Full Point Encoding for Local Feature Aggregation in 3-D Point Clouds

Yong He, Hongshan Yu, Zhengeng Yang, Xiaoyan Liu, Wei Sun, and Ajmal Mian, *Senior Member, IEEE*

***Abstract*— Point cloud processing methods exploit local point features and global context through aggregation which does not explicitly model the internal correlations between local and global features. To address this problem, we propose full point encoding which is applicable to convolution and transformer architectures. Specifically, we propose full point convolution (FuPConv) and full point transformer (FPTransformer) architectures. The key idea is to adaptively learn the weights from local and global geometric connections, where the connections are established through local and global correlation functions, respectively. FuPConv and FPTransformer simultaneously model the local and global geometric relationships as well as their internal correlations, demonstrating strong generalization ability and high performance. FuPConv is incorporated in classical hierarchical network architectures to achieve local and global shape-aware learning. In FPTransformer, we introduce full point position encoding in self-attention, that hierarchically encodes each point position in the global and local receptive field. We also propose a shape-aware downsampling block that takes into account the local shape and the global context. Experimental comparison to existing methods on benchmark datasets shows the efficacy of FuPConv and FPTransformer for semantic segmentation, object detection, classification, and normal estimation tasks. In particular, we achieve state-of-the-art semantic segmentation results of 76.8% mIoU on S3DIS sixfold and 73.1% on S3DIS Area 5. Our code is available at https://github.com/hnuhyuwa/FullPointTransformer.**

Manuscript received 2 March 2023; revised 24 November 2023 and 8 April 2024; accepted 1 June 2024. This work was supported in part by the National Natural Science Foundation of China under Grant U2013203, Grant 62373140, Grant U21A20487, and Grant 62103137; in part by the National Key Research and Development Program under Grant 2023YFB4704500; in part by the Project of Science Fund for Distinguished Young Scholars of Hunan Province under Grant 2021JJ10024; in part by the Leading Talents in Science and Technology Innovation of Hunan Province under Grant 2023RC1040; in part by the Project of Science Fund of Hunan Province under Grant 2022JJ30024; in part by the Project of Talent Innovation and Sharing Alliance of Quanzhou City under Grant 2021C062L; in part by the Key Research and Development Project of Science and Technology Plan of Hunan Province under Grant 2022GK2014; and in part by the Natural Science Fund of Hunan Province under Grant 2022JJ40100. The work of Ajmal Mian was supported by the Australian Research Council Future Fellowship Award by the Australian Government under Project FT210100268. *(Yong He and Hongshan Yu contributed equally to this work.) (Corresponding author: Hongshan Yu.)*

Yong He, Hongshan Yu, Xiaoyan Liu, and Wei Sun are with the College of Electrical and Information Engineering, School of Robotics, Quanzhou Institute of Industrial Design and Machine Intelligence Innovation, Hunan University, Yuelu, Changsha 410082, China (e-mail: h.yong@hnu.edu.cn; yuhongshancn@hotmail.com; xiaoyan.liu@hnu.edu.cn; david-sun@126.com).

Zhengeng Yang is with the College of Engineering and Design, Hunan Normal University, Yuelu, Changsha 410082, China (e-mail: yzg050215@163.com).

Ajmal Mian is with the Department of Computer Science, The University of Western Australia, Perth, WA 6009, Australia (e-mail: ajmal.mian@uwa.edu.au).

Digital Object Identifier 10.1109/TNNLS.2024.3409891

***Index Terms*— 3D point clouds, convolution, deep learning, global context, local features, transformer.**

I. Introduction

POINT cloud processing has drawn considerable research interest due to its wide range of applications in autonomous driving [1], [2], [3], robotics [4], and industrial automation [5]. Learning effective features from raw point clouds is difficult due to its irregular nature. Early methods transformed points into regular grids (e.g., multiview images [6], [7], voxels [8], [9]), for seamless application of grid convolutions. However, the discretization process inevitably sacrifices important geometric information, distorts object shapes, and results in a huge computational overhead.

To learn features from raw point clouds, the pioneering work PointNet [10] employs shared multilayer perceptrons (MLPs) on each point and uses maxpooling to aggregate the features into a global representation. Such a design ignores local structures that are crucial for shape representation. To alleviate this problem, PointNet++ [11] additionally exploits local aggregation and adopts MLPs to learn local features. However, the local aggregator still treats the points independently, losing sight of the overall shape.

To gain a deeper understanding of the shape inherent in irregular point clouds, it is essential to employ the correlation function to quantify the connection among points, allowing the model to leverage the fine-grained details (e.g., edge, corner, and surface) and contextual information (e.g., scene layout). Inspired by 2-D convolutions, some methods exploit point convolutions that learn the convolutional weights from local geometric connections. Early point convolutions use MLPs [12], [13], [14], [15], [16] as the convolutional weight function to learn weights from point coordinates. Other works approximate the weight functions as correlation functions [17], [18], [19], [20]. Some methods associate coefficients (derived from point coordinates) [15], [16], [21] with weight functions to adjust the learned weights.

Self-attention was introduced to focus on important details in point clouds. Early methods in this category usually operate on global receptive field [22], [23], [24], [25], [26], [27] and learn the attention maps through a scalar dot-product. Differently, point transformer (PT) [28] employs self-attention to the local receptive field and uses vector attention instead of scalar dot-product for learning the local geometric connections. Applying aggregation modules on local regions can effectively learn the local structure, but falls short of capturing the global context. To learn both

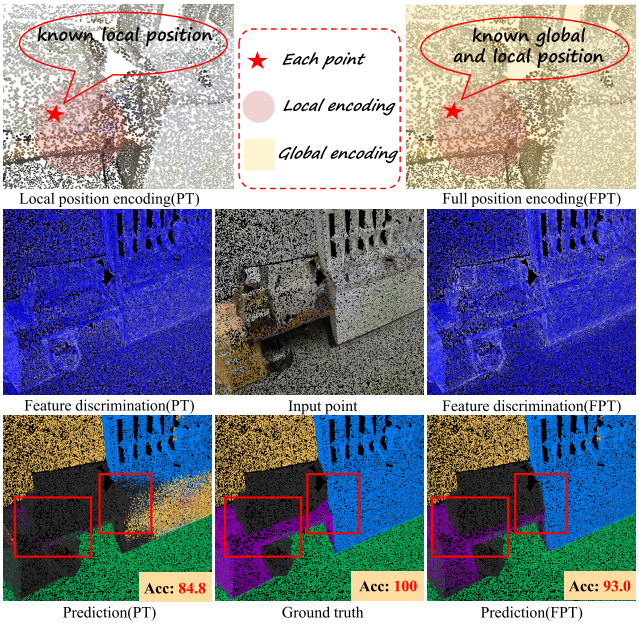

Fig. 1. Full point position encoding makes each point aware of its position in the global and local receptive field (top), helping the proposed FPTransformer to simultaneously model global and local geometric cues as well as their internal correlations. FPTransformer enhances discriminating features of long-range shapes, e.g., edges, improving full shape awareness.

local and global features, some methods integrate local and nonlocal feature aggregation modules into their network in a serial [26], [29] or parallel [27] manner. However, such pipelines still ignore the internal correlations between the local and global features local and the global features besides significantly increasing the network parameters and leading to poor generalization. Although hierarchical architecture can help local aggregation to get a global receptive field and learn global features, this has three limitations: shallow encoding layers over limited local receptive field can not learn the super long-range features of high-resolution points; deep encoding layers over global receptive field struggle to accurately learn super long-range features of low-resolution points; the robustness of global features learned by deep encoding layers highly depends on the local features learned by shallow encoding layers.

We propose full point position encoding, applicable to convolution and transformer modules. Specifically, we propose full point convolution (FuPConv) and Full PT (FPTransformer) that exploit the local features and global context along with their interactions. FuPConv learns the global points layout in the global receptive field by a global correlation function, and then learns the local points layout in the local receptive field by a local correlation function, in a hierarchical manner. In FPTransformer, we introduce a novel position encoding scheme into the self-attention that fully encodes the position information in the local as well as the global receptive field in a hierarchical manner. This makes each point aware of its position in the global and local structure. To summarize, the full point encoding in the FuPConv and FPTransformer learns global features within the global receptive field, then groups these global features into the local receptive field, and finally learns local features based on these grouped global features over the local receptive field. By alternately encoding features in the global and local receptive field within every local aggregation, full point encoding makes each point aware of its position in the global and local structure, helping local aggregation to better connect the global features with the local features over multiple different resolution points. Fig. 1 intuitively compares the local position encoding (LPE) in PT [28] to full point position encoding in our FPTransformer. FPTransformer enhances the feature discrimination of long-range shapes (e.g., edges), improving full shape awareness. In FPTransformer, we also propose a shape-aware downsampling block that integrates hierarchical MLPs with the farthest point subsampling to maintain the shape information in terms of point positions as well as features. Our contributions are summarized as follows.

1) We derive a general formulation for local feature aggregation methods, including local point-wise MLP, point convolution, and PT, to highlight their limitations.
2) Based on the above general formulation, we propose a full point encoding method so as to simultaneously model local and global geometric features of point clouds along with their internal correlations. Using our novel encoding, we propose two network architectures, namely, FuPConv and FPTransformer and show promising results with both.
3) We propose a learnable downsampling block that performs local and global shape-aware downsampling by incorporating the full point position encoding of the proposed FPTransformer into point-wise MLPs.

We conduct extensive experiments on benchmark datasets to show the efficacy of our proposed methods and their strong generalization ability to different tasks such as semantic segmentation, object detection, classification, and normal estimation. FuPConv achieves competitive results on semantic segmentation, classification, and normal estimation tasks compared to various point convolution-based methods. FPTransformer achieves state-of-the-art semantic segmentation performance with 76.8% mIoU on S3DIS sixfold and 73.1% mIoU on S3DIS Area 5. Incorporating FPTransformer into existing detection networks gives considerable performance improvement on ScanNetv2 and KITTI datasets.

II. Related Work

A. Point-Wise MLPs

To maximally preserve the geometric information, recent deep neural networks prefer to directly process raw point clouds. The pioneering work PointNet [10] uses shared MLPs to exploit point-wise features and adopts a symmetric function (i.e., maxpooling) to aggregate these features into global features. However, it fails to consider the geometric relationships of local points. PointNet++ [11] addresses this issue by adopting a hierarchical network that benefits from sampling and grouping. Subsequent works successively propose efficient grouping [16], [24], [27], [30], [31] and sampling [26], [32], [33], [34], [35] methods to improve performance. However, MLP still treats each local point individually and ignores their geometric connections.

Follow-up works construct geometric connections between points to enrich point-wise features and then apply shared

MLPs to them. For instance, some methods [36], [37] handcraft the geometric connections through curves, triangles, umbrella orientation, or affine transformation. Graphs are also used to connect local points [26], [38], [39] or global points [40], [41], [42], [43], for subsequent geometric representation, i.e., edge, contour, curvature, and connectivity. Although geometric connections do not largely increase the learnable network parameters, the parameters of these handcrafted representations or graphs must be optimized for different datasets with varying densities or shape styles.

B. Point Convolution

Inspired by 2-D convolutions, various works successfully proposed novel convolutions on points or their graphs that dynamically learn convolutional weights through functions that operate on local geometric connections. Hence, the weight functions enable convolutions to be aware of the overall object shape. Early methods paid more attention to the weight function design. Most convolutional weight functions are approximated by MLP [12], [15], [16], [44], [45]. Other approaches treat weight functions as local correlation functions such as spline function [19], a family of polynomial functions [20], or standard unparameterized Fourier function [46] to learn the convolution weights from the local geometric connections.

Unlike the above dynamic convolution kernels, KPConv [18] and KCNet [17] fixed the convolution kernel for robustness to varying point density. These networks predefine the kernel points on the local region and learn convolutional weights on the kernel points from their geometric connections to local points using linear and Gaussian correlation functions, respectively. Here, the number and position of kernel points need to be optimized for different datasets.

Convolutional weights are generally learned from local geometric connections by weight functions. Hence, convolutional weight learning highly depends on geometric connections. Some works construct additional low-level geometric connections (e.g., relative positions, 3-D Euclidean distances [44], [45]) to enrich the input to the weight function. Another line of works [15], [16], [21] associates a coefficient (derived from point coordinates) with the weight function to adjust the learned convolutional weights. Convolutional weights learned from low-level geometric connections cannot embed the global context in the convolution operation. Besides, point cloud resolution decreases at deeper layers in many networks, and the geometric connections constructed by the sparse points may get distorted, leading to nonrobust weights learned by the weight function.

C. Point Transformer

Early self-attention modules operate on global points [22], [23], [24], [25], [26], [27] to learn the geometric point connections (i.e., attention map) through scalar dot-product. Such a pipeline suffers from high computational costs and struggles to learn large and complex 3-D scenes. PT [28], [47], on the other hand, employs self-attention to local points and uses vector attention instead to construct the geometric connections between points. This not only requires fewer computations but also helps the PT to learn robust attention weights from high-level geometric connections while encoding local point geometry. The success of PT shows the importance of point position encoding. Sparse convolution [48] performs convolution on all valid neighborhood voxels that are efficiently found using a hash table. Similarly, fast PT [49] leverages local self-attention with a voxel hashing architecture.

Applying the above local aggregation modules can learn the local structure well. However, the local regions divide the global scene into several subscenes, breaking the semantic continuity and integrity. To exploit the long-range features, the stratified transformer [50] densely selects nearby points over a cubic window and sparsely selects distant points. This stratified strategy enlarges the effective receptive field without too much computational overhead. However, by using a window-based approach, the Stratified Transformer focuses on an expanded local region rather than the global region. Moreover, the hyper-parameters such as the window size and number of distant points must be optimized for different datasets with varying densities. Consequently, super long-range features (e.g., line features) are not learned effectively because the local features aggregation module cannot exploit the global context. To address this issue, works [26], [27], [51] separately extract local features and global context by local and nonlocal aggregation, and then combine them in a serial and parallel manner. However, these approaches require enormous computations and are still unable to extract the inherent relationship between local features and global context. Motivated by these problems, we propose FPTransformer, to simultaneously exploit local features, global context, and their correlations.

III. Method

We revisit the three types of local point aggregations, i.e., local point-wise MLP, point convolution, and PT, and then derive a general formulation in Section III-A. Based on the general formulation, we present FuPConv in Section III-B and FPTransformer in Section III-C, followed by our down sampling block in Section III-D and finally, in Section III-E, we give our network details.

A. Rethinking Local Point Aggregation

Assume we have an unordered point cloud $p_i \in \mathbb{R}^{N\times 3}$ and its corresponding features $f_i \in \mathbb{R}^{N\times C}$. Here, p_i is a position vector, f_i is a feature vector and may contain additional attributes such as color or surface normal. N and C are the number of points and feature channels, respectively. We denote the K neighbors of p_i as $p_{ij} \in \mathbb{R}^{K\times 3}$ and their corresponding features are $f_{ij} \in \mathbb{R}^{K\times C}$.

1) Local Point-Wise MLP: The general formulation of local point-wise MLP on a point set p_i can be expressed as

$$\mathcal{G}_i = \Lambda_{j=1}^{K} \mathrm{MLP}(f_{ij}) \tag{1}$$

where MLP is a point-to-point mapping function that maps input features f_{ij} to output features $\mathcal{G}_i$. $\Lambda_{j=1}^{K}$ is a symmetric operation (SOP), e.g., maxpool. The input points have no

communication with each other. To establish communication, some works use a function to construct geometric connections between points to enhance point-wise features and then employ an MLP on these enriched point-wise features

$$\mathcal{G}_i = \Lambda_{j=1}^{K} \mathrm{MLP}\big(\mathcal{F}_1\big(p_{ij}, f_{ij}\big)\big). \tag{2}$$

Here, $\mathcal{F}_1(\cdot)$ is as a geometric construction function (e.g., hand-crafted geometric descriptor, graph filter, MLP). These functions do not largely increase the network parameters, however, their parameters still need to be optimized, which impacts their ability to generalize across different datasets with varying point densities or shapes.

2) Local Point Convolution: Point convolutions introduce the weight function to convolutions such that it learns weights from local point coordinates to dynamically adjust the point-wise features, expressed as

$$\mathcal{G}_i = \sum_{j=1}^{K} \mathcal{W}\big(p_{ij}\big) f_{ij} \tag{3}$$

where $\mathcal{W}(\cdot)$ is a weight function. Learning weights from the lowest level geometric information (i.e., point coordinates) may not lead to robust convolutional weights. Therefore, some works [15], [16], [21] associate coefficients with weight functions that further adjust the learned weights. Others [44], [45] propose geometric construction functions that incorporate additional low-level geometric information (e.g., Euclidean distance, position difference, and feature difference) to enrich the input to the weight functions so that they can learn more robust convolutional weights. These are, respectively, expressed as

$$\mathcal{G}_i = \sum_{j=1}^{K} \mathcal{C}\big(p_{ij}, f_{ij}\big)\mathcal{W}\big(p_{ij}\big) f_{ij} \tag{4}$$

$$\mathcal{G}_i = \sum_{j=1}^{K} \mathcal{W}\big(\mathcal{F}_2\big(p_{ij}, f_{ij}\big)\big) f_{ij}. \tag{5}$$

Here $\mathcal{C}(\cdot)$ is the coefficient (also derived from point coordinates or features) of the weight function, and $\mathcal{F}_2(\cdot)$ is defined as a geometric construction function to construct the rich low-level geometric information, such as distance, coordinate difference, and feature difference.

The resolution of the point cloud decreases at the deeper encoder layers in a hierarchical network leading to the distortion of low-level geometric connections derived from sparse points. Learning convolutional weights from such distorted geometric information can lead to nonrobust weights.

3) Local PT: Local PT solves the above problems and can be expressed as

$$\mathcal{G}_i = \sum_{j=1}^{K} \mathcal{W}\big(\mathcal{F}_2\big(f_{ij}\big) + \delta\big(p_{ij}\big)\big)\big(\mathcal{F}_1\big(f_{ij}\big) + \delta\big(p_{ij}\big)\big) \tag{6}$$

where $+$ is the addition operation, $\delta(\cdot)$ is the LPE that maps the point position in the local coordinate system from low-dimension (3-D) space to high-dimension space. This makes each local point well aware of its position in the local shape. As illustrated in Fig. 2, the design of local feature aggregations share the similar idea that they utilize the robust weights/attention stream to adjust the feature stream. In general, the weight/attention learned from richer local geometric correlations makes the aggregated local features more robust. Observing the above formulations of the three classical local feature aggregation methods, we introduce a *general formulation* for local feature aggregation as

$$\mathcal{G}_i = \sum_{j=1}^{K} \mathcal{C}\big(p_{ij}, f_{ij}\big)\mathcal{W}\big(\mathcal{F}_2\big(p_{ij}, f_{ij}\big) + \delta\big(p_{ij}\big)\big) \times \big(\mathcal{F}_1\big(p_{ij}, f_{ij}\big) + \delta\big(p_{ij}\big)\big). \tag{7}$$

Although existing local aggregation methods get promising performance on local point structures, they have two limitations, 1) they pay little to no attention to the global geometric structure and 2) they completely ignore the internal connections between global and local structures.

B. Full Point Convolution

To exploit local features as well as the global context features along with their correlations, the FuPConv is expressed as

$$\mathcal{G}(i) = \sum_{j=1}^{K} \mathcal{W}_c\big(\mathcal{S}_2\big(p_i, p_{ij}, \mathcal{S}_1\big(p_i, p_{\mathrm{in}}\big)\big)\big) f_{ij} \tag{8}$$

where S_1 and S_2 denote the global and local correlation functions. $\mathcal{W}_c$ is an adaptive weight function that learns the convolutional weights from local and global geometric connections. p_{in} is the global neighborhood points of p_i.

1) Global Correlation Function: The goal of the global correlation function is to construct the geometric connection between global and local points, enriching the local points with global information. We define the global correlation function as

$$\mathcal{S}_1(\cdot) = \sum_{n=1}^{N} \mathcal{R}\big(p_i, p_{\mathrm{in}}\big) \tag{9}$$

where $\mathcal{R}(\cdot)$ is the relation between p_{in} and each point p_i, which should be higher when p_{in} is closer to p_i. Inspired by [18], we propose global linear relation function

$$\mathcal{R}(\cdot) = \max\left(0, 1 - \frac{\|p_i - p_{\mathrm{in}}\|}{\sigma}\right) \tag{10}$$

where $\|\cdot\|$ is the Euclidean distance between global points and local points. σ is the influence coefficient that controls the influence of global points to each point. We set the correlation of global point to center point as $\mathcal{S}_{1i}$ and its corresponding global correlation to neighborhood points as $\mathcal{S}_{1ij}$.

2) Local Correlation Function: Learning convolutional weights highly depends on the local geometric connections. Therefore, we construct sufficient geometric connections using a local connection function. We define the local correlation function as

$$\mathcal{S}_2(\cdot) = p_{ij} + \big(p_{ij} - p_i\big) + \big\|p_{ij} - p_i\big\| + \big(\mathcal{S}_{1i} - \mathcal{S}_{1ij}\big) \tag{11}$$

where p_i is the center point position, $p_{ij} - p_i$ is the position difference, $\|p_j - p_i\|$ is the 3-D Euclidean distance and

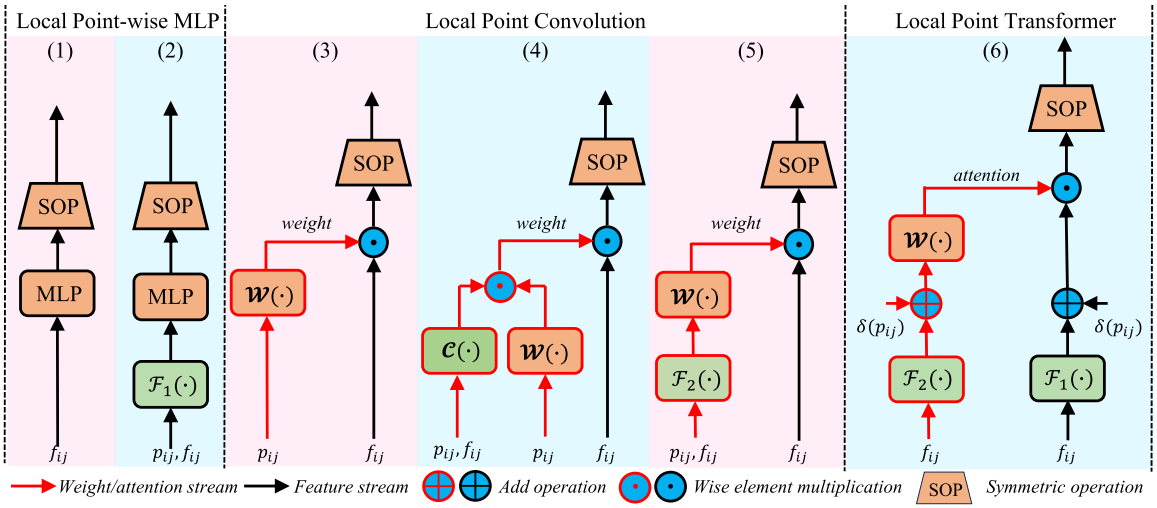

Fig. 2. Architecture of three local feature aggregations. The local connections (i.e., convolution weights/attention weights) learned by weight/attention function $\mathcal{W}(\cdot)$ adjust the features adaptively, improving the local shape awareness of feature aggregations.

$+$ denotes concatenation. Note that there is no learnable parameter in local and global correlation functions, but they require point distances. These distances are inherited from the farthest point sampling (FPS) in the same encoding layer.

3) Efficient Weight Function: The goal of adaptive weight function $\mathcal{W}_c(\cdot)$ is to learn the kernel weights. The output of the adaptive weight function is

$$\mathcal{W}_c = \rho_c\big(\phi_c\big(\mathcal{S}_2\big(p_i, p_{ij}, \mathcal{S}_1(p_i, p_{\text{in}})\big)\big)\big) \tag{12}$$

where ϕ_c is implemented with MLPs: $\mathbb{R}^{N\times K\times 8} \rightarrow \mathbb{R}^{N\times K\times C\times C}$ and ρ_c indicates softmax normalization. Learning a mount of weights from limited geometric connections $\in \mathbb{R}^{K\times 8}$ is inefficient. To this end, we formulate an efficient FuPConv based on the following lemma.

Lemma: FuPConv is equivalent to the following formulation: $\mathcal{G} = \text{Max}(\mathcal{T}_{c2} \otimes \text{Conv}_{1\times 1}(\mathcal{T}_{c1}, \mathcal{F}_c))$, where $\text{Conv}_{1\times 1}$ is 1×1 convolution, $\otimes$ is matrix multiplication and Max is maxpooling operation, $\mathcal{T}_{c1} \in \mathbb{R}^{C\times C_m\times C}$ is the kernels of 1×1 convolution, and $\mathcal{T}_{c2} \in \mathbb{R}^{K\times C_m}$ is the weight matrix learned by adaptive weight function.

Proof: To better understand the FuPConv reformulation, set attention weight matrix $\mathcal{W}_c \in \mathbb{R}^{N\times K\times C}$ as $\mathcal{W}_c(i, j, c)|i = 1, \ldots, N, j = 1, \ldots, K, c = 1, \ldots, C$, and set $f_{ij} \in \mathbb{R}^{N\times K\times C}$ as $\mathcal{F}_c(i, j, c)|i = 1, \ldots, N, j = 1, \ldots, K, c = 1, \ldots, C$, where i, j, and c are the index of the global, neighbor points, and input feature channels. According to (8), FuPConv can be expressed as

$$\mathcal{G}(i) = \sum_{j=1}^{K}\sum_{c=1}^{C} \mathcal{W}_c(i, j, c)\mathcal{F}_c(i, j, c). \tag{13}$$

Since the weight function is approximated by MLPs implemented as 1×1 convolutions, the weight matrix generated by the weight function can be expressed as

$$\mathcal{W}_c(i, j, c_{\text{in}}) = \sum_{c_m=1}^{C_m} \mathcal{T}_{c2}(i, j, c_m)\mathcal{T}_{c1}^{\text{T}}(i, c, c_m) \tag{14}$$

where c_{mid} and C_{mid} are the index and number of output channels of the middle layer. Substituting (14) into (13), we get

$$\begin{aligned}\mathcal{G}(i) &= \sum_{j=1}^{K}\sum_{c=1}^{C} \mathcal{W}_c(i, j, c)\mathcal{F}_c(i, j, c)\\ &= \sum_{j=1}^{K}\sum_{c=1}^{C}\left(\sum_{c_{\text{mid}}=1}^{C_m} \mathcal{T}_{c2}(i, j, c_m)\mathcal{T}_{c1}^{\text{T}}(i, c, c_m)\right)\mathcal{F}_c(i, j, c)\\ &= \sum_{j=1}^{K}\sum_{c_m=1}^{C_m} \mathcal{T}_{c2}(i, j, c_m)\sum_{c=1}^{C}\big(\mathcal{T}_{c1}^{\text{T}}(c, c_{\text{mid}})\mathcal{F}_c(i, j, c)\big)\\ &= \text{Sum}(\mathcal{T}_{c2} \otimes \text{Conv}_{1\times 1}(\mathcal{T}_{c1}, \mathcal{F}_c)).\end{aligned} \tag{15}$$

According to the above reformulation, FuPConv comprises two operations including one 1×1 convolution and one matrix multiplication. Using this formulation, we divide the kernel weight matrix $\mathcal{W}_c \in \mathbb{R}^{N\times K\times C\times C}$ into two parts: 1×1 convolution weight matrix $\mathcal{T}_{c1} \in \mathbb{R}^{C\times(C_m\times C)}$ and weight matrix $\mathcal{T}_{c2} \in \mathbb{R}^{N\times K\times C_m}$, where the $\mathcal{T}_{c1}$ are learned in data-driven manner, and $\mathcal{T}_{c2}$ are efficiently learned though the adaptive weight function according to the local and global geometric connections. Here, the adaptive weight function can be defined as

$$\mathcal{T}_2 = \rho_{\text{ce}}\big(\phi_{\text{ce}}\big(\mathcal{S}_2\big(p_i, p_{ij}, \mathcal{S}_1(p_i, p_{\text{in}})\big)\big)\big) \tag{16}$$

where ϕ_{ce} is a nonlinear function implemented with MLPs: $\mathbb{R}^{N\times K\times 8} \rightarrow \mathbb{R}^{N\times K\times C_m}$. ρ_{ce} indicates softmax normalization. The weight function transfers $K \times 8$ dimension geometric connection information to $N \times K \times C_m$ dimension weight matrix, where C_m is smaller than C. With the above reformulation, convolution can learn more robust weights from limited geometric connections. Our reformulated FuPConv is shown in Fig. 3(a). Compared to the simple point convolution [see Fig. 2(3)], the FuPConv differs in two key aspects: 1) the simple point convolution learns the convolutional weights from the local geometric connections, while the FuPConv learns the weights from the rich geometric connections constructed by global and local correlation function. 2) FuPConv reformulates the convolution operation into two operations including one

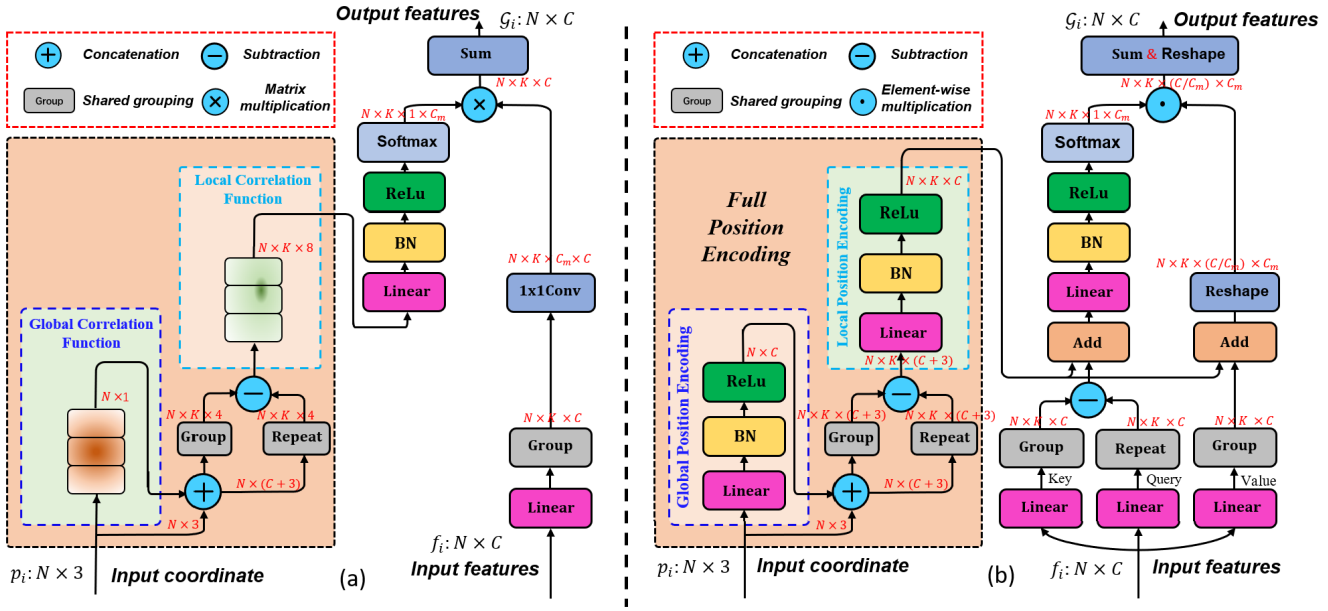

Fig. 3. (a) Proposed FuPConv block which takes the $N \times 3$ input point coordinates and $N \times C$ features from the previous layer to output $N \times C$ features. FuPConv incorporates full point correlation (global, local, and global–local correlation) into each point. (b) Proposed FPTransformer block which takes the $N \times 3$ input point coordinates and $N \times C$ features from the previous layer to output $N \times C$ features. FPTransformer incorporates FPE into each point.

1×1 convolution and one matrix multiplication, learning more robust weights.

C. Full PT

We also bring this idea into PT, we propose an FPTransformer that exploits local features as well as the global context and their internal correlations. Give a point (p_i, f_i), we use three linear projections to project the point features f_i to the query q_i, key k_i, and value feature vectors v_i, expressed as

$$q_i = W_q f_i, \quad k_i = W_k f_i, \quad v_i = W_v f_i \tag{17}$$

where W_q, W_k, and W_v ($\mathbb{R}^{N\times C} \rightarrow \mathbb{R}^{N\times C}$) are projection functions implemented as linear layers. The FPTransformer applied on the point (p_i, f_i) and its corresponding point set (p_{ij}, f_{ij}) can be formulated as

$$\mathcal{G}_i = \sum_{j=1}^{K} \mathcal{W}_a\left(\left(q_i - k_{ij}\right) + \delta^a_{\text{full}}\right)\left(v_{ij} + \delta^a_{\text{full}}\right). \tag{18}$$

Here, $\sum$ refers to an aggregation function such as summation, δ^a_{full} is the full position encoding (FPE) for FPTransformer, and $\mathcal{W}_a(\cdot)$ is the attention function (i.e., weight function) that learns the attention map from the geometric connection between key point and query point.

1) Full Position Encoding: Point features are derived from their coordinates and already contain position information, however, this information may get diluted in deep aggregation layers. Therefore, we add fine-grained position information to features in each aggregation layer. Another limitation of existing position encoding methods is that they tend to map relative positions in the local coordinate system from low dimensional (3-D) space to higher dimensions. This enhances the awareness of local geometric connections but does not encode the global context. We propose full point position encoding that includes global and LPE. We define global position encoding (GPE) on point $p_i \in \mathbb{R}^{N\times 3}$ as

$$\delta^a_{\text{global},i} = \left[\phi^a_{\text{global}}(p_i), p_i\right] \tag{19}$$

the GPE function ϕ^a_{global} is an MLP: $\mathbb{R}^{N\times 3} \rightarrow \mathbb{R}^{N\times C}$. $[,]$ is concatenation operation. The output of GPE is $\delta^a_{\text{global},i} \in \mathbb{R}^{N\times(C+3)}$. We denote the neighborhood point position of $\delta^a_{\text{global},i}$ as $\delta^a_{\text{global},ij} | j = 1, 2, \ldots, K$, where K is the number of neighborhood points. The FPE on local points $\delta^a_{\text{global},ij} \in \mathbb{R}^{N\times K\times(C+3)}$ is defined as

$$\delta^a_{\text{full}} = \phi^a_{\text{local}}\left(\delta^a_{\text{global},i} - \delta^a_{\text{global},ij}\right) \tag{20}$$

where the LPE function ϕ^a_{local} is an MLP: $\mathbb{R}^{N\times K\times(C+3)} \rightarrow \mathbb{R}^{N\times K\times C}$. The output of FPE is $\delta^a_{\text{full}} \in \mathbb{R}^{N\times K\times C}$. With this hierarchical position encoding, each point perceives high-dimensional position information in its global and local receptive fields.

2) Efficient Attention Function: The attention function plays an important role in our FPTransformer by encoding the geometric connections (i.e., features difference) between query and key points into the attention map. The attention function is formulated as

$$\mathcal{W}_a(\cdot) = \rho_a\left(\phi_a\left(\left(q_i - k_{ij}\right) + \delta^a_{\text{full}}\right)\right) \tag{21}$$

where ϕ_a is an MLP ($\mathbb{R}^{N\times K\times C} \rightarrow \mathbb{R}^{N\times K\times C}$) and ρ_a is the softmax normalization to keep the attention weights in the range (0, 1). The attention function transfers $N \times K \times C$ geometric connection information to an attention weight matrix $\mathcal{W}_a$ of the same dimension. Since, the generation of attention weight matrix $\mathcal{W}_a \in \mathbb{R}^{N\times K\times C}$ requires large memory, we formulate an efficient attention layer as per the following lemma.

Lemma: FPTransformer is equivalent to the following reformulation: $\mathcal{G}_i = \text{Sum}(\mathcal{T}_a \odot \mathcal{F}_a)$, where $\mathcal{T}_a \in \mathbb{R}^{N\times K\times 1\times C_m}$

are the attention weights obtained from the attention function, $\mathcal{F}_a \in \mathbb{R}^{N\times K\times C/C_m\times C_m}$ is the reshaped feature. $\odot$ is the element-wise multiplication and Sum is the summation.

Proof: To better understand the FPTransformer reformulation, set attention weight matrix $\mathcal{W}_a \in \mathbb{R}^{N\times K\times C}$ as $\mathcal{W}_a(i,j,c)|i = 1,\ldots,N, j = 1,\ldots,K, c = 1,\ldots,C$, and set $(v_{ij}+\delta^a_{\text{full}}) \in \mathbb{R}^{N\times K\times C}$ as $\mathcal{V}(i,j,c)|i = 1,\ldots,N$, $j = 1,\ldots,K, c = 1,\ldots,C$, where i, j, and c are the index of the global, neighbor points, and input feature channels. As per to (18), FPTransformer can be expressed as

$$\begin{aligned}\mathcal{G}_i &= \sum_{j=1}^{K}\sum_{c=1}^{C}\mathcal{W}_a(i,j,c)\odot\mathcal{V}(i,j,c)\\ &= \sum_{j=1}^{K}\sum_{c/c_m\times c_m=1}^{C/C_m\times C_m}\mathcal{W}_a(i,j,c/c_m\times c_m)\odot\mathcal{V}(i,j,c/c_m\times c_m)\\ &= \sum_{j=1}^{K}\sum_{c/c_m=1}^{C/C_m}\sum_{c_m=1}^{C_m}\mathcal{W}_a(i,j,c/c_m,c_m)\odot\mathcal{V}(i,j,c/c_m,c_m).\end{aligned} \tag{22}$$

Here, C_m is the number of middle channels ($C_m < C$), and c_m is the index of the middle channel. We set $\{\mathcal{W}_a(i,j,1,c_m)|i = 1,\ldots,N, j = 1,\ldots,K, c_m = 1,\ldots,C_m\} \in \mathbb{R}^{C/C_m}$ as a vector from attention weight matrix $\mathcal{W}_a \in \mathbb{R}^{N\times K\times C/C_m\times C_m}$, and make these C/C_m number of vectors share the same attention weights. Hence, (22) can be expressed as

$$\begin{aligned}\mathcal{G}_i &= \sum_{j=1}^{K}\sum_{c/c_m=1}^{C/C_m}\sum_{c_m=1}^{C_m}\mathcal{W}_a(i,j,c/c_m,c_m)\odot\mathcal{V}(i,j,c/c_m,c_m)\\ &= \sum_{j=1}^{K}\sum_{c/c_m=1}^{C/C_m}\sum_{c_m=1}^{C_m}\mathcal{W}_a(i,j,1,c_m)\odot\mathcal{V}(i,j,c/c_m,c_m)\\ &= \text{Sum}(\mathcal{T}_a\odot\mathcal{F}_a).\end{aligned} \tag{23}$$

Here, $\mathcal{F}_a \in \mathbb{R}^{N\times K\times C/C_m\times C_m}$ is the reshaped feature. $\mathcal{T}_a \in \mathbb{R}^{N\times K\times 1\times C_m}$ are the attention weights obtained from the efficient attention function defined as

$$\mathcal{W}_a(\cdot) = \rho_{\text{ae}}\big(\phi_{\text{ae}}\big((q_i - k_{ij}) + \delta^a_{\text{full}}\big)\big) \tag{24}$$

where ϕ_{ae} is an MLP ($\mathbb{R}^{N\times K\times C} \to \mathbb{R}^{N\times K\times C_m}$) and ρ_{ae} is softmax normalization. The attention function transfers $N \times K \times C$ dimensional geometric connection information to a $N \times K \times C_m$ dimensional attention weight matrix. Similarly, using this reformulation, the transformer can learn more robust weights from limited geometric information. Details of C_m are further discussed in Section IV-E. Our reformulated FPTransformer is shown in Fig. 3(b). Compared to the simple PT [see Fig. 2(6)], the FPTransformer differs in two key aspects: 1) the simple PT with the LPE only has local geometric awareness. In contrast, the FPTransformer with the full point position encoding has local and global geometric awareness. 2) FPTransformer introduces the attention-sharing mechanism into vector attention, learning more robust weights.

D. Shape-Aware Downsampling (SADS) Block

The shape information of the subsampled point cloud is represented not only by the point positions but also by the point features. Classical farthest-point sampling can effectively maintain shape information at the point position level. However, the low-resolution subsampled point cloud may suffer from shape distortion at the position level, and may also face challenges in preserving shape details at the feature level.

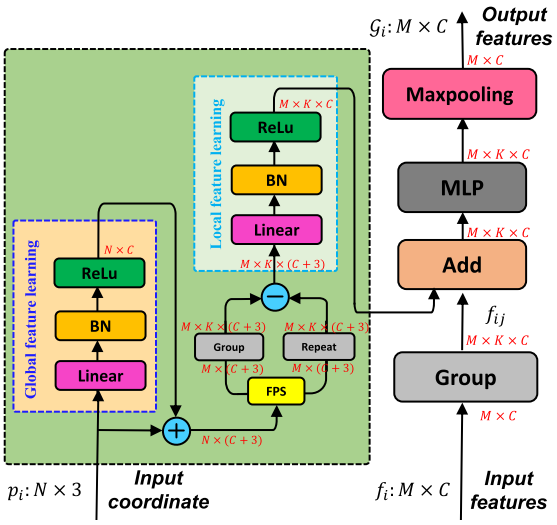

Fig. 4. Proposed SADS block. Grouping is done based on the current features to which the FPE is added for subsequent downsampling.

To overcome this problem, we propose a SADS block with learnable parameters (see Fig. 4). SADS incorporates a low-level feature learner to maximally preserve the low-level features (i.e., shape details) of the scene. Our downsampling block can be expressed as

$$\mathcal{G}_i = \max\big(\text{MLP}\big(f_{ij} + H\big)\big) \tag{25}$$

where $f_{ij} \in \mathbb{R}^{M\times K\times C}$ are the features from the grouping operation, M is the number of points sampled by FPS, H are the hierarchical features of points (see Fig. 4), and "max" is maxpooling. The hierarchical features basically include low-level global and local features. The global features on point $p_i \in \mathbb{R}^{N\times 3}$ are learned by

$$f^d_{\text{global},i} = \Big[\phi^d_{\text{global}}(p_i),\, p_i\Big] \tag{26}$$

where ϕ^d_{global} is an MLP: $\mathbb{R}^{N\times 3} \to \mathbb{R}^{N\times C}$. The low-level global features of point p_i are $f^d_{\text{global},i} \in \mathbb{R}^{N\times(C+3)}$. After sampling and grouping, we denote the neighborhood point position of sampled point $f^d_{\text{global},i} \in \mathbb{R}^{M\times(C+3)}$ as $f^d_{\text{global},ij} \in \mathbb{R}^{M\times K\times(C+3)}$. The hierarchical features H are defined as

$$H = \phi^d_{\text{local}}\Big(f^d_{\text{global},i} - f^d_{\text{global},ij}\Big) \tag{27}$$

where the LPE function is an MLP: $\mathbb{R}^{M\times K\times(C+3)} \to \mathbb{R}^{M\times K\times C}$. This way, simple but effective hierarchical MLPs are integrated into a conventional sampling block, to ensure that each subsampled point integrates the most prominent shape information at both the position and feature levels.

E. Network Architecture

For FuPConv, we choose the basic PointNet++ [11] as the backbone and replace the MLPs with FuPConv to form

TABLE I

SEMANTIC SEGMENTATION RESULTS ON S3DIS DATASET AREA-5. WE REPORT THE MIOU, MACC, AND OA. THE BEST RESULT IS IN **BOLD**, AND THE SECOND BEST IS UNDERLINED. "*" MEANS THAT THE MODEL USED TEST TIME AUGMENTATIONS

Year	Method	mIoU	mAcc	OA	ceil.	floor	wall	beam	col.	wind.	door	chair	table	book	sofa	board	clut.	Para.	FLOPs
2020 PAMI	SPH3D-GCN [52]	59.5	65.9	–	93.3	97.1	81.1	0.0	33.2	45.8	43.8	79.7	86.9	33.2	71.5	54.1	53.7	–	–
2020 CVPR	PointASNL [27]	62.6	68.5	87.7	94.3	98.4	79.1	0.0	26.7	55.2	66.2	86.8	83.3	68.3	47.6	56.4	52.1	22.4M	19.1G
2020 CVPR	SegGCN [21]	63.6	70.4	–	93.7	98.6	80.6	0.0	28.5	42.6	74.5	80.9	88.7	69.0	71.3	44.4	54.3	–	–
2021 CVPR	PAConv [44]	66.58	73.00	–	94.55	98.59	82.37	0.00	26.43	57.96	59.96	**89.73**	80.44	74.32	69.80	73.50	57.72	–	–
2021 CVPR	BAAF-Net [53]	65.4	73.1	88.9	92.9	97.9	82.3	0.0	23.1	**65.5**	64.9	87.5	78.5	70.7	61.4	68.7	57.2	5.0M	–
2022 CVPR	CBL [54]	69.4	75.2	90.6	93.9	98.4	84.2	0.0	37.0	57.7	71.9	81.8	91.7	75.6	**77.8**	69.1	**62.9**	18.6M	–
2022 CVPR	RepSurf-U [36]	68.9	76.0	90.2	–	–	–	–	–	–	–	–	–	–	–	–	–	1.0M	1.0G
2022 CVPR	Strat. Transformer* [50]	72.0	78.1	91.5	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
2022 ECCV	PointMixer [55]	71.4	77.4	–	94.2	98.2	86.0	0.0	43.8	62.1	**78.5**	82.2	90.8	79.8	73.9	78.5	59.4	–	–
2022 NIPS	PointNeXt [56]	71.1	77.2	91.0	94.2	98.5	84.4	0.0	37.7	59.3	74.0	91.6	83.1	77.2	77.4	78.8	60.6	41.6M	84.8G
2022 NIPS	PointTransformerV2* [47]	71.6	77.9	91.1	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
2023 CSVT	LCPFormer [57]	70.2	76.8	90.8	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
2023 TNNLS	PicassoNet++ [58]	71.0	77.2	91.3	94.4	98.4	87.5	0.0	46.9	63.7	75.5	81.4	90.3	71.3	76.2	76.7	61.1	–	–
2018 NIPS	PointNet++ [11]	53.4	62.9	–	89.1	98.1	73.7	0.00	3.0	58.2	21.1	67.0	78.6	44.6	60.8	56.5	43.2	1.0M	7.2G
	FuPConv(ours)	68.4(+15)	75.0(+12.1)	–	93.18	98.54	85.18	0.00	31.11	58.61	77.24	82.58	89.24	67.86	73.45	74.95	56.89	2.5M	6.5G
2021 ICCV	Point Transformer [28]	70.4	76.5	90.8	94.0	98.5	86.3	0.0	38.0	63.4	74.3	82.4	89.1	80.2	74.3	76.0	59.3	7.8M	5.6G
	FPTransformer(ours)	72.2(+1.8)	78.5(+2.0)	91.5(+0.7)	94.6	**98.7**	87.2	0.0	44.5	64.8	77.0	82.9	92.3	79.3	74.6	81.2	61.8	10.9M	8.3G
	FPTransformer*(ours)	**73.1**(+2.7)	**78.8** (+2.3)	**91.7**(+0.9)	**95.0**	**98.7**	**87.8**	0.0	**47.4**	64.9	75.7	83.9	**92.7**	**85.6**	74.6	**82.0**	61.8	10.9M	8.3G

TABLE II

SEMANTIC SEGMENTATION RESULTS ON S3DIS WITH SIXFOLD CROSS VALIDATION

Year	Method	mIoU	mAcc	OA	ceil.	floor	wall	beam	col.	wind.	door	chair	table	book	sofa	board	clut.
2019 CVPR	PointWeb [38]	66.7	76.2	87.3	93.5	94.2	80.8	52.4	41.3	64.9	68.1	67.1	71.4	62.7	50.3	62.2	58.5
2019 ICCV	KPConv [18]	70.6	79.1	–	93.6	92.4	83.1	63.9	54.3	66.1	76.6	64.0	57.8	74.9	69.3	61.3	60.3
2020 PAMI	SPH3D-GCN [52]	68.9	77.9	88.6	93.3	96.2	81.9	58.6	55.9	55.9	71.7	82.4	72.1	64.5	48.5	54.8	60.4
2020 CVPR	PointASNL [27]	68.7	79.0	88.8	95.3	97.9	81.9	47.0	48.0	67.3	70.5	77.8	71.3	60.4	50.7	63.0	62.8
2021 CVPR	PAConv [44]	69.31	78.65	–	94.30	93.46	82.80	56.88	45.74	65.21	74.90	59.74	74.60	67.41	61.78	65.79	58.36
2021 CVPR	SCF-Net [59]	71.6	82.7	88.4	93.3	96.4	80.9	**64.9**	47.4	64.5	70.1	81.6	71.4	64.4	67.2	67.5	60.9
2021 CVPR	BAAF-Net [53]	72.2	83.1	88.9	93.3	96.8	81.6	61.9	49.5	65.4	73.3	**83.7**	72.0	64.3	67.5	67.0	62.4
2022 NIPS	PointNeXt [56]	74.9	83.0	90.3	–	–	–	–	–	–	–	–	–	–	–	–	–
2022 CVPR	RepSurf-U [36]	74.3	82.6	90.8	–	–	–	–	–	–	–	–	–	–	–	–	–
2022 CVPR	CBL [54]	73.1	79.4	89.6	94.1	94.2	85.5	50.4	58.8	**70.3**	78.3	75.0	75.7	74.0	**71.8**	60.0	62.4
2021 ICCV	Point Transformer [28]	73.5	81.9	90.2	94.3	97.5	84.7	55.6	58.1	66.1	78.2	74.1	77.6	71.2	67.3	65.7	64.8
	FPTransformer(ours)	76.0(+2.5)	84.0(+2.1)	91.0(+0.8)	95.0	97.6	86.0	59.7	60.2	68.3	**78.6**	75.2	84.3	75.2	69.6	71.5	67.0
	FPTransformer*(ours)	**76.8**(+3.3)	**84.4**(+2.5)	**91.5**(+1.3)	**95.2**	**97.7**	**86.4**	60.1	**62.0**	68.8	78.4	76.2	**85.0**	**77.5**	70.2	**72.2**	**67.5**

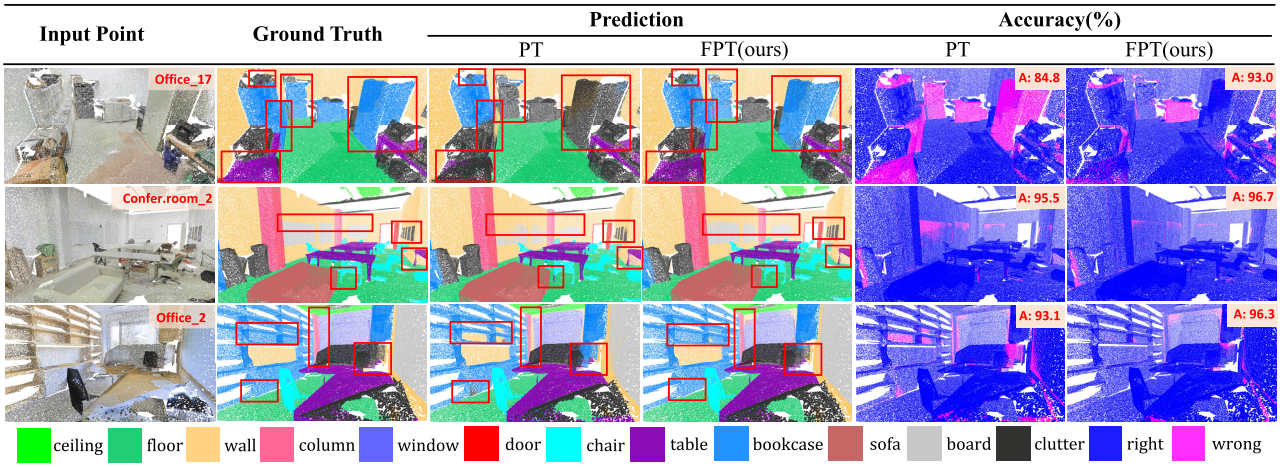

Fig. 5. Visualization of semantic segmentation results on S3DIS Area-5 without test time augmentations. The red boxes highlight areas in the scenes where our proposed FPTransformer performs particularly better than the PT.

the new network architectures for three tasks, i.e., semantic segmentation, classification, and normal estimation.

For FPTransformer and SADS block, we use a U-Net-like architecture with five encoding and decoding layers and skip connections, covering semantic segmentation, 3-D detection, and classification. The first encoding and decoding layers consist of one MLP and FPTransformer. Other encoding layers contain one SADS block and several FPTransformer blocks (details in Section IV). We set the feature dimensions C as [32, 64, 128, 256, 512] and the middle feature dimension C_m as [4, 8, 16, 32, 64] for the five encoding and decoding layers. The ratio of up/down sampling is set as [1, 4, 4, 4, 4] for both encoder/decoder. Decoding layers, besides the first, contain one upsampling block and one FPTransformer. For segmentation, we add an MLP at the end to predict the final point-wise labels. For detection, we use the network as a 3-D backbone (excluding the prediction layer). We adopt the shared grouping that uses the same neighbor indices in the proposed modules (i.e., FuPConv, FPTransformer, and SADS), to improve the efficiency of the proposed full point encoding.

IV. EXPERIMENTAL RESULTS

We evaluate our network on semantic segmentation, 3-D detection, shape classification, and normal estimation tasks and perform detailed ablation studies to demonstrate the

effectiveness and robustness of the proposed FuPConv, FPTransformer, and SADS.

A. Semantic Segmentation

1) Datasets: We evaluate our network on two large-scale indoor scene datasets, S3DIS and ScanNet. S3DIS [60] contains colored point clouds annotated point-wise with 13 classes. It covers 271 rooms from six large-scale indoor scenes (a total of 6020 m^2). We conduct sixfold cross-validation on S3DIS and, in line with other works, conduct more extensive comparisons on Area 5 as the test set, which is not in the same building as the other areas. ScanNet [61] contains colored point clouds of indoor scenes with point-wise semantic labels of 20 object categories. It is split into 1201 scenes for training and 312 for validation. For the FuPConv network, we adopt a data preprocessing strategy similar to PointNet++ and divide the entire point cloud scene into several blocks to reduce the number of input points. Each block contains 4096 points only. Similarly, for the FPTransformer network, we employ a voxel-based downsampling similar to PT [28], to decrease the overall number of points in the point cloud. In addition, we heavily utilize shared grouping operations across our proposed FuPConv, FPTransformer, and SADS modules, which share a set of indices obtained through the K-nearest neighbor (KNN) search.

2) Network Configuration: For semantic segmentation on S3DIS, we set the FPTransformer block in the five encoding layers depths [1, 2, 2, 6, 2]. We adopt the SGD optimizer and weight decay as 0.9 and 0.0001. The base learning rate is set as 0.5 and the learning rate is scheduled by the MultiStepLR every 30 epochs. We train and test the model with batch size 16 on 4 GPUs and batch size 4 on a single GPU, respectively. We adopt downsampling, scaling, contrast, translation, jitter, and chromatic translation to preprocess training data. We set the voxel size as 4 cm and the maximum number of voxels to 80 000. On ScanNet, we set the FPTransformer block in the five encoding layer depths as [1, 3, 3, 9, 3]. The base learning rate is set as 0.1. We use downsampling, rotation, flip, scaling, and jitter to preprocess training data. We set the voxel size as 2 cm and the maximum number of voxels as 120 000.

3) Results: We compare our method with the recent state-of-the-art on three metrics, i.e., mean class-wise intersection over union (mIoU), mean class-wise accuracy (mAcc), and network parameters (Para.). Table I provides detailed results on S3DIS Area-5. Our network equipped with FPTransformer and SADS achieves the best performance 73.1%, 78.8%, and 91.7% in terms of mIoU, mAcc, and OA, respectively. It also achieves competitive results (top 2) on 9 out of the 13 categories including *ceiling, floor, wall, column, window, door, table, board, and clutter*. Moreover, compared to the classical vector attention-based method (i.e., PT), the performance of our method exceeds it by a large margin on some long-range shape classes such as *column, door, table, and board*. Compared to the previous state-of-the-art vector attention-based method (e.g., PT and PT V2), window attention-based method (e.g., stratified transformer), and point-wise MLP-based method (e.g., PointNeXt), our FPTransformer gets significant improvements on all three metrics, i.e., mIoU, mAcc, and OA. Besides, our FuPConv achieves the best performance among classical Convolution-based methods such as KPConv and PAConv and exceeds the baseline PointNet++ 15% in the term of mIoU.

TABLE III
SEMANTIC SEGMENTATION RESULTS (MIOU) ON SCANNET

Year	Method	Input	Val(%)	Test(%)
2020 CVPR	PointASNL [27]	point	63.5	66.6
2019 ICCV	MVPNet [63]	point	66.4	–
2019 ICCV	KPConv [18]	point	69.2	68.6
2019 3DV	JointPointBased [64]	point	69.2	63.4
2022 CVPR	RepSurf-U [36]	point	70.0	–
2022 CVPR	Stratified Transformer* [50]	point	74.3	73.7
2021 CVPR	BPNet [62]	point+image	72.5	74.9
2019 CVPR	MinkowskiNet [48]	voxel	72.2	73.6
2022 CVPR	FastPointTransformer [49]	voxel	72.0	–
2022 NIPS	PointTransformerV2* [47]	point	75.4	75.2
2023 TNNLS	PicassoNet++ [58]	mesh	–	69.2
2021 CVPR	PointTransformer [28]	point	70.6	–
	FPTransformer* (ours)	point	**75.6**(+5.0)	**75.5**

We report results in Table II for the sixfold validation setting on S3DIS dataset. Our method performs the best, achieving state-of-the-art results on all three metrics, i.e., 76.8% mIoU, 84.4% mAcc, and 91.5% OA. Notably, our method achieves the best performance on 8 out of the 13 categories and achieves the second-best performance on another three categories. Fig. 5 shows visualizations of our results on S3DIS Area 5 without test time augmentations in comparison to PT. We can see that our method is more robust to long-range shapes such as bookcases and boards.

Table III shows semantic segmentation results on ScanNet validation set as well as test set, and compares the proposed FPTransformer to existing state-of-the-art. Compared to PT (which also uses vector attention), our method gets significant improvement of +5.0 in mIoU (see Table III last row). Our method even outperforms the multimodal BPNet [62]. Compared to the Stratified Transformer, our method gets +1.3% and +1.8% higher mIoU on the validation and test sets, respectively. Compared to the PT V2, our method gets better performance (+0.2%, +0.3% mIoU) on validation and test, respectively.

B. Three-Dimensional Object Detection

We conduct experiments for 3-D object detection to show the generalization ability of the FPTransformer.

1) Datasets: We conduct experiments on two popular datasets: ScanNetV2 [61] and KITTI [67]. ScanNetV2 consists of 1513 indoor scenes and 18 object classes. On this dataset, we adopt the mean Average Precision (mAP) metric under the threshold of 0.25 (mAP@0.25) and 0.5 (mAP@0.5) without considering the bounding box orientations. KITTI is a large-scale outdoor dataset captured with a LiDAR sensor. It has 7518 tests and 7481 training samples (further divided into train-validation splits). We calculate mAP for easy, moderate, and hard cases, at 11 and 40 recall positions following the official KITTI protocol.

2) Network Configuration: For ScanNetV2, we select two detection architectures: VoteNet [74] and Group-free-3D [75]. For KITTI, we choose PointRCNN [65] and PVRCNN [66].

TABLE IV

THREE-DIMENSIONAL OBJECT DETECTION RESULTS ON THE KITTI VALIDATION SET. WE REPORT MAP AT 11 AND 40 RECALL POSITIONS, SIMILAR TO PRIOR WORKS [65], [66]

Methods	3D Backbone	Car(%)			Pedestrian(%)			Cyclist(%)			Recall
		Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard	
PointRCNN [65]	PN++(MSG)	88.72	78.55	77.72	62.64	55.06	50.83	**85.68**	**71.22**	**65.63**	R@11
PointRCNN [65]	PT	88.89	78.62	77.94	65.53	58.24	55.76	83.95	69.45	64.97	
PointRCNN(ours)	**FPTransformer**	**89.05**(+0.33)	**78.89**(+0.44)	**78.44**(+0.72)	**68.78**(+6.14)	**61.45**(+6.39)	**56.57**(+5.74)	84.10(-1.58)	70.15(-1.07)	65.55(-0.08)	
PV-RCNN [66]	Spconv + MLP	89.23	83.23	78.81	67.29	61.01	56.40	87.16	73.45	68.98	
PV-RCNN [66]	Spconv + PT	89.10	82.98	77.76	67.75	61.20	56.43	86.64	72.54	69.11	
PV-RCNN (ours)	Spconv + **FPTrans.**	**89.43**(+0.2)	**83.60**(+0.37)	**78.84**(+0.03)	**68.21**(+0.92)	**61.45**(+0.44)	**56.62**(+0.22)	86.59(-0.57)	72.32(-1.13)	**69.27**(+0.29)	
PointRCNN [65]	PN++(MSG)	91.28	80.47	78.02	62.80	55.41	48.83	90.58	70.98	66.65	R@40
PointRCNN [65]	PT	91.10	80.23	77.96	65.74	58.65	53.45	90.68	71.04	66.84	
PointRCNN (ours)	**FPTransformer**	**91.30**(+0.02)	**81.87**(+1.40)	**80.14**(+2.12)	**68.55**(+5.75)	**61.98**(+6.57)	**56.17**(+7.34)	**91.76**(+1.18)	**71.45**(+0.57)	**67.24**(+0.59)	
PV-RCNN [66]	Spconv + MLP	92.00	84.38	82.42	68.19	60.55	55.55	90.39	70.38	65.97	
PV-RCNN [66]	Spconv + PT	92.04	84.45	82.43	68.23	60.76	56.23	91.45	71.23	67.56	
PV-RCNN (ours)	Spconv + **FPTrans.**	**92.15**(+0.15)	**84.74**(+0.36)	**82.62**(+0.20)	**68.24**(+0.05)	**61.10**(+0.55)	**57.32**(+1.77)	**92.71**(+2.32)	**72.50**(+2.12)	**69.26**(+3.29)	

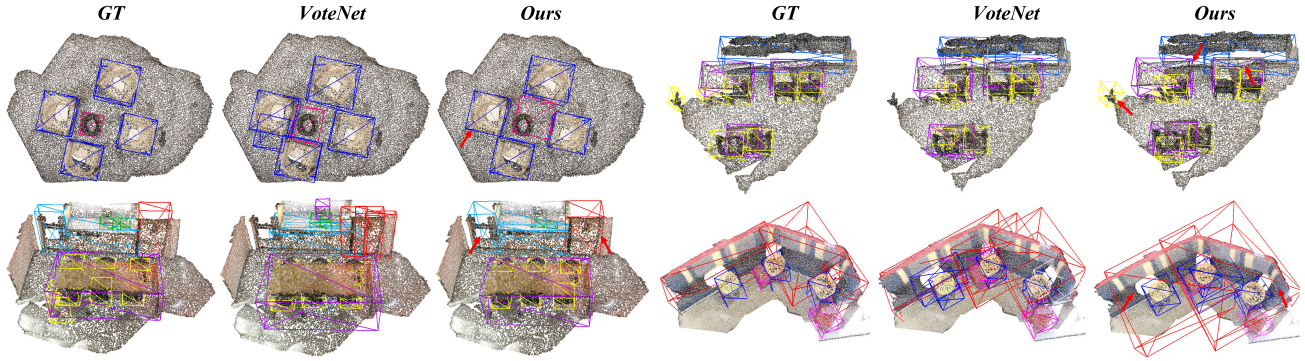

Fig. 6. Visualization of detection results on ScanNetV2. The red arrows indicate objects in the scenes where the VoteNet equipped with FPTransformer performs better than the original VoteNet.

TABLE V

DETECTION RESULTS ON SCANNETV2 VALIDATION SET FROM THE BENCHMARK WEBSITE. WE REPORT MAP AT THRESHOLDS OF 0.25 AND 0.5 IOU. "**" DENOTES MODEL REPRODUCED BY [68]

Year	Method	3D Backbone	mAP0.25	mAP0.5
2020 ECCV	H3DNet [69]	4×PointNet++	67.2	48.1
2021 ICCV	3DETR [70]	Transformer	65.0	47.0
2021 CVPR	BRNet [71]	4×PointNet++	66.1	50.9
2022 CVPR	TokenFusion [72]	2×PointNet++	70.8	54.2
2022 CVPR	RBGNet [73]	PointNet++	70.6	**55.2**
2022 CVPR	RepSurf-U [36]	PointNet++	71.2	54.8
2019 CVPR	VoteNet [74]	PointNet++	58.6	33.5
	VoteNet (ours)	**FPTransformer**	62.3(+3.7)	38.6(+5.5)
2019 CVPR	VoteNet** [74]	PointNet++	62.9	39.9
	VoteNet** (ours)	**FPTransformer**	65.2(+2.3)	45.1(+5.2)
2021 CVPR	Group-Free-3D(6,256) [75]	PointNet++	67.3	48.9
	Group-Free-3D(6,256) (ours)	**FPTransformer**	69.2(+1.9)	50.7(+1.8)
2021 CVPR	Group-Free-3D(12,512) [75]	2× PointNet++	69.1	52.8
	Group-Free-3D(12,512)(ours)	**FPTransformer**	**71.5**(+2.4)	54.3(+1.5)

VoteNet, Group-Free-3D, and PointRCNN have a similar network architecture. They use a 3-D backbone to extract point features for subsequent object bounding box prediction. We replace their 3-D backbones with our network. PVRCNN integrates voxel and point features, where the voxel features are learned by 3-D sparse convolution using multiple encoding layers and summarized into a small set of key points via the voxel set abstraction module. We replace the abstraction module with our FPTransformer. During training, we keep the same parameters of the original PointRCNN and PVRCNN networks. However, for VoteNet, we change the number of input points to 40 960 and epochs to 260. For Group-Free-3D, we only adopt the point coordinates as input features.

3) Results: Table IV shows results on the KITTI dataset. Our method consistently improves the performance of both detectors for Recall@40 and improves 13 out of 18 cases for Recall@11. PointRCNN with FPTransformer obtains remarkable improvements of 5.75%, 6.57%, and 7.34% (Recall@40) on the easy, moderate, and hard cases of the "Pedestrian" class. PVRCNN with FPTransformer obtains significant improvements of 1.77% and 3.29% for the hard cases of "Pedestrian," and "Cyclist."

As shown in Table V, when FPTransformer is plugged into the VoteNet and Group-Free-3D networks, the detection performance improves significantly on ScanNetV2. Specifically, the VoteNet official model with FPTransformer obtains 3.7% mAP@0.25 and 5.5% mAP@0.5 improvements. Similarly, the VoteNet reproduced by [68] with FPTransformer obtains 2.3% mAP@0.25 and 5.2% mAP@0.5 improvements on the MMDetection3D platform. As shown in Fig. 6, FPtransformer helps VoteNet to reduce false positive detection. Our FPTransformer backbone also improves the accuracy of both versions of the Group-Free-3D detection network. Specifically, the Group-Free-3D (12 decoder layers and 512 object candidates) with our FPTransformer backbone gains 2.4% mAP@0.25 and 1.5% mAP@0.5 improvement, outperforming all prior models equipped with transformer modules such as 3DETR [70] and TokenFusion [72], on the mAP@0.25 metric.

C. Classification

1) Synthetic Data: We first evaluate FuPConv and FPTransformer on ModelNet40 [76] which comprises 12 311 CAD models from 40 categories. The data is divided into 9843 training and 2468 test models. We uniformly sample 1024 points from each model and only use their (x, y, z) coordinates as input. The training data is augmented by

TABLE VI

CLASSIFICATION RESULTS ON MODELNET40 AND SCANOBJECTNN DATASET. OUR NETWORK ACHIEVES THE BEST OA. "x, y, z" AND "n" REPRESENT COORDINATES AND NORMAL VECTOR. "K" STANDS FOR ONE THOUSAND. "PN++" STANDS FOR POINTNET++

Method	Input	#Points	OA(%)	
			ModelNet40	ScanObjectNN
PointWeb [38]	xyz, n	1K	92.3	-
PointConv [15]	xyz, n	1K	92.5	-
SpiderCNN [20]	xyz, n	5K	92.4	-
KPConv [18]	xyz	7K	92.9	-
PointASNL [27]	xyz, n	1K	93.2	-
PRANet [29]	xyz	2K	93.7	82.1
RS-CNN [45]	xyz	1K	92.9	-
RS-CNN* [45]	xyz	1K	93.6	-
3DmFV [78]	xyz	1K	91.4	63.0
BGA-DGCN [77]	xyz	1K	-	79.9
BGA-PN++ [77]	xyz	1K	-	80.2
DRNet [79]	xyz	1K	93.1	80.3
GBNet [80]	xyz	1K	93.8	80.5
PointASNL [27]	xyz	1K	92.9	-
PRANet [29]	xyz	1K	93.2	81.0
PointMLP [37]	xyz	1K	94.1	85.4
PointTransformerV2* [47]	xyz	1K	94.2	–
PointNext [56]	xyz	1K	93.2	**87.7**
LCPformer [57]	xyz	1K	93.6	–
PointNet++ [11]	xyz	1K	90.7	77.9
FuPConv(ours)	xyz	1K	93.6 (+2.9)	84.6 (+6.7)
FuPConv*(ours)	xyz	1K	93.9 (+3.2)	85.6 (+7.7)
PointTransformer [28]	xyz	1K	93.7	–
FPTransformer(ours)	xyz	1K	94.1(+0.4)	86.0
FPTransformer*(ours)	xyz	1K	**94.3**(+0.6)	86.5

randomly translating in the range $[-0.2, 0.2]$, scaling in the range $[0.67, 1.5]$.

2) Real-World Data: We also evaluate our network on the real-world ScanObjectNN dataset [77] which includes 15 000 objects categorized into 15 classes. Unlike the synthetic ModelNet40 objects, these objects have occlusion, background noise, deformed geometric shapes, and nonuniform surface density providing a more challenging scenario. We conduct experiments on its hardest perturbed variant (i.e., PB_T50_RS variant). We uniformly sample 1024 points from each object and only use their (x, y, z) coordinates as input. The training data is augmented similarly to ModeleNet40.

3) Network Configuration: We use the same network configuration for ModelNet40 and ScanOjectNN. During training, we use an SGD optimizer with 0.9 momentum and 0.1 initial learning rate to train our model for 350 epochs with a batch size of 32. We adopt cosine annealing to dynamically adjust the learning rate when it drops to 0.001 and use a dropout ratio of 0.4.

4) Results: We compare our method with representative state-of-the-art methods in Table VI using the overall accuracy (OA) metric. For better comparison, we also show the input data type and number of input points for each method. The FuPConv network achieves competitive OA of 93.9% on ModelNet40 and 85.6% on ScanObjectNN, giving significant improvements of 3.2% and 7.7% over the backbone PointNet++ [11]. On ModelNet40, our network outperforms the classical local point convolution KPConv [18], by 1% even though KPConv uses 7000 input points and our network only uses 1024 points. Moreover, our network outperforms RS-CNN when the voting strategy is not used.

The FPTransformer network achieves the stat-of-the-art OA of 94.3% on ModelNet40 and 86.5% on ScanObjectNN.

TABLE VII

NORMAL ESTIMATION RESULTS ON MODELNET40. "x, y, z" REPRESENTS COORDINATES AND "K" STANDS FOR THOUSAND

Method	Input	#Point	Error ↓
PointNet [10]	xyz	1K	0.47
PointNet++ [11]	xyz	1K	0.29
DGCNN [41]	xyz	1K	0.29
PCNN [81]	xyz	1K	0.19
RS-CNN [45]	xyz	1K	0.15
FuPConv(ours)	xyz	1K	0.12
FPTransformer(ours)	xyz	1K	**0.10**

On ModelNet40. Our network outperforms the vector attention-based method (e.g., PT) by 0.6% and scalar attention-based method (e.g., PointASNL) by 1.4%, even though PointASNL additionally uses surface normals as input. With the test time augmentation, our method gets better performance (+0.1% OA) compared to PT V2. Our network exceeds the previous state-of-the-art MLP-based method PointNext by 1.1%

On ScanObjectNN, our network gets the second highest accuracy of 86.5%, outperforming most of existing methods, and exceeding MLP-based method PointMLP by 1.1%, using the same number of input points. Superior performance on real-world datasets indicates that our method is more suitable for practical applications.

D. Normal Estimation

1) Data: Surface normal estimation in point clouds is significant to 3-D reconstruction and rendering. We take normal estimation as a supervised regression task and use the semantic segmentation architecture to achieve it. We conduct the experiment on the ModelNel40 [76], where each point is labeled with its three-directional normal. During training, we uniformly sample 1024 points from each model and only use their (x, y, z) coordinates as input.

2) Network Configuration: The normal estimation network has a similar architecture to the semantic segmentation network apart from the final softmax layer. The K-nearest neighborhood in encoding layers is set to 16. We use the SGD optimizer with 0.9 momentum and 0.05 initial learning rate to train our network for 200 epochs with a batch size of 32. The cosine annealing starts to dynamically adjust the learning rate when it drops to 0.0005.

3) Results: Table VII summarizes our normal estimation results. FuPConv achieves the competitive performance with an error of 0.12. It reduces the error of the backbone PointNet++ [11] by 58.6% and reduces the error compared to prior state-of-the-art RS-CNN [45] by 20%. FPTransformer achieves state-of-the-art performance with a minimum error of 0.10.

E. Ablation Studies

We conduct ablation studies to demonstrate the effectiveness of FPTransformer and SADS block. The first three ablation experiments are performed on S3DIS Area 5 [60], and the fourth and fifth ablation experiment is performed on ScanNet [61] (segmentation dataset). We conduct the last ablation study

TABLE VIII
SEGMENTATION PERFORMANCE OF OUR MODEL ON S3DIS AREA FIVE WITH DIFFERENT POSITION ENCODING. THE NETWORK DOES NOT INCLUDE SADS BLOCK

Encoder	Strategy	mIoU(%)	Encoder	Strategy	mIoU(%)
Sinusoidal	LPE	69.8	MLP	LPE	70.2
Sinusoidal	GPE	68.4	MLP	GPE	69.3
Sinusoidal	FPE	70.5	MLP	FPE	**71.5**

TABLE IX
ABLATION STUDY ON THE NUMBER C_m OF MIDDLE CHANNELS IN EFFICIENT ATTENTION FUNCTION. "M" MEANS MILLION

case	C_m	mIoU(%)	mAcc(%)	OA(%)	Para.
1	[8, 16, 32, 64, 128]	70.7	77.5	90.8	11.1M
2	[4, 8, 16, 32, 64]	**72.2**	**78.5**	**91.5**	10.9M
3	[2, 4, 8, 16, 32]	70.2	77.4	91.3	10.7M
4	[8, 8, 16, 16, 32]	70.8	77.0	90.8	**10.7**M

TABLE X
ABLATION STUDY FOR DIFFERENT SAMPLING BLOCKS. △ STANDS FOR THE DIFFERENCE. "M" MEANS MILLION

Sampling Block	mIoU	△*mIoU*	mAcc	△*mAcc*	OA	△*OA*	Para	△*Para*
GDS	69.2	–	76.3	–	89.3	–	**10.5**M	–
TDS	71.5	*+2.3*	78.2	*+1.9*	91.1	*+0.8*	10.7M	*+0.2*M
SADS (ours)	**72.2**	*+3.0*	**78.5**	*+2.2*	**91.5**	*+1.2*	10.9M	*+0.4*M

TABLE XI
ABLATION STUDY FOR OBJECT DETECTION ON SCANNETV2 USING DIFFERENT NUMBER OF NEIGHBORHOOD POINTS

case	Neighborhood points	mAP@0.25	mAP@0.5
1	[8,64,16,16,16]	60.72	37.71
2	[8,64,32,16,16]	59.41	37.72
3	[8,32,16,16,16]	62.15	**38.87**
4	[8,32,8,8,8]	59.23	36.32
5	[8,28,8,8,8]	**62.31**	38.58
6	[8,24,8,8,8]	59.30	37.02

TABLE XII
CLASSIFICATION AND SEMANTIC SEGMENTATION RESULTS OF FUPCONV WITH DIFFERENT INFLUENCE COEFFICIENT σ

σ	**ScanObjectNN**(OA%)	σ	**S3DIS**(mIoU%)
0.8	84.07	0.1	64.98
1.0	84.39	0.2	**66.80**
1.2	**84.60**	0.4	66.71
1.4	83.48	0.6	64.78
1.6	83.41	0.8	65.89

on ScanObjectNN [77] and S3DIS [60] to determine the optimal coefficient σ in FuPConv.

1) Position Encoding: We study the effects of different encoding strategies used for the position encoding in our transformer encoder. We compare the proposed FPE with some classical position encoding strategies such as LPE and GPE. For each of the three cases, we test with learnable MLP or nonlearnable Sinusoidal [82] position encoding. The results are shown in Table VIII. We can see that the performance of GPE is lower than that of LPE. The underlying cause is that GPE lacks geometric connection information. When an FPE strategy is incorporated into the Transformer, the network achieves the highest performance. Our results also show that the MLP encoder is more flexible than Sinusoidal encoder. This indicates our proposed PT with FPE has more geometric awareness.

2) Efficient Attention: We study the effect of middle channel numbers on our efficient attention function. For this, we test four cases in encoding and decoding layers. Table IX compares the mIoU, mAcc, OA, and network parameters (Para.) of different cases. As we can see, the difference in the number of parameters is not much but when the number of middle channels is set as [4, 8, 16, 32, 64], the network gets the best results on all three metrics. More middle channel numbers slightly increase network parameters and the network can not find a good local optimum with limited training. Conversely, fewer middle channel numbers weaken the geometric encoding ability of the transformer. Overall, the performance of the network remains stable, even for large variations in the channel numbers.

3) Sampling Block: To prove the effectiveness of our proposed SADS block, we compare it with two types of downsampling blocks including the baseline general downsampling (GDS) block and the transition downsampling (TDS) block [28]. GDS consists of one sampling, one grouping, and maxpooling operation. Table X shows the performance of our network with different sampling blocks. Compared to the TDS, the network integrated with the SADS block gets a higher improvement (+3%, +2.2%, and +1.2%) in terms of mIoU, mAcc, and OA with only a minor increase in parameters (+0.4M).

4) Neighborhood Point: Table XI provides detection results of VoteNet [74] integrated with FPTransformer for the different number of neighborhood points. We use the ScanNetV2 dataset for this experiment and reproduce VoteNet using its official code as well as integrate the FPTransformer into it. As we can see when the neighborhood points are set as [8, 28, 8, 8, 8], the network achieves the best performance of 62.31% on the mAP@0.25 metric and when the neighborhood points are set as [8, 32, 16, 16, 16], the network gets the best performance of 38.87% on the mAP@0.5 metric.

5) Coefficient σ: We vary the coefficient σ and observe the performance of our network on ScanObjectNN and S3DIS datasets. As shown in Table XII, our network is not sensitive to the choice of σ, and the best performance is achieved on ScanObjectNN and S3DIS with coefficient 1.2 and 0.2, respectively.

F. Robustness Analysis

1) Robustness to Density: We compare the robustness of our models to point density with several typical baselines such as PointNet [10], PointNet++ [11], DGCNN [41], as well as classical convolutional network such as RS-CNN [45], PointASNL [27]. For a fair comparison, all the networks are trained on modelnet40_normal_resampled dataset [76] with 1024 points using only coordinates as the input. During the test, we use downsampled points of 1024, 512, 256, 128, and 64 as input to the trained model. Results are shown in Fig. 7. As the input points get sparse, the classification accuracy of all networks drops. Overall, our FuPConv and FPTransformer remain more robust than other networks.

2) Robustness to Transformation: To demonstrate the robustness of our FPTransformer, we evaluate its performance on S3DIS under a variety of perturbations in the test data, including permutation, translation, scaling, and jitter.

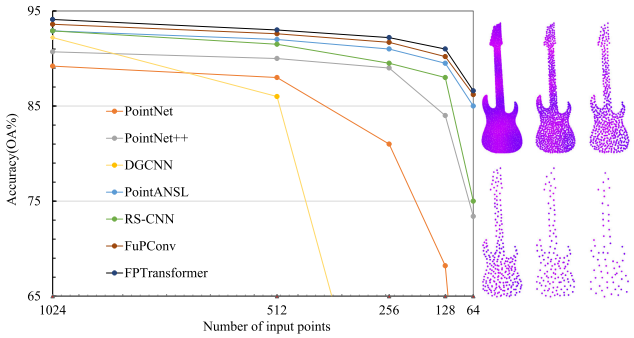

Fig. 7. FuPConv shows the highest robustness to decreasing the density of input points. All models were trained with 1024 points. An example guitar is shown for illustration.

TABLE XIII
ROBUSTNESS STUDY FOR RANDOM POINT PERMUTATIONS

Methods	None	Perm.	Translation		Scaling		Jitter
			+ 0.2	- 0.2	× 0.8	× 1.2	
S3DIS Dataset mIoU(%)							
PointNet [69]	57.75	59.71	22.33	29.85	56.24	59.74	59.04
MinkowskiNet [48]	64.68	64.56	64.59	64.96	59.60	61.93	58.96
PAConv [71]	65.63	65.64	55.81	57.42	64.20	63.94	65.12
PT [28]	70.36	70.45	70.44	70.43	65.73	66.15	59.67
STrans. [50]	71.96	72.02	71.99	71.93	70.42	71.21	72.02
FPTransformer(ours)	**72.21**	**72.23**	**72.31**	**72.42**	**72.09**	**71.48**	**72.31**
ModelNet40 Dataset OA(%)							
PointNet++ [11]	92.1	92.1	90.7	90.8	91.2	91.0	91.0
DGCNN [41]	92.5	92.5	92.3	92.3	92.1	92.3	91.5
PointConv [15]	91.8	91.8	91.8	91.8	89.9	90.6	90.6
FuPConv(ours)	**93.6**	**93.6**	**93.5**	**93.5**	**92.4**	**92.8**	**91.6**

TABLE XIV
ROBUSTNESS TO BACKGROUND NOISE ON SCANOBJECTNN WITHOUT VOTING STRATEGY [45]

Method	obj_nobg	obj_bg	OA drop(%)
3DmFV [78]	69.8	63.0	6.8↓
PointNet [10]	74.4	68.2	6.2↓
PointNet++ [11]	80.2	77.9	2.3↓
SpiderCNN [20]	76.9	73.7	3.2↓
DGCNN [15]	81.5	78.1	3.4↓
PointCNN [12]	80.8	78.5	2.3↓
FuPConv	85.3	84.6	**0.7**↓
FPTransformer	**87.6**	**86.0**	1.6↓

As shown in Table XIII (top), our method's performance remains extremely stable under various transformations. Specially, the performance even improves (+0.21%, +0.27%, and +0.10% mIoU) under the −0.2 translation in x-, y-, and z-axis, and ×1.2 scaling and jitter.

We further evaluate the transformation robustness of our FuPConv on ModelNet40 at test time. Table XIII (bottom) shows that all methods are invariant to permutations. In terms of sensitivity to point scaling, FuPConv performs relatively better when the scaling range is decreased. FuPConv achieves the best accuracy under all transformations.

3) Robustness to Noise: To verify the robustness of FuPConv and FPTransformer to noise, we conduct experiments on the PB_T50_RS variant with background noise ("obj_bg") and without background noise ("obj_nobg") of ScanObjectNN. Table XIV compares the results of our models with some baselines provided in [77]. The OA of all networks decreases when trained and tested in the presence of background noise. However, our model gets the highest accuracy and the lowest performance drops 0.7% and 1.6% from "obj_nobg" variant to "obj_bg" variant, outperforming all compared networks by a large margin.

V. CONCLUSION

We proposed a novel full point encoding to simultaneously explore the local and global features of point clouds as well as their internal correlations by encoding the point positions on the global and local receptive field in a hierarchical manner. Using the proposed full point encoding, we designed FuPConv and FPTransformer that use the rich geometric connections exploited by full point encoding to derive robust convolutional weights and attention weights, respectively. We proposed SADS to maintain the shape information at the point position and feature levels. We evaluated their performance across various tasks such as semantic segmentation, 3-D detection, classification, and normal estimation. Extensive experiments on challenging benchmarks, as well as thorough ablation studies and theoretical analysis, show the robustness and effectiveness of our method on real-world datasets. It is important to exploit the internal connections between local features and global features for irregular data such as point clouds. We hope that our idea of full-point encoding will inspire the research community to rethink local and global feature extraction as a single step.

REFERENCES

[1] Y. Chen, H. Li, R. Gao, and D. Zhao, "Boost 3-D object detection via point clouds segmentation and fused 3-D GIoU-L1 loss," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 2, pp. 762–773, Feb. 2022.

[2] J. Gao, X. Yan, W. Zhao, Z. Lyu, Y. Liao, and C. Zheng, "Spatio-temporal contextual learning for single object tracking on point clouds," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, 2023. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10011208

[3] D. W. Shu and J. Kwon, "Hierarchical bidirected graph convolutions for large-scale 3-D point cloud place recognition," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, 2023. [Online]. Available: https://ieeexplore.ieee.org/document/10021981

[4] Z. Du, H. Ye, and F. Cao, "A novel local–global graph convolutional method for point cloud semantic segmentation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 4, pp. 4798–4812, Apr. 2024.

[5] M. U. Khalid, J. M. Hager, W. Kraus, M. F. Huber, and M. Toussaint, "Deep workpiece region segmentation for bin picking," in *Proc. IEEE 15th Int. Conf. Autom. Sci. Eng. (CASE)*, Aug. 2019, pp. 1138–1144.

[6] F. J. Lawin, M. Danelljan, P. Tosteberg, G. Bhat, F. S. Khan, and M. Felsberg, "Deep projective 3D semantic segmentation," in *Proc. Int. Conf. Pattern Recognit. Image Anal.* Cham, Switzerland: Springer, Jun. 2017, pp. 95–107.

[7] A. Boulch, J. Guerry, B. Le Saux, and N. Audebert, "SnapNet: 3D point cloud semantic labeling with 2D deep segmentation networks," *Comput. Graph.*, vol. 71, pp. 189–198, Apr. 2018.

[8] L. Tchapmi, C. Choy, I. Armeni, J. Gwak, and S. Savarese, "SEGCloud: Semantic segmentation of 3D point clouds," in *Proc. Int. Conf. 3D Vis. (3DV)*, Oct. 2017, pp. 537–547.

[9] D. Maturana and S. Scherer, "VoxNet: A 3D convolutional neural network for real-time object recognition," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2015, pp. 922–928.

[10] 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. (CVPR)*, Jul. 2017, pp. 652–660.

[11] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep hierarchical feature learning on point sets in a metric space," 2017, *arXiv:1706.02413*.

[12] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen, "PointCNN: Convolution on X-transformed points," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, vol. 31, 2018, pp. 820–830.

[13] M. Simonovsky and N. Komodakis, "Dynamic edge-conditioned filters in convolutional neural networks on graphs," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 3693–3702.

[14] S. Wang, S. Suo, W. Ma, A. Pokrovsky, and R. Urtasun, "Deep parametric continuous convolutional neural networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 2589–2597.

[15] W. Wu, Z. Qi, and L. Fuxin, "PointConv: Deep convolutional networks on 3D point clouds," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 9621–9630.

[16] P. Hermosilla, T. Ritschel, P.-P. Vázquez, À. Vinacua, and T. Ropinski, "Monte Carlo convolution for learning on non-uniformly sampled point clouds," *ACM Trans. Graph.*, vol. 37, no. 6, pp. 1–12, Dec. 2018.

[17] Y. Shen, C. Feng, Y. Yang, and D. Tian, "Mining point cloud local structures by kernel correlation and graph pooling," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4548–4557.

[18] 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 Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 6411–6420.

[19] M. Fey, J. E. Lenssen, F. Weichert, and H. Müller, "SplineCNN: Fast geometric deep learning with continuous B-spline kernels," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 869–877.

[20] 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.*, 2018, pp. 87–102.

[21] H. Lei, N. Akhtar, and A. Mian, "SegGCN: Efficient 3D point cloud segmentation with fuzzy spherical kernel," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 11608–11617.

[22] S. Xie, S. Liu, Z. Chen, and Z. Tu, "Attentional ShapeContextNet for point cloud recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4606–4615.

[23] X. Liu, Z. Han, Y.-S. Liu, and M. Zwicker, "Point2sequence: Learning the shape representation of 3D point clouds with an attention-based sequence to sequence network," in *Proc. AAAI Conf. Artif. Intell.*, Jul. 2019, vol. 33, no. 1, pp. 8778–8785.

[24] J. Yang et al., "Modeling point clouds with self-attention and Gumbel subset sampling," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 3323–3332.

[25] J. Lee, Y. Lee, J. Kim, A. R. Kosiorek, S. Choi, and Y. W. Teho, "Set Transformer: A framework for attention-based permutation-invariant neural networks," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2019, pp. 3744–3753.

[26] M. Feng, L. Zhang, X. Lin, S. Z. Gilani, and A. Mian, "Point attention network for semantic segmentation of 3D point clouds," *Pattern Recognit.*, vol. 107, Nov. 2020, Art. no. 107446.

[27] X. Yan, C. Zheng, Z. Li, S. Wang, and S. Cui, "PointASNL: Robust point clouds processing using nonlocal neural networks with adaptive sampling," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 5589–5598.

[28] H. Zhao, L. Jiang, J. Jia, P. Torr, and V. Koltun, "Point transformer," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 16259–16268.

[29] S. Cheng, X. Chen, X. He, Z. Liu, and X. Bai, "PRA-Net: Point relation-aware network for 3D point cloud analysis," *IEEE Trans. Image Process.*, vol. 30, pp. 4436–4448, 2021.

[30] Q. Hu et al., "RandLA-Net: Efficient semantic segmentation of large-scale point clouds," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 11108–11117.

[31] F. Groh, P. Wieschollek, and H. P. Lensch, "Flex-convolution," in *Proc. Asian Conf. Comput. Vis.* Cham, Switzerland: Springer, May 2019, pp. 105–122.

[32] J. Li, B. M. Chen, and G. H. Lee, "SO-Net: Self-organizing network for point cloud analysis," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 9397–9406.

[33] F. Engelmann, T. Kontogianni, J. Schult, and B. Leibe, "Know what your neighbors do: 3D semantic segmentation of point clouds," in *Proc. Eur. Conf. Comput. Vis. Workshop*. Cham, Switzerland: Springer, Jan. 2019, pp. 395–409.

[34] F. Engelmann, T. Kontogianni, A. Hermans, and B. Leibe, "Exploring spatial context for 3D semantic segmentation of point clouds," in *Proc. IEEE Int. Conf. Comput. Vis. Workshops (ICCVW)*, Oct. 2017, pp. 716–724.

[35] Z. Zhang, B. Hua, and S. Yeung, "ShellNet: Efficient point cloud convolutional neural networks using concentric shells statistics," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 1607–1616.

[36] H. Ran, J. Liu, and C. Wang, "Surface representation for point clouds," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 18920–18930.

[37] 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," in *Proc. Int. Conf. Learn. Represent.*, 2022. [Online]. Available: https://openreview.net/forum?id=3Pbra-_u76D

[38] H. Zhao, L. Jiang, C.-W. Fu, and J. Jia, "PointWeb: Enhancing local neighborhood features for point cloud processing," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 5565–5573.

[39] L. Jiang, H. Zhao, S. Liu, X. Shen, C.-W. Fu, and J. Jia, "Hierarchical point-edge interaction network for point cloud semantic segmentation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 10433–10441.

[40] M. Xu, J. Zhang, Z. Zhou, M. Xu, X. Qi, and Y. Qiao, "Learning geometry-disentangled representation for complementary understanding of 3D object point cloud," in *Proc. AAAI Conf. Artif. Intell.*, 2021, vol. 35, no. 4, pp. 3056–3064.

[41] 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.

[42] R. Klokov and V. Lempitsky, "Escape from cells: Deep Kd-networks for the recognition of 3D point cloud models," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 863–872.

[43] M. Xu, Z. Zhou, and Y. Qiao, "Geometry sharing network for 3D point cloud classification and segmentation," in *Proc. AAAI Conf. Artif. Intell.*, 2020, vol. 34, no. 7, pp. 12500–12507.

[44] 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 Recognit. (CVPR)*, Jun. 2021, pp. 3173–3182.

[45] Y. Liu, B. Fan, S. Xiang, and C. Pan, "Relation-shape convolutional neural network for point cloud analysis," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 8895–8904.

[46] C. Wang, B. Samari, and K. Siddiqi, "Local spectral graph convolution for point set feature learning," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 52–66.

[47] X. Wu, Y. Lao, L. Jiang, X. Liu, and H. Zhao, "Point transformer v2: Grouped vector attention and partition-based pooling," 2022, *arXiv:2210.05666*.

[48] C. Choy, J. Gwak, and S. Savarese, "4D spatio-temporal ConvNets: Minkowski convolutional neural networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 3075–3084.

[49] C. Park, Y. Jeong, M. Cho, and J. Park, "Fast point transformer," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 16949–16958.

[50] X. Lai et al., "Stratified transformer for 3D point cloud segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 8500–8509.

[51] Y. Ma, Y. Guo, H. Liu, Y. Lei, and G. Wen, "Global context reasoning for semantic segmentation of 3D point clouds," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Mar. 2020, pp. 2931–2940.

[52] H. Lei, N. Akhtar, and A. Mian, "Spherical kernel for efficient graph convolution on 3D point clouds," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 10, pp. 3664–3680, Oct. 2021.

[53] S. Qiu, S. Anwar, and N. Barnes, "Semantic segmentation for real point cloud scenes via bilateral augmentation and adaptive fusion," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 1757–1767.

[54] L. Tang, Y. Zhan, Z. Chen, B. Yu, and D. Tao, "Contrastive boundary learning for point cloud segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 8489–8499.

[55] J. Choe, C. Park, F. Rameau, J. Park, and I. S. Kweon, "PointMixer: MLP-mixer for point cloud understanding," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*. Cham, Switzerland: Springer, Oct. 2022, pp. 620–640.

[56] G. Qian et al., "PointNext: Revisiting PointNet++ with improved training and scaling strategies," in *Proc. Adv. Neural Inf. Process. Syst.*, 2022, pp. 23192–23204.

[57] Z. Huang, Z. Zhao, B. Li, and J. Han, "LCPFormer: Towards effective 3D point cloud analysis via local context propagation in transformers," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 33, no. 9, pp. 4985–4996, Sep. 2023.

[58] H. Lei, N. Akhtar, M. Shah, and A. Mian, "Mesh convolution with continuous filters for 3-D surface parsing," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, 2023. [Online]. Available: https://ieeexplore.ieee.org/document/10149824

[59] S. Fan, Q. Dong, F. Zhu, Y. Lv, P. Ye, and F.-Y. Wang, "SCF-Net: Learning spatial contextual features for large-scale point cloud segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 14504–14513.
[60] I. Armeni et al., "3D semantic parsing of large-scale indoor spaces," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 1534–1543.
[61] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, "ScanNet: Richly-annotated 3D reconstructions of indoor scenes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 5828–5839.
[62] W. Hu, H. Zhao, L. Jiang, J. Jia, and T.-T. Wong, "Bidirectional projection network for cross dimension scene understanding," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 14373–14382.
[63] M. Jaritz, J. Gu, and H. Su, "Multi-view PointNet for 3D scene understanding," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. Workshop (ICCVW)*, Oct. 2019, pp. 3995–4003.
[64] H.-Y. Chiang, Y.-L. Lin, Y.-C. Liu, and W. H. Hsu, "A unified point-based framework for 3D segmentation," in *Proc. Int. Conf. 3D Vis. (3DV)*, Sep. 2019, pp. 155–163.
[65] S. Shi, X. Wang, and H. Li, "PointRCNN: 3D object proposal generation and detection from point cloud," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 770–779.
[66] S. Shi et al., "PV-RCNN: Point-voxel feature set abstraction for 3D object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 10529–10538.
[67] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *Int. J. Robot. Res.*, vol. 32, no. 11, pp. 1231–1237, 2013.
[68] MMDetection3D Contributors. (2020). *MMDetection3D: OpenMMLab Next-Generation Platform for General 3D Object Detection*. [Online]. Available: https://github.com/open-mmlab/mmdetection3d
[69] Z. Zhang, B. Sun, H. Yang, and Q. Huang, "H3DNet: 3D object detection using hybrid geometric primitives," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, Oct. 2020, pp. 311–329.
[70] I. Misra, R. Girdhar, and A. Joulin, "An end-to-end transformer model for 3D object detection," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 2906–2917.
[71] B. Cheng, L. Sheng, S. Shi, M. Yang, and D. Xu, "Back-tracing representative points for voting-based 3D object detection in point clouds," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 8963–8972.
[72] Y. Wang, X. Chen, L. Cao, W. Huang, F. Sun, and Y. Wang, "Multimodal token fusion for vision transformers," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2022, pp. 12186–12195.
[73] H. Wang et al., "RBGNet: Ray-based grouping for 3D object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2022, pp. 1100–1109.
[74] C. R. Qi, O. Litany, K. He, and L. Guibas, "Deep Hough voting for 3D object detection in point clouds," in *Proc. ICCV*, Oct. 2019, pp. 9277–9286.
[75] Z. Liu, Z. Zhang, Y. Cao, H. Hu, and X. Tong, "Group-free 3D object detection via transformers," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 2949–2958.
[76] Z. Wu et al., "3D ShapeNets: A deep representation for volumetric shapes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1912–1920.
[77] M. A. Uy, Q. Pham, B. Hua, T. Nguyen, and S. Yeung, "Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 1588–1597.
[78] Y. Ben-Shabat, M. Lindenbaum, and A. Fischer, "3DmFV: Three-dimensional point cloud classification in real-time using convolutional neural networks," *IEEE Robot. Autom. Lett.*, vol. 3, no. 4, pp. 3145–3152, Oct. 2018.
[79] S. Qiu, S. Anwar, and N. Barnes, "Dense-resolution network for point cloud classification and segmentation," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Jan. 2021, pp. 3813–3822.
[80] S. Qiu, S. Anwar, and N. Barnes, "Geometric back-projection network for point cloud classification," *IEEE Trans. Multimedia*, vol. 24, pp. 1943–1955, 2022.
[81] M. Atzmon, H. Maron, and Y. Lipman, "Point convolutional neural networks by extension operators," *ACM Trans. Graph.*, vol. 37, no. 4, pp. 1–12, Aug. 2018.
[82] Z. Liu, H. Hu, Y. Cao, Z. Zhang, and X. Tong, "A closer look at local aggregation operators in point cloud analysis," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, Nov. 2020, pp. 326–342.

Yong He received the B.S. degree from the Anhui University of Science and Technology, Huainan, Anhui, China, in 2015, and the M.S. degree from the China University of Mining and Technology, Xuzhou, Jiangsu, China, in 2018. He is currently pursuing the Ph.D. degree with Hunan University, Changsha, China.

He is also a Visiting Scholar with The University of Western Australia, Perth, WA, Australia. His research interests include computer vision, point cloud analysis, and deep learning.

Hongshan Yu received the B.S., M.S., and Ph.D. degrees in control science and technology from the Department of Electrical and Information Engineering, Hunan University, Changsha, China, in 2001, 2004, and 2007, respectively.

From 2011 to 2012, he was a Post-Doctoral Researcher with the Laboratory for Computational Neuroscience, University of Pittsburgh, Pittsburgh, PA, USA. He is currently a Professor with Hunan University and the Associate Dean with the National Engineering Laboratory for Robot Visual Perception and Control. His research interests include autonomous mobile robots and machine vision.

Zhengeng Yang received the B.S. and M.S. degrees from Central South University, Changsha, China, in 2009 and 2012, respectively, and the Ph.D. degree from Hunan University, Changsha, in 2020.

He was a Visiting Scholar with the University of Pittsburgh, Pittsburgh, PA, USA, from 2018 to 2020. He was a Post-Doctoral Researcher at Hunan University from 2020 to 2023. He is currently an Associate Professor with the College of Engineering and Design, Hunan Normal University, Changsha. His research interests include computer vision, image analysis, and machine learning.

Xiaoyan Liu received the Ph.D. degree in process and system engineering from Otto von Guericke University, Magdeburg, Germany, in 2005.

She is currently a Professor with Hunan University, Changsha, China. Her research interests include machine vision and pattern recognition.

Wei Sun received the M.S. and Ph.D. degrees in control science and technology from Hunan University, Changsha, China, in 1999 and 2002, respectively.

He is currently a Professor with Hunan University. His research interests include artificial intelligence, robot control, complex mechanical and electrical control systems, and automotive electronics.

Ajmal Mian (Senior Member, IEEE) is currently a Professor of computer science with The University of Western Australia, Perth, WA, USA. He received three esteemed national fellowships and several major research grants from the Australian Research Council, the National Health and Medical Research Council, and the U.S. Department of Defense DARPA. His research interests include 3-D computer vision, machine learning, point cloud analysis, human action, and video analysis.

Mr. Mian is a fellow of the International Association for Pattern Recognition. He received several awards including the HBF Mid-Career Scientist of the Year Award in 2022 and the West Australian Early Career Scientist of the Year Award in 2012. He serves as a Senior Editor for IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS and an Associate Editor for *Pattern Recognition* journal.