 32-filter convolutions, which are then doubled in subsequent blocks. This module offers a regular-structured embedding volumetric grid that supports 3D convolutions in hierarchically capturing global information. To extract features of high-resolution inputs, this module is utilized in conjunction with the feature encoding network. To keep local fine details in early encoder layers, at the same spatial resolution, we connect the encoder network's encoded features to equivalent features extracted from the detail grid feature extraction module.

3.4. Feature fusion unit

The feature fusion unit is made up of two consecutive convolutional layers. We used $3 \times 3 \times 3$ convolutions twice, with BN and ReLU in between, and a stride in each convolution to help manage overfitting. The proposed DGFE module and the encoding network outputs are fused using a cross-product in the feature fusion unit, as shown in Figure 4, to produce a feature with improved contextual representation.

3.5. Network overview

We built a 3D convolutional network with fixed points inside each grid cell, which aids in the learning of local approximation functions in high-order that better capture local shape features. Figure 1 presents a diagram of the proposed architecture. The network is made up of two major modules. A feature encoding network that serves as the foundation for extracting features from the input grid, as shown in Figure 1A in Section 3.2, and detail grid feature extraction (DGFE) module which comprises various convolution blocks accompanied with an operation of max-pooling to help in representing several relational features and pull out features from the input (Section 3.3). We hierarchically combine these two modules to form the proposed improved fused feature network. The proposed DGFE module and the encoding network outputs are fused in the feature fusion unit containing two consecutive convolutional layers (Figure 1C) to produce a feature with improved contextual representation by utilizing both local and global shape structures.

The point cloud is first normalized within the unit box. In each grid, the coordinates of the points are piled as features. accordingly, given the appropriate x, y, and z coordinates, a K-point grid has features 3K. In theory, by dividing the sum of points (P) by its grid cells, K can be approximated. To acquire classification scores, the resulting fused feature can be categorized using two fully connected layers. Finally, one additional fully connected layer is added, along with a softmax, which aids in regressing the likelihood in every group. The whole layer's nodes correspond to the set of categories of objects inside the dataset. To generate the segmentation, the segmentation network decodes the retrieved features. To create the output, this network upsamples and combines the features. For every cell inside the grid, this network produces K+1 labels, as for K points in that cell equivalent to K labels and one more label level cell. Obtaining ground truth labels of object components, we chose its greater label among the labels of points within every cell. Unoccupied Cells are tagged "no label." Before actually acquiring the object part, we perform a deconvolution operation by concatenating the feature obtained from the feature fusion unit, with the feature retrieved out of each block of the feature encoding network.

4. Experiments

In this section, a number of datasets including ModelNet10 and ModelNet40 (Wu et al., 2015) for object classification and part segmentation on ShapeNetPart (Yi et al., 2016) were used to assess the performance of the proposed network. We discuss the dataset's specifics and the evaluation metrics in Section 4.1. The implementation protocol discussion presented in Section 4.2. In Sections 4.3, 4.4, and 4.5, we discuss some experimental results from applying the proposed network to classify shapes on ModelNet, measure precision-recall on ModelNet10, and segment parts on ShapeNetPart. In Section 4.6, we demonstrate the advantages of the proposed method by conducting a good set of ablation experimental tests to evaluate various setup adjustments.

4.1. Datasets and evaluation metrics

ModelNet dataset: This is indeed a notable dataset. It comprises two datasets with CAD models in 10 and 40 categories, respectively. ModelNet10 is made up of 4,899 object instances including 2,468 training samples and 909 testing samples. ModelNet40 is made up of 12,311 object instances, 9,843 of which are in the training set and 3,991 samples in the testing set. For object classification on the ModelNet dataset, we employed accuracy as the assessment metric.

ShapeNetPart dataset: There are 16,881 shapes in this dataset, divided into 16 categories and annotated with a combined amount of 50 components. A considerable share of shape categories is partitioned into 2–5 segments. We, then, used mean intersection over union (mIoU) for evaluation. For every part shape within the object category, we calculate the union of prediction and ground truth. The mIoU was computed using Equation 6 as follows:

$$mIoU = \frac{X}{X + G - P} \tag{6}$$

where G, P, and X denote the number of ground truth points, predicted positive points, and true positive points, respectively. The mIoU is obtained by taking the average of each class's IoU.

4.2. Implementation protocol

In Python, the proposed method was implemented using the Tensorflow deep learning library. Each experiment is conducted on an Nvidia Geforce Titan GTX GPU, CUDA 10.1, and CuDNN 7.1 with RAM of 12 GB. For the classification task, we test with various

parameters setup including different grid sizes and K values. Each point's location is jittered with a standard deviation of 0.02. The batch size is 32, and batch normalization is used for all layers. For both the segmentation and classification tasks, we used the cross-entropy loss to improve the discrimination of the class features. We utilized an initial learning rate of 10^{-4} and employ Adam optimizer (Kingma and Ba, 2015).

Loss function: Over the years, a wide range of loss functions have been proposed to perform 3D shape analysis tasks. For example, the cross-entropy loss was already been utilized successfully in many shape analysis tasks. Although the network can be trained using cross-entropy loss alone, we employ a combination of Shape loss (Wei et al., 2020) and modified cross-entropy loss (Huang et al., 2019) to make the class features more discriminatory. The Shape Loss is given as follows:

$$L_{shape} = L_s\left(C(S), M\right) \tag{7}$$

where M is the shapes's class label, L_s is a cross-entropy loss based on shape feature S, and C is a classifier.

Moreover, the cross-entropy loss is given as follows:

$$L_{cross-entropy} = \frac{1}{n}\sum_{y}\left(zlogQ + (1-z)log(1-Q)\right) \tag{8}$$

For each sample, $Q \in [0, 1]$ is the likelihood of the network output and z represents the class ground truth. To minimize the weight of easily categorized samples, the cross-entropy function can be reshaped by inserting a hyperparameter that aids in weight balancing.

$$L_{cross-entropy} = \frac{1}{n}\sum_{y}\left[z(1-Q)^{\gamma} logQ + (1-Q)Q^{\gamma} log(1-Q)\right] \tag{9}$$

Once a sample is successfully identified, $Q \rightarrow 1$, the factor $(1-Q) \rightarrow 0$; Alternatively, when Q is small, the factor $(1-Q)$ approaches 1. Our total loss is the combination of this two losses as follows:

$$L_{total} = L_{shape} + L_{cross-entropy} \tag{10}$$

4.3. Classification on ModelNet

We use the PointNet (Charles et al., 2017) convention to prepare the data. Input points are set to 1,024 by default. Furthermore, we improve performance by incorporating more points and surface normal. To analyze various models to varying degrees of speed and accuracy, the network is trained with varying settings to balance speed and performance (Section 4.6). The variants are in different grid sizes and K values.

4.3.1. Classification on ModelNet10

Comparison: The proposed improved fused feature residual network approach was compared with a number of state-of-the-art methods, as shown in Table 1. The proposed method outperforms the majority of previous voxel-based techniques in terms of "overall accuracy" including VoxNet (Maturana and Scherer, 2015), 3DShapeNets (Wu et al., 2015), 3DGAN (Wu et al., 2016), VSL (Liu et al., 2018), and BV-CNN's (Ma et al., 2017). Although VRN (Brock et al., 2016), which combines many networks, outperforms our method in ModelNet classification, their network structure is quite complex, with each network being trained separately and taking many days to complete, making them unsuitable for large datasets. When compared with point cloud-based methods, the proposed method outperforms many of them, including Dominguez et al. (2018), OctNet (Riegler et al., 2017), ECC (Simonovsky and Komodakis, 2017), DGCB-Net (Tian et al., 2020), and VACWGAN-GP (Ergün and Sahillioglu, 2023). The DGFE module helps 3D convolutions hierarchically acquire global information, allowing the network to capture the contextual neighborhood of points. Despite using viewpoints in a predefined sequence, as opposed to any random views by DeepPano (Shi et al., 2015), Gan classifier (Varga et al., 2020), GPSP-DWRN (Long et al., 2021), OrthographicNet (Kasaei, 2019), PANORAMA-NN (Sfikas et al., 2017), and SeqViews2SeqLabels (Han et al., 2019) both of which are multi-view techniques, the method outperforms these approaches, making it suitable for high resolution input. The proposed method also outperforms PolyNet (Yavartanoo et al., 2021), a mesh-based 3D representation network that combined the features in a much smaller dimension using PolyShape's multi-resolution structure.

TABLE 1 Object classification accuracy (%) on ModelNet10.

Method	Input	Acc (%)
VoxNet (Maturana and Scherer, 2015)	Volume	92.0
3DShapeNet (Wu et al., 2015)	Volume	83.5
3DGAN (Wu et al., 2016)	Volume	91.0
VSL (Liu et al., 2018)	Volume	91.0
BV-CNNs (Ma et al., 2017)	Volume	92.3
VRN (Brock et al., 2016)	Volume	97.1
PolyNet (Yavartanoo et al., 2021)	Mesh	94.9
DeepPano (Shi et al., 2015)	Image	85.4
OrthographicNet (Kasaei, 2019)	Image	88.5
PANORAMA-NN (Sfikas et al., 2017)	Image	91.1
SeqViews2SeqLabels (Han et al., 2019)	Image	94.8
Geometry-image (Sinha et al., 2016)	Image	88.4
Gan Classifier (Varga et al., 2020)	Image	89.2
GPSP-DWRN (Long et al., 2021)	Image	92.4
G3DNet (Dominguez et al., 2018)	Point	93.1
OctNet (Riegler et al., 2017)	Point	90.4
ECC (Simonovsky and Komodakis, 2017)	Point	90.0
DGCB-Net (Tian et al., 2020)	Point	94.6
VACWGAN-GP (Ergün and Sahillioglu, 2023)	Point	91.7
(Ours)	Point	**95.6**

The bold values used to differentiate our results from the rest of the other methods.

TABLE 2 Object classification accuracy (%) on ModelNet40.

Method	Input	Acc (%)
VoxNet (Maturana and Scherer, 2015)	Volume	83.0
3DShapeNet (Wu et al., 2015)	Volume	77.0
3DGAN (Wu et al., 2016)	Volume	83.3
VSL (Liu et al., 2018)	Volume	84.5
BV-CNNs (Ma et al., 2017)	Volume	85.4
VRN (Brock et al., 2016)	Volume	95.5
NormalNet (Wang et al., 2019a)	Volume	88.6
DeepNN (Gao et al., 2022)	Mesh	91.0
PolyNet (Yavartanoo et al., 2021)	Mesh	82.8
GIFT (Bai et al., 2016)	Image	83.1
DeepPano (Shi et al., 2015)	Image	77.6
OrthographicNet (Kasaei, 2019)	Image	88.5
SeqViews2SeqLabels (Han et al., 2019)	Image	93.0
Geometry-image (Sinha et al., 2016)	Image	83.9
PointNet (Charles et al., 2017)	Point	89.2
PointConT (Liu et al., 2023)	Points	93.5
RECON (Qi et al., 2023)	Point	93.9
Pointwise (Hua et al., 2018)	Point	86.1
NPCEM (Song et al., 2020)	Point	89.4
ECC (Simonovsky and Komodakis, 2017)	Point	83.2
DGCB-Net (Tian et al., 2020)	Point	92.9
3DCTN (Lu et al., 2022)	Point	91.2
VACWGAN-GP (Ergün and Sahillioglu, 2023)	Point	81.3
(Ours)	Point	**93.1**

The bold values used to differentiate our results from the rest of the other methods.

4.3.2. Classification on ModelNet40

Comparison: We further tested the effectiveness and applicability of the proposed approach using the ModelNet40 dataset. Table 2 compares the classification accuracy of the proposed method to that of alternative scalable 3D representations techniques on the ModelNet40 datasets. As observed, the proposed method performs better than VoxNet (Maturana and Scherer, 2015), 3DGAN (Wu et al., 2016), 3DShapeNets (Wu et al., 2015), NormalNet, VACWGAN-GP (Wang et al., 2019a; Ergün and Sahillioglu, 2023), DPRNet (Arshad et al., 2019), Pointwise (Hua et al., 2018), BV-CNN's (Ma et al., 2017), NPCEM (Song et al., 2020), ECC (Simonovsky and Komodakis, 2017), PointNet (Charles et al., 2017), Geometry image (Sinha et al., 2016), VSL (Liu et al., 2018), GIFT (Bai et al., 2016), FPNN (Li et al., 2016), DGCB-Net (Tian et al., 2020), and DeepNN (Gao et al., 2022) that utilized mesh 3D data. The recent RECON (Qi et al., 2023) and PointConT (Liu et al., 2023) slightly outperformed our technique, which could be attributed to their usage of transformers and pre-train models. The improved fused feature residual network offers a significant advantage over the bulk of voxel and point cloud-based approaches, as shown in Table 2. The proposed method

TABLE 3 ModelNet40 per-class classification comparison between PointNet, Pointwise, DPRNet, and (ours).

Methods	Ours	PointNet	Pointwise	DPRNet
Avg. class	**87.4**	86.2	81.4	81.9
Airplane	100	100	100	100
Bathtub	90.0	80.0	82.0	76.0
Bed	94.0	94.0	93.0	95.0
Bench	80.0	75.0	68.4	80.0
Bookshelf	88.0	93.0	91.8	85.0
Bottle	98.0	94.0	93.9	95.0
Bowl	95.0	100	95.0	95.0
Car	99.0	97.9	95.6	91.0
Chair	97.0	96.0	96.0	97.0
Cone	100	100	80.0	90.0
Cup	90.0	70.0	60.0	70.0
Curtain	85.0	90.0	80.0	80.0
Desk	77.0	79.0	76.7	86.0
Door	92.0	95.0	75.0	85.0
Dresser	74.0	65.1	67.4	60.5
Flowerpot	44.6	30.0	10.0	25.0
Glassbox	91.0	94.0	80.8	86.0
Guiter	99.0	100	98.0	100
Keyboard	100	100	100	100
Lamp	87.0	90.0	83.3	80.0
Laptop	86.0	100	95.0	100
Mental	87.0	96.0	93.9	93.0
Monitor	71.0	95.0	92.9	96.0
Nightstand	65.0	82.6	70.2	70.9
Person	90.0	85.0	89.5	90.0
Piano	91.0	88.8	84.5	83.0
Plant	91.0	73.0	78.8	83.0
Radio	88.0	70.0	65.0	55.0
Range hood	96.0	91.0	88.9	89.9
Sink	85.0	80.0	65.0	70.0
Sofa	93.0	96.0	96.0	93.0
Stairs	90.0	85.0	80.0	75.0
Stool	90.0	90.0	83.3	70.0
Table	98.0	88.0	90.9	77.0
Tent	85.0	95.0	90.0	90.0
Toilet	98.0	99.0	94.9	95.0
TV stand	80.0	87.0	84.5	89.0
Vase	83.0	78.8	81.3	80.0
Wardrobe	65.0	60.0	30.0	20.0
Xbox	90.0	70.0	75.0	80.0

The bold values used to differentiate our results from the rest of the other methods.

performs below VRN (Brock et al., 2016), which makes usage of 24 rotating replicas for training and voting when compared with non-voxel-based approaches. Additionally, the proposed method outperformed PolyNet (Yavartanoo et al., 2021), a mesh-based 3D representation network that integrated the features in a much fewer dimension using PolyShape's multi-resolution structure. It is also worth noting that the improved fused feature residual network proposed already has a high level of accuracy, with a score of above 90%. This may be attributed to the fact that our feature encoding network together with the DGFE module, directly extracts features from the input grid and represents an organized structure of numerous feature representations.

4.3.3. ModelNet40 per-class classification accuracy comparison

Table 3 and Figure 5 compared the per-class accuracies of the proposed method to PointNet (Charles et al., 2017), Pointwise (Hua et al., 2018), and DPRNet (Arshad et al., 2019) on ModelNet40 dataset. As shown in Table 3 and Figure 5, using residual learning and extracting detail features improves per class classification accuracy. The proposed method outperforms PointNet, Pointwise, and DPRNet in key classes such as bathhub, car, bottle dresser, flowerpot, cup, and radio. In terms of average class performance, the method outperformed PointNet (1.2%), Pointwise (6%), and DPRNet (5.5%). Table 3 illustrates it.

4.4. Precision-recall on ModelNet10

Precision is a metric that assesses the accuracy of predictions, i.e., the percentage of correct predictions. It determines how many of the model's predictions were actually right. The precision was computed using Equation 11 as follows:

$$P = \frac{T_P}{T_P + F_P} \tag{11}$$

where T_P is true positive while F_P is false positive (predicted as positive but was incorrect). In the case of recall, it determines how well all of the positives are found which is given as follows:

$$R = \frac{T_P}{T_P + F_N} \tag{12}$$

where F_N is false negatives (unable to predict the presence of an object). The mAP is calculated as the average precision of all classes in the dataset while the F1-score is the harmonic mean of the precision and recall. We used these metrics to assess the efficacy and robustness of the proposed method. We used a grid size of $32 \times 32 \times 32$ and kept the value of K at 8. As shown in Figure 6, the model can learn all 10 object class categories with high precision and recall on the ModelNet10 dataset, with 100% precision on bathtub and chair and 100% recall on bed and toilet. We can also observe that the four classes with the lowest precision and recall (desk, table, nightstand, and dresser) are highly similar which makes them difficult to distinguish even by a human expert. As shown in Figure 6, we observed that the proposed approach successfully generated results with (1) more than 90% precision on the bed, monitor, sofa, table, and toilet and more than 80% on the remaining classes, (2) 90% or higher recall of bathtub, chair, monitor, sofa, and table with more than 80% on the desk, dresser, and nightstand, and (3) 90% or higher F1-score of the bathtub, bed, chair, monitor, sofa, toilet, and table with more than 80% on the desk, dresser, and nightstand. This demonstrates that our model can learn discriminative features from 3D shapes directly across several classes.

To calculate the mAP, we perform several experiments, one of which involved using $16 \times 16 \times 16$ voxel size combined with sampling 8 points per grid. The model was trained using ModelNet10 from scratch, which achieved a 90.2% mAP score. We, then, reduced the learning rate by half (0.5^{-5}) and retrained the model. The effect of fine-tuning improves the mAP to 90.7%. Another experiment was using a $32 \times 32 \times 32$ grid size with the same points per grid. We train the model using the same procedure in the first experiment. We achieved 92.5% with 0.1^{-4} learning rate, and after reducing the learning rate to half and retraining the model, the result improves to 93.3%. With mAP scores of 93.3%, our model surpasses 3DShapeNets (Wu et al., 2015), PANORAMA-ENN (Sfikas et al., 2017), DeepPano (Shi et al., 2015), PolyNet (Yavartanoo et al., 2021), Multimodal (Chen et al., 2021), SeqViews2SeqLabels (Han et al., 2019), Geometry image (Sinha et al., 2016), and GIFT (Bai et al., 2016) on the ModelNet10 dataset, as shown in Table 4. Even while SeqViews2SeqLabels (Han et al., 2019) has the advantage of pre-existing 2D networks that have been pre-trained on big datasets such as ImageNet1K, we achieved a higher mean average precious mAP with 1.9% margin on ModelNet10. To further illustrate the effectiveness of the improved fused feature network, Figure 7 shows the confusion matrix. The confusion matrix was normalized to 100%. We can see that most objects from all classes are recognized correctly.

4.5. Part segmentation on ShapeNetPart

Part segmentation seems to be more difficult than classification tasks and is regarded as every-point classification. Given a triangular mesh or point cloud representation of a 3D object, the purpose of part segmentation is to give each point or triangle face a part category which makes it more challenging than object classification because of the fine-grained and dense predictions. We used the metric procedure from PointNet++ (Qi et al., 2017). For every part shape within the object category, we calculate the union of prediction and ground truth. Figure 8 shows some ShapeNetPart dataset segmentation results from our method. As observed, in most cases, the proposed method results are visually appealing.

Comparison: The segmentation performance of the proposed method is compared with that of various deep learning methods, as shown in Table 5. Although OCNN and RS-Net (Huang et al., 2018) exceed ours in terms of mIoU of all shapes, the improved fused feature residual network outperforms OCNN in specific categories, such as bag, cap, rocket, lamp, and motorbike, and achieves comparable results in the remaining categories. While OCNN has the best IoU, it also uses a conditional dense random field to rectify their network output which serve as a post-processing step, whereas our approach has no similar strategy.

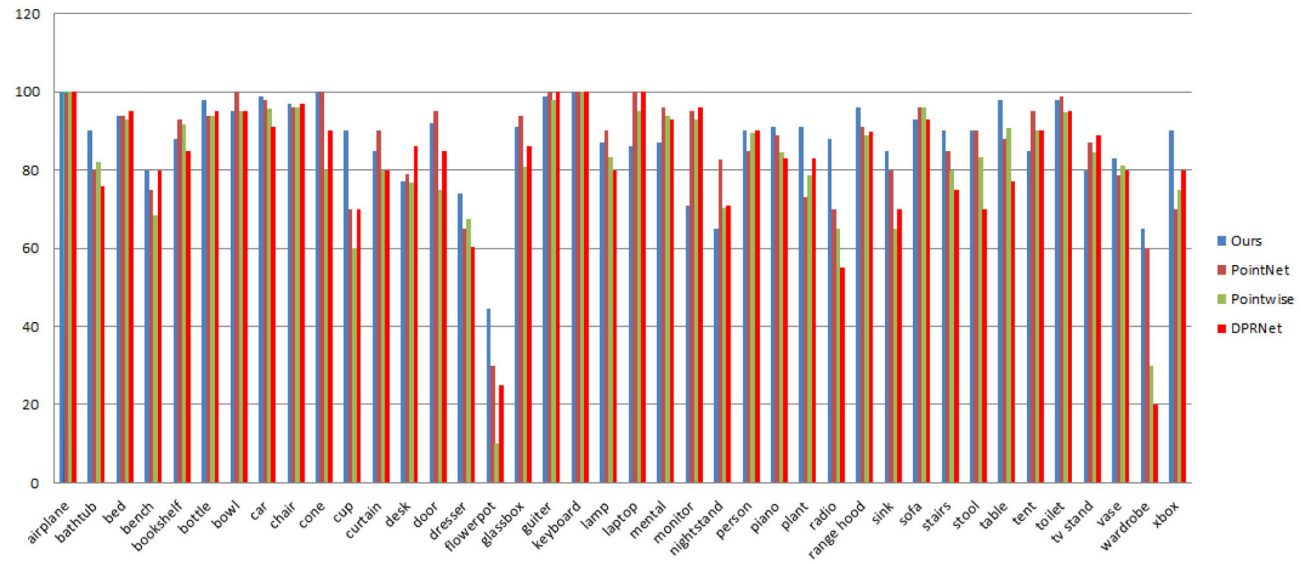

FIGURE 5
ModelNet40 per-class classification accuracy comparisons between PointNet, Pointwise, DPRNet, and (proposed).

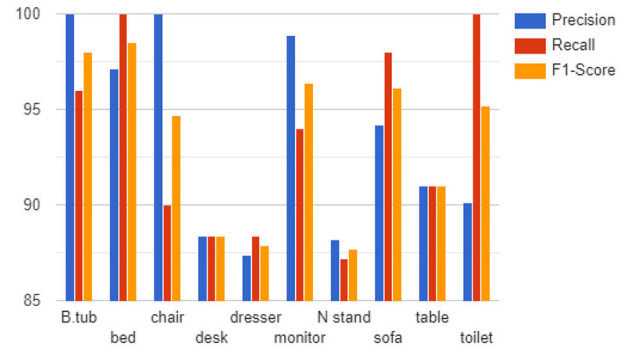

FIGURE 6
Precision, recall, and F1-score on ModelNet10.

TABLE 4 Mean average precision mAP (%) on ModelNet10.

Method	mAP (%)
3DShapeNet (Wu et al., 2015)	68.3
DeepPano (Shi et al., 2015)	84.1
PANORAMA-ENN (Sfikas et al., 2017)	93.2
SeqViews2SeqLabels (Han et al., 2019)	91.4
Geometry-image (Sinha et al., 2016)	88.4
GIFT (Bai et al., 2016)	91.1
PolyNet (Yavartanoo et al., 2021)	84.6
(Ours) ($16 \times 16 \times 16 - grid$)	**90.7**
(Ours) ($32 \times 32 \times 32 - grid$)	**93.3**

The bold values used to differentiate our results from the rest of the other methods.

4.6. Ablation experiments

Here, we conduct some ablation experimental tests to assess various setup modifications and highlight the benefits of the improved fused feature network. The experiments were carried out using the ModelNet10 (Wu et al., 2015) dataset.

4.6.1. Effects of extracted features in the DGFE module

We present an ablation test on ModelNet10 classification to demonstrate the impact of the DGFE module's extracted features. Specifically, we experimented with many variables, including different grid sizes and K values. In the first settings, using a grid size of $16 \times 16 \times 16$ and increasing the value of K from 2 to 8, the classification accuracy increased from 88.1% with K = 2 to 90.5% with K = 8. In the second attempt, we used a grid size of $32 \times 32 \times 32$ and kept the values of K between 2 and 8, and the classification accuracy increased from 90.1% with K = 2 to 91.8% with K = 8. We end up using the later attempt to set the DVFE module in our approach which yields the best model result of 95.6%. Figure 9 displays the results. It shows how the proposed DGFE module encourages correlation among different point cloud regions and is useful for modeling the entire point cloud spatial distribution.

4.6.2. Effects of feature encoding network

This section analyzes the significance of the encoding branch in the proposed approach. After removing the encoding branch, the network is trained using only the DVFE module and KNN search, to sample the local region in each grid cell. We, then, repeated the tests using the same configuration as the previous ablation experiment, with a grid size of 16x16x16 and K = 2. The classification accuracy was 90.1% with K = 2 and 91.1% with K = 8. The classification accuracy improved from 91.3% with K = 2 to 92.4% with K = 8 when utilizing a grid size of $32 \times 32 \times 32$. The results are shown in Figure 9. The model design aids in

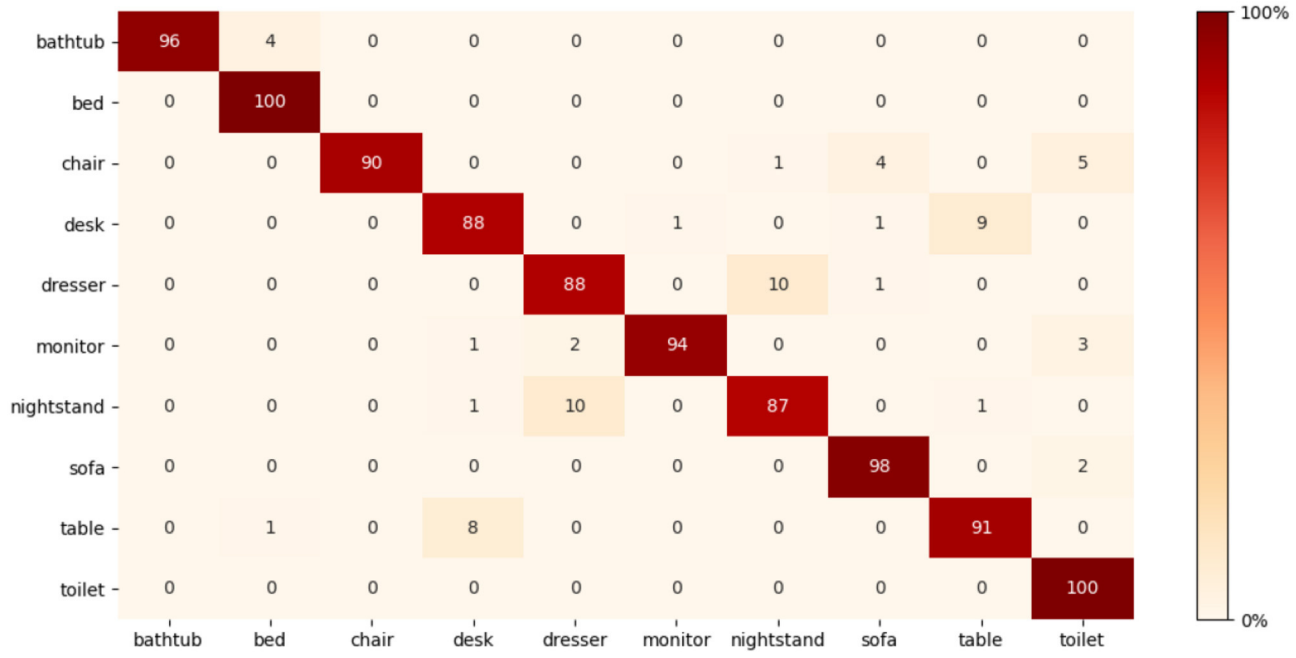

FIGURE 7
Confusion matrix on ModelNet10.

the efficient encoding of features from the input grid and DVFE module. The output features are combined to complement one another. Figure 9 demonstrates the accuracy achieved by inserting the feature encoding network into the whole network, which results in boosting the classification accuracy. The next experiments investigate the sensitivities of the feature encoding units which consist of two units (Feature Encoding Block FEB Unit A and Feature Encoding Block FEB Unit B) with layer skips containing BN and ReLU in between. In each unit, we start with $3 \times 3 \times 3$ convolutions twice, followed by $1 \times 1 \times 1$ convolutions. The main difference between the units is in the application of BN, a regularly used technique to speed up and stabilize the learning process of deep neural networks, and Relu, which has the advantage of allowing complicated correlations in the data to be learned. To test how resilient our approaches are to changes of this type, we swapped the units in different orders. With a $32 \times 32 \times 32$ grid size and $K = 8$, we apply four possible combinations, such as ABAB, BABA, AABB, and BBAA. We train the model from the scratch. As shown in Table 6, the classification accuracy is fairly stable across the different combinations. The combination of ABAB has the highest accuracy and the lowest total log loss, with AABB coming in second. Although the two other combinations, BABA and BBAA, have lower accuracy, their overall performance is generally stable. The above result seems to indicate that, in line with He et al. (2016a), adding BN after addition forces skip connections to perturb the output, which is problematic. The main advantage of applying BN before addition here is that it speeds up training and allows a wider range of learning rates without sacrificing training convergence.

4.6.3. Time complexity

Table 7 compares the average testing time for classification and segmentation with other similar methods. TensorFlow 1.1 is used to record forward time using Nvidia Geforce Titan GTX GPU. The proposed method requires less testing time than many other methods, such as (Leng et al., 2016; Charles et al., 2017; Huang et al., 2018), DGCNN (Wang Y. et al., 2019), SpecGCN (Wang et al., 2018), and 3D-UNet (Cicek et al., 2016), because of its strong data closeness and consistency. Because zeros are padded to empty voxel, the proposed voxelization and sampling approaches both include random memory accesses, which help to decrease unnecessary computation. As observed, using the same voxel resolution of 32^3, the proposed improved fused feature residual network is faster than the 3DCNN (Leng et al., 2016) method and still outperforms it in terms of mIoU, as shown in Table 5. Another advantage of this strategy is that the same number of points is kept in each grid cell while still being able to describe neighborhood information. Now lets analyze the approach to the PointNet++ (Qi et al., 2017), set abstraction module. If we have a batch of 2,048 points with 64-channel characteristics, the technique can model the entire point cloud, but the SA module must aggressively downsample the input, resulting in information loss. The proposed method does not necessitate dynamic kernel computing, which is typically rather expensive. Even though RSNet (Huang et al., 2018) outperformed ours in terms of Mean IoU by 0.7%, the proposed improved fused feature residual network is much faster and requires less memory consumption, as shown in Table 7.

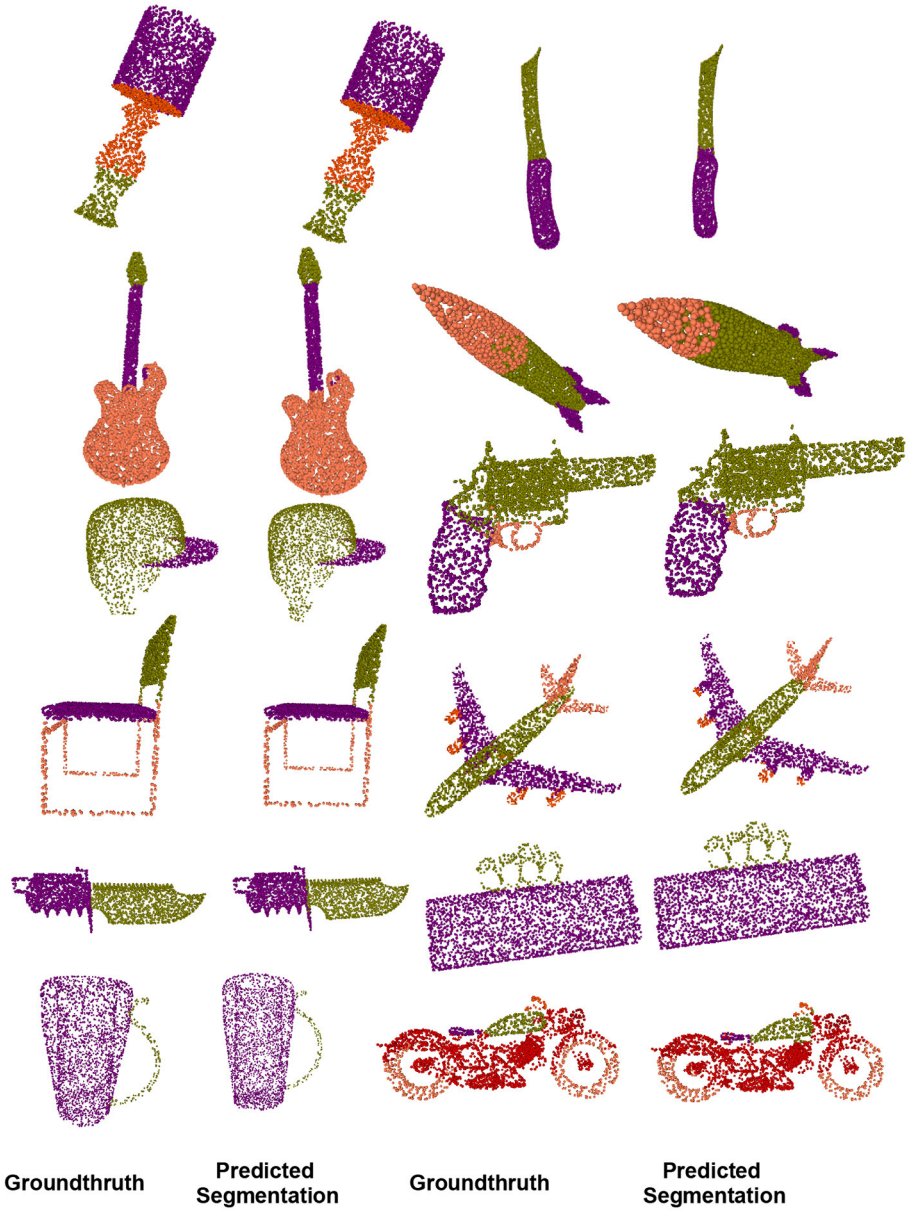

FIGURE 8
On the ShapeNet-part dataset, we compared the visual results of our object part segmentation with groundthruth.

4.6.4. Effects of neighborhood query

In this section, we experiment with ball query and sift query, two other popular neighbor querying methods to sample local areas and experiment with general search radius. For all experiments, we use a $32 \times 32 \times 32$ grid size with a $K = 8$ value on the ModelNet10 dataset. Table 8 shows that KNN is more effective for our strategy. The sift query is the most inefficient method when compared with the KNN and ball query.

5. Conclusion and future work

In this study, we proposed the detail grid feature extraction (DGFE) module which is a highly efficient module. This module assists 3D convolutions in hierarchically capturing global information, reducing the grid size in each spatial dimension and managing overfitting by gradually lowering the spatial dimension of the representation, making it practical for high-resolution 3D objects. Furthermore, we design a feature encoding network that uses two different building blocks with layer skips containing batch normalization and non-linearity ReLU in between, resulting in fewer layers in the early training phase which helps speed learning and reduces the effect of gradients vanishing since there are few layers through which to propagate. The outputs of the two modules are fused in the feature fusion unit to produce a feature with improved contextual representation by utilizing both local and global shape structures. We built a network called improved fused feature

TABLE 5 Segmentation results of different methods on ShapeNet-part dataset (Yi et al., 2016).

Methods	(Ours)	P.Net	ShapeNet	KD-Net	MRTNet	3DCNN	RS-Net	O-CNN
mIoU	**84.2**	83.7	81.4	77.2	83.0	79.4	84.9	85.9
Airplane	83.8	83.4	81	79.9	81.0	75.1	82.7	85.5
Bag	88.9	78.7	78.4	71.2	76.7	72.8	86.4	87.1
Cap	91.9	82.5	77.7	80.9	87.0	73.3	84.1	84.7
Car	72	74.9	75.7	68.8	73.8	70.0	78.2	77.0
Chair	88	89.6	87.6	88.0	89.1	87.2	90.4	91.1
Earphone	47.0	73.0	61.9	72.4	67.6	63.5	69.3	85.1
Guitar	86.8	91.5	92	88.9	90.6	88.4	91.4	91.9
Knife	86.7	85.9	85.4	86.4	85.4	79.6	87.0	87.4
Lamp	89.8	80.8	82.5	79.8	80.6	74.4	83.5	83.3
Laptop	60.8	95.3	95.7	94.9	95.1	93.9	95.4	95.4
Motorbike	93.7	65.2	70.6	55.8	64.4	58.7	66.0	56.9
Mug	94.4	93.0	91.9	86.5	91.8	91.8	92.6	96.2
Pistol	80	81.2	85.9	79.3	79.7	76.4	81.8	81.6
Rocket	86.1	57.9	53.1	50.4	57.0	51.2	56.1	53.5
Skateboard	70.1	72.8	69.8	71.1	69.1	65.3	75.8	74.1
Table	74.1	80.6	75.3	80.2	80.6	77.1	82.2	84.4

The bold values used to differentiate our results from the rest of the other methods.

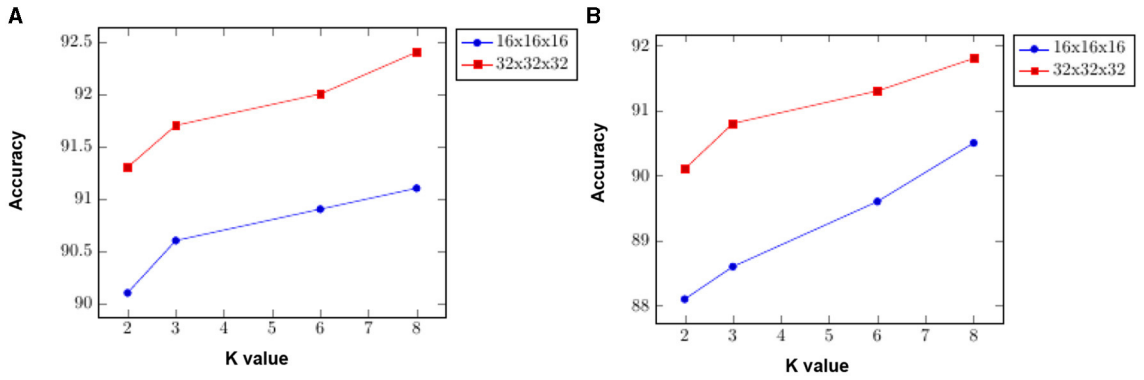

FIGURE 9
To highlight the influence of both the DGFE module and the feature encoding network, a ModelNet10 classification ablation test is presented. We experimented with some variables including different grid sizes and K values. **(A)** Shows how the feature encoding network performs with 16 × 16 × 16 and 32 × 32 × 32 grid sizes and different values of K; **(B)** demonstrates the performance of the DGFE module's effects of extracted features on 16 × 16 × 16 and 32 × 32 × 32 grid sizes with 2, 3, 6, and 8 K values.

residual network using the modules that have been proposed, which achieve a notable balance of accuracy and speed. In both ModelNet10 and ModelNet40 datasets, the proposed improved fused feature residual network offers a significant advantage over the bulk of voxel and point cloud-based approaches, as shown in Tables 1, 2. Due to its scalability and efficiency, the proposed method can be used in extracting large-scale features of high-resolution inputs.

Although our method performs well with normal datasets, we note that when noise is added to the datasets, the performance drops, for example, when Gaussian noise is added to the 3D models, the performance decreases despite applying different parameters. In future, instead of directly sampling points, we will use sparse convolutions to convert them to a small number of voxels and sample non-empty voxels to ensure that precise point positions are retained.

In addition, numerous mechanisms for attention employed in transformer approaches are adaptable and offer a high potential for future advances. We think cutting-edge outcomes can be attained by extending generic point cloud processing innovation

TABLE 6 Different combinations of feature encoding units on ModelNet10.

FEB unit	Acc (%)	Logloss
ABAB	95.6	2.22
BABA	93.8	2.38
AABB	94.54	2.25
BBAA	93.94	2.32

TABLE 7 Average testing time of our method with others on ModelNet40.

Method	Classification (ms)	Segmentation (ms)
PointNet++ (Qi et al., 2017)	163	-
3DCNN (Leng et al., 2016)	49	137
SpecGCN (Wang et al., 2018)	11254	-
DGCNN (Wang Y. et al., 2019)	52	87.8
3D-UNet (Cicek et al., 2016)	-	682.1
RSNet (Huang et al., 2018)	-	74.6
(Ours)	**28**	**19**

The bold values used to differentiate our results from the rest of the other methods.

TABLE 8 Effects of neighborhood query on ModelNet10 classification.

Sift query		Ball query		KNN
$r = 0.1$	$r = 0.2$	$r = 0.1$	$r = 0.2$	
90.8%	91.0%	92.6%	93.0%	95.6%

to transformer techniques. For instance, one possible option we are looking at is by swapping out the feature extraction module in our network design for one that is transformer/attention-based. Instead of just depending on transformers to extract features, we can conduct local feature extraction using non-transformer-based approaches and then couple it with a transformer for global feature interaction which will lead to the extraction of more fine grain features.

Data availability statement

The raw data supporting the conclusions of this article will be made available by the authors, without undue reservation.

Author contributions

AG: conceptualization of this study, methodology, writing—original draft preparation, and software. CL: conceptualization, software, supervision, resources, project administration, and funding acquisition. HJ, YN, and MA: data curation, writing—reviewing and editing, and software. HC: data curation, software, and supervision. All authors contributed to the article and approved the submitted version.

Funding

This study was supported by the Fujian Province University Key Lab for the Analysis and Application of Industry Big Data, Fujian Key Lab of Agriculture IOT Application, and IOT Application Engineering Research Center of Fujian Province Colleges and Universities.

Conflict of interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's note

All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.

References

Arshad, S., Shahzad, M., Riaz, Q., and Fraz, M. (2019). DPRNet: deep 3D point based residual network for semantic segmentation and classification of 3D point clouds. *IEEE Access* 7, 68892–68904. doi: 10.1109/ACCESS.2019.2918862

Atzmon, M., Maron, H., and Lipman, Y. (2018). Point convolutional neural networks by extension operators. *ACM Trans. Graph.* 37, 1–12. doi: 10.1145/3197517.3201301

Bai, S., Bai, X., Zhou, Z., Zhang, Z., and Latecki, L. J. (2016)."GIFT: A real-time and scalable 3D shape search engine," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (Las Vegas, NV: IEEE), 5023–5032. doi: 10.1109/CVPR.2016.543

Bello, S. A., Wang, C., Wambugu, N. M., and Adam, J. M. (2021). FFpointNet: local and global fused feature for 3D point clouds analysis. *Neurocomputing* 461, 55–62. doi: 10.1016/j.neucom.2021.07.044

Bello, Saifullahi, A., Yu, S., Wang, C., Adam, Jibril, M., and Li, J. (2020). Review: deep learning on 3D point clouds. *Remot. Sens.* 12, 11. doi: 10.3390/rs12111729

Brock, A., Lim, T., Ritchie, J., and Weston, N. (2016). Generative and discriminative voxel modeling with convolutional neural networks. *ArXiv*. doi: 10.48550/arXiv.1608.04236

Charles, R., Su, H., Kaichun, M., and Guibas, L. (2017). "PointNet: Deep learning on point sets for 3D classification and segmentation," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (Honolulu, HI: IEEE), 77–85. doi: 10.1109/CVPR.2017.16

Chen, Z., Jing, L., Liang, Y., Tian, Y., and Li, B. (2021). Multimodal semi-supervised learning for 3D objects. *ArXiv*. doi: 10.48550/arXiv.2110.11601

Chiotellis, I., Triebel, R., Windheuser, T., and Cremers, D. (2016). "Non-rigid 3D shape retrieval via large margin nearest neighbor embedding," in *European Conference on Computer Vision (ECCV)* (Amsterdam). doi: 10.1007/978-3-319-46475-6_21

Choy, C., Danfei, X., JunYoung, G., Kevin, C., and Savarese, S. (2016). "3D-R2N2: A unified approach for single and multi-view 3D object reconstruction," in *European Conference on Computer Vision (ECCV)* (Amsterdam).

Cicek, Ö., Abdulkadir, A., Lienkamp, S. S., Brox, T., and Ronneberger, O. (2016). 3D U-Net: Learning dense volumetric segmentation from sparse annotation. *ArXiv*. doi: 10.48550/arXiv.1606.06650

Dominguez, M., Dhamdhere, R., Petkar, A., Jain, S., Sah, S., and Ptucha, R. (2018). "General-purpose deep point cloud feature extractor," in *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)* (Lake Tahoe, NV: IEEE), 1972–1981. doi: 10.1109/WACV.2018.00218

Eldar, Y., Lindenbaum, M., Porat, M., and Zeevi, Y. (1997). The farthest point strategy for progressive image sampling. *IEEE Trans. Image Process.* 9, 1305–1315. doi: 10.1109/83.623193

Elhassan, M. A., Huang, C., Yang, C., and Munea, T. L. (2021). DSANet: dilated spatial attention for real-time semantic segmentation in urban street scenes. *Expert Syst. Appl.* 183, 115090. doi: 10.1016/j.eswa.2021.115090

Ergün, O., and Sahillioglu, Y. (2023). 3D point cloud classification with ACGAN-3D and VACWGAN-GP. *Turk. J. Electr. Eng. Comput. Sci.* 31, 381–395. doi: 10.55730/1300-0632.3990

Gao, M., Ruan, N., Shi, J., and Zhou, W. (2022). Deep neural network for 3D shape classification based on mesh feature. *Sensors* 22, 187040. doi: 10.3390/s22187040

Gezawa, A. S., Bello, Z. A., Wang, Q., and Yunqi, L. (2021). A voxelized point clouds representation for object classification and segmentation on 3D data. *J. Supercomput.* 21, 1–22. doi: 10.1007/s11227-021-03899-x

Gezawa, Sulaiman, A., Zhang, Y., Wang, Q., and Lei, Y. (2020). A review on deep learning approaches for 3D data representations in retrieval and classifications. *IEEE Access* 8, 57566–57593. doi: 10.1109/ACCESS.2020.2982196

Han, Z., Shang, M., Liu, Z., Vong, C.-M., Liu, Y.-S., Zwicker, M., et al. (2019). SeqViews2SeqLabels: learning 3D global features via aggregating sequential views by RNN with attention. *IEEE Trans. Image Process.* 28, 658–672. doi: 10.1109/TIP.2018.2868426

He, K., Zhang, X., Ren, S., and Sun, J. (2016a). "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (Las Vegas, NV: IEEE), 770–778. doi: 10.1109/CVPR.2016.90

He, K., Zhang, X., Ren, S., and Sun, J. (2016b). Identity mappings in deep residual networks. *ArXiv*. doi: 10.48550/arXiv.1603.05027

Hua, B.-S., Tran, M.-K., and Yueng, S.-K. (2018). "Pointwise convolutional neural networks," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (Salt Lake City, UT: IEEE), 984–993. doi: 10.1109/CVPR.2018.00109

Huang, F., Xu, C., Tu, X., and Li, S. (2019). Weight loss for point clouds classification. *J. Phys.* 1229, e012045. doi: 10.1088/1742-6596/1229/1/012045

Huang, G., Liu, Z., van der Maaten, L., and Weinberger, K. Q. (2017). "Densely connected convolutional networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (Honolulu, HI: IEEE), 2261–2269. doi: 10.1109/CVPR.2017.243

Huang, Q., Wang, W., and Neumann, U. (2018). "Recurrent slice networks for 3D segmentation of point clouds," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (Salt Lake City, UT: IEEE), 2626–2635. doi: 10.1109/CVPR.2018.00278

Ioffe, S., and Szegedy, C. (2015). Batch normalization: accelerating deep network training by reducing internal covariate shift. *ArXiv*. doi: 10.48550/arXiv.1502.03167

Kasaei, H. (2019). OrthographicNet: a deep learning approach for 3d object recognition in open-ended domains. *ArXiv*. doi: 10.48550/arXiv.1902.03057

Kingma, D. P., and Ba, J. (2015). Adam: a method for stochastic optimization. *CoRR*. doi: 10.48550/arXiv.1412.6980

Klokov, R., and Lempitsky, V. (2017). "Escape from cells: Deep Kd-networks for the recognition of 3D point cloud models," in *2017 IEEE International Conference on Computer Vision (ICCV)* (Venice: IEEE), 863–872. doi: 10.1109/ICCV.2017.99

Kohonen, T. (1998). The self-organizing map. *Neurocomputing* 21, 1–6. doi: 10.1016/S0925-2312(98)00030-7

Kuangen, Z., Ming, H., Wang, J., de Silva, C. W., and Fu, C. (2019). Linked dynamic graph CNN: learning on point cloud via linking hierarchical features. *ArXiv*. doi: 10.48550/arXiv.1904.10014

Landrieu, L., and Simonovsky, M. (2018). "Large-scale point cloud semantic segmentation with superpoint graphs," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (Salt Lake City, UT: IEEE), 4558–4567. doi: 10.1109/CVPR.2018.00479

Le, T., and Duan, Y. (2018). "PointGrid: A deep network for 3D shape understanding," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (Salt Lake City, UT: IEEE), 9204–9214. doi: 10.1109/CVPR.2018.00959

Leng, B., Liu, Y., Yu, K., Zhang, X., and Xiong, Z. (2016). 3D object understanding with 3D convolutional neural networks. *Inf. Sci.* 366, 188–201. doi: 10.1016/j.ins.2015.08.007

Li, G., Muller, M., Qian, G., Delgadillo, I. C., Abualshour, A., Thabet, A., et al. (2023). DeepGCNs: making GCNs go as deep as CNNs. *IEEE Trans. Pattern Anal. Mach. Intell.* 45, 6923–6939. doi: 10.1109/TPAMI.2021.3074057

Li, J., Chen, B. M., and Lee, G. H. (2018). "SO-Net: Self-organizing network for point cloud analysis," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (Salt Lake City, UT: IEEE), 9397–9406. doi: 10.1109/CVPR.2018.00979

Li, Y., Bu, R., Sun, M., Wu, W., Di, X., and Chen, B. (2018). "PointCNN: convolution on x-transformed points," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems (NIPS'18)* (Red Hook, NY: Curran Associates Inc), 828–838.

Li, Y., Pirk, S., Su, H., Qi, C., and Guibas, L. (2016). FPNN: field probing neural networks for 3D data. *ArXiv*. doi: 10.48550/arXiv.1605.06240

Liu, S., Giles, L., and Ororbia, A. (2018). "Learning a hierarchical latent-variable model of 3D shapes," in *2018 International Conference on 3D Vision* (Verona), 542–551. doi: 10.1109/3DV.2018.00068

Liu, Y., Wang, B., Lv, Y., Li, L., and Wang, F. (2023). Point cloud classification using content-based transformer via clustering in feature space. *ArXiv*. doi: 10.48550/arXiv.2303.04599

Long, H., Lee, S.-H., and Kwon, K.-R. (2021). A deep learning method for 3D object classification and retrieval using the global point signature plus and deep wide residual network. *Sensors* 21, 82644. doi: 10.3390/s21082644

Lu, D., Xie, Q., Gao, K., Xu, L., and Li, J. (2022). 3DCTN: 3D convolution-transformer network for point cloud classification. *IEEE Trans. Intell. Transport. Syst.* 23, 24854–24865. doi: 10.1109/TITS.2022.3198836

Ma, C., An, W., Lei, Y., and Guo, Y. (2017). "BV-CNNS: binary volumetric convolutional networks for 3D object recognition," in *British Machine Vision Conference 2017, BMVC 2017* (London: BMVA Press).

Maturana, D., and Scherer, S. (2015). "VoxNet: A 3D Convolutional Neural Network for real-time object recognition," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (Hamburg: IEEE), 922–928. doi: 10.1109/IROS.2015.7353481

Nair, V., and Hinton, G. E. (2010). "Rectified linear units improve restricted Boltzmann machines," in *Proceedings of the 27th International Conference on International Conference on Machine Learning* (Madison, WI: Omnipress), 807–814.

Qi, C., Yi, L., Hao, S., and Guibas, L. (2017). "Pointnet^{++}: deep hierarchical feature learning on point sets in a metric space," in *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17)* (Red Hook, NY: Curran Associates Inc), 5105–5114.

Qi, Z., Dong, R., Fan, G., Ge, Z., Zhang, X., Ma, K., et al. (2023). Contrast with reconstruct: contrastive 3D representation learning guided by generative pretraining. *ArXiv*. doi: 10.48550/arXiv.2302.02318

Qiangeng, X., Weiyue, W., Duygu, C., Mech, R., and Neumann, U. (2019). "DISN: deep implicit surface network for high-quality single-view 3D reconstruction," in *Proceedings of the 33rd International Conference on Neural Information Processing Systems* (Red Hook, NY: Curran Associates Inc), 492–502.

Riegler, G., Ulusoy, A. O., and Geiger, A. (2017). "OctNet: Learning deep 3D representations at high resolutions," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (Honolulu, HI: IEEE), 6620–6629. doi: 10.1109/CVPR.2017.701

Sfikas, K., Theoharis, T., Pratikakis, I. (2017). "Exploiting the PANORAMA representation for convolutional neural network classification and retrieval," in *Proceedings of the Workshop on 3D Object Retrieval (3Dor '17)* (Goslar: Eurographics Association), 1-7. doi: 10.2312/3dor.20171045

Shi, B., Bai, S., Zhou, Z., and Bai, X. (2015). DeepPano: deep panoramic representation for 3-D shape recognition. *IEEE Sign. Process. Lett.* 22, 2339–2343. doi: 10.1109/LSP.2015.2480802

Simonovsky, M., and Komodakis, N. (2017). "Dynamic edge-conditioned filters in convolutional neural networks on graphs," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (Honolulu, HI: IEEE), 29–38. doi: 10.1109/CVPR.2017.11

Sinha, A., Bai, J., and Ramani, K. (2016). "Deep learning 3D shape surfaces using geometry images," in *European Conference on Computer Vision (ECCV)* (Amsterdam).

Song, Y., Gao, L., Li, X., and Shen, W. (2020). A novel point cloud encoding method based on local information for 3D classification and segmentation. *Sensors* 20, 92501. doi: 10.3390/s20092501

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., et al. (2015). "Going deeper with convolutions," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (Boston, MA: IEEE), 1–9. doi: 10.1109/CVPR.2015.7298594

Thomas, H., Qi, C. R., Deschaud, J.-E., Marcotegui, B., Goulette, F., and Guibas, L. (2019). "KPConv: Flexible and deformable convolution for point clouds," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)* (Seoul: IEEE), 6410–6419. doi: 10.1109/ICCV.2019.00651

Tian, Y., Chen, L., Song, W., Sung, Y., and Woo, S. (2020). DGCB-Net: dynamic graph convolutional broad network for 3D object recognition in point cloud. *Remote. Sens.* 13, 66. doi: 10.3390/rs13010066

Varga, M., Jadlovský, J., and Jadlovska, S. (2020). Generative enhancement of 3D image classifiers. *Appl. Sci.* 2020, 10217433. doi: 10.3390/app10217433

Wang, C., Cheng, M., Sohel, F., Bennamoun, M., and Li, J. (2019a). NormalNet: a voxel-based CNN for 3D object classification and retrieval. *Neurocomputing* 323, 139–147. doi: 10.1016/j.neucom.2018.09.075

Wang, C., Samari, B., and Siddiqi, K. (2018). "Local spectral graph convolution for point set feature learning," in *Computer Vision – ECCV 2018: 15th European Conference, Munich, Germany, September 8–14, 2018, Proceedings, Part IV* (Berlin; Heidelberg: Springer-Verlag), 56–71. doi: 10.1007/978-3-030-01225-0_4

Wang, D. Z., and Posner, I. (2015). "Voting for voting in online point cloud object detection," in *Robotics: Science and Systems* (Rome), 10–15607.

Wang, H., Shi, C., Shi, S., Lei, M., Wang, S., He, D., et al. (2023). DSVT: dynamic sparse voxel transformer with rotated sets. *ArXiv*. doi: 10.48550/arXiv.2301.06051

Wang, L., Huang, Y., Hou, Y., Zhang, S., and Shan, J. (2019). "Graph attention convolution for point cloud semantic segmentation," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (Long Beach, CA: IEEE), 10288–10297. doi: 10.1109/CVPR.2019.01054

Wang, Y., Sun, Y., Liu, Z., Sarma, S. E., Bronstein, M. M., and Solomon, J. M. (2019). Dynamic graph cnn for learning on point clouds. *ACM Trans. Graph.* 38, 1–12. doi: 10.1145/3326362

Wei, X., Yu, R., and Sun, J. (2020). "View-GCN: View-based graph convolutional network for 3D shape analysis," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (Seattle, WA: IEEE), 1847–1856. doi: 10.1109/CVPR42600.2020.00192

Wu, J., Zhang, C., Xue, T., Freeman, B., and Tenenbaum, J. (2016). "Learning a probabilistic latent space of object shapes via 3D generative-adversarial modeling," in *Proceedings of the 30th International Conference on Neural Information Processing Systems (NIPS'16)* (Red Hook, NY: Curran Associates Inc), 82–90.

Wu, W., Qi, Z., and Fuxin, L. (2019). "PointConv: Deep convolutional networks on 3D point clouds," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (Long Beach, CA: IEEE), 9613–9622. doi: 10.1109/CVPR.2019.00985

Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., et al. (2015). "3D ShapeNets: A deep representation for volumetric shapes," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (Boston, MA: IEEE), 1912–1920. doi: 10.1109/CVPR.2015.7298801

Yang, H., Wang, W., Chen, M., Lin, B., He, T., Chen, H., et al. (2023). PVT-SSD: single-stage 3d object detector with point-voxel transformer. *ArXiv*. doi: 10.48550/arXiv.2305.06621

Yang, J., Zhang, Q., Ni, B., Li, L., Liu, J., Zhou, M., et al. (2019). "Modeling point clouds with self-attention and gumbel subset sampling," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (Long Beach, CA: IEEE), 3318–3327, doi: 10.1109/CVPR.2019.00344

Yavartanoo, M., Hung, S.-H., Neshatavar, R., Zhang, Y., and Lee, K. M. (2021). "PolyNet: Polynomial neural network for 3D shape recognition with polyshape representation," in *2021 International Conference on 3D Vision (3DV)* (London), 1014–1023, doi: 10.1109/3DV53792.2021.00109

Yi, L., Kim, V. G., Ceylan, D., Shen, I.-C., Yan, M., Su, H., et al. (2016). A scalable active framework for region annotation in 3D shape collections. *ACM Trans. Graph.* 35, 1–12. doi: 10.1145/2980179.2980238

Yifan, X., Tianqi, F., Mingye, X., Long, Z., and Qiao, Y. (2018). "SpiderCNN: deep learning on point sets with parameterized convolutional filters," in *European Conference on Computer Vision (ECCV)* (Munich).

Zhijian, L., Haotian, T., Yujun, L., and Song, H. (2019). "Point-voxel CNN for efficient 3D deep learning," in *Proceedings of the 33rd International Conference on Neural Information Processing Systems* (Red Hook, NY: Curran Associates Inc), 965–975.

Zhou, Y., and Tuzel, O. (2018). "VoxelNet: End-to-end learning for point cloud based 3D object detection," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (Salt Lake City, UT: IEEE), 4490–4499. doi: 10.1109/CVPR.2018.00472