PointLGGS: Parallel Local-Global Dual-Path for Efficient Point Cloud Classification

Min Yuan[1], Jiale Li[1], Qiuhai Zhao[1,4], Lei Pan[2], and Hongyong Leng[1,3,4(✉)]

[1] School of Software, Xinjiang University, Ürümqi 830091, Xinjiang, China
[2] Norweststar Information Technology Co., Ltd., Ürümqi 830000, Xinjiang, China
[3] School of Computer Science, Beijing Institute of Technology, Beijing 100081, China
[4] Xinjiang Engineering Research Center of Big Data and Intelligent Software, Ürümqi 830091, Xinjiang, China
leng@xju.edu.cn

Abstract. Despite significant advances in point cloud-based 3D vision research, existing methods often fail to achieve efficient collaborative modeling of both fine-grained local geometry and long-range global semantics, which limits further gains in classification accuracy. To address this limitation, this paper proposes a lightweight point cloud classification network named PointLGGS. Its three core contributions are summarized as follows. First, a Local Geometric Feature Extractor (LGFE) is designed to accurately capture fine-grained structures within local point cloud patches through relative position encoding and max-pooling operations. Second, a Global Semantic Feature Extractor (GSFE) is constructed by integrating dual mechanisms: Patch-Point Self-Attention (PP-SA) and Channel Self-Attention (CSA). This design jointly models long-range dependencies between patches and contextual correlations across channel dimensions, enabling efficient global semantic extraction. Third, using the LG-GS block as a fundamental building unit, a dual-path parallel architecture is employed to couple the LGFE and GSFE. Deep integration of local and global features is achieved via adaptive fusion, resulting in robust point cloud representations. Experimental results show that PointLGGS achieves classification accuracy of 94.0% and 90.7% on the ModelNet40 and ScanObjectNN datasets, respectively, with only 4.6 million parameters, thereby striking an excellent balance between classification accuracy and model efficiency.

Keywords: point cloud classification, lightweight network, dual-path architecture, local-global feature fusion, attention mechanism

1 Introduction

The rapid advancement of 3D sensor technologies has established point cloud data as a fundamental format for representing the spatial structure of 3D objects. It finds increasingly wide application in diverse fields such as autonomous driving [1–3], robotics [4–6], augmented/virtual reality (AR/VR), and industrial inspection. Compared to traditional 2D images, point clouds provide richer spatial and geometric

information, thereby offering a more comprehensive perspective for understanding and analyzing complex scenes. However, the inherent disorder, unstructured nature, and sparsity of point clouds pose a significant challenge in computer vision: the effective extraction of discriminative features.

Existing point cloud classification methods are primarily categorized into two types. One is represented by PointNet++[7], which focuses on extracting local geometric features to identify structural details of objects. The other is exemplified by DGCNN[8], which emphasizes modeling global semantic features to enhance classification performance by capturing the overall shape and long-range contextual dependencies via dynamic graph convolution. However, these two approaches often fail to achieve collaborative modeling of both local details and global context. Overemphasizing local features can result in an inadequate understanding of the overall structure, whereas excessive focus on global features may cause critical local geometric details to be overlooked. This imbalance between local and global feature extraction severely limits model performance in complex scenes and fine-grained classification tasks. In recent years, Transformer[9] and self-attention mechanisms, such as those in PointTransformer[10], have demonstrated significant potential in point cloud processing by effectively capturing long-range global dependencies. Nevertheless, these methods still exhibit insufficient capability in modeling local geometric details, which typically leads to degraded perception of local structures.

To tackle the aforementioned challenges, this paper proposes a lightweight point cloud classification network, **PointLGGS**, which simultaneously achieves fine-grained geometric mining within local point cloud patches and long-range global semantic modeling across patches. This design effectively balances model accuracy and computational efficiency. The network employs the **LG-GS block** as its fundamental building unit and adopts a dual-path parallel architecture. This architecture decouples the extraction of inter-patch local geometric features and inter-patch global semantic features into separate processing streams. Specifically, the local pathway utilizes the **Local Geometric Feature Extractor(LGFE)** module to capture fine-grained geometric details within patches accurately, leveraging relative position encoding and max-pooling operations. The global pathway incorporates the **Global Semantic Feature Extractor(GSFE)** module. This module integrates the dual attention mechanisms of Patch-Point Self-Attention (PP-SA) and Channel Self-Attention (CSA) to efficiently model long-range dependencies between patches and contextual correlations along the channel dimension. Finally, the model constructs robust, globally-aware point cloud representations through the adaptive fusion of features from these two pathways.

The main contributions of this paper are summarized as follows:

• We propose the PointLGGS lightweight point cloud classification network. Centered around the LG-GS block, it adopts a dual-path parallel architecture to separately extract inter-patch local geometric features and inter-patch global semantic features. Deep collaboration between the two is achieved via adaptive fusion, ensuring classification accuracy while maintaining a lightweight model.

• We design a Local Geometric Feature Extractor (LGFE). Based on relative position encoding and max-pooling operations, it accurately captures fine-grained geometric structures within point cloud patches.

• We construct a Global Semantic Feature Extractor (GSFE). By fusing the dual attention mechanisms of PP-SA and CSA, it efficiently models long-range dependencies between patches and feature correlations across channels, effectively enhancing global context representation capabilities.

2 Related Work

Point cloud classification is a fundamental task in 3D vision, with its core challenge lying in handling the inherent disorder and unstructured nature of point clouds to learn discriminative feature representations representations. Based on differences in data processing paradigms, existing methods can be roughly categorized into two main types: multi-view/voxelization methods and direct point-based methods.

2.1 Multi-View and Voxelization Methods

To circumvent the irregular nature of point clouds, early methods typically converted them into regular representations. Multi-view methods project 3D point clouds into 2D images from different viewpoints, then leverage mature 2DCNNs for feature extraction. The representative work MVCNN[11] aggregates features through multi-view projection and view pooling. While such methods can reuse the advantages of image networks, the projection process leads to loss of geometric information, making it difficult to capture fine local structures. Moreover, increasing the number of views significantly increases computational cost. Subsequent works like ViewGCN[12] used Graph Neural Networks to model interview relationships to enhance global representation, but still did not solve the problems of local detail loss and high computational cost.

Voxelization methods quantize point clouds into 3D regular grids, then employ 3D CNNs for feature learning. VoxNet[13] is the pioneering work in this line. While voxelization structures point clouds, it faces a sharp trade-off between resolution and resource consumption. Low resolution loses geometric details, while high resolution incurs huge storage and computational burdens. To mitigate this, works like OctNet[14] employed octrees for adaptive grid partitioning. However, the inherent error introduced by voxel quantization remains difficult to avoid, limiting their performance in fine-grained geometric perception tasks. Both types of methods struggle to achieve an effective balance between geometric information retention, computational efficiency, and storage overhead.

2.2 Point-Based Methods

Operating directly on raw point clouds while preserving complete geometric information, point-based methods have become the mainstream research paradigm. PointNet [15] first utilized shared MLPs and symmetric functions to process point sets directly, but it lacked explicit modeling of local structures. Building on this, PointNet++ [7] achieved local geometric feature extraction through hierarchical sampling and neighborhood grouping, while DGCNN [8] proposed dynamic graph convolution (EdgeConv) to dynamically construct topology in feature space and

capture local correlations. These methods significantly improved local feature modeling capabilities, but still exhibited deficiencies in capturing global context and achieving local-global feature synergy. Subsequent works like PAConv [16], SpiderCNN [17], and PointMLP [18] optimized local feature learning from the perspectives of convolution paradigms and geometric module design, yet their ability to model long-range dependencies in point clouds remained relatively limited.

In recent years, thanks to its powerful global modeling capabilities, the Transformer architecture [9] has been widely introduced into point cloud processing. Works like PCT [19] and PointTransformer [10] directly applied self-

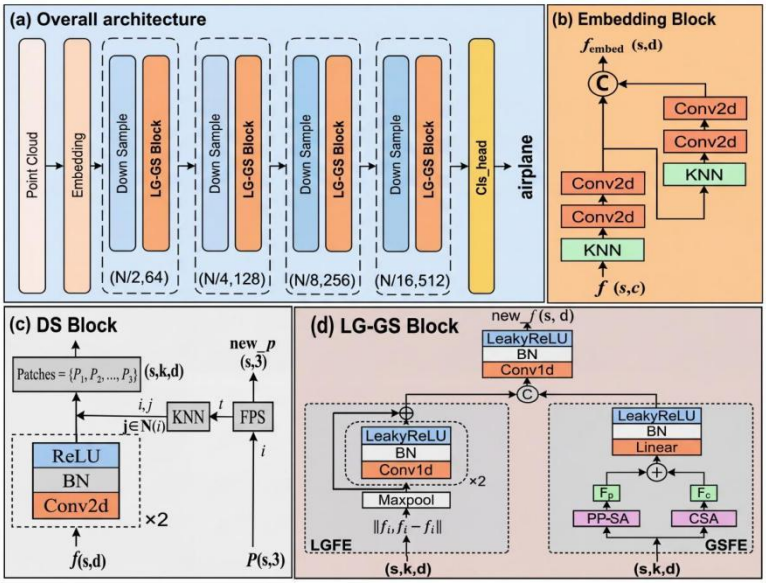

Fig. 1. PointLGGS Framework and Components. (a) Overall architecture. (b) Embedding block. (c) Down-sampling module (DS Block). (d) Local-Global feature fusion block (LG-GS Block)

Attention mechanisms to points or point clusters, effectively modeling global semantic dependencies. However, pure Transformer models are often computationally expensive and exhibit relatively weak perception of fine-grained local geometric structures. To address this, hybrid architectures like PointTransformerv2[20] and 3DCTN [21] attempted to fuse local convolution and global attention, aiming to capture both local details and global context simultaneously.

Despite the many advances in existing research, how to achieve efficient and deep synergistic fusion of **local geometric details** and **global semantic context** under a **lightweight network architecture** remains a key challenge in the field of point cloud classification. To address this core challenge, this paper proposes the PointLGGS network, which constructs a parallel dual-path architecture to separately extract local geometric and global semantic features. Combined with efficient dual attention mechanisms and feature fusion strategies, it achieves a better balance between classification accuracy and model efficiency.

3 Method

This section introduces the overall architecture of the PointLGGS network and its core unit—the Local-Global feature fusion block (LG-GS). This block is composed of the Local Geometric Feature Extractor (LGFE) and the Global Semantic Feature Extractor (GSFE) in a dual-path collaborative manner.

3.1 PointLGGS Framework

The PointLGGS network adopts a hierarchical architecture aimed at efficiently and collaboratively modeling the local geometric details and global semantic context of point clouds through progressive feature extraction and fusion. Its overall pipeline follows the paradigm of "Embedding → Multilevel Down-sampling and Feature Fusion → Classification", gradually reducing point cloud resolution while increasing the abstraction level and representational capacity of features, as shown in Fig.1(a).

Given an input point cloud $f \in R^{s\times c}$ containing $s = 1024$ points, the network first maps the original 3D coordinates to a high-dimensional feature representation through the **Embedding Block** (detailed structure in Fig.1(b)). This module employs the center_diff mode, based on$k = 32$ nearest neighbor grouping, to compute the relative coordinates between the center point and its neighbors, expanding the 3D absolute coordinates into a 6D local geometric descriptor composed of absolute and relative coordinates concatenated. Subsequently, neighborhood information is aggregated through two "grouping → 2D convolution → max-pooling" operations, finally outputting 128-dimensional point-level features:

$$f_{embed} = Embedding(f), f_{embed} \in R^{s\times d} \quad (1)$$

where $d = 128$ is the output feature dimension. The feature f_{embed} provides an initial representation rich in local structure for subsequent processing.

Subsequently, the features pass through **four consecutive processing stages**, gradually building a multi-scale feature pyramid. Each stage sequentially performs down-sampling (DS Block) and **Local-Global Feature Fusion (LG-GS Block)** operations. Across the four stages, the number of points is **halved successively, and the feature dimension is doubled**, finally outputting a feature dimension of 8C (C=64) at the fourth stage.

Down-sampling Module (DS Block): Its structure is shown in Fig. 1(c). At the beginning of each stage, the module processes the input point cloud spatial coordinates $\mathbf{P} \in \mathbb{R}^{s\times 3}$and features $\mathbf{F}_{in} \in \mathbb{R}^{s\times d/2}$ in parallel: on one hand, it obtains $s/2$ key point coordinates $\mathbf{P}_{fps} \in \mathbb{R}^{(s/2)\times 3}$ via Farthest Point Sampling (FPS); on the other hand, it uses a lightweight feature transformation module Convs to increase the dimensionality of the input features. This module contains two identical sequential operations, each consisting of: 2D convolution (Conv2d), batch normalization (BN), and ReLU activation function. Finally, with $\mathbf{P}_{fps}$ as

centers, structured patch features $\mathbf{F}_{patch}$ (corresponding to Patches in Fig. 1(c)) are constructed via K-Nearest Neighbors (KNN) search, which are input to the subsequent LG-GS block.

$$\mathrm{F_{out}} = \mathrm{Convs}(\mathrm{F_{in}}), \quad \mathrm{F_{in}} \in \mathrm{R}^{\mathrm{s}\times\mathrm{d}/2}, \mathrm{F_{out}} \in \mathrm{R}^{\mathrm{s}\times\mathrm{d}} \tag{2}$$

$$F_{patch} = KNN(P_{fps}, F_{out}), \quad F_{patch} \in R^{(s/2)\times k\times d} \tag{3}$$

where s denotes the total number of input points at the current stage, $s/2$ is the number of points after FPS down-sampling (total number of patches), $\mathbf{P}_{fps}$ are the spatial coordinates of the sampled points, k is the number of points in the KNN neighborhood grouping, and $d = 2^{m-1}C$ is the patch feature dimension.

Local-Global Feature Fusion Block (LG-GS Block): The structured patches obtained from the down-sampling module are fed into the core Local-Global Feature Fusion Block (LG-GS Block), whose structure is shown in Fig. 1(d). This block adopts a parallel architecture, consisting of the Local Geometric Feature Extractor (LGFE) and the Global Semantic Feature Extractor (GSFE). The former captures fine-grained local geometric structures, while the latter models long-range contextual dependencies. Feature enhancement is achieved through **feature fusion**. Their detailed designs are elaborated in Sections 3.2 and 3.3.

Finally, the output features from the fourth stage are passed through a global pooling layer and then mapped to the final classification result by a classification head containing two linear layers.

3.2 Local Geometric Feature Extractor (LGFE)

This module is designed to efficiently capture the fine-grained geometric structure of local regions in point clouds. As shown in Fig. 1(d), its input is the structured patch feature tensor $\mathbf{F}_{patch} \in \mathbb{R}^{s\times k\times d}$ output by the down-sampling module, where s is the number of patches, k is the number of neighbor points per patch, and $d = 2^{m-1}C$ is the input feature dimension. The specific processing flow of LGFE is as follows:

First, to enhance geometric awareness, we compute the relative coordinates of each neighborhood point $\mathbf{p}_{ij}$ within a patch relative to its center point $\hat{\mathbf{p}}_i$:

$$\Delta\mathrm{p_{ij}} = \mathrm{p_{ij}} - \hat{\mathrm{p}}_\mathrm{i}, \quad \forall \mathrm{p_{ij}} \in \mathcal{N}(\hat{\mathrm{p}}_\mathrm{i}) \tag{4}$$

where $\mathcal{N}(\hat{\mathbf{p}}_i)$ denotes the neighborhood set of the center point $\hat{\mathbf{p}}_i$, We concatenate this relative position information with the original patch features, injecting fine-grained local spatial geometric constraints into subsequent feature encoding, thereby strengthening the model's perception of local structure.

Subsequently, the patch center point feature $\mathbf{F}_{centroid} \in \mathbb{R}^{s\times 1\times d}$ is extracted. The relative features of neighborhood points relative to the center point are computed, and both are fused to enhance local structure representation:

$$\mathrm{F_{relative}} = \mathrm{F_{patch}} - \mathrm{F_{centroid}} \tag{5}$$

$$F_{enhanced} = F_{centroid} \oplus F_{relative} \tag{6}$$

where $\mathbf{F}_{relative}$ represents the relative features, $\oplus$ denotes the feature concatenation operation, and $\mathbf{F}_{enhanced}$ is the locally enhanced feature.Max pooling (Max-Pool) is applied along the neighborhood dimension k to aggregate features within each patch, preserving the most significant local structural responses:

$$F_{pooled} = \mathrm{MaxPool}_k(F_{enhanced}), \quad F_{pooled} \in R^{s\times 2d} \tag{7}$$

Finally, a Residual Enhancer containing residual connections is used to perform nonlinear transformation on the aggregated features, outputting the enhanced local geometric features:

$$F_{local} = \mathrm{ResEnhancer}(F_{pooled}), \quad F_{local} \in \mathbb{R}^{s\times d} \tag{8}$$

This design enables the LGFE module to accurately extract and enhance the fine-grained geometric structure information within each local patch.

3.3 Global Semantic Feature Extractor (GSFE)

The GSFE module aims to efficiently capture the global semantic context and long-range dependencies of point cloud data. As shown in Fig. 1(d), its input is the patch-level aggregated feature $\mathbf{F} \in \mathbb{R}^{s\times d}$, where s is the number of patches and $d = 2^{m-1}C$ is the feature dimension. This module fuses dual mechanisms: Patch-Point Self-Attention (PP-SA) and Channel Self-Attention (CSA), modeling inter-patch and inter-channel dependencies respectively.

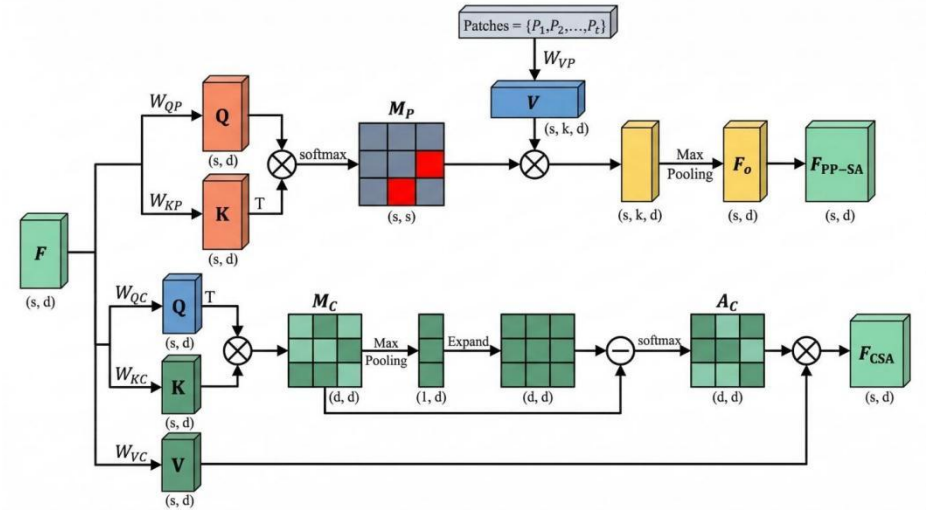

Fig. 2. The dual attention mechanisms in GSFE. Top: Patch-Point Self-Attention (PPSA), modeling dependencies between patches. Bottom: Channel Self-Attention (CSA), modeling correlations across feature channels.

Patch-Point Self-Attention (PP-SA) The PP-SA mechanism (detailed data flow in Fig. 3) aims to modulate and fuse local neighborhood features by leveraging global relationships between patches. Its input consists of two parts: the patch-level aggregated feature $\mathbf{F}$, and the neighborhood features $\mathbf{N} = \mathbf{F}_{patch} \in \mathbb{R}^{s\times k\times d}$, where k is the number of neighbor points within a single patch.

First, the query (Query) and key (Key) matrices are generated by projecting $\mathbf{F}$, while the neighborhood features $\mathbf{N}$ are projected as the value (Value) matrix:

$$\mathrm{Q} = \mathrm{FW_Q}, \quad \mathrm{K} = \mathrm{FW_K}, \quad \mathrm{V} = \mathrm{NW_V} \tag{9}$$

where $\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V \in \mathbb{R}^{d\times d}$ are learnable weight matrices. The inter-patch attention map $\mathbf{M}_p \in \mathbb{R}^{s\times s}$ is calculated as follows:

$$\mathrm{M_p} = \mathrm{Softmax}\left(\frac{\mathrm{QK^T}}{\sqrt{\mathrm{d}}} + \mathrm{B}\right) \tag{10}$$

Here, $\mathbf{B} \in \mathbb{R}^{s\times s}$ is a learnable bias based on the relative positions of patches, used to inject spatial structural information.

The value matrix $\mathbf{V}$ is aggregated with weights from the attention map $\mathbf{M}_p$. For a target patch i, its feature is obtained by the weighted sum of the value vectors from all patches j according to $M_{p,ij}$:

$$\mathrm{F_{weighted}[i]} = \sum_{\mathrm{j=1}}^{\mathrm{s}} \mathrm{M_{p,ij}} \cdot \mathrm{V[j]}, \quad \mathrm{F_{weighted}} \in \mathbb{R}^{\mathrm{s\times k\times d}} \tag{11}$$

Finally, max pooling is performed along the neighborhood dimension k to extract the most significant feature response for each patch, yielding the PP-SA output:

$$\mathrm{F_{PP-SA}} = \mathrm{MaxPool_k(F_{weighted})}, \quad \mathrm{F_{PP-SA}} \in \mathbb{R}^{\mathrm{s\times d}} \tag{12}$$

Channel Self-Attention (CSA) The CSA mechanism (detailed data flow in Fig. 3) aims to model correlations between different feature channels, refining contextual information along the channel dimension by enhancing cross-channel interaction. Given the input feature $\mathbf{F}$, we first project it into query, key, and value matrices for the channel dimension:

$$\mathrm{Q_c} = \mathrm{FW_{Q_c}}, \quad \mathrm{K_c} = \mathrm{FW_{K_c}}, \quad \mathrm{V_c} = \mathrm{FW_{V_c}} \tag{13}$$

where $\mathrm{W_{Q_c}}, \mathrm{W_{K_c}}, \mathrm{W_{V_c}} \in \mathbb{R}^{\mathrm{d\times d}}$ are learnable weight matrices. The channel attention map $\mathbf{M}_c \in \mathbb{R}^{d\times d}$ is obtained by calculating the similarity between channels:

$$\mathrm{M_c} = \mathrm{K_c^T Q_c} \tag{14}$$

To emphasize differentiated information and avoid aggregating redundant features, we compute the channel affinity matrix $\mathbf{A}_c \in \mathbb{R}^{d\times d}$ based on $\mathbf{M}_c$:

$$\mathrm{A_c} = \mathrm{Softmax(expand(maxpool(M_c)) - M_c)} \tag{15}$$

where max-pool $(\cdot)$ denotes the row-wise maximum operation on the matrix, and expand $(\cdot)$ expands this maximum value vector into a $d \times d$ matrix along the column direction. This design ensures that a larger value in $\mathbf{A}_c$ corresponds to a more significant semantic difference between two channels, thereby focusing attention on non-redundant channels.

Finally, the channel affinity matrix is used to perform weighted aggregation on the value matrix $\mathbf{V}_c$, yielding the CSA output features:

$$\mathrm{F_{CSA}} = \mathrm{V_c A_c}, \quad \mathrm{F_{CSA}} \in \mathbb{R}^{\mathrm{s\times d}} \tag{16}$$

Dual-Path Feature Fusion The output features of PP-SA and CSA capture inter-patch dependencies and inter-channel context information, respectively. They are fused via element-wise addition to obtain the enhanced global semantic features:

$$\mathrm{F_{global}} = \mathrm{F_{PP-SA}} + \mathrm{F_{CSA}} \tag{17}$$

To stabilize the training process and preserve original feature information, a residual connection is introduced at the end of the module:

$$\mathbf{F}_{output} = \mathbf{F} + LBR(\mathbf{F}_{global}) \tag{18}$$

where LBR $(\cdot)$ denotes a feature transformation module composed of a Linear layer, Batch Normalization, and ReLU activation function. $\mathbf{F}_{output} \in \mathbb{R}^{s\times d}$ is the final output feature of the GSFE module.

4 Experiments

This section details the experimental results of the proposed PointLGGS model on point cloud classification tasks. All experiments were conducted using the PyTorch framework on a single NVIDIA RTX 3090 GPU. To ensure reproducibility and fair comparison, we adopted a unified and strict training configuration: the Stochastic Gradient Descent (SGD) optimizer with momentum and weight decay set to 0.9 and 0.0001, respectively; the learning rate followed a cosine annealing schedule with an initial value of 0.001, accompanied by a 10 epoch linear warm-up; the loss function was cross-entropy loss with label smoothing; all experiments used fixed random seeds to eliminate randomness; the model input was uniformly sampled 1024 points. The data augmentation strategy included conventional random scaling and translation, with additional RSMix introduced to enhance model generalization.

Adaptive hyper-parameters were used for different dataset characteristics: on ModelNet40[22], the batch size was 32, training for 300 epochs, and the test batch size was 16; on the more challenging real-world dataset ScanObjectNN[23], the batch size was set to 64, training for 400 epochs, and the test batch size was 32.

Evaluation Metrics: To objectively and comprehensively measure model performance, we report the mean per-class accuracy (mean Accuracy, mAcc) and the overall accuracy (Overall Accuracy, OA), defined as follows:

$$\mathrm{mAcc} = \frac{1}{\mathrm{N}}\sum_{\mathrm{i}=1}^{\mathrm{N}} \frac{\mathrm{T_i}}{\mathrm{M_i}}, \quad \mathrm{OA} = \frac{\mathrm{T_{all}}}{\mathrm{M_{all}}} \tag{19}$$

where N is the total number of categories in the dataset, Mi is the number of samples in the ith class, and Ti is the number of correctly classified samples in the ith class. Let $T_{all} = \sum_{i=1}^{N} T_i$ be the total number of correctly classified samples, and $M_{all} = \sum_{i=1}^{N} M_i$ be the total number of samples in the dataset.

Furthermore, to fully validate the model's performance, we compare our method with two mainstream categories of point cloud classification methods: MLP/CNN-based methods for local feature modeling, and Transformer-based methods for global context modeling.

4.1 ModelNet40 Classification Experiment

Table 1. Shape Classification Results on the ModelNet40 Dataset

Method	OA(%)	mAcc(%)	Params(M)
MLP/CNN-based Methods			
PointNet[15]	89.2	86.2	3.5
PointNet++[7]	90.7	—	1.5
PointCNN[16]	92.2	88.1	0.6
DGCNN[8]	92.9	90.2	1.8
SimpleView[24]	93.0	90.5	0.8
MVTN[25]	93.8	**92.2**	11.2
DeepGCN[26]	93.6	90.9	2.2
KPConv[27]	92.9	—	14.3
CurveNet[28]	93.8	—	2.0
PointMLP[18]	**94.5**	91.4	12.6
PointNeXt[29]	93.7	90.9	1.4
RepSurf[30]	94.0	91.1	1.48
LinNet[31]	93.6	91.0	1.4
Transformer-based Methods			
Transformer[9]	91.4	—	22.1
PointTransformer[10]	93.7	90.6	22.1
PCT[19]	93.2	90.3	2.8
Point-BERT[32]	93.2	—	22.1
Point-MAE[33]	93.8	—	22.1
Point-M2AE[34]	94.0	—	15.3
3DCTN[21]	93.2	91.6	4.21
GBNet[35]	93.8	91.0	8.38
APES[36]	93.8	—	**0.35**
PointLGGS	94.0	91.6	4.6

We evaluated the proposed PointLGGS model on shape classification using the ModelNet40[22] dataset. This dataset is a widely used benchmark in point cloud classification, containing 12, 311 CAD models from 40 categories, with 9, 840 for training and 2, 468 for testing. Quantitative results are summarized in Table 1.

To intuitively validate the feature decoupling effect and working mechanism of our PointLGGS framework, we visualize the learned features on an airplane sample from ModelNet40 as shown in Fig. 3. Specifically, the global semantic feature extractor (GSFE) focuses on discriminative semantic regions such as the tail, fuselage and wing roots, which provide critical clues for accurate category recognition. In contrast, the local geometric feature extractor (LGFE) concentrates on geometric contours and structural variations, enabling the model to capture fine-grained shape information for robust representation. Moreover, the t-SNE visualization illustrates that features from GSFE and LGFE form two distinct and well-separated clusters, which convincingly verifies that the dual branches learn decoupled and non-redundant representations.

As shown in the table, PointLGGS achieves excellent comprehensive performance. Compared to the pioneering work PointNet, thanks to the dual-path design for local-global collaboration, PointLGGS shows a significant improvement of 4.8% in OA. Compared to methods that also focus on local geometric modeling, such as DGCNN and PointCNN, PointLGGS effectively captures long-range dependencies through the Patch-Point Self-Attention in the GSFE module, outperforming them by 1.1% and 1.8% in OA, respectively. Compared to pure attention-based methods (e.g., PointTransformer[10]), PointLGGS employs an efficient lightweight design in the LGFE module, maintaining powerful global

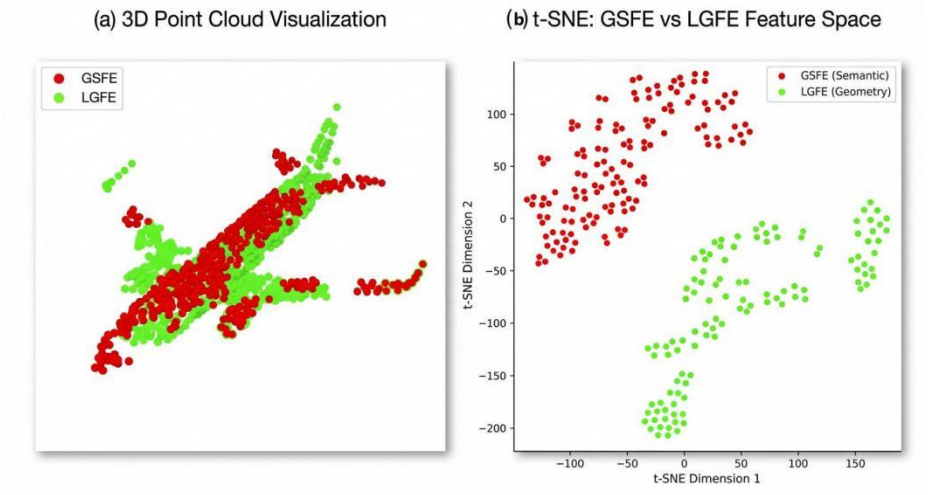

Fig. 3. PointLGGS feature visualization on an airplane sample from ModelNet40.(a) GSFE focuses on discriminative semantic regions for classification, while LGFE captures geometric contours to model shape information.(b) t-SNE visualization shows separated clusters of GSFE and LGFE features, verifying their decoupled and non-redundant properties.

modeling capabilities while significantly reducing model complexity and computational cost. Ultimately, with only 4.6M parameters, PointLGGS achieves an overall accuracy of 94.0%, striking a better balance between accuracy and efficiency.

4.2 ScanObjectNN Classification Experiment

We further evaluated the model's robustness on the more challenging real-world dataset ScanObjectNN[23]. This dataset contains point clouds segmented from real scanned scenes, covering 15 object categories with a total of 15, 000 samples. Due to significant background noise, occlusion, and geometric incompleteness in the point clouds, this dataset poses higher demands on the model's discriminative and generalization capabilities. Following the standard split, the training and test sets contain 2, 321 and 581 samples, respectively.

Table 2. Shape Classification Results on the ScanObjectNN (PB-T50-RS) Dataset

Method	OA(%)	mAcc(%)	Params(M)
MLP/CNN-based Methods			
PointNet[15]	68.0	63.4	3.5
PointNet++[7]	77.9	75.4	1.5
PointCNN[16]	78.5	75.1	**0.6**
DGCNN[8]	78.1	73.6	1.8
MVTN[25]	82.8	—	11.2
PointMLP[18]	85.4	83.9	12.6
PointNeXt[29]	87.7	85.8	1.4
RepSurf[30]	86.0	83.1	1.48
PointMetaBase[37]	87.9	—	—
Transformer-based Methods			
PCT[19]	85.1	82.8	2.8
3DCTN[21]	81.5	79.5	4.21
Point-BERT[32]	83.1	—	22.1
Point-MAE[33]	85.2	—	22.1
Point-M2AE[34]	86.4	—	15.3
GBNet[35]	80.5	77.8	8.38
PTv3[38]	86.4	83.9	—
ACT[39]	88.2	—	12.0
MaskFeat3D[40]	88.4	—	14.5
PointLGGS	88.2	87.1	4.6
PointLGGS*	**90.7**	**89.4**	4.6

The experimental results are shown in Table 2. PointLGGS demonstrates excellent classification performance on this dataset. Without employing test-time voting augmentation, the model already achieves 88.2% OA and 87.1% mAcc; with voting ensemble, performance further improves to 90.7% OA and 89.4% mAcc, significantly outperforming existing mainstream methods. This result fully validates the

effectiveness of the proposed parallel dual-path architecture: the LGFE module enhances the model's robustness to noise and occlusion through fine-grained local geometric modeling; the GSFE module compensates for the lack of local information through global context modeling. Their collaborative operation enables PointLGGS to effectively adapt to the irregularity and complexity of real-world point clouds, providing reliable technical support for practical 3D perception applications.

4.3 Ablation Study

To verify the effectiveness of each core component of PointLGGS, we conducted ablation experiments on the ScanObjectNN[23] dataset, with consistent training settings across all experiments. As previously clarified, the GSFE module consists of two sub-modules, PP-SA and CSA; therefore, this experiment also verifies the rationality of the internal structure of GSFE.

Component Ablation: As shown in Table 3, the LGFE module is responsible for fine-grained local geometric feature extraction. Its removal causes OA and mAcc to drop by 5.3% and 6.1% respectively, confirming its crucial role in mining local structures. **RES (Residual Enhancer)** corresponds to the Res-Enhancer in Equation (8), used for feature enhancement and mitigating gradient vanishing. Its removal leads to a decrease of 0.8% in OA and 0.7% in mAcc. As core sub-modules of GSFE, PP-SA and CSA collaboratively achieve global semantic modeling: PP-SA aggregates global information by modeling long-range global dependencies between patches. Its removal results in a decrease of 0.6% in OA and 0.5% in mAcc. CSA is responsible for channel feature calibration. Its removal causes a decrease of 0.4% in OA and 0.5% in mAcc. The four components work synergistically to ensure the excellent performance of PointLGGS.

Table 3. Ablation Study Results for PointLGGS Components (ScanObjectNN)

ID	LGFE	RES	PP-SA	CSA	OA(%)	mAcc(%)	GFLOPs	GMACs	Params(M)
I	✓	✓			86.40	85.40	**2.54**	**1.25**	**2.89**
II	✓	✓	✓		87.20	86.10	2.67	1.31	3.59
III	✓	✓	✓	✓	87.80	86.60	3.06	1.51	4.20
IV	✓				**88.20**	**87.10**	3.21	1.58	4.60
V VI	✓	✓	✓	✓	87.50	86.10	2.95	1.45	4.29
VII	✓		✓	✓	87.50	86.30	3.09	1.52	3.94
				✓	82.90	81.00	3.15	1.55	3.67

Number of Stages: To evaluate the impact of model depth on performance, we tested configurations with 3, 4, and 5 stages. The results are shown in Table 4. The model performs best with 4 stages, achieving 88.20% OA and 87.10% mAcc. When the number of stages is reduced to 3, the model's expressive capacity is limited, and OA drops to 87.40%. Increasing the number of stages to 5 significantly increases the parameter count from 4.60M to 17.25M, but OA actually decreases to 87.80%, likely due to over-fitting or gradient vanishing. The results indicate that the 4-stage configuration achieves the best balance between model expressive capacity and computational efficiency, effectively extracting and fusing multi-scale features.

Table 4. Ablation Study Results for the Number of Stages (ScanObjectNN)

Stages	OA(%)	mAcc(%)	GFLOPs	GMACs	Params(M)
3	87.40	85.90	**2.79**	**1.37**	**1.42**
4	**88.20**	**87.10**	3.21	1.58	4.60
5	87.80	86.60	4.02	2.01	17.25

Patch Size: To investigate the impact of local feature extraction granularity on model performance, we analyzed configurations with different patch_size values. The experimental results are shown in Table 5. The model achieves optimal performance when patch_size is set to 20, with an overall accuracy (OA) of 88.20% and a mean accuracy (mAcc) of 87.10%. Smaller patch sizes (8, 16) or overly large sizes (32) both lead to a decrease in model performance, validating that a moderate local region division can more efficiently capture the geometric structure of point clouds, which aligns with the design intention. Furthermore, computational costs (GFLOPs, GMACs) and parameter counts remain constant across different configurations, indicating that this parameter only affects feature representation quality without increasing computational complexity.

Table 5. Ablation Study Results for Patch Size (ScanObjectNN)

patch_size	OA(%)	mAcc(%)	GFLOPs	GMACs	Params(M)
8	88.10	87.00	3.21	1.58	4.60
16	88.00	86.70	3.21	1.58	4.60
20	**88.20**	**87.10**	3.21	1.58	4.60
32	87.80	86.40	3.21	1.58	4.60

5 Conclusion

This paper addresses the challenges in point cloud classification where efficient collaboration between local geometry and global semantics is difficult, and a balance between accuracy and efficiency is hard to achieve. We propose a lightweight dual-path parallel network, **PointLGGS**. This method uses the LG-GS block as its basic unit and adopts a parallel architecture to decouple the extraction process of local geometric features and global semantic features, alleviating the feature bias and efficiency loss inherent in traditional serial structures. Specifically, the LGFE module enhances fine-grained geometric perception through relative position encoding, while the GSFE module efficiently models long-range dependencies and channel correlations via the dual attention mechanisms of PP-SA and CSA. Robust point cloud representations are formed through the adaptive fusion of these two pathways. Experimental results on the ModelNet40 and ScanObjectNN benchmark datasets demonstrate that the proposed method achieves excellent classification performance while maintaining a lightweight architecture, effectively balancing classification accuracy, computational cost, and parameter count. Future work will focus on

dynamic feature selection and adaptive cross-path interaction to further improve the model's generalization capability and robustness in complex and low-quality real-world point cloud scenarios.

Acknowledgments. This work was supported in part by the Finance Science and Technology Project of Xinjiang Uyghur Autonomous Region (No. 2023B01029-2), the '2+5' Key Talent Program of the Xinjiang Uygur Autonomous Region (Project Title: 'Research on Key Technologies for Design and Construction of Prefabricated Bridges in Xinjiang'), and the Xinjiang Transportation Prefabricated Construction Key Technology Innovation Team (Leading Talent) (No. XJJTZZB-FWCG-202401-0002).

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Z. Chen, J. Zhang, and D. Tao, "Progressive LiDAR adaptation for road detection, " IEEE/CAA J. Autom. Sinica, vol. 6, no. 3, pp. 693–702, 2019.
2. T.-H. Chen and T. S. Chang, "RangeSeg: Rangeaware real time segmentation of 3D LiDAR point clouds, " IEEE Trans. Intell. Veh., vol. 7, no. 1, pp. 93–101, 2022.
3. C. Zhao, C. Fu, J. M. Dolan, and J. Wang, "L-shape fittingbased vehicle pose estimation and tracking using 3D-LiDAR, " IEEE Trans. Intell. Veh., vol. 6, no. 4, pp. 787–798, 2021.
4. X. Bai, Z. Hu, X. Zhu, Q. Huang, Y. Chen, H. Fu, and C.-L. Tai, "Transfusion: Robust lidarcamera fusion for 3d object detection with transformers, " in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., pp. 1090–1099, 2022.
5. C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas, "Frustum pointnets for 3d object detection from rgbd data, " in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., pp. 918–927, 2018.
6. S. Wang, S. Suo, W.-C. Ma, A. Pokrovsky, and R. Urtasun, "Deep parametric continuous convolutional neural networks, " in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., pp. 2589–2597, 2018.
7. C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space, " in Adv. Neural Inf. Process. Syst., pp. 5099–5108, 2017.
8. Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph CNN for learning on point clouds, " ACM Trans. Graph., vol. 38, no. 5, pp. 1–12, 2019.
9. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need, " in Adv. Neural Inf. Process. Syst., pp. 5998–6008, 2017.
10. H. Zhao, L. Jiang, J. Jia, P. H. S. Torr, and V. Koltun, "Point transformer, " in Proc. IEEE/CVF Int. Conf. Comput. Vis., pp. 16259–16268, 2021.
11. H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller, "Multiview convolutional neural networks for 3D shape recognition, " in Proc. IEEE Int. Conf. Comput. Vis., pp. 945–953, 2015.
12. X. Wei, R. Yu, and J. Sun, "View-GCN: Viewbased graph convolutional network for 3D shape analysis, " in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., pp. 1850–1859, 2020.
13. D. Maturana and S. Scherer, "Voxnet: A 3D convolutional neural network for realtime object recognition, " in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., pp. 922–928, 2015.

14. G. Riegler, A. O. Ulusoy, and A. Geiger, “Octnet: Learning deep 3D representations at high resolutions, ” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., pp. 3577–3586, 2017.
15. C. R. Qi, H. Su, K. Mo, and L. J. Guibas, “Pointnet: Deep learning on point sets for 3d classification and segmentation, ” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., pp. 652–660, 2017.
16. M. Xu, R. Ding, H. Zhao, and X. Qi, “PAConv: Position adaptive convolution with dynamic kernel assembling on point clouds, ” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognition., pp. 3173–3182, 2021.
17. Y. Xu, T. Fan, M. Xu, L. Zeng, and Y. Qiao, “SpiderCNN: Deep learning on point sets with parameterized convolutional filters, ” in Proc. Eur. Conf. Comput. Vis., pp. 87–102, 2018.
18. X. Ma, C. Qin, H. You, H. Ran, and Y. Fu, “Rethinking network design and local geometry in point cloud: A simple residual MLP framework, ” arXiv preprint arXiv:2202.07123, 2022.
19. M.-H. Guo, J.-X. Cai, Z.-N. Liu, T.-J. Mu, R. R. Martin, and S.-M. Hu, “PCT: Point cloud transformer, ” Comput. Vis. Media, vol. 7, no. 2, pp. 187–199, 2021.
20. H. Zhao, L. Jiang, J. Jia, and V. Koltun, “Point Transformer V2: Grouped Vector Attention and Partitionbased Pooling, ” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., pp. 16010–16019, 2022.
21. D. Lu, J. Xie, K. Lin, X. Zhou, and L. Wang, “3DCTN: 3D convolutiontransformer network for point cloud classification, ” IEEE Trans. Intell. Transp. Syst., vol. 23, no. 12, pp. 24854– 24865, 2022.
22. Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, “3D ShapeNets: A deep representation for volumetric shapes, ” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., pp. 1912–1920, 2015.
23. M. A. Uy, Q.-H. Pham, B.-S. Hua, D. T. Nguyen, and S.-K. Yeung, “Revisiting point cloud classification: A new benchmark dataset and classification model on realworld data, ” in Proc. IEEE Int. Conf. Comput. Vis., pp. 1588–1597, 2019.
24. A. Goyal, H. Law, B. Liu, A. Newell, and J. Deng, “Revisiting point cloud shape classification with a simple and effective baseline, ” in Proc. Int. Conf. Mach. Learn., pp. 3801–3812, 2021.
25. A. Hamdi, S. Giancola, and B. Ghanem, “MVTN: Multiview transformation network for 3D shape recognition, ” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., pp. 1–11, 2021.
26. G. Li, M. Müller, A. Thabet, and B. Ghanem, “DeepGCNs: Can GCNs go as deep as CNNs, ” in Proc. IEEE/CVF Int. Conf. Comput. Vis., pp. 9267–9266, 2019.
27. H. Thomas, C. R. Qi, J.-E. Deschaud, B. Marcotegui, F. Goulette, and L. J. Guibas, “KPConv: Flexible and deformable convolution for point clouds, ” in Proc. IEEE/CVF Int. Conf. Comput. Vis., pp. 6411–6420, 2019.
28. T. Xiang, C. Zhang, Y. Song, J. Yu, and W. Cai, “Walk in the cloud: Learning curves for point clouds shape analysis, ” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., pp. 915–924, 2021.
29. Q. Li, Y. Shen, L. Chen, S. Chen, and Y. Wang, “PointNeXt: Revisiting PointNet++ with improved training and scaling strategies, ” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., pp. 19488–19497, 2022.
30. H. Ran, J. Liu, and C. Wang, “Surface representation for point clouds, ” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., pp. 18942–18952, 2022.
31. H. Deng, K. Jing, S. Cheng, C. Liu, J. Ru, J. Bo, and L. Wang, “Linnet: Linear network for efficient point cloud representation learning, ” in Adv. Neural Inf. Process. Syst., vol. 37, pp. 43189–43209, 2024.

32. X. Yu, L. Tang, Y. Rao, T. Huang, J. Zhou, and J. Lu, “Point-BERT: Pretraining 3D point cloud transformers with masked point modeling, ” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., pp. 19313–19322, 2022.
33. Y. Pang, W. Wang, F. E. H. Tay, W. Liu, Y. Tian, and L. Yuan, “Masked autoencoders for point cloud selfsupervised learning, ” in Proc. Eur. Conf. Comput. Vis., pp. 604–621, 2022.
34. R. Zhang, Z. Guo, P. Gao, R. Fang, B. Zhao, D. Wang, Y. Qiao, and H. Li, “Point-M2AE: Multiscale masked autoencoders for hierarchical point cloud pretraining, ” in Adv. Neural Inf. Process. Syst., vol. 35, pp. 27061–27074, 2022.
35. S. Qiu, S. Anwar, and N. Barnes, “Geometric backprojection network for point cloud classification, ” IEEE Trans. Multimed., vol. 24, pp. 1943–1955, 2022.
36. C. Wu, J. Zheng, J. Pfrommer, and J. Beyerer, “Attentionbased point cloud edge sampling, ” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., pp. 13479–13488, 2023.
37. H. Lin, X. Zheng, L. Li, F. Chao, S. Wang, Y. Wang, Y. Tian, and R. Ji, “Meta architecture for point cloud analysis, ” arXiv preprint arXiv:2211.14462, 2022.
38. X. Wu, L. Jiang, P.-S. Wang, Z. Liu, X. Liu, Y. Qiao, W. Ouyang, T. He, and H. Zhao, “Point Transformer V3: Simpler, Faster, Stronger, ” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., pp. 20641–20651, 2024.
39. Y. Li, R. Yu, G. Li, and L. Gu, “Attention Clustering Transformer for Point Cloud Analysis, ” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., pp. 19468–19477, 2022.
40. C. Chen, S. Li, Y. Wang, and H. Li, “Masked Feature Prediction for Self-Supervised Visual Pre-Training, ” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., pp. 19478–19487, 2022.