Learning Composite Representations for Point Cloud Scene Understanding

Haibo Qiu

Supervisor: Prof. Dacheng Tao
Auxiliary Supervisor: Dr. Baosheng Yu

A thesis submitted in fulfilment of
the requirements for the degree of
Doctor of Philosophy

School of Computer Science
Faculty of Engineering
The University of Sydney
Australia

March 2024

To my family.

Statement of Originality

This is to certify that to the best of my knowledge, the content of this thesis is my own work. This thesis has not been submitted for any degree or other purposes.

I certify that the intellectual content of this thesis is the product of my own work and that all the assistance received in preparing this thesis and sources have been acknowledged.

Haibo Qiu
School of Computer Science
Faculty of Engineering
The University of Sydney
31 December 2023

Authorship Attribution Statement

This thesis was conducted at the University of Sydney, under the supervision of Prof. Dacheng Tao, between 2020 and 2023. The main results presented in this dissertation were first introduced in the following publications:

(1) **Haibo Qiu**, Baosheng Yu and Dacheng Tao. GFNet: Geometric Flow Network for 3D Point Cloud Semantic Segmentation. In: *Transactions on Machine Learning Research* (2022). ISSN: 2835-8856. Presented in Chapter 3. I designed the research, implemented the systems, conducted the experiments, and wrote the draft of the paper.

(2) **Haibo Qiu**, Baosheng Yu and Dacheng Tao. Collect-and-Distribute Transformer for 3D Point Cloud Analysis. *Under review*. Presented in Chapter 4. I designed the research, implemented the systems, conducted the experiments, and wrote the draft of the paper.

(3) **Haibo Qiu**, Baosheng Yu, Yixin Chen and Dacheng Tao. PointHR: Exploring High-Resolution Architectures for 3D Point Cloud Segmentation. *Under review.* Presented in Chapter 5. I designed the research, implemented the systems, conducted the experiments, and wrote the draft of the paper.

In addition to the statements above, in cases where I am not the corresponding author of a published item, permission to include the published material has been granted by the corresponding author.

Haibo Qiu
School of Computer Science
Faculty of Engineering
The University of Sydney
31 December 2023

As supervisor for the candidature upon which this thesis is based, I can confirm that the authorship attribution statements above are correct.

Dacheng Tao
School of Computer Science
Faculty of Engineering
The University of Sydney
31 December 2023

Acknowledgements

I extend my deepest gratitude to Professor Dacheng Tao, my esteemed mentor, whose invaluable guidance and unwavering support have been instrumental throughout this academic journey. I am truly honored to have had Professor Tao as my supervisor. His profound expertise and encouragement have significantly enriched my research endeavors, enabling me to achieve substantial progress in my field of study.

I am also indebted to Dr. Baosheng Yu, my co-supervisor, for his patient discussions and invaluable insights that have helped me navigate and surmount the challenges encountered during my doctoral research.

Furthermore, I wish to express my heartfelt appreciation to other mentors who have offered their guidance during my doctoral tenure: Dr. Chunyu Wang, Prof. Wenjun Zeng, Dr. Dihong Gong, Dr. Zhifeng Li and Dr. Wei Liu. Their mentorship and teachings have played a pivotal role in shaping my academic growth and understanding.

I am immensely grateful to numerous colleagues and friends: Dr. Sen Zhang, Dr. Yongcheng Jing, Dr. Hao Guan, Dr. Yajing Kong, Dr. Zhihao Cheng, Dr. Yufei Xu, Dr. Qiming Zhang, Yan Sun, Dr. Zhi Hou, Cheng Wen, Shiye Lei, Xu Zhang, Liyao Tang, Xikun Zhang, Dr. Jinlong Fan, Dr. Qi Zheng, Haimei Zhao, Sihan Ma, Dr. Jiaxian Guo, Dr. Xinqi Zhu, Kaining Zhang, Qian Cheng, Yang Qian, Huihui Gong, Linwei Tao, Dr. Chaoyue Wang, Dr. Jing Zhang, Dr. Yibing Zhan, Xiaofei Liu, Dr. Yan Wang, Dr. Jianwei Yu, Qu Cui, Dr. Mengyang Liu, Dr. Heng Chang, Dr. Yizhou Zhou, Huayang Li, Dr. Songxiang Liu, Dr. Lingfeng Guo, Dr. Yujia Zhang, Wenxuan Wang, Yijia Zhao, Yu He, Ziyang Chen, Jiayi Jiang, Jiannan Xiang, Xiangqian Sun, Jiacheng Li, and Jialun

Peng. Their unwavering support and companionship have been a constant source of motivation, propelling me forward during difficult times and making this milestone attainable.

Lastly, I would like to extend my special thanks to my parents, Minghe Qiu and Pingxiang Deng. Their steadfast support and encouragement have been my rock throughout these years. Additionally, I am profoundly grateful to my girlfriend, Dr. Kailin Gao. Despite the challenges of our time apart during my doctoral pursuits, her unwavering understanding and support have been an invaluable source of strength and motivation.

I owe a debt of gratitude to each and every individual mentioned above, as well as to those not explicitly named, for their indispensable contributions to my academic and personal growth. This dissertation would not have been possible without their support, guidance, and encouragement.

Abstract

In recent years, there has been significant interest in 3D point clouds from academia and industry. Unlike 2D images, point clouds consist of 3D points with Cartesian coordinates, offering an accurate, view-invariant representation of real-world scenes. Understanding 3D point cloud scenes is crucial for applications like autonomous driving, robotics, and AR/VR. However, comprehending these scenes is challenging due to their complex spatial structures and objects at various scales. Previous methods have often focused on learning representations that capture only local details or a specific scale, leading to suboptimal results.

This thesis aims to advance learning composite representations for scene understanding by integrating multiple clues. It explores composite learning schemes from two angles: the data side and the model side. From the data perspective, it suggests projecting 3D point clouds into various 2D views and using multi-view feature fusion to learn composite representations. An end-to-end trainable geometric flow network is introduced to achieve this, enabling the learning and fusion of multi-view representations. The model side investigates the design of local basic operators and a global framework. The thesis introduces the collect-and-distribute block for local operators, which capture short- and long-range contexts simultaneously, effectively learning composite representations that incorporate sufficient contextual information. For the global framework, it addresses the challenge of multi-scale objects by employing high-resolution architectures that maintain high resolutions throughout the network and facilitate communication of multiple resolution features, efficiently learning multi-scale composite representations. Extensive experiments and analysis on popular benchmarks demonstrate the effectiveness of these approaches.

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

Scene understanding is a fundamental and widely studied topic in the field of computer vision. Its primary objective is to comprehensively perceive and comprehend a given scene, consequently providing essential groundwork for downstream applications such as autonomous driving [2, 67] and virtual reality [3, 18]. The core task involved in scene understanding is semantic segmentation, which aims to partition and analyze an input scene (*e.g.*, the RGB image) into distinct regions associated with specific semantic categories such as sky, road, person, table, and bed. Depending on the type of input scene, whether it be images or point clouds, semantic segmentation can be further categorized into 2D and 3D branches. The main distinction lies in the fact that 2D semantic segmentation assigns a semantic label to each pixel in an image, while 3D semantic segmentation predicts the semantic category for each individual point in a point cloud.

Due to the emergence of Deep Learning [46, 99, 57] in recent years, significant progress has been achieved in the field of 2D semantic segmentation [74, 98, 13, 152]. For instance, the Fully Convolutional Network (FCN) [74] is a pivotal approach that eliminates fully connected layers, enabling the handling of inputs with arbitrary sizes. FCN also introduces skip connections, combining deep, coarse, semantic information with shallow, fine, appearance information to learn multi-scale information effectively. Another noteworthy architecture is the U-Net [98], which consists of a contracting path to capture context and a symmetric expanding path for precise localization, resulting in a u-shaped structure. The inclusion of skip connections between these paths facilitates the propagation of context information to higher resolution layers. Additionally, DeepLab [13] proposes the use of atrous convolution to effectively enlarge the receptive field, followed by atrous spatial pyramid pooling (ASPP) to capture objects and image context at multiple scales. Moreover, PSPNet [152] introduces a pyramid pooling module (PPM) to capture both global and sub-region context information for handling various categories with different scales.

Compared to 2D images, such as RGB images, point clouds consist of a collection of 3D points represented by Cartesian coordinates (x, y, z). This representation provides an accurate geometry perspective aligned with the 3D real world. Consequently, semantic segmentation of 3D point clouds plays a crucial role, particularly in safety-critical applications like autonomous driving [67, 2] and robotics [64, 136]. There are three primary challenges that need to be addressed in 3D point cloud scene understanding:

- **Complex space structure:** The arrangement and structure of objects in a 3D point cloud scene are typically intricate and mixed. For instance, in a traffic scene, cars and pedestrians are intertwined, while the surrounding buildings possess diverse shapes.
- **Insufficient contextual information:** When the model can only perceive the local region and fail to catch sufficient contexts, it becomes challenging to differentiate between two categories with partially similar shapes. For example, distinguishing between the flat plane of a table and the floor poses a difficulty in predicting the semantic label for each point.
- **Multi-scale objects:** 3D scenes often encompass objects of varying scales. For instance, an office room might feature a small cup on a large table.

This thesis aims to address the aforementioned challenges from the perspective of representation learning. Previous works, such as PointNet [83], have made significant contributions by utilizing multi-layer perceptrons (MLPs) to directly process point clouds. PointNet++ [84] further advances this approach by introducing a hierarchical structure that captures contextual information. In contrast, PointCNN [66] learns an x-transformation to align input points, followed by typical convolution layers. Additionally, KPConv [108] introduces a novel point convolution operator known as kernel point convolution, which processes neighboring points using spatially located weights. Recently, transformer architectures have emerged for point cloud segmentation [58, 81, 151]. Point Transformer [151] proposes the use of a vector self-attention mechanism to aggregate neighbor features and capture local dependencies. However, these existing methods either focus solely on capturing local details or a specific scale of objects, thus resulting in suboptimal results.

Therefore, this thesis proposes to learn a composite representation that combines multiple clues to facilitate a comprehensive understanding of the target scene. To achieve this goal, the thesis explores representation learning from both the data transformation and model construction perspectives. Specifically, three complementary works with different approaches are introduced: multi-view data fusion, short-long range operator, and multi-scale framework. These works aim to enhance the perception of space structure, capture sufficient short-long contextual clues, and effectively capture multi-scale objects, respectively. Together, they contribute to the advancement of learning effective representations for point cloud scene understanding.

1.1 Problem Formulation

A 3D point cloud typically comprises a set of N points denoted as $\boldsymbol{P} \in \mathbb{R}^{N \times C_{raw}}$, where C_{raw} represents the feature dimension of each point. This dimension includes Cartesian coordinates (x, y, z) and other attributes like RGB values. The objective of 3D point cloud semantic segmentation is to assign a semantic category label (*i.e.*, sky, table, chair) to each point $(x, y, z) \in \boldsymbol{P}$. The final predictions have the same spatial dimension as the raw points, denoted as $\boldsymbol{Y} \in \mathbb{N}^N$, where each entry $y \in \{0, \cdots, C-1\}$ and C represents the total number of semantic categories. In some cases, for a comprehensive scene understanding, it is necessary to classify 3D objects, where the goal shifts to predicting a single semantic label for the entire point cloud $\boldsymbol{P}$.

The general pipeline for processing point clouds involves several steps. Initially, the raw point cloud $\boldsymbol{P}$ is transformed into an input tensor $\boldsymbol{F}_{input}$. One common transformation approach is to project the 3D point cloud onto a 2D image and utilize well-established 2D networks [74, 152, 98] for further processing. Another approach involves voxelizing the 3D point cloud into structural grids and applying 3D convolutions [103, 25, 22, 23]. Alternatively, a simple normalization step can be applied, and the points can be directly used as input. Next, a model is constructed and trained to map the input tensor $\boldsymbol{F}_{input}$ to an output representation $\boldsymbol{F}_{output} \in \mathbb{R}^{N \times C_{fea}}$, where C_{fea} represents the feature dimension. Finally, a classifier is applied to obtain the semantic predictions for each point.

This thesis aims to learn the composite representation $\boldsymbol{F}_{output}$ from both the data and model aspects. Regarding the data side, it proposes to project the point cloud into multiple 2D views and learn a multi-view composite representation across these views to perceive the structure of the space from different angles. On the model side, the thesis first investigates the basic processing block of the model and introduces the short-long context operator to capture sufficient short-long contextual information. Then, it explores the overall model framework and designs a multi-scale architecture to learn a multi-scale composite representation, enabling the capture of rich multi-scale objects. These three works complement each other and collectively tackle the challenges related to 3D point cloud scene understanding.

1.2 Contributions

The contributions of this thesis can be summarized as follows:

- **Comprehensive analysis of the characteristics of 3D point cloud and proposal of learning different composite representations to address three important challenges in 3D point cloud scene understanding.** These challenges include complex space structures, insufficient contextual information, and multi-scale objects.
- **Development of a novel multi-view fusion scheme to handle the complex structure of 3D scenes.** This scheme involves projecting point clouds into multiple 2D views and introducing a new GFNet (Geometric Flow Network) to simultaneously fuse features from these views. The scheme includes a geometric flow module (GFM) that allows geometric correspondence information to flow across different views, enabling comprehensive observation and perception of the spatial structure.
- **Introduction of an innovative block that integrates short-range and long-range clues to capture contextual information effectively.** This block incorporates a novel collect-and-distribute mechanism to learn composite representations that capture both short- and long-range contexts. Additionally, it utilizes context-aware position encoding to enhance position clues and facilitate communication between points.
- **Exploration of an elaborate high-resolution framework to capture multi-scale features.** This framework aims to maintain multiple resolutions simultaneously, allowing for frequent communication between different resolutions within each stage. This facilitates the learning of composite representations that capture information at multiple scales. The framework is designed with unified sequence and resampling operators, enabling the use of off-the-shelf point cloud blocks and layers without requiring additional efforts.

1.3 Thesis Outline

The remainder of this thesis is organized into five chapters, with the main content of each chapter summarized as follows:

Chapter 2: Background

This chapter presents an overview of the literature that is relevant to this thesis, with a primary focus on 3D scene understanding. Additionally, it includes a review of closely related 2D scene understanding approaches, encompassing considerations such as multi-scale analysis and contextual information. It is worth noting that previous approaches to 3D scene understanding are examined from three main perspectives: projected representation learning, voxelized representation learning, and vanilla representation learning.

Chapter 3: Multi-View Composite Representation Learning

This chapter explores the use of observations from multiple different views to comprehensively perceive the complex space structure of 3D point clouds. A novel Geometric Flow Network (GFNet) is introduced for multi-view composite representation learning. GFNet incorporates a geometric flow module that enables the flow of geometric correspondence information across different views. The design of GFNet allows for training and testing in an end-to-end paradigm, using grid sampling and KPConv to avoid time-consuming and non-differentiable post-processing. Extensive experiments on SemanticKITTI and nuScenes, two popular large-scale point cloud semantic segmentation benchmarks, demonstrate the effectiveness of GFNet.

Chapter 4: Contextual Composite Representation Learning

This chapter focuses on capturing sufficient contextual information by integrating short-range and long-range clues. A novel collect-and-distribute mechanism is proposed, which builds the CDFormer architecture to capture both short-

and long-range contexts simultaneously. This mechanism effectively learns local-global contextual composite representations of point clouds. Additionally, context-aware position encoding enhances position clues dynamically and facilitates communication among point features. Comprehensive experiments on segmentation and classification tasks using datasets such as S3DIS, ScanNetV2, ShapeNetPart, ModelNet40, and ScanObjectNN demonstrate the superiority of CDFormer and the benefits of each design choice.

Chapter 5: Multi-Scale Composite Representation Learning

This chapter investigates multi-scale composite representation learning to effectively capture multi-scale objects. The PointHR framework, designed for point cloud segmentation, maintains multiple resolutions simultaneously and facilitates frequent communication between all resolutions within each stage. The proposed framework formulates PointHR in a unified way using a sequence operator and a resampling operator, allowing for the use of off-the-shelf point cloud blocks and modules without additional efforts. Additionally, pre-computed indices for knn collection and resampling operations are proposed to avoid on-the-fly computations, resulting in efficient utilization of high-resolution architectures. Comprehensive experiments on S3DIS and ScanNetV2 datasets demonstrate the effectiveness of high-resolution architectures for 3D dense point cloud analysis and establish new state-of-the-art performances.

Chapter 6: Conclusions

This chapter provides a summary of the contributions and implications of the research conducted in the thesis. It also outlines future directions for further exploration in the field.

CHAPTER 2

Background

This chapter begins by providing a brief overview of the literature on 2D scene understanding that is relevant to this thesis. Specifically, it examines the literature from various angles such as multi-scale issues and contextual information. Subsequently, the chapter delves into a comprehensive investigation of 3D scene understanding methods, with a primary focus on the perspective of representation learning.

2.1 2D Scene Understanding

In recent years, due to the significant advancement of Deep Learning, deep convolutional neural networks (CNNs) become the dominant approach in various computer vision tasks, such as image classification [57, 99, 46] and object detection [37, 96]. Similarly, Long *et al.* [74] propose a milestone work in scene parsing field, named Fully Convolutional Network (FCN), which discards the fully connected layers to handle input of arbitrary size and to compute efficiently for dense predictions. However, there are still three main challenges to be tackled: 1) objects to be parsed are multi-scale; and 2) contextual information is not sufficiently explored and utilized. In the following, I will categorize related literature into two groups that address the above two concerns, and revisit them in detail.

2.1.1 Multi-Scale Learning

The objects to be parsed in the scene are always multi-scale, *e.g.*, a small cup is on the big table, which is not handled well by standard convolution. A straightforward way to deal with this is to feed the DCNN with rescaled versions of the same image and then aggregate the feature maps [15]. However, it is less computationally efficient. FCN [74] proposes to combine deep, coarse, semantic information with shallow, fine, appearance information with skip connections to learn multi-scale information. Similarly, UNet [98] consists of a contracting path to capture context and a symmetry expanding path for precise localization, yielding a u-shaped architecture. The skip connections between two paths is to fuse multi-scale information from different layers.

DeepLab [13] proposes atrous convolution to effectively enlarge the receptive field and the following atrous spatial pyramid pooling (ASPP) to capture objects as well as image context at multiple scales. Later, DeepLabV3 [14] augments the atrous spatial pyramid pooling (ASPP) with image-level features which encode global context to further boost the recognition accuracy for small objects. DeepLabv3+ [16] proposes a novel encoder-decoder structure by extending DeepLabV3 with a simple yet effective decoder module, which can capture multi-scale information by gradually recovering the spatial information. PSPNet [152] devises a pyramid pooling module (PPM) to capture the global and sub-region context information to handle various categories with different scales. It also adopts a deeply auxiliary loss to effectively train the network.

2.1.2 Contextual Information Learning

Except for accurately locating the objects, correctly classifying the object is also the key part of scene parsing, which can be facilitated by utilizing contextual information. However, the contextual information is not sufficiently explored due to the local characteristic of convolution operations in DCNN. Hence, [74, 98] propagate context information from lower to higher resolution layers to facilitate the final dense predictions. DeepLab [13] introduces atrous convolution to effectively enlarge the receptive field to incorporate larger context without increasing the parameters or computations. PSPNet [152] proposes a pyramid pooling module (PPM) to capture the global and sub-region context information.

However, those methods can either not generate dense contextual information, or not satisfy the requirement that different pixels need different contextual dependencies. EncNet [146] introduces the Context Encoding Module to capture the semantic context of scenes and selectively highlights class-dependent features. It also proposes a Semantic Encoding Loss (SE-Loss) which predicts the presences of object categories in the scene, to incorporate global context. Given that SE-Loss considers big and small objects equally, resulting in the improvement of the small objects in segmentation. APCNet [45] proposes Adaptive Context Modules to utilize local and global representation to estimate affinity weights for local regions, allowing to construct adaptive and multi-scale contextual representations for segmentation task. DMNet [44] devises the Dynamic Convolutional Modules, which exploit a set of context-aware filters (*i.e.*, adaptive to image contents) to estimate semantic representations for specific scales. PSANet [153] proposes the point-wise spatial attention to aggregate long-range contextual information. Each position in the feature map is connected with all other ones through self-adaptively predicted attention maps. Furthermore, a bi-directional information propagation path is designed for a comprehensive understanding of complex scenes. DANet [34] designs a dual self-attention module, *i.e.*, position attention module and channel attention module, to capture the semantic interdependencies in spatial and channel dimensions respectively.

Furthermore, NonLocal Net [117] presents non-local operations, which computes the response at a position as a weighted sum of the features at all positions, to capture long-range dependencies. Self-attention in [113] can be viewed as a form of the non-local operation. Later, several work [63, 51, 156, 11, 140] modify the dot-product based attention from non-local block [117] to largely reduce the computation and memory consumption. EMANet [63] reformulates the self-attention mechanism into an expectation-maximization iteration manner, which can learn a more compact basis set to largely reduce the computational complexity. CCNet [51] proposes the criss-cross attention module which aggregates contextual information in horizontal and vertical directions (*i.e.*, criss-cross path). By taking a further recurrent operation, each pixel can finally capture the full-image dependencies from all pixels. It is GPU memory friendly and computation efficient comparing to the non-local blocks [117]. ANN [156] embeds a pyramid pooling module [152] into the non-local block [117] to largely reduce the computation and memory consumption, which is called as APNB. Then the AFNB adapts APNB to fuse the features of different stages of a deep network, bringing a considerable improvement over the baseline model. GCNet [11] observes the attention maps (from non-local network [117]) for different query positions are almost the same, indicating only query-independent dependency is learned. Thus, they simplify the non-local block by explicitly using a query-independent attention map for all query positions, which is significantly less computation but is observed with almost no decrease in accuracy. DNLNet [140] disentangles the dot-product based attention from non-local block [117] into two terms: a whitened pairwise term that accounts for the impact of one pixel specifically on another pixel, and a unary term that represents the influence of one pixel generally over all the pixels. After joint learning of them, the whitened pairwise term captures within-region relationships while the unary term learns salient boundaries.

2.2 3D Scene Understanding

Recently, there has been an increasing interest in 3D point cloud scene understanding because the deployment of all kinds of 3D sensors makes point clouds based 3D data easily accessible, and also due to its wide range of applications in 3D modeling, autonomous driving, robotics, building modeling, etc. Given that the point cloud data has the irregular and unordered format, it is non-trivial to utilize off-the-shelf deep learning technologies to learn the effective point cloud representation for scene understanding. Recent methods usually explore those representation learning from the perspectives of either voxelization, single/multi-view projections, or vanilla point-based operations, which will be fully investigated in the following.

2.2.1 Voxelized Representation Learning

They [19, 103, 132, 144, 155] first voxelize point clouds to regular grids and process with 3D convolutions. FusionNet [144] first proposes to aggregate neighborhood features by both Voxel and Point Feature Aggregation, and then design the Inner-Voxel Aggregation for fine-grain point-level feature aggregations and label predictions to address the ambiguous/wrong predictions when voxels consist of points from different classes. Cylinder3D [155] introduces the cylindrical partition and asymmetrical 3D convolution networks to explore the 3D geometric pattern and tackle these difficulties caused by sparsity and varying density of point clouds. Moreover, a point-wise refinement module is employed to alleviate the interference of lossy voxel-based label encoding. SPVNAS [103] proposes Sparse Point-Voxel Convolution (SPVConv), which is a lightweight 3D module consisting of the vanilla Sparse Convolution and the high-resolution point-based branch. Furthermore, 3D Neural Architecture Search (3D-NAS) is presented to obtain the efficient and effective architecture for semantic segmentation. AF2S3Net [19] designs an AF2M to capture the global context and local details and an AFSM to learn inter-relationships between channels across multi-scale feature maps from AF2M. RPVNet [129] propose a novel Range-Point-Voxel fusion network with multiple and mutual information interactions among these three views, and design a gated fusion module (GFM) to adaptively merge the three features based on concurrent inputs. The proposed fusion strategy uses points as middle hosts to bidirectional propagate features between range-pixels, Voxel-cells and vanilla points. However, the distributions of large-scale outdoor scenes (*e.g.*, SemanticKITTI [7]) are extremely sparse, and the computations grow cubically when increasing the voxel resolution.

2.2.2 Projected Representation Learning

Point clouds are first projected to 2D images, *e.g.*, range-view (RV) [5, 26, 79, 120, 121, 127] and bird's-eye-view (BEV) [148], and then processed using 2D convolutions. For example, MVCNN [101] first projects 3D point clouds into 2D images from multiple perspectives, and then employ CNN to extract features for each view, following a view pooling layer to aggregate features from multi-views, resulting in a multi-view convolutional neural network. MVPNet [53] aggregates 2D multi-view RGB image features into 3D point clouds, and then uses a point based network to fuse the features in 3D canonical space to predict 3D semantic labels. RangeNet++ [79] adopts a DarkNet [95] as the backbone to process RV images, and uses a KNN for post-processing. SqueezeSeg [120] takes range-images as input, which are transformed from LiDAR point clouds by spherical projection and directly outputs a point-wise label map, which is then refined by a conditional random field (CRF) implemented as a recurrent layer. SalsaNext [26] introduces a new context module that utilizes a residual dilated convolution stack to merge receptive fields at different scales. 3D-MiniNet [5] proposes a learning-based projection module for extracting local and global information from 3D data, which is then inputted into a 2D FCNN with range images to produce semantic segmentation predictions. In contrast, PolarNet [148] employs a polar-based birds-eye-view (BEV) instead of the standard 2D Cartesian BEV projections to more effectively capture the imbalanced spatial distribution of point clouds.

Among projection-based methods, applying multi-view projection can leverage rich complementary information [4, 12, 36, 69], while previous works usually process RV/BEV individually in sequence [12, 36] or perform a vanilla late fusion [4, 69]. For example, MVLidarNet [12] first obtains predictions from the RV image, which are then projected to BEV as initial features to learn representation by feature pyramid networks. Differently, TornadoNet [36] conducts in reverse order by devising a pillar-projection-learning module (PPL) to extract features from BEV, and then placing these features into RV, modeled by an encoder-decoder CNN. On the other side, MPF [4] utilizes two different models to separately process RV and BEV, and then combines the predicted softmax probabilities from two branches as final predictions. AMVNet [69] takes a further step, *i.e.*, after obtaining the separate predictions from RV and BEV, it adopts a point head [83] to refine the uncertain predictions, which are defined by the disagreements of two branches.

2.2.3 Vanilla Representation Learning

Recent methods mainly devise novel point operations/architectures to directly learn vanilla representations from raw points [48, 66, 83, 84, 108, 118, 105], including mlp-based [48, 83, 84], cnn-based [66], graph-based methods [108, 118], and transformer-based [41, 58, 81, 151]. Specifically, PointNet [83] is a pioneering network architecture that directly takes point clouds as input and outputs either class labels for the entire input or per point segment/part labels for each point of the input. To deal with the permutation invariance of point clouds, PointNet proposes to adopt multi-layer perceptron (MLP) to extract features followed by a single symmetric function *i.e.*, max pooling. For transformation invariance, they introduce a data-dependent spatial transformer network that attempts to canonicalize the data before the PointNet processes them. PointNet++ [84] introduces a hierarchical neural network that applies PointNet recursively on a nested partitioning of the input point set. By exploiting metric space distances, PointNet++ learns local features with increasing contextual scales and adaptively combines features from multiple scales. PointCNN [66] learns an X -transformation from the input points to simultaneously re-weight and arrange the associated features of each point. Element-wise product and sum operations of the typical convolution operator are subsequently applied on the X -transformed features to leverage neighbor correlations. SO-Net [60] handles the permutation invariance of orderless point clouds by building a Self-Organizing Map (SOM) to fix the order of points. DGCNN [118] proposes a new neural network module called EdgeConv, which can better capture local geometric features of point clouds while still maintaining permutation invariance, and act on graphs dynamically computed in each layer of the network. RandLA-Net [48] identities random point sampling instead of more complex point selection approaches as the most suitable component for efficient learning on large-scale point clouds. Additional, it proposes an effective local feature aggregation module that preserves complex local structures by progressively increasing the receptive field for each point. KPConv [108] introduces a new point convolution operator named Kernel Point Convolution (KPConv) to directly takes points in the neighborhood as input and processes them with spatially located weights. KPConv favors radius neighborhoods instead of k-nearest-neighbors (KNN), which is not robust in non-uniform sampling settings as in [47]. Furthermore, a deformable version of this convolution operator was also introduced that learns local shifts to make them adapt to point cloud geometry.

Recently, the popular transformer architecture has been introduced to point cloud tasks [58, 81, 124, 151]. PCT [41] presents the offset-attention mechanism, which replaces the original self-attention. PTv1 [151] proposes vector attention to aggregate neighbor features. PTv2 [124] further introduces grouped vector attention to more efficiently learn discriminative representations while avoiding overfitting. Stratified Transformer [58] employs local window-based self-attention and captures long-range contexts by sampling nearby points densely and distant points sparsely.

CHAPTER 3

Multi-View Composite Representation Learning

This chapter aims to gain a comprehensive understanding of the intricate spatial structure of 3D point clouds through the lens of data transformation. In particular, the approach involves transforming the point cloud data into multiple distinct views and subsequently learning a composite representation that encompasses all these views.

Different views capture different information of point clouds and thus are complementary to each other. However, recent projection-based methods for point cloud semantic segmentation usually utilize a vanilla late fusion strategy for the predictions of different views, failing to explore the complementary information from a geometric perspective during the representation learning. In this chapter, we introduce a geometric flow network (GFNet) to explore the geometric correspondence between different views in an align-before-fuse manner. Specifically, we devise a novel geometric flow module (GFM) to bidirectionally align and propagate the complementary information across different views according to geometric relationships under the end-to-end learning scheme. We perform extensive experiments on two widely used benchmark datasets, SemanticKITTI and nuScenes, to demonstrate the effectiveness of our GFNet for project-based point cloud semantic segmentation. Concretely, GFNet not only significantly boosts the performance of each individual view but also achieves state-of-the-art results over all existing projection-based models. Code is available at `https://github.com/haibo-qiu/GFNet`.

3.1 Introduction

3D point cloud analysis has drawn increasing attention from both academic and industrial communities, since the wide deployments of lidar sensors have made it possible to obtain abundant 3D point cloud data [7, 10]. Compared to 2D images (*e.g.*, RGB images), a point cloud can capture precise structures of objects, thus providing a geometry-accurate perspective representation, intrinsically in line with the 3D real world. Point cloud semantic segmentation, aiming to assign a semantic label to each point, is fundamental to scene understanding, which enables intelligent agents to precisely perceive not only the objects but also the dynamically changing environment. Therefore, point cloud semantic segmentation plays a crucial role, especially in safety-critical applications such as autonomous driving [67, 2] and robotics [64, 136].

Unlike structural pixels in an image, a point cloud is a set of points represented by (x, y, z) coordinates without a specific order, and extremely sparse for in-the-wild scenes. Hence, it is non-trivial to utilize off-the-shelf deep learning technologies on images for point cloud analysis. Recent point cloud segmentation methods usually address the above-mentioned sparse distributed issue from the perspectives of either voxelization, single/multi-view projections, or novel point-based operations. However, voxel-based methods mainly suffer from heavy computations while point-based operations struggle to efficiently capture the neighbour information, especially when dealing with large-scale outdoor scenes [7, 10]. With the great success of fully convolutional networks for image-based semantic segmentation [13, 14, 74, 152], projection-based methods have recently received increasing attention. Figure 3.1 illustrates two widely used projected views, *i.e.*, range-view (RV) [79] and bird's-eye-view (BEV) [148]. Single view based methods can only learn view-specific representations [5, 26, 127], failing to handle those occluded points during projection. For example, the RV in Figure 3.2 shows a occluded tail phenomenon (*i.e.*, the distant occluded

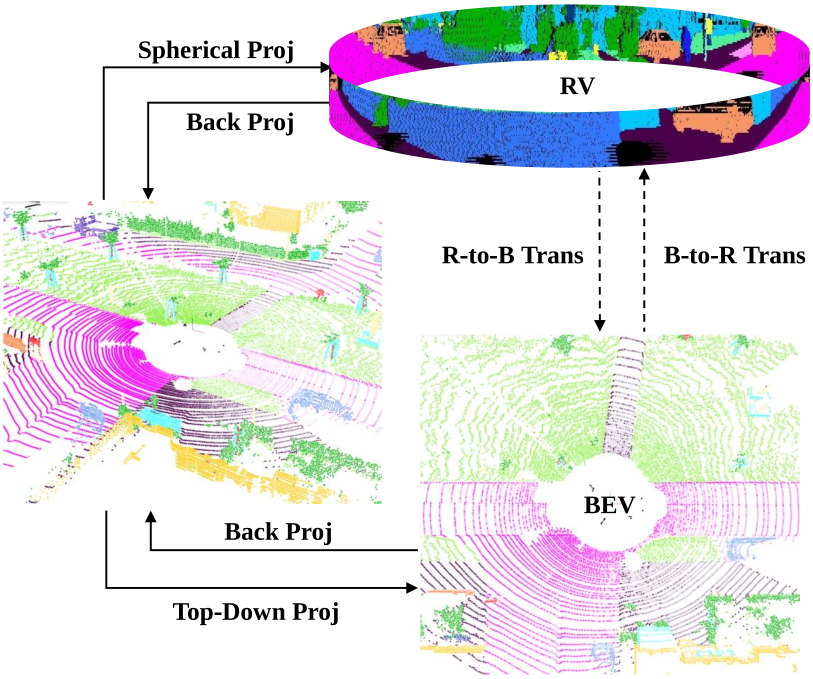


FIGURE 3.1. Geometric bidirectional transformation diagram between range-view (RV) and bird's-eye-view (BEV).

points are assigned with the labels of near displayed points) in the red rectangle areas. To address this problem, recent methods resort to multi-view models to incorporate complementary information over different views, which usually deal with RV/BEV in sequence [12, 36] or perform a vanilla late fusion [4, 69]. However, existing methods fail to probe the intrinsically geometric connections of RV/BEV during the representation learning.

As we can see from Figure 3.1, to find the geometric correspondence between two views (the dash line), we can utilize the original point cloud as a bridge, *e.g.*, the transformation RV to BEV can be obtained from two transformations (the solid line): 1) from RV to point cloud; and 2) from point cloud to BEV. Inspired by this, we introduce a novel geometric flow network (GFNet) to simultaneously learn view-specific representations and explore the geometric correspondences between RV and BEV in an end-to-end learnable manner. Specifically, we first propose to adopt two branches to process RV and BEV inputs, where each branch follows an encoder-decoder architecture using a ResNet [46] as the backbone. We then devise a geometric flow module (GFM), which is then applied at multiple levels of feature representations, to bidirectionally align and propagate geometric information across two projection views, aiming to learn more discriminative representations. Figure 3.2 illustrates an example of propagating the information from BEV to RV which benefits handling those occluded points by RV projection. In addition, inspired by [55], we also use KPConv [108] at the top of GFNet to replace a KNN post-processing, thus making it easy to train the overall multi-view point cloud semantic segmentation pipeline in an end-to-end paradigm. The main contributions of this chapter are summarized as follows:

- We introduce a novel GFNet to simultaneously learn and fuse multi-view representations, where the proposed geometric flow module (GFM) enables the geometric correspondence information to flow across different views.
- We devise two branches for RV and BEV with KNN post-processing replaced by KPConv, making the proposed GFNet end-to-end trainable.
- Extensive experiments are performed on two popular large-scale point cloud semantic segmentation benchmarks, *i.e.*, SemanticKITTI and nuScenes, to demonstrate the effectiveness of GFNet, which achieves state-of-the-art performance over all existing projection-based models.

3.2 Related Work

In this section, we review recent point cloud semantic segmentation literature from the perspectives of point-based, voxel-based, and projection-based methods. In addition, we discuss more recently hybrid methods which simultaneously use multiple formats/modalities. Among all projection-based methods, we mainly focus on the multi-view projection-based methods.

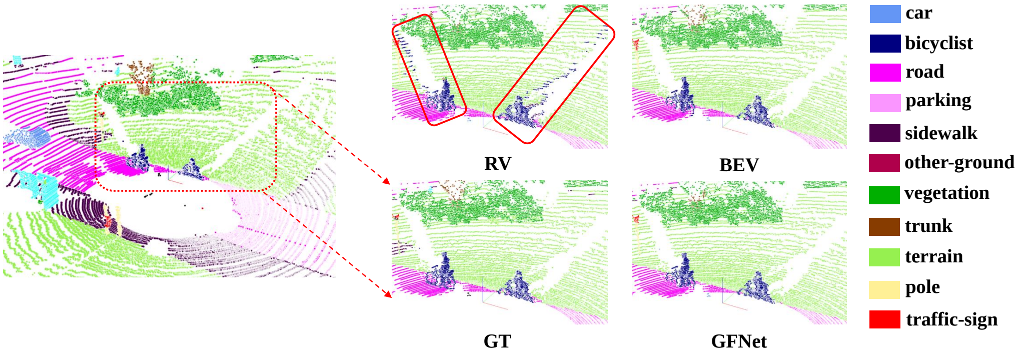


FIGURE 3.2. The distant occluded points caused by RV projection are misclassified as the labels of near displayed points in the red rectangle areas, while they are totally captured by BEV. By propagating the information between BEV and RV, this issue can be well addressed by our GFNet.

Point-based Methods. Recent methods mainly devise novel point operations or architectures to directly learn representations from raw points [48, 66, 83, 84, 108, 118, 105], including mlp-based [48, 83, 84], cnn-based [66], and graph-based methods [108, 118]. Specifically, PointNet [83] is a pioneer that directly processes point cloud with multi-layer perceptron (MLP), which is improved by PointNet++ [84] using a hierarchical neural network to learn local features. PointCNN [66] learns a X-transformation from the input points for alignment, followed by typical convolution layers. DGCNN [118] proposes a new graph convolution module called EdgeConv to capture local geometric features. RandLA-Net [48] employs random point sampling with an effective local feature aggregation module to persevere the local information. KPConv [108] introduces a new point convolution operator named Kernel Point Convolution to directly take neighbouring points as input and processes with spatially located weights. Nevertheless, the irregular and disordered characteristics of point clouds make it inefficient to capture the neighbour information.

Voxel-based Methods. They [19, 103, 132, 144, 155] first voxelize point clouds to regular grids and process with 3D convolutions. Cylinder3D [155] introduces the cylindrical partition and asymmetrical 3D convolution networks to tackle the issues of sparsity and varying density of point clouds. SPVNAS [103] proposes Sparse Point-Voxel Convolution (SPVConv), which is a lightweight 3D module consisting of the vanilla Sparse Convolution and the high-resolution point-based branch. Furthermore, 3D Neural Architecture Search (3D-NAS) is presented to obtain the efficient and effective architecture for semantic segmentation. AF2S3Net [19] designs an AF2M to capture the global context and local details and an AFSM to learn inter-relationships between channels across multi-scale feature maps from AF2M. However, the distributions of large-scale outdoor scenes (*e.g.*, SemanticKITTI [7]) are extremely sparse, and the computations grow cubically when increasing the voxel resolution.

Hybrid Methods. Recent methods [129, 138, 133] usually focus on simultaneously using multiple formats/modalities (*e.g.*, voxel, points or natural images) to learn discriminative representations. DRINet++ [138] proposes Sparse Feature Encoder to extract local context information from voxelized grids, and Sparse Geometry Feature Enhancement to enhance the geometric characteristics of sparse points using multi-scale sparse projection and attentive multi-scale fusion. RPVNet [129] explores multiple and mutual information interactions among three views (*i.e.*, projection, voxel and point), following by a gated fusion module to adaptively merge the three features based on concurrent inputs. 2DPASS [133] assists raw points with 2D natural images. It distills richer semantic and structural information from 2D images without strict paired data constraints to the pure 3D point network, by leveraging an auxiliary modal fusion and multi-scale fusion-to-single knowledge distillation (MSFSKD).

Projection-based Methods. Point clouds are first projected to 2D images, *e.g.*, range-view (RV) [5, 26, 79, 120, 121, 127] and bird's-eye-view (BEV) [148], and then processed using 2D convolutions. For example, RangeNet++ [79] adopts a DarkNet [95] as the backbone to process RV images, and uses a KNN for post-processing. SqueezeSegV3 [127], standing on the shoulders of [120, 121], employs a spatially-adaptive Convolution (SAC) to adopt different filters for different locations according to input RV images. SalsaNext [26] introduces a new context module which consists of a residual dilated convolution stack to fuse receptive fields at various scales. On the other hand, PolarNet [148] uses a polar-based birds-eye-view (BEV) instead of the standard 2D Cartesian BEV projections to better model the imbalanced spatial distribution of point clouds.

Among projection-based methods, applying multi-view projection can leverage rich complementary information [4, 12, 36, 69], while previous works usually process RV/BEV individually in sequence [12, 36] or perform a vanilla late fusion [4, 69]. For example, MVLidarNet [12] first obtains predictions from the RV image, which are then projected to BEV as initial features to learn representation by feature pyramid networks. Differently, TornadoNet [36] conducts in reverse order by devising a pillar-projection-learning module (PPL) to extract features from BEV, and then placing these features into RV, modeled by an encoder-decoder CNN. On the other side, MPF [4] utilizes two different models to separately process RV and BEV, and then combines the predicted softmax probabilities from two branches as final predictions. AMVNet [69] takes a further step, *i.e.*, after obtaining the separate predictions from RV and BEV, it adopts a point head [83] to refine the uncertain predictions, which are defined by the disagreements of two branches. Whereas, our GFNet enables geometric correspondence information to flow between RV/BEV at multi-levels during end-to-end learning, leading to a more discriminative representation and better performances.

3.3 Method

In this section, we first provide an overview of point cloud semantic segmentation and the proposed geometric flow network (GFNet). We then introduce projection-based point cloud segmentation using range-view (RV) and bird's-eye-view (BEV) in detail. After that, we describe the proposed geometric flow module

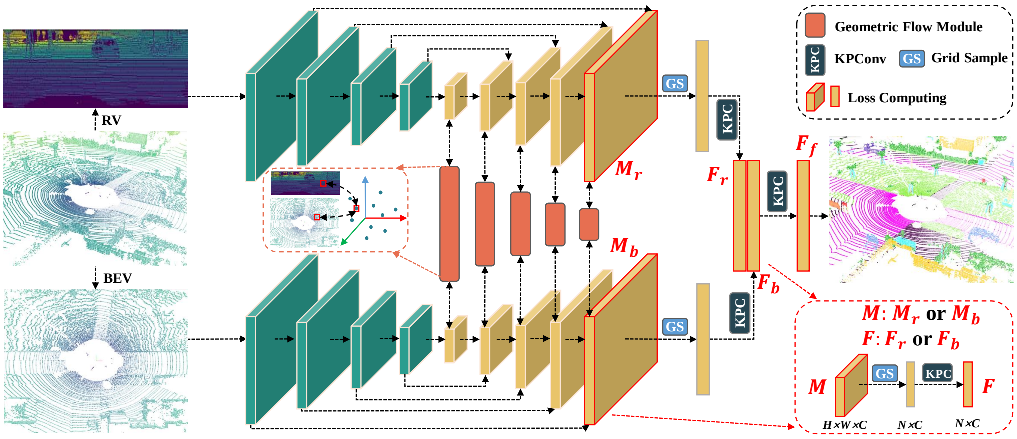


FIGURE 3.3. The overview of geometric flow network (GFNet). Point clouds are first projected to range-view (RV) and bird's-eye-view (BEV) using spherical and top-down projections, respectively. Then two branches with the proposed geometric flow module (GFM) handle RV/BEV to generate feature maps ($H \times W \times C$). Finally, grid sampling based on corresponding projection relationships is utilized to get the probability ($N \times C$) for each point, and the fused prediction $\mathbf{F_f}$ is obtained by applying kpconv on the concatenation of RV/BEV. Note that the subplot in bottom right corner illustrates how grid sampling works with dimension changing annotation, *i.e.*, sampling $\mathbf{F}$ ($\mathbf{F_r}$ or $\mathbf{F_b}$) with $N \times C$ from $\mathbf{M}$ ($\mathbf{M_r}$ or $\mathbf{M_b}$) with $H \times W \times C$.

(GFM), including geometric alignment and attention fusion. Lastly, the end-to-end optimization of GFNet is depicted.

3.3.1 Overview

Given a lidar point cloud with N 3D points $\mathbf{P} \in \mathbb{R}^{N \times 4}$, we then have the format of each point as $(x, y, z, remission)$, where (x, y, z) is the cartesian coordinate of the point relative to the lidar sensor and $remission$ indicates the intensity of returning laser beam. The goal of point cloud semantic segmentation is to assign all points in $\mathbf{P}$ with accurate semantic labels, *i.e.*, $\mathbf{Q} \in \mathbb{N}^N$. For projection-based point cloud semantic segmentation, we also need to transform the ground truth labels $\mathbf{Q}$ to the projected views during training, *i.e.*, $\mathbf{Q}_r$ for RV and $\mathbf{Q}_b$ for BEV.

The overall pipeline of GFNet is illustrated in Figure 3.3. Specifically, a point cloud $\mathbf{P}$ is first transformed to range-view (RV) as $\mathbf{I}_r$ and bird's-eye-view (BEV) as $\mathbf{I}_b$ using spherical and top-down projections, respectively. We then have two sub-network branches with encoder-decoder architectures to take RV/BEV images as inputs and generate dense predictions, which are referred to as the probability maps for each semantic class. The proposed geometric flow module (GFM) is incorporated into each layer of the decoder, bidirectionally propagating feature information according to the geometric correspondences across two views. After that, we obtain the classification probabilities of all points by applying a grid sampling on the dense probability maps, which is based on the projection relationship between a specific view and the original point cloud, as illustrated in the bottom right corner of Figure 3.3. Inspired by [55], we also introduce KPConv [108] on the top of the proposed GFNet to replace the KNN operation and capture the accurate neighbour information in a learnable way. By doing this, the overall multi-view point cloud semantic segmentation pipeline can be trained in an end-to-end manner.

3.3.2 Multi-View Projection

For projection-based methods, a point cloud $\mathbf{P} \in \mathbb{R}^{N \times 4}$ needs to be transformed to an image $\mathbf{I} \in \mathbb{R}^{HW \times C}$ first to leverage deep neural networks primarily developed for 2D visual recognition, where H and W indicate the spatial size of projected images and C is the number of channels. Different projections are corresponding to different transformations, *i.e.*, $\mathcal{P} : \mathbb{R}^{N \times 4} \mapsto \mathbb{R}^{HW \times C}$. In this chapter, we adopt two widely-used projected views for point cloud analysis, *i.e.*, range-view (RV) and bird's-eye-view (BEV). As shown in Figure 3.3, we aim to learn effective representations from two different views, RV and BEV, using the proposed two-branch networks with an encoder-decoder architecture in each branch. We describe the details of multi-view projection as follows.

Range-View (RV). To learn effective representations from RV images, spherical projection is required to first project a point cloud $\mathbf{P}$ to a 2D RV image [79]. Specifically, we first project a point (x, y, z) from the cartesian space to the spherical space as follows:

$$\begin{bmatrix} \psi \\ \phi \\ r \end{bmatrix} = \begin{bmatrix} arctan(y, x) \\ arcsin(z/\sqrt{x^2 + y^2 + z^2}) \\ \sqrt{x^2 + y^2 + z^2} \end{bmatrix}, \tag{3.1}$$

where ψ, ϕ, and r indicate azimuthal angle, polar angle, and radial distance (*i.e.*, the range of each point), respectively. We then have the pixel coordinate of (x, y, z) in the projected 2D range image as

$$\begin{bmatrix} \widetilde{u} \\ \widetilde{v} \end{bmatrix} = \begin{bmatrix} (1 - \psi/\pi)/2 \cdot W \\ (f_{up} - \phi)/f \cdot H \end{bmatrix}, \tag{3.2}$$

where (H, W) represent the spatial size of range image, and $f = f_{up} - f_{down}$ is the vertical field-of-view of the lidar sensor. For each projected pixel (u, v) (discretized from $(\widetilde{u}, \widetilde{v})$), we take the $(x, y, z, r, remission)$ as its feature, leading to a range image with the size of $(H, W, 5)$. In addition, an improved range-projected method is proposed by [111], which further unfolds the point clouds following the captured order by the lidar sensor, leading to smoother projected images and a higher valid projection rate[1]. If not otherwise stated, we adopt this improved range projection [111] in all our experiments.

Bird's-Eye-View (BEV). To learn effective representations from BEV images, top-down orthogonal projection is employed to transform a point cloud into a BEV image [17]. Furthermore, the polar coordinate system is introduced to

[1] Please refer to Appendix. 3.6.1 for more details.

replace the cartesian system by [148], which can be formulated as follows:

$$\begin{bmatrix} \widetilde{u} \\ \widetilde{v} \end{bmatrix} = \begin{bmatrix} \sqrt{x^2 + y^2}\cos(\arctan(y,x)) \\ \sqrt{x^2 + y^2}\sin(\arctan(y,x)) \end{bmatrix} = polar(x,y), \tag{3.3}$$

where $polar(\cdot)$ is the coordinate transformation from cartesian system to polar system. Following [148], we use nine features to describe each pixel (u, v) (by discretizing $(\widetilde{u}, \widetilde{v})$ to $[0, H-1]$ and $[0, W-1]$) in BEV image, including three relative cylindrical distance, three cylindrical coordinates, two cartesian coordinates and one remission, which can be constructed as follows:

$$[\Delta cylindrical(x,y,z),\ cylindrical(x,y,z),\ x,\ y,\ remission)], \tag{3.4}$$

where $cylindrical(x,y,z) = [polar(x,y), z]$ represents the cylindrical coordinates, $\Delta cylindrical(x,y,z)$ are the relative distances to the center of the BEV grid, and each BEV image thus has the shape of $(H, W, 9)$.

3.3.3 Geometric Flow Module

Intuitively, RV and BEV contain different view information of the original point cloud through different projections, leading to different information loss on different classes. For example, RV is good at those tiny or vertically-extended objects such as *motorcycle* and *person*, while BEV is sensitive to those objects with large and discriminative spatial size on the x-y plane. To sufficiently investigate the complementary information from RV/BEV, we explore them from a geometric perspective. Specifically, we devise a geometric flow module (GFM), which is based on the geometric correspondences between RV and BEV, to bidirectionally propagate the complementary information across different views. As illustrated in Figure 3.4, the first step is referred to as **Geometric Alignment**, which aligns the feature of source view (RV or BEV) to the target view using their geometric transformation; then the second step is called **Attention Fusion**, which applies the self-attention and the residual connection to combine the aligned

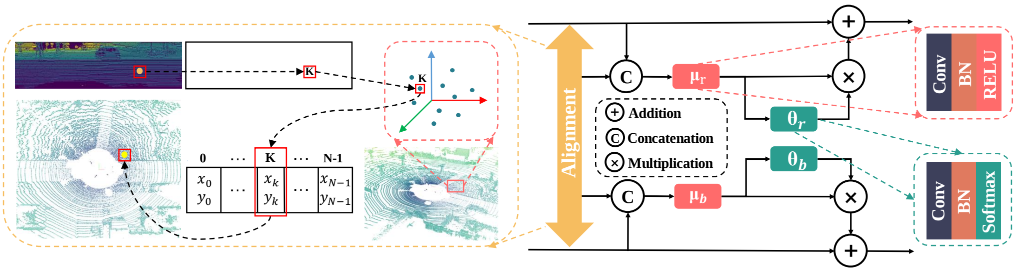


FIGURE 3.4. An overview of the proposed geometric flow module (GFM). It contains two main steps, *i.e.*, geometric alignment and attention fusion, which first aligns the feature from the source view (RV or BEV) to the target view using their geometric correspondences, and then applies self-attention and residual connections to combine view-specific features with the flowed information. Note that μ_r and θ_r share the same architecture but not weights with μ_b and θ_b, respectively.

feature representation with the original one. We describe the above-mentioned key steps of the proposed GFM module in detail as follows.

Geometric Alignment. The key idea lies in the geometric transformation matrices between two views, *i.e.*, $\mathbf{T}_{R\to B}$ (from RV to BEV) and $\mathbf{T}_{B\to R}$ (from BEV to RV). To obtain these transformation matrices, we propose to utilize the original point cloud as an intermediary agent. Specifically, from Eq.(3.1) and (3.2), we have the transformation from RV to the point cloud $\mathbf{P}$ as follows:

$$\mathbf{T}_{R\to P} = \begin{bmatrix} n_{0,0} & \cdots & n_{0,W_r-1} \\ \vdots & \vdots & \vdots \\ n_{H_r-1,0} & \cdots & n_{H_r-1,W_r-1} \end{bmatrix}, \tag{3.5}$$

where $\mathbf{T}_{R\to P} \in \mathbb{Z}^{H_r\times W_r}$, (H_r, W_r) are the spatial size of 2D RV image, and $\{(n_{i,j})|\ 0 <= i <= H_r - 1,\ 0 <= j <= W_r - 1\}$ is the $(n_{i,j})_{th}$ point which projects on (i, j) coordinates. Note that if multiple points project to the same pixel, the point with smaller range is kept; If a pixel is not projected by any points, then its $n_{i,j}$ is assigned as -1. We then have the transformation from $\mathbf{P}$

to BEV image according to Eq.(3.3):

$$\mathbf{T}_{P\to B} = \begin{bmatrix} \mathbf{u}_0 & \cdots & \mathbf{u}_{N-1} \end{bmatrix}^T \tag{3.6}$$

$$= \begin{bmatrix} u_0 & \cdots & u_{N-1} \\ v_0 & \cdots & v_{N-1} \end{bmatrix}^T, \tag{3.7}$$

where $\mathbf{T}_{P\to B} \in \mathbb{Z}^{N\times 2}$, and $\{\mathbf{u}_k = (u_k, v_k) |\, 0 <= k <= N-1\}$ are the projected pixel coordinates of 2D BEV image, corresponding to the k_{th} point.

Algorithm 1: Geometric Flow Module (BEV → RV)

```
**Input:** RV feature map $M_r : [H_r, W_r, C_r]$, BEV feature map
$M_b : [H_b, W_b, C_b]$, $\mathbf{T}_{R\to B} : [H_r, W_r, 2]$.
**Output:** Fused RV feature map $M^r_{fused} : [H_r, W_r, C_r]$.
**Step 1: Geometric Alignment**
    • Zero-initializing aligned feature $M_{b\to r}$ with the shape of
      $[H_r, W_r, C_b]$;
    • **foreach** $(i, j) \in [1 : W_r] \times [1 : H_r]$ **do**
          $\mathbf{u} = \mathbf{T}_{R\to B}[i, j]$
          $u, v = \mathbf{u}$
          $M_{b\to r}[i, j, :] = M_b[u, v, :]$
**Step 2: Attention Fusion**
    • Concatenating $M_r$ and $M_{b\to r}$ along the channel dimension as
      $M_{concat} : [H_r, W_r, C_r + C_b]$
    • Applying the self-attention module to get
      $M_{atten} = \mu(M_{concat}) \cdot \theta(\mu(M_{concat}))$ with the shape of $[H_r, W_r, C_r]$
    • Employing residual connection $M^r_{fused} = M_r + M_{atten}$
```

Lastly, we calculate the transformed matrix $\mathbf{T}_{R\to B}$ via $\mathbf{T}_{R\to P}$ and $\mathbf{T}_{P\to B}$. In particular, for each pixel (i, j) in RV image, we first get its 3D point $n_{i,j} = \mathbf{T}_{R\to P}[i, j]$, then project $n_{i,j}$ to BEV image to obtain the corresponding pixel $\mathbf{T}_{P\to B}[n_{i,j}] = \mathbf{u}_{n_{i,j}}$. Now, we obtain $\mathbf{T}_{R\to B} \in \mathbb{Z}^{H_r\times W_r\times 2}$ as:

$$\mathbf{T}_{R\to B} = \begin{bmatrix} \mathbf{u}_{n_{0,0}} & \cdots & \mathbf{u}_{n_{0,W_r-1}} \\ \vdots & \vdots & \vdots \\ \mathbf{u}_{n_{H_r-1,0}} & \cdots & \mathbf{u}_{n_{H_r-1,W_r-1}} \end{bmatrix}, \tag{3.8}$$

Once obtaining $\mathbf{T}_{R\to B}$, we can then align BEV features to RV features as follows: for each location (i, j) in RV image, the (u, v) coordinates in BEV image can be fetched via $\mathbf{T}_{R\to B}[i, j]$, and then we fuse the feature in (u, v) to (i, j) to get aligned feature $F_{b\to r}$. To align features from RV to BEV, we can operate in a similar way with $\mathbf{T}_{B\to R} \in \mathbb{Z}^{H_b \times W_b \times 2}$.

Attention Fusion. After the geometric feature alignment, we employ an attention fusion module to obtain the fused feature by concatenating the aligned feature and the target feature, which is followed by two convolution operations $\mu(\cdot)$ and $\theta(\cdot)$. They have simple architectures "Conv-BN-RELU" and "Conv-BN-Softmax" respectively, where the softmax function in θ is to map values to $[0, 1]$ as attention weights. The attention feature is finally combined with the target feature using a residual connection. We demonstrate the overall process of fusing BEV to RV, including geometric alignment and attention fusion modules, in Algorithm 1. The geometric flow from RV to BEV can be calculated similarly.

3.3.4 Optimization

Given $\mathbf{Q_r}$ as the labels for RV image $\mathbf{I_r}$ and $\mathbf{Q_b}$ for BEV image $\mathbf{I_b}$, which are projected from the original point cloud label $\mathbf{Q}$, we then have the 2D predictions $\mathbf{M_r}$ for RV and $\mathbf{M_b}$ for BEV, respectively. After that, we obtain the 3D predictions via grid sampling and KPConv, *i.e.*, $\mathbf{F_r}$ for RV and $\mathbf{F_b}$ for BEV. After fusion, we get the final 3D predictions $\mathbf{F_f}$. For simplicity and better illustration, we also highlight all predictions, *i.e.*, $\mathbf{M_r}, \mathbf{M_b}$ and $\mathbf{F_r}, \mathbf{F_b}, \mathbf{F_f}$, in Figure 3.3. To train the proposed GFNet, we first use the loss functions $\mathcal{L}_{2D}$ and $\mathcal{L}_{3D}$ for 2D and 3D predictions, respectively, as follows:

$$\mathcal{L}_{2D} = \rho \cdot \mathcal{L}_{CL}(\mathbf{M_r}, \mathbf{Q_r}) + \sigma \cdot \mathcal{L}_{CL}(\mathbf{M_b}, \mathbf{Q_b}), \tag{3.9}$$

and

$$\mathcal{L}_{3D} = \beta \cdot \mathcal{L}_{CE}(\mathbf{F_r}, \mathbf{Q}) + \gamma \cdot \mathcal{L}_{CE}(\mathbf{F_b}, \mathbf{Q}), \tag{3.10}$$

where $\mathcal{L}_{CE}$ indicates the typical cross entropy loss function while $\mathcal{L}_{CL}$ is the combination of the cross entropy loss and the Lovasz-Softmax loss [8] with weights $1:1$. We then apply the cross entropy loss on the final 3D predictions $\mathbf{F_f}$, that is, the overall loss function $\mathcal{L}_{total}$ can be evaluated as:

$$\mathcal{L}_{total} = \alpha \cdot \mathcal{L}_{CE}(\mathbf{F_f}, \mathbf{Q}) + \mathcal{L}_{3D} + \mathcal{L}_{2D}, \tag{3.11}$$

where $\lambda \doteq [\alpha, \beta, \gamma, \rho, \sigma]$ indicates the weight coefficient of different losses, and we investigate the influences of different loss terms in Sec. 3.4.4.

3.4 Experiments

In the section, we first introduce the adopted SemanticKITTI [7] and nuScenes [10] datasets and the mean IoU and accuracy metric for point cloud segmentation. We then provide the implementation details of GFNet, including the network architectures and training settings. After that, we perform extensive experiments to demonstrate the effectiveness of GFM and analyze the influences of different hyper-parameters in GFNet. Lastly, we compare the proposed GFNet with recent state-of-the-art point/projection-based methods to show our superiority.

3.4.1 Datasets and Evaluation Metrics

SemanticKITTI [7], derived from the KITTI Vision Benchmark [35], provides dense point-wise annotations for semantic segmentation task. The dataset presents 19 challenging classes and contains 43551 lidar scans from 22 sequences collected with a Velodyne HDL-64E lidar, where each scan contains approximately 130k points. Following [7, 79], these 22 sequences are divided into 3 sets, *i.e.*, training set (00 to 10 except 08 with 19130 scans), validation set (08 with 4071 scans) and testing set (11 to 21 with 20351 scans). We perform extensive experiments on the validation set to analyze the proposed method, and also report performance on the test set by submitting the result to the official test server.

nuScenes [10] is a large-scale autonomous driving dataset, containing 1000 driving scenes of 20 second length in Boston and Singapore. Specifically, all driving scenes are officially divided into training (850 scenes) and validation set (150 scenes). By merging similar classes and removing rare classes, point cloud semantic segmentation task uses 16 classes, including 10 foreground and 6 background classes. We use the official test server to report the final performance on test set.

Evaluation Metrics. Following [7], we use mean intersection-over-union (mIoU) over all classes as the evaluation metric. Mathematically, the mIoU can be defined as:

$$mIoU = \frac{1}{C}\sum_{c=1}^{C}\frac{TP_c}{TP_c + FP_c + FN_c}, \tag{3.12}$$

where TP_c, FP_c, and FN_c represent the numbers of true positive, false positive, and false negative predictions for the given class c, respectively, and C is the number of classes. For a comprehensive comparison, we also report the accuracy among all samples, which can be formulated as:

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}. \tag{3.13}$$

3.4.2 Implementation Details

For SemanticKITTI, we use two branches to learn representations from RV/BEV in an end-to-end trainable way, where each branch follows an encoder-decoder architecture with a ResNet-34 [46] as the backbone. The ASPP module [14] is also used between the encoder and the decoder. The proposed geometric flow module (GFM) is incorporated into each upsampling layer. Note that the elements of $\mathbf{T}_{R\to B}, \mathbf{T}_{B\to R}$ fed into GFM are scaled linearly according to the current flowing feature resolution. For RV branch, point clouds are first projected to a range image with the resolution $[64, 2048]$, which is sequentially upsampled bilinearly to $[64 \times 2S, 2048 \times S]$ where S is a scale factor. During training, a horizontal $1/4$ random crop of RV image, ***i.e.***, $[128S, 512S]$, is used as data augmentation. On the other hand, we adopt polar partition [148] for BEV, and use a polar grid size of $[480, 360, 32]$ to cover a space of $[radius : (3m, 50m), z : (-3m, 1.5m)]$ relative to the lidar sensor. The grid first goes through a mini PointNet [83] to obtain the maximum feature activations along the z axis, leading to a reduced resolution $[480, 360]$ for BEV branch. We employ a SGD optimizer with momentum 0.9 and the weight decay $1e-4$. We use the cosine learning rate schedule [76] with warmup at the first epoch to 0.1. The backbone network is initialized using the pretrained weights from ImageNet [28]. By default, we use $\lambda = [2.0, 2.0, 2.0, 1.0, 1.0]$ as the loss weight for Eq.3.11. We train the proposed GFNet for 150 epochs using the batch size 16 on four NVIDIA A100-SXM4-40GB GPUs with AMD EPYC 7742 64-Core Processor.

For nuScenes, we adopt [79] to project point clouds to a RV image with the resolution $[32, 1024]$ which is then upsampled bilinearly to $[32 \times 3S, 1024 \times S]$ where $S = 4$ in our experiments. Besides, a polar grid size of $[480, 360, 32]$ is used to cover a relative space of $[radius : (0m, 50m), z : (-5m, 3m)]$ for BEV branch. We train the model for total 400 epoch with batch size 56 using 8 NVIDIA A100-SXM4-40GB GPUs under AMD EPYC 7742 64-Core Processor. We adopt cosine learning rate schedule [76] with warmup at the first 10 epoch to 0.2. Other settings are kept the same with SemanticKITTI.

3.4.3 Evaluation on GFM

In this part, we show the effectiveness of the proposed geometric flow module (GFM) as well as its influences on each single branch. As shown in Figure 3.3, we denote the results from $\mathbf{F_r}$ and $\mathbf{F_b}$ as *RV-Flow* and *BEV-Flow*, respectively, in regard to the information flow between RV and BEV brought by GFM. The predictions from $\mathbf{F_f}$ (obtained by applying KPConv on the concatenation of $\mathbf{F_r}$

TABLE 3.1. Quantitative comparisons in terms of mIoU to demonstrate the effectiveness of GFM on the validation set of SemanticKITTI.

Method	car	bicycle	motorcycle	truck	other-vehicle	person	bicyclist	motorcyclist	road	parking	sidewalk	other-ground	building	fence	vegetation	trunk	terrain	pole	traffic-sign	**mIoU**
RV-Single	93.7	48.7	57.7	32.4	40.5	69.2	79.9	0.0	95.9	53.4	83.9	0.1	89.2	59.0	87.8	66.1	75.3	64.0	45.2	60.1
RV-Flow	93.8	45.0	58.8	69.9	31.6	63.6	73.8	0.0	95.6	52.9	83.6	0.3	90.3	62.1	88.0	64.3	75.8	63.2	47.4	**61.1**
BEV-Single	93.6	29.9	42.4	64.8	26.8	48.1	74.0	0.0	94.0	45.9	80.7	1.4	89.2	46.5	86.9	61.4	74.9	56.8	41.6	55.7
BEV-Flow	93.7	43.7	61.2	74.0	31.0	61.6	80.6	0.0	95.3	53.1	82.8	0.2	90.8	61.4	88.0	63.1	75.6	58.9	43.1	**61.0**
GFNet	94.2	49.7	63.2	74.9	32.1	69.3	83.2	0.0	95.7	53.8	83.8	0.2	91.2	62.9	88.5	66.1	76.2	64.1	48.3	**63.0**

and $\mathbf{F}_{\mathbf{b}}$) are actually our final results, termed as *GFNet*. Note that the above results are evaluated using $\lambda = [2, 2, 2, 1, 1]$ for Eq.3.11. In addition, we train also each single branch separately without GFM modules, i.e., using $\lambda = [0, 2, 0, 1, 0]$ and $\lambda = [0, 0, 2, 0, 1]$ for *RV-Single* and *BEV-Single*, respectively.

We compare the performances of *RV/BEV-Single* and *BEV/BEV-Flow* in Table 3.1. Specifically, we find that both RV and BEV branches have been improved by a clear margin when incorporating with the proposed GFM module, e.g., $55.7\% \rightarrow 61.0\%$ for BEV. Intuitively, RV is good at those vertically-extended objects like *motorcycle* and *person*, while BEV is sensitive to the classes with large and discriminative spatial size on the x-y plane. For example, *RV-Single* only achieves 32.4% on *truck* while *BEV-Single* obtains 64.8%, which is also illustrated by the first row of Figure 3.5 where RV predicts *truck* as a mixture of *truck, car* and *other-vehicle*, but BEV acts much well. This is partially because *truck* is more discriminative on x-y plane (captured by BEV) than vertical direction (captured by RV) compared to *car, other-vehicle*. With the information flow from BEV to RV using GFM, *RV-Flow* significantly boosts the performance

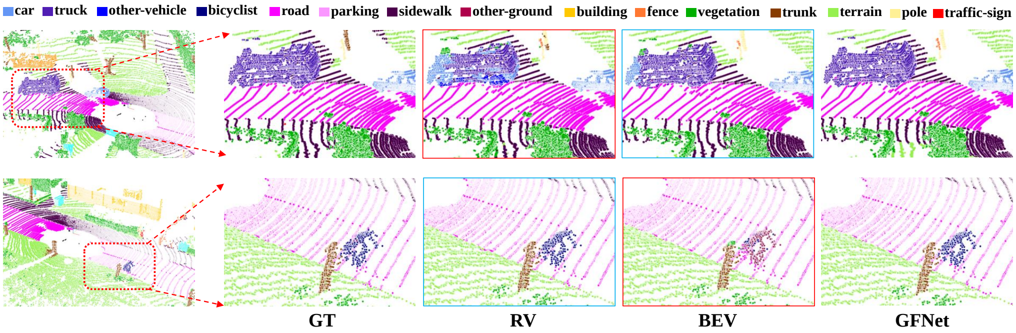


FIGURE 3.5. Visualization of RV and BEV. The view with the cyan contour helps the one with red. By incorporating both RV and BEV, our GFNet makes more accurate predictions.

from 32.4% to 69.9%. A similar phenomenon can be observed in the second row of Figure 3.5, where BEV misclassifies *bicyclist* as *trunk*, since both of them are vertically-extended and also very close to each other, while RV predicts precisely. With the help of RV, *BEV-Flow* dramatically improves the performance from 55.7% to 61.0%. When further applying KPConv on the concatenation of *RV/BEV-Flow*, the proposed *GFNet* achieves the best performance 63.0%. These results demonstrate that the proposed GFM can effectively propagate complementary information between RV and BEV to boost the performance of each other, as well as the final performance.

3.4.4 Ablation Studies

TABLE 3.2. Ablation studies of attention in GFM on the SemanticKITTI val set.

attention		mIoU
sigmoid	softmax	
		62.0
✓		62.9
	✓	63.0

TABLE 3.3. Ablation studies of attention in GFM, loss weight coefficient λ and scale factor S under $\triangle = 1, \bigtriangledown = 2$ on the SemanticKITTI val set.

cfg	α	β	γ	ρ	σ	S	mIoU
a	$\triangle$					3	61.7
b	$\triangle$	$\triangle$	$\triangle$			3	61.8
c	$\triangle$	$\triangle$	$\triangle$	$\triangle$	$\triangle$	3	62.4
d	$\bigtriangledown$	$\bigtriangledown$	$\bigtriangledown$	$\triangle$	$\triangle$	3	63.0
e	$\bigtriangledown$	$\bigtriangledown$	$\bigtriangledown$	$\triangle$	$\triangle$	2	61.7
f	$\bigtriangledown$	$\bigtriangledown$	$\bigtriangledown$	$\triangle$	$\triangle$	4	63.2

In this subsection, we first explore the impacts of attention mechanism in GFM, the loss weights λ defined in Eq.3.11; and the scale factor S introduced in Sec. 3.4.2. In the default setting, we use the softmax attention with $\lambda = [2, 2, 2, 1, 1]$ and $S = 3$.

As shown in Table 3.2, without attention mechanism (*i.e.*, no $\theta(\cdot)$ and $\otimes$ in Figure 3.4), the performance 62.0% is obviously inferior to the counterparts 62.9% and 63.0%, indicating that the attention operation helps to focus on the strengths instead of weaknesses of source view when fusing it into target view. If not otherwise stated, we use the softmax attention in our experiments. We evaluate the influences of $\lambda \doteq [\alpha, \beta, \gamma, \rho, \sigma]$ in Table 3.3, where $\triangle = 1, \bigtriangledown = 2$, e.g., we have $\lambda \doteq [\alpha, \beta, \gamma, \rho, \sigma] = [2.0, 2.0, 2.0, 1.0, 1.0]$ the configuration d. Specifically, when comparing the configurations a to b and c, we see that that additional supervisions on dense 2D and each branch RV/BEV 3D predictions further improve model performance. When comparing c and d, a large weight on 3D prediction brings a better result. Therefore, if not otherwise stated, we adopt $\lambda = [2.0, 2.0, 2.0, 1.0, 1.0]$ for remaining experiments. The scale factor S in Sec. 3.4.2 indicates the resolution of RV image, e.g., when $S = 3$, we have $[128S, 512S] = [384, 1536]$ and $[128S, 2048S] = [383, 6144]$ for training and testing, respectively. When comparing d and e in Table 3.3, we find that a higher resolution significantly improves model performance, from 61.7% to

TABLE 3.4. Comparisons under mIoU, Accuracy and Frame Per Second (FPS) on SemanticKITTI test set. Note that the results of methods with * are obtained from RangeNet++ [79]. From top to down (light green → blue → gray), the methods are grouped into point-based, projection-based and multi-view fusion models.

Method	car	bicycle	motorcycle	truck	other-vehicle	person	bicyclist	motorcyclist	road	parking	sidewalk	other-ground	building	fence	vegetation	trunk	terrain	pole	traffic-sign	**mIoU**	**Accuracy**	**FPS**
PointNet* [83]	46.3	1.3	0.3	0.1	0.8	0.2	0.2	0.0	61.6	15.8	35.7	1.4	41.4	12.9	31.0	4.6	17.6	2.4	3.7	14.6	-	2
PointNet++* [84]	53.7	1.9	0.2	0.9	0.2	0.9	1.0	0.0	72.0	18.7	41.8	5.6	62.3	16.9	46.5	13.8	30.0	6.0	8.9	20.1	-	0.1
TangentConv* [106]	86.8	1.3	12.7	11.6	10.2	17.1	20.2	0.5	82.9	15.2	61.7	9.0	82.8	44.2	75.5	42.5	55.5	30.2	22.2	35.9	-	0.3
PointASNL [134]	87.9	0	25.1	39.0	29.2	34.2	57.6	0	87.4	24.3	74.3	1.8	83.1	43.9	84.1	52.2	70.6	57.8	36.9	46.8	-	-
RandLa-Net [48]	94.2	26.0	25.8	40.1	38.9	49.2	48.2	7.2	90.7	60.3	73.7	20.4	86.9	56.3	81.4	61.3	66.8	49.2	47.7	53.9	88.8	22
KPConv [108]	96.0	30.2	42.5	33.4	44.3	61.5	61.6	11.8	88.8	61.3	72.7	31.6	90.5	64.2	84.8	69.2	69.1	56.4	47.4	58.8	90.3	-
SqueezeSeg* [120]	68.3	18.1	5.1	4.1	4.8	16.5	17.3	1.2	84.9	28.4	54.7	4.6	61.5	29.2	59.6	25.5	54.7	11.2	36.3	30.8	-	55
SqueezeSegV2* [121]	81.8	18.5	17.9	13.4	14.0	20.1	25.1	3.9	88.6	45.8	67.6	17.7	73.7	41.1	71.8	35.8	60.2	20.2	36.3	39.7	-	50
SalsaNet [2]	87.5	26.2	24.6	24.0	17.5	33.2	31.1	8.4	89.7	51.7	70.7	19.7	82.8	48.0	73.0	40.0	61.7	31.3	41.9	45.4	-	-
RangeNet++ [79]	91.4	25.7	34.4	25.7	23.0	38.3	38.8	4.8	91.8	65.0	75.2	27.8	87.4	58.6	80.5	55.1	64.6	47.9	55.9	52.2	89.0	12
PolarNet [148]	93.8	40.3	30.1	22.9	28.5	43.2	40.2	5.6	90.8	61.7	74.4	21.7	90.0	61.3	84.0	65.5	67.8	51.8	57.5	54.3	90.0	16
3D-MiniNet-KNN [5]	90.5	42.3	42.1	28.5	29.4	47.8	44.1	14.5	91.6	64.2	74.5	25.4	89.4	60.8	82.8	60.8	66.7	48.0	56.6	55.8	89.7	28
SqueezeSegV3 [127]	92.5	38.7	36.5	29.6	33.0	45.6	46.2	20.1	91.7	63.4	74.8	26.4	89.0	59.4	82.0	58.7	65.4	49.6	58.9	55.9	89.5	6
SalsaNext [26]	91.9	48.3	38.6	38.9	31.9	60.2	59.0	19.4	91.7	63.7	75.8	29.1	90.2	64.2	81.8	63.6	66.5	54.3	62.1	59.5	90.0	24
MVLidarNet [12]	87.1	34.9	32.9	23.7	24.9	44.5	44.3	23.1	90.3	56.7	73.0	19.1	85.6	53.0	80.9	59.4	63.9	49.9	51.1	52.5	88.0	92
MPF [4]	93.4	30.2	38.3	26.1	28.5	48.1	46.1	18.1	90.6	62.3	74.5	30.6	88.5	59.7	83.5	59.7	69.2	49.7	58.1	55.5	-	21
TORNADONet [36]	94.2	55.7	48.1	40.0	38.2	63.6	60.1	34.9	89.7	66.3	74.5	28.7	91.3	65.6	85.6	67.0	71.5	58.0	65.9	63.1	90.7	4
AMVNet [69]	96.2	59.9	54.2	48.8	45.7	71.0	65.7	11.0	90.1	71.0	75.8	32.4	92.4	69.1	85.6	71.7	69.6	62.7	67.2	65.3	91.3	-
GFNet (ours)	96.0	53.2	48.3	31.7	47.3	62.8	57.3	44.7	93.6	72.5	80.8	31.2	94.0	73.9	85.2	71.1	69.3	61.8	68.0	65.4	92.4	10

63.0%, further enlarging S from 3 to 4 only brings a slightly better performance. For a better speed-accuracy tradeoff, we use $S = 3$ in the default setting.

3.4.5 Benchmark

SemanticKITTI. For fair comparison with recent methods, we follow the same setting in [7, 55], *i.e.*, both training and validation splits are used for training when evaluating on the test server. As shown in Table 3.4, GFNet achieves the new state-of-the-art performance 65.4% mIoU, significantly surpassing point-based methods (*e.g.*, 58.8% for KPConv [108]) and single view models (*e.g.*, 59.5% for SalsaNext [26]). For multi-view approaches [4, 12, 36, 69], GFNet outperforms recent methods [4, 12, 36] by a large margin. Comparing with AMVNet [69], GFNet clearly outperforms it on the point-wise accuracy, i.e., 92.4% *vs.* 91.3%. The superior performance of GFNet shows the effectiveness of bidirectionally aligning and propagating geometric information between RV/BEV. Notably, AMVNet requires to train models for RV/BEV branch as well as their post-processing point head separately, while GFNet is end-to-end trainable. Additionally, since the acquisition frequency of the Velodyne HDL-64E LiDAR sensor (used by SemanticKITTI) is 10 Hz, GFNet can thus run in real-time, i.e., 10 FPS.

nuScenes. To evaluate the generalizability of GFNet, we also report the performance on the testset in Table 3.5 by submitting results to the test server. Similarly, GFNet achieves superior mIoU performance 76.1%, which remarkably outperforms PolarNet [148] and tights AMVNet [69]. However, our result 90.4% under Frequency Weighted IoU (FW IoU) beats 89.5% from AMVNet [69], which is consistent with the accuracy comparison on SemanticKITTI. It also reveals that GFNet performs much better on frequent classes while somewhat struggles on those rare/small classes. Despite the good performance of GFNet, it is also interesting to further improve GFNet by addressing rare/small classes from the perspectives of data sampling/augmentation and loss function.

TABLE 3.5. Comparisons on nuScenes (the test set) under mIoU and Frequency Weighted IoU (or FW IoU).

Method	barrier	bicycle	bus	car	const-vehicle	motorcycle	pedestrian	traffic-cone	trailer
PolarNet [148]	72.2	16.8	77.0	86.5	51.1	69.7	64.8	54.1	69.7
AMVNet [69]	79.8	32.4	82.2	86.4	62.5	81.9	75.3	72.3	83.5
GFNet (ours)	81.1	31.6	76.0	90.5	60.2	80.7	75.3	71.8	82.5

Method	truck	dri-surface	other-flat	sidewalk	terrain	manmade	vegetation	**mIoU**	**FW IoU**
PolarNet [148]	63.4	96.6	67.1	77.7	72.1	87.1	84.4	69.4	87.4
AMVNet [69]	65.1	97.4	67.0	78.8	74.6	90.8	87.9	76.1	89.5
GFNet (ours)	65.1	97.8	67.0	80.4	76.2	91.8	88.9	76.1	90.4

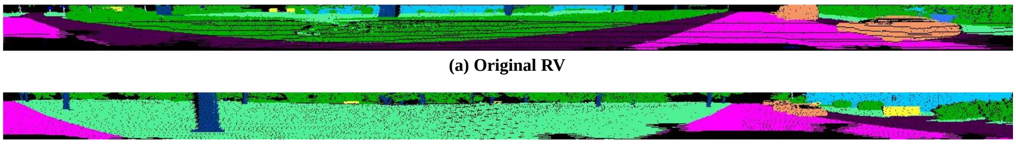
3.5 Summary

In this chapter, we propose a novel geometric flow network (GFNet) to learn effective view representations from RV and BEV. To enable propagating the complementary information across different views, we devise a geometric flow module (GFM) to bidirectionally align and fuse different view representations via geometric correspondences. Additionally, by incorporating grid sampling and KPConv to avoid time-consuming and non-differentiable post-processing, GFNet can be trained in an end-to-end paradigm. Extensive experiments on SemanticKITTI and nuScenes confirm the effectiveness of GFM and demonstrate the new state-of-the-art performance on projection-based point cloud semantic segmentation.

There are some limitations of the proposed method, since it builds upon two specific point cloud projection methods. Specifically, both RV and BEV may not be applicable to indoor datasets such as S3DIS [6]. For example, in a indoor scene of the bookcase, common objects such as table and chair are distinguishable and meaningful in the vertical direction, while the height information is missing for BEV. Also, RV image requires a scan cycle by the lidar sensor, which typically appears in outdoor scenarios such as autonomous driving (Please also refer to Appendix. 3.6.2 for more details and figures). Additionally, we apply the proposed geometric flow module in each decoder layer (*i.e.*, the upsampling layers), while popular point cloud object detection frameworks don't have such a decoder structure. Therefore, it is also non-trivial to directly apply the proposed method for object detection, which will be the subject of future studies.

3.6 Appendix

3.6.1 RV Projection



(a) Original RV

(b) Improved RV

FIGURE 3.6. Range images from original RV [79] and improved RV [111]. As we can see, [111] obtains smoother projected image than [79].

TABLE 3.6. Valid projection rate (%) when using two different RV projections [79, 111] to generate images of 64×2048 size.

Method	Train	Val
Original RV [79]	72.47	72.12
Improved RV [111]	**83.69**	**83.51**

We project 3D point cloud $\mathbf{P}$ to a 2D RV image with the size of (H, W): due to the 2D-to-3D ambiguity, there are pixels that are not projected by any points, while multiple points might be projected to the same pixel. We define the valid projection rate as the ratio of valid pixels (*i.e.*, projected by as least one point) comparing to total pixels HW. Specifically, more valid pixels usually result in a smoother projected image, *i.e.*, the higher valid projection rate, the better. We compare two different projection methods [79, 111] in terms of valid projection rate in 3.6. As we can observe, [111] significantly outperforms [79] by over 11% in both train and val set. In addition, we visualize the RV images obtained by [79, 111] separately in Figure 3.6. Obviously, the RV image generated by [111] is clearly smoother than [79]. Therefore, we use [111] in all experiments if not states otherwise.

3.6.2 RV under Indoor and Outdoor Scenes

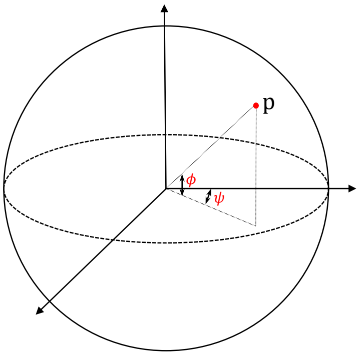


FIGURE 3.7. Range-view projection (*i.e.*, spherical projection). p is a 3D point, ψ, ϕ are the azimuthal angle, polar angle of p.

In this section, we first described the RV projection (*i.e.*, spherical projection) in detail. We then show the difference between the point clouds collected in outdoor and indoor scenes. Lastly, we discuss why RV projection is not suitable for indoor scenes.

As shown in Figure 3.7, given a 3D point p, we first obtain the corresponding ψ, ϕ for spherical projection. We then normalize ϕ, ψ to $[0, 1]$ and map them to 2D RV image with size $[H, W]$ according to Eq. 3.2. However, point clouds in outdoor and indoor scenes are usually collected in different ways. For example, SemanticKITTI is collected via a Velodyne HDL-64E lidar on the top of the driving car, which launches lasers to all-around (360°) horizontal directions and a certain degree $[f_{down}, f_{up}]$ vertical directions. When applied RV projection, a meaningful projected cylindrical image can be obtained (please refer to Figure 3.1). But for indoor dataset like S3DIS [6], it scans the entire room in any directions with a Matterport [52] scanner to generate point clouds. In addition, those indoor objects are much more dense than outdoor objects. If we still want to

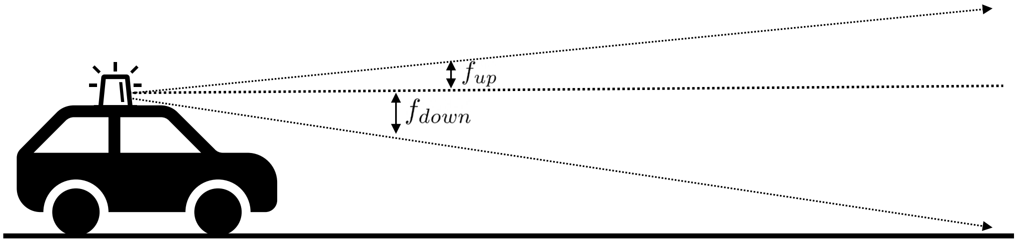


FIGURE 3.8. A simple diagram of the working mechanism when the lidar sensor collects point clouds. The lidar launches lasers to all-around (360°) horizontal directions and a certain degree $[f_{down}, f_{up}]$ vertical directions. Note that the vertical field of view $f = f_{up} - f_{down}$ where f_{down} is negative.

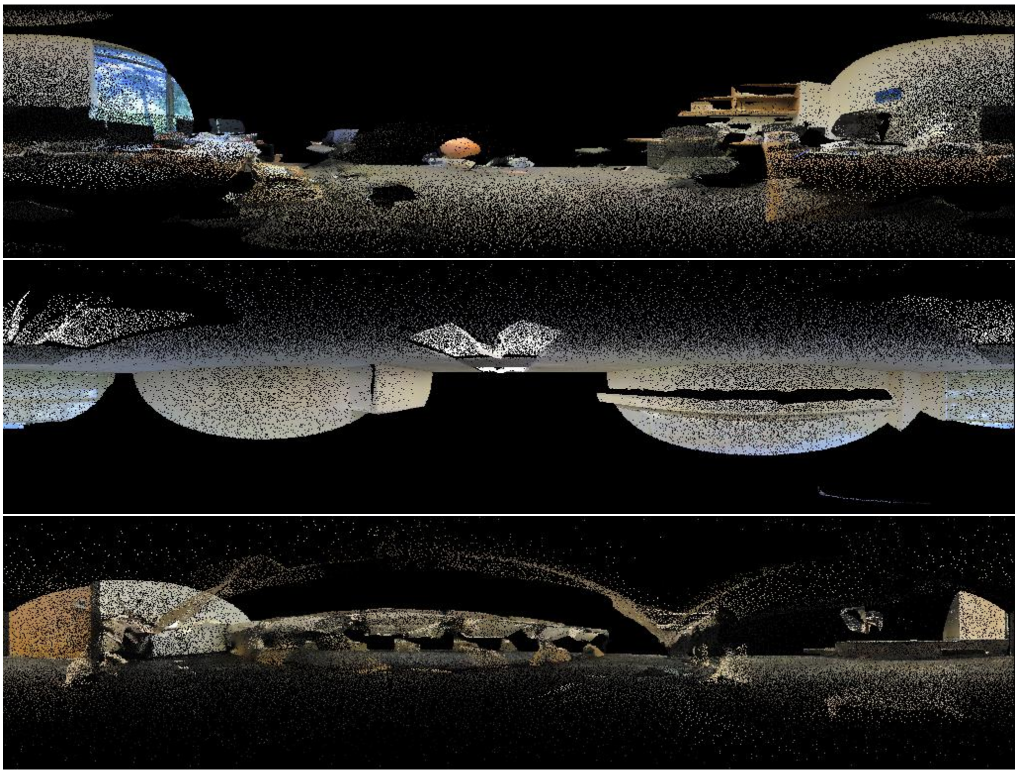
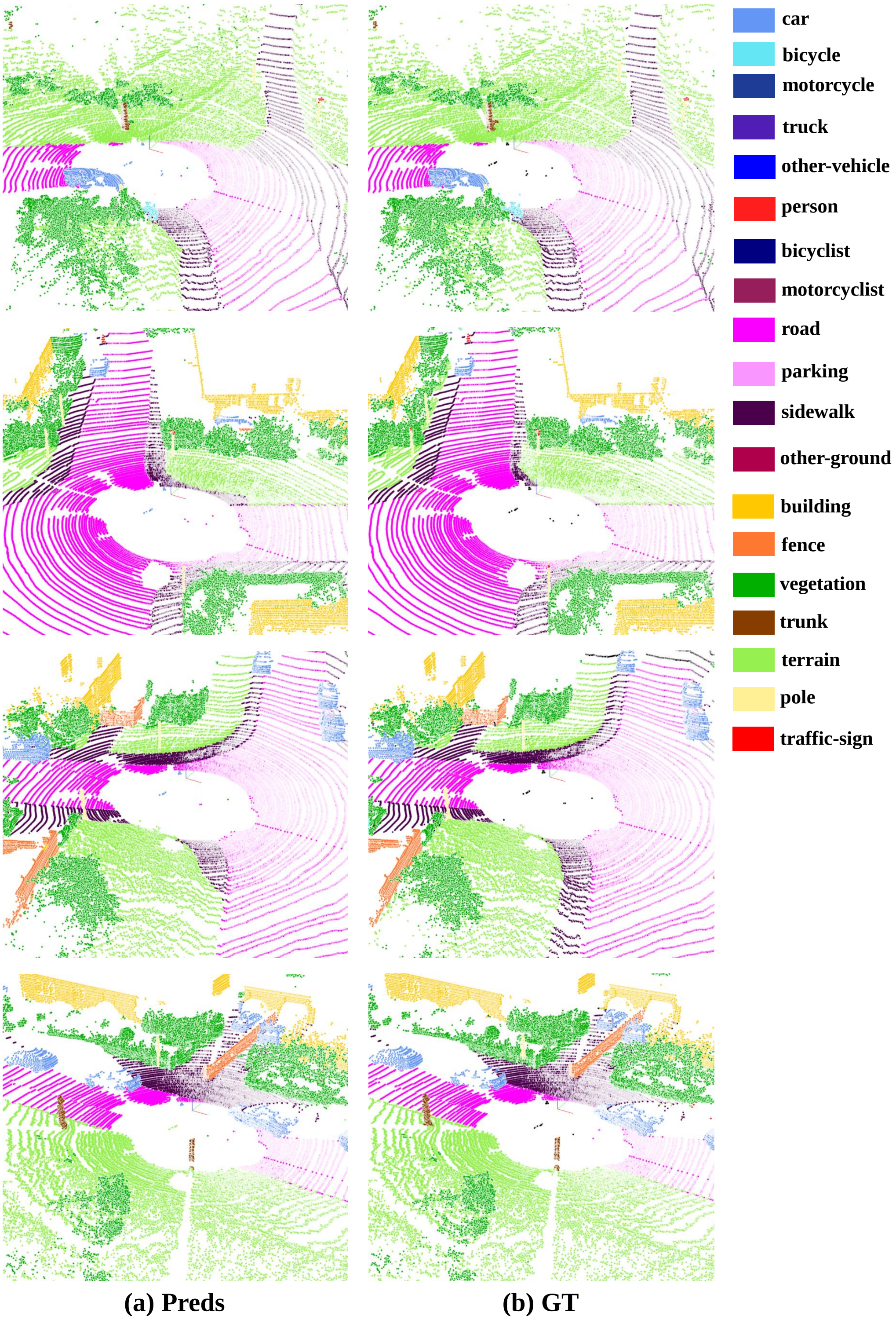


FIGURE 3.9. Three visualizations of samples from S3DIS [6] using RV projection.

use RV projection, it will lead to a severe distort image. We have also made some attempts using RV projection for S3DIS, but obtained meaningless images as in Figure 3.9. That is also the reason why existing projection-based methods [79, 120, 26] only employ RV projection in outdoor lidar-collected point clouds. As for indoor datasets like S3DIS, the mainstream methods [83, 84, 108] usually take raw points as input directly given that their size is much smaller than outdoor scenes.

3.6.3 Visualization

Figure 3.10 illustrates comparisons between GFNet and ground truth on complex scenes, revealing the excellent performance of GFNet. We also provide a GIF image, *i.e.*, `figs/vis.gif`, at https://github.com/haibo-qiu/GFNet for more visualizations.



(a) Preds **(b) GT**

FIGURE 3.10. Visualization of the predictions from our GFNet comparing to GT.

CHAPTER 4

Contextual Composite Representation Learning

This chapter primarily concentrates on capturing adequate contextual information by integrating short-range and long-range clues derived from the model's basic block angle. To achieve this, a novel collect-and-distribute mechanism is proposed, which effectively learns composite representations of point clouds that incorporate both local and global contextual information.

Specifically, in this chapter, we propose a new transformer network equipped with a collect-and-distribute mechanism to communicate short- and long-range contexts of point clouds, which we refer to as CDFormer. In particular, we first employ self-attention to capture short-range interactions within each local patch, and the updated local features are then collected into a set of proxy reference points from which we can extract long-range contexts. Afterward, we distribute the learned long-range contexts back to local points via cross-attention. To address the position clues for short- and long-range contexts, we additionally introduce the context-aware position encoding to facilitate position-aware communications between points. We perform experiments on five popular point cloud datasets, namely ModelNet40, ScanObjectNN, ShapeNetPart, S3DIS and ScanNetV2, for classification and segmentation. Results show the effectiveness of the proposed CDFormer, delivering several new state-of-the-art performances on point cloud classification and segmentation tasks. The source code is available at `https://github.com/haibo-qiu/CDFormer`.

4.1 Introduction

Point clouds have been extensively investigated in recent years, mainly due to their numerous promising real-world applications, such as autonomous driving [2, 67] and robotics [64, 136]. Unlike 2D images, a point cloud is a set of 3D points distributed in an irregular and disordered manner, where each point is usually characterized by its Cartesian coordinates (x, y, z). These fundamental differences make it non-trivial to utilize off-the-shelf 2D deep architectures for point cloud analysis. Therefore, many recent methods have developed different deep architectures to handle the special characteristics of point clouds, including mlp-based [78, 83, 84], cnn-based [66], and graph-based models [108, 118]. These architectures aim to address the challenges posed by the irregularity and unordered nature of point clouds and enable effective point cloud analysis.

One of the main challenges in point cloud analysis is efficiently exploring local and global features while considering the irregular and disordered characteristics of point clouds. Recently, transformer architectures, which enable effective local/global-range learning via the attention mechanism, have become popular in both natural language processing [9, 29, 113] and computer vision [30, 71, 116, 115]. Inspired by this, transformer architectures have been further introduced for point cloud analysis [41, 58, 81, 151]. However, the vanilla self-attention module in transformer has a time complexity of $\mathcal{O}(N^2)$ when operating on a sequence of tokens with length N. When taking each point as a token, the $\mathcal{O}(N^2)$ complexity becomes unaffordable, as there are tens of thousands of points for each point cloud in real-world applications [6]. To address this challenge, [151] introduces vector self-attention in a local way, avoiding $\mathcal{O}(N^2)$ complexity by only interacting features with K neighbors (*e.g.*, $K = 16$), while failing to capture long-range contexts. On the other hand, [58] employs local window-based self-attention with a shifted window strategy similar to [71] and proposes to capture long-range contexts by sampling nearby points densely and distant

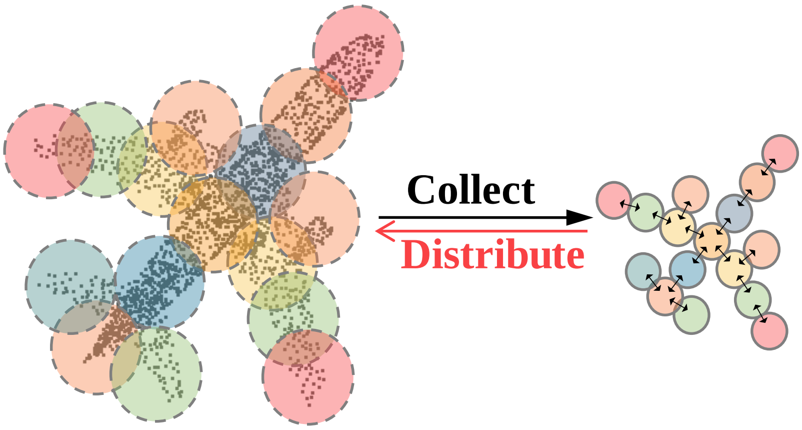


FIGURE 4.1. An illustration of the proposed collect-and-distribute mechanism.

points sparsely. Nevertheless, due to the heterogeneous density distribution of point clouds, the fixed-size window partition employed in [58] leads to a diverse number of points in each local window, thus requiring complicated and sophisticated designs during implementation.

In this chapter, we propose a new collect-and-distribute transformer, CDFormer, to learn both short- and long-range contexts for 3D point cloud analysis. Specifically, we first divide the point cloud into a set of local patches using K nearest neighbor points, instead of a fixed window partition [58]. Each local patch contains the same number of points, enabling direct modeling by popular deep learning packages [1, 82] without custom operations and avoids the prohibitive $\mathcal{O}(N^2)$ time complexity. Besides local self-attention for short-range interactions within each local patch, we introduce a collect-and-distribute mechanism. This mechanism first collects local patch information to a set of proxy reference points, explores long-range contexts among these reference points, and distributes the collected long-range contexts back to local points through cross-attention between reference points and local points. An illustration of the proposed collect-and-distribute mechanism is shown in Figure 4.1. To enhance local-global structure learning, the positional information is critical for transformers employed in point clouds [58, 151]. Therefore, we introduce context-aware position encoding for CDFormer, where the relative position information interacts with the input features to dynamically enhance the positional clues.

Our main contributions can be summarized as follows:

- We propose CDFormer to capture short- and long-range contexts simultaneously with a novel collect-and-distribute mechanism, effectively learning local-global structures.
- We introduce context-aware position encoding to enhance position clues and facilitate communications within points.
- We perform extensive experiments on five well-known point cloud datasets: ModelNet40 [125] and ScanObjectNN [112] for classification, along with ShapeNetPart [139], S3DIS [6], and ScanNetV2 [27] for segmentation. The experimental results and analysis demonstrate the effectiveness of CDFormer and achieve state-of-the-art performances in point cloud processing.

4.2 Related Work

Transformer Architectures. Transformer architectures have emerged as a dominant framework for natural language processing in recent years [29, 113]. In addition, they have seen widespread exploration in the realm of vision tasks, with ViT [30] being a groundbreaking work that divides images into local patches and treats each patch as a token. Building upon the success of ViT, numerous subsequent works have been proposed that either explore hierarchical architectures with multi-scale resolutions [32, 40, 65, 71, 116, 115] or incorporate local-global information [24, 62, 135]. For instance, PVT [116] devises a progressive shrinking pyramid to effectively explore multi-resolution features while HRViT [40] integrates high-resolution multi-branch architectures to learn multiplicative scale representations. Twins-SVT [24] incorporates locally-grouped self-attention and global sub-sampled attention to capture local-global contexts. Swin Transformer [71] introduces a hierarchical transformer architecture equipped with the shifted window strategy to enable cross-window communications. SepViT [62] proposes a self-attention mechanism that is depthwise separable to facilitate efficient local and global information exchange within a single attention block.

Point Cloud Analysis. The mainstream point cloud analysis methods can be roughly divided into three directories: point-based [41, 58, 66, 78, 81, 86, 83, 84, 87, 104, 108, 118, 128, 151], voxel-based [19, 103, 155], and projection-based [26, 89, 79, 148]. For a better trade-off between complexity and efficiency, we focus primarily on point-based approaches, where existing methods usually devise novel operations/architectures for raw points, including mlp-based [78, 86, 83, 84], cnn-based [66], graph-based [108, 118], and transformer-based [41, 58, 81, 151]. A pioneering work in this field is PointNet [83], which directly processes point clouds using multi-layer perceptrons (MLPs). This approach was further improved upon by PointNet++[84], which introduced a hierarchical structure for processing point clouds. PointNext[86] proposes even more improved training strategies that significantly improve upon PointNet++[84]. PointCNN[66] learns an x-transformation from the input points for alignment, which is followed by typical convolution layers. In contrast, KPConv [108] introduces kernel point convolution, a new point convolution operator, that takes neighboring points as input and processes them with spatially located weights. Recently, transformer architectures have been introduced for point cloud analysis [41, 58, 81, 151]. PCT [41] presents the offset-attention mechanism, which replaces the original self-attention. Point Transformer [151] proposes to introduce a vector self-attention mechanism to aggregate neighbor features but fails to capture long-range dependencies. Stratified Transformer [58] captures long-range contexts by sampling nearby points densely and distant points sparsely with the shifted window strategy, as used in Swin Transformer [71]. However, due to the varying density distribution of point clouds, partitioning windows in a fixed size leads to varying point counts in different local windows, requiring sophisticated designs to address this issue.

4.3 Method

In this section, we first provide an overview of the proposed CDFormer for 3D point cloud analysis. We then introduce the patch division and the collect-and-distribute mechanism in detail. Lastly, we discuss the proposed context-aware position encoding.

4.3.1 Overview

A 3D point cloud usually consists of a set of N points $\boldsymbol{P} \in \mathbb{R}^{N\times 3}$, where each point is featured by the Cartesian coordinate (x, y, z). The objective of typical point cloud analysis tasks is to assign semantic labels to the point cloud. For example, for point cloud classification, the goal is to predict a single semantic label $Q \in \{0, \cdots, U-1\}$ for the whole point cloud $\boldsymbol{P}$, where U is the total number of semantic categories. For point cloud segmentation, the goal is to assign a semantic label for each point in the point cloud $\boldsymbol{P}$.

The main CDFormer framework for 3D point cloud analysis is illustrated by Figure 4.2. Since a point cloud may also contain extra features such as color, we consider the input feature of a general point cloud as $\boldsymbol{X} \in \mathbb{R}^{N\times C}$, where C indicates the number of feature channels. Specifically, we first utilize a KPConv [108] layer as the embedding layer to aggregate local information for raw point embeddings to obtain the embedded features with the size of $N \times C_1$. After that, the main backbone network is a stack of multiple the proposed collect-and-distribute blocks or CD Blocks, where each block first divides all points into local patches and then explores short- and long-range contexts in a collect-and-distribute manner as follows: 1) a local self-attention is first used to learn

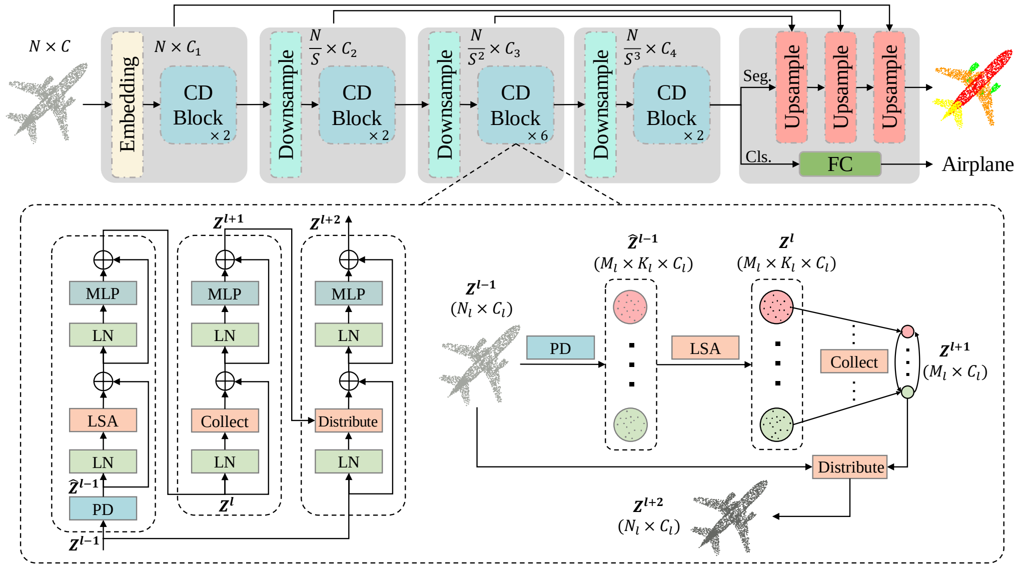


FIGURE 4.2. The main collect-and-distribute transformer framework. Note that **PD** refers to patch division and **LSA** indicates the local self-attention defined by Equation 4.1 and 4.2.

short-range relations in each patch of points; 2) the learned local features are then collected to a set of proxy reference points which then communicate with each other to capture long-range contexts; and 3) the learned long-range contexts are finally distributed back to the original local patch points such that the learned point embeddings are equipped with both short- and long-range information. CDFormer is built by stacking multiple downsampling layers and CD Blocks.

4.3.2 Patch Division

For a typical point cloud with tens of thousands of points [6], it is computationally prohibitive to consider each point as a token: given a sequence of tokens with the length N, the time complexity of the self-attention in transformer is $\mathcal{O}(N^2)$. Therefore, following [30, 80, 141], we divide a point cloud into multiple local patches, *e.g.*, M patches with K points in each patch, and then employ self-attention within each local patch instead of all points. Therefore, with a proper patch division, it becomes acceptable with a linear time complexity $\mathcal{O}(MK^2)$.

We describe the patch division process used in our method as follows. Given a scale factor S, the furthest point sampling algorithm (FPS) [31, 84] is applied on the point features $\boldsymbol{X} \in \mathbb{R}^{N \times C}$ to obtain patch centers $\bar{\boldsymbol{X}} \in \mathbb{R}^{M \times C}$ where $M = N/S$. For each local patch, we group the K nearest neighbors around each patch center and reformulate M patches as $\hat{\boldsymbol{X}} \in \mathbb{R}^{M \times K \times C}$, and then apply the local self-attention (LSA) to extract local features. Specifically, for the i_{th} patch where $i \in [0, \cdots, M-1]$, let $\hat{\boldsymbol{x}} \in \mathbb{R}^{K \times C}$ denote its representation, which is then fed into MLPs to generate the query token $\boldsymbol{Q}_{lsa} = \boldsymbol{W}^{q}_{lsa}\hat{\boldsymbol{x}}$, the key token $\boldsymbol{K}_{lsa} = \boldsymbol{W}^{k}_{lsa}\hat{\boldsymbol{x}}$, and the value token $\boldsymbol{V}_{lsa} = \boldsymbol{W}^{v}_{lsa}\hat{\boldsymbol{x}}$, respectively. The output $\bar{\boldsymbol{z}} \in \mathbb{R}^{K \times C}$ can thus be calculated as follows:

$$\bar{\boldsymbol{z}} = \boldsymbol{W}^{attn}_{lsa}\boldsymbol{V}_{lsa}, \tag{4.1}$$

$$\boldsymbol{W}^{attn}_{lsa} = \text{Softmax}((\boldsymbol{Q}_{lsa}\boldsymbol{K}^{\top}_{lsa})/\sqrt{C}). \tag{4.2}$$

All M patches features can be similarly obtained as $\bar{\boldsymbol{Z}} \in \mathbb{R}^{M \times K \times C}$. Notably, we do not discuss the position encoding here and leave it in Sec 4.3.4. Lastly, we have the complexity as follows: the local self-attention in each patch only has the time complexity $\mathcal{O}(K^2)$, and the overall time complexity of M patches become affordable $\mathcal{O}(MK^2) = \mathcal{O}(NK^2/S)$. Through LSA, all points in each local patch can communicate with each other to capture short-range dependencies.

4.3.3 Collect-and-Distribute Mechanism

As the aforementioned local self attention only models the short-range information in each local patch, we then introduce how to explore long-range contexts with the proposed collect-and-distribute mechanism. Specifically, we first collect those communicated local feature as a set of proxy reference points such that each local patch directly corresponds to a specific proxy point. To capture the long-range dependencies, a neighbor self-attention (NSA) is then applied on those proxy references to allow feature propagation among neighbors. By doing this, we can extract long-range contexts with a reduced linear time complexity

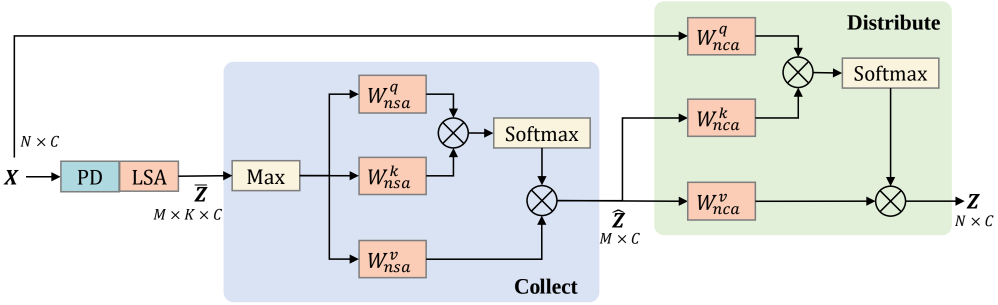


FIGURE 4.3. Collect-and-distribute block. Note that **Collect** involves the neighbor self-attention (nsa) defined by Equation 4.3 and 4.4; **Distribute** includes the neighbor cross-attention (nca) formulated in Equation 4.5 and 4.6.

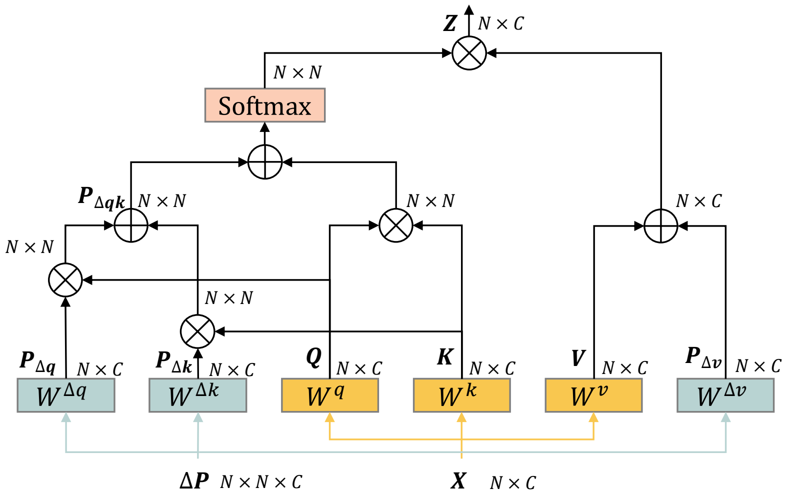


FIGURE 4.4. Context-aware position encoding.

since each proxy represents a patch of points. Lastly, those enhanced proxies distribute back to the local points via cross-attention to achieve short- and long-range contexts communications. An illustration of the proposed block is shown in Figure 4.3, and we depict each step in detail as follows.

Collect. Given local patches $\bar{\boldsymbol{Z}} \in \mathbb{R}^{M \times K \times C}$, we collect local patch information from all K local points to a proxy reference point via a max-pooling operation, $\boldsymbol{R} = \text{maxpool}(\bar{\boldsymbol{Z}}) \in \mathbb{R}^{M \times C}$. Next, we consider the K nearest neighbors for each proxy $\hat{\boldsymbol{R}} \in \mathbb{R}^{M \times K \times C}$ to capture the long-range contexts by employing a neighboring self-attention between $\boldsymbol{R}$ and $\hat{\boldsymbol{R}}$, *i.e.*, a self-attention between each proxy and its K neighbors followed by a sum operation to generate the output $\hat{\boldsymbol{Z}} \in \mathbb{R}^{M\times C}$. We first generate the feature embeddings with MLPs for query, key, and value by $\boldsymbol{Q}_{nsa} = \boldsymbol{W}_{nsa}^{q}\boldsymbol{R}$, $\boldsymbol{K}_{nsa} = \boldsymbol{W}_{nsa}^{k}\hat{\boldsymbol{R}}$, and $\boldsymbol{V}_{nsa} = \boldsymbol{W}_{nsa}^{v}\hat{\boldsymbol{R}}$, respectively. Then we can formulate the process as follows:

$$\hat{\boldsymbol{Z}} = \text{Sum}(\boldsymbol{W}_{nsa}^{attn}\boldsymbol{V}_{nsa}) \tag{4.3}$$

$$\boldsymbol{W}_{nsa}^{attn} = \text{Softmax}((\boldsymbol{Q}_{nsa}\boldsymbol{K}_{nsa}^{\top})/\sqrt{C}). \tag{4.4}$$

Notably, the time complexity $\mathcal{O}(NK^2/S)$ is linear to N because the NSA is only applied on K neighbors instead of all proxies.

Distribute. We distribute the long-range information in $\hat{\boldsymbol{Z}} \in \mathbb{R}^{M\times C}$ back to local patch points for joint short- and long-range contexts via a neighbor cross-attention (NCA). In particular, given vanilla point features $\boldsymbol{X} \in \mathbb{R}^{N\times C}$ and enhanced proxies $\hat{\boldsymbol{Z}} \in \mathbb{R}^{M\times C}$, we first group the K nearest neighbor proxies $\boldsymbol{E} \in \mathbb{R}^{N\times K\times C}$ for each point. After that, we employ the neighbor cross-attention by regarding $\boldsymbol{X}$ as the query, $\boldsymbol{E}$ as the key and value, which is fed into a successive sum operation to obtain the final output $\boldsymbol{Z} \in \mathbb{R}^{N\times C}$. We first get the embeddings with MLPs by $\boldsymbol{Q}_{nca} = \boldsymbol{W}_{nca}^{q}\boldsymbol{X}$, $\boldsymbol{K}_{nca} = \boldsymbol{W}_{nca}^{k}\boldsymbol{E}$, and $\boldsymbol{V}_{nca} = \boldsymbol{W}_{nca}^{v}\boldsymbol{E}$, respectively. After that, we can calculate $\boldsymbol{Z}$ as follows:

$$\boldsymbol{Z} = \text{Sum}(\boldsymbol{W}_{nca}^{attn}\boldsymbol{V}_{nca}) \tag{4.5}$$

$$\boldsymbol{W}_{nca}^{attn} = \text{Softmax}((\boldsymbol{Q}_{nca}\boldsymbol{K}_{nca}^{\top})/\sqrt{C}). \tag{4.6}$$

In this way, each point communicates with K enhanced proxies, *i.e.*, approximate K^2 original proxies, and K^3 original points. Therefore, the long-range dependencies can be effectively distributed back to local points, *i.e.*, both short- and long-range contexts are fused into the learned representations.

4.3.4 Context-Aware Position Encoding

The position information of input tokens is necessary for transformer architectures [30, 113], while 3D point cloud naturally contains the (x, y, z) coordinates as position. Therefore, it might be straightforward to remove position encoding in the transformer-based architecture for point clouds. However, when the network goes deeper, the coordinates information can not keep precisely intact [58, 151]. Inspired by [122], we propose to further enhance the position clues in point cloud transformer via a context-aware position encoding (CAPE). Specifically, CAPE is calculated by interacting relative position differences with the current features, thus can simultaneously handle the unordered characteristic of the point cloud and adaptively enhance the position information. An illustration of the proposed context-aware position encoding is shown in Figure 4.4.

Given the input $\boldsymbol{X} \in \mathbb{R}^{N \times C}$ and the relative position differences $\Delta \boldsymbol{P} \in \mathbb{R}^{N \times N \times 3}$, we first obtain the $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}$ and their position embeddings $\boldsymbol{P}_{\Delta q}, \boldsymbol{P}_{\Delta k}, \boldsymbol{P}_{\Delta v}$ via MLPs. We then calculate the CAPE by:

$$\boldsymbol{P}_{\Delta qk} = \boldsymbol{P}_{\Delta q} \boldsymbol{Q}^{\top} + \boldsymbol{P}_{\Delta k} \boldsymbol{K}^{\top}, \tag{4.7}$$

such that it is aware of the input features and can dynamically magnify the position information to effectively facilitate the communication of points. Lastly, we obtain the output $\boldsymbol{Z} \in \mathbb{R}^{N \times C}$ as follows:

$$\boldsymbol{Z} = \boldsymbol{W}_{attn} \cdot (\boldsymbol{V} + \boldsymbol{P}_{\Delta v}) \tag{4.8}$$

$$\boldsymbol{W}_{attn} = \text{Softmax}((\boldsymbol{Q}\boldsymbol{K}^{\top} + \boldsymbol{P}_{\Delta qk}) / \sqrt{C}). \tag{4.9}$$

Notably, if not otherwise stated, we use CAPE in all attention layers, including LSA, NSA, and NCA, to better capture short- and long-range contexts.

4.4 Experiments

In this section, we first introduce the implementation details. We then evaluate the proposed CDFormer on ModelNet40 [125] and ScanObjectNN [112] for point cloud classification, ShapeNetPart [139] for point cloud part segmentation, as well as S3DIS [6] and ScanNetV2 [27] for point cloud scene segmentation. Lastly, we perform comprehensive ablation studies on each component.

4.4.1 Implementation Details.

For ModelNet40 [125], following [86], we use fewer blocks in each stage $[1, 1, 3, 1]$ with larger feature dimensions $\mathcal{C} = [64, 128, 256, 512]$ and larger number of heads $\mathcal{H} = [4, 8, 16, 32]$. We simply use $K = 16$ neighbours in all blocks. Cosine annealing schedule with learning rate 0.001 is adopted for training 600 epochs with batch size 32. We use AdamW optimizer [75] with the weight decay 0.05. For each data sample, only 1,024 points with (x, y, z) are used for training and testing, and common data augmentations, including shift, scale and cutmix [147], are employed. The downsampling scale S of each stage is set to 4. For ScanObjectNN [112], we follow [86] to use point resampling to adapt 1,024 points for training, and only the hardest perturbed variant (PB_T50_RS) is considered in our experiment. We keep other training configurations the same as those in ModelNet40.

For S3DIS [6], following the practice in [58, 151], we first apply the grid sampling on the raw input points with the grid size 0.04m. We adopt the encoder with four stages with $[2, 2, 6, 2]$ blocks, the number of channels $\mathcal{C} = [C_1, C_2, C_3, C_4] = [48, 96, 192, 384]$ and the number of heads $\mathcal{H} = [H_1, H_2, H_3, H_4] = [3, 6, 12, 24]$. For simplicity, we set neighbours $K = 16$ in all blocks. During the training process, the maximum number of input points is set to 80,000. Meanwhile, we set the downsampling scale S of each stage to 8. We use AdamW optimizer [75] with the weight decay 0.01. All models are trained for 100 epochs, and the learning rate starts from 0.01 and drops by 1/10 at 60 and 80 epochs. We use four V100 GPUs with the batch size of 8. Following [86], we use the cross-entropy loss with label smoothing, and popular data augmentations including jitter, scale, rotate, and color drop. For ScanNetV2 [27], we use grid sampling with a size of 0.02m on the raw point clouds, following [124]. We employ the OneCycleLR [100] scheduler, which increases the learning rate from 0.0005 to 0.005 in the first 5 epochs and then uses cosine annealing to decrease it to 0 over the remaining 95 epochs. Additionally, we repeat the training set (1,201 scenes) 9 times to obtain 10,809 samples during the training process. For ShapeNetPart [139], we follow [86] to use 2,048 points for training and testing. Considering that each point cloud has fewer points than it in S3DIS [6], we reduce the downsampling scale S to 4 and use the batch size 80. The model is trained for 300 epochs, and the learning rate starts from 0.01 and drops by 1/10 at 210 and 270 epochs. Other settings are same as those in S3DIS.

4.4.2 Point Cloud Classification

Dataset and Metrics. ModelNet40 [125] is a canonical dataset for object shape classification, which consists of 9,843 training and 2,468 testing CAD models belonging to 40 categories. We report the results of overall accuracy (OA). ScanObjectNN [112] is a more challenging real-world benchmark in terms of background, noise, and occlusions. It contains totally 15,000 objects from 15 classes. We report the results of the mean of class-wise accuracy (mAcc) and overall accuracy (OA) on hardest perturbed variant, *i.e.*, PB_T50_RS.

Results. Table 4.1 shows the results on ModelNet40, where the proposed CDFormer achieves comparable performance 94.0% with other state-of-the-arts by only taking 1,024 points as input. In Table 4.2, we also evaluate the proposed method on the most challenging variant (PB_T50_RS) of ScanObjectNN, where CDFomer achieves 87.2 ± 0.3 and 88.4 ± 0.2 on mAcc and OA, outperforming

TABLE 4.1. Results on ModelNet40.

Method	#Points	OA (%)
PointNet [83]	1024	89.2
PointNet++ [84]	1024	90.7
PointNet++ [84]	5000	91.9
PointCNN [66]	1024	92.5
PointConv [123]	1024	92.5
A-CNN [56]	1024	92.6
KPConv [108]	7000	92.9
DGCNN [118]	1024	92.9
PointASNL [134]	1024	92.9
PointNext [86]	1024	93.2
PosPool [70]	5000	93.2
PCT [41]	1024	93.2
SO-Net [60]	5000	93.4
PT [151]	1024	93.7
GBNet [91]	1024	93.8
PA-DGC [130]	1024	93.9
PointMLP [78]	1024	**94.1**
CDFormer (ours)	1024	94.0

all previous methods in terms of both performance and stability. Specifically, PointMLP [78] obtains 94.1% on ModelNet40, slightly higher than 94.0% by CDFormer. I believe that this small difference of 0.1% may be due to the dataset's simplicity and randomness; most recent methods saturate around 94.0%. Additionally, we surpass PointMLP by over 3% in ScanObjectNN and also show more stable performance as indicated by the smaller standard derivation. PointNeXt [86] is the current state-of-the-art method, but we still outperform it by a clear margin, especially on mAcc (87.2% *vs.* 85.8%). Also, the smallest gap between mean of class-wise accuracy (mAcc) and overall accuracy (OA) achieved by CDFormer implies that our approach shows excellent robust performance on each class instead of biasing to a specific category. We owe it to the proposed collect-and-distribute mechanism for effectively capturing both short- and long-range contexts, which can handle different categories with different scales and shapes.

TABLE 4.2. Results on ScanObjectNN.

Method	mAcc (%)	OA (%)
PointNet [83]	63.4	68.2
SpiderCNN [131]	69.8	73.7
PointNet++ [84]	75.4	77.9
DGCNN [118]	73.6	78.1
PointCNN [66]	75.1	78.5
BGA-DGCNN [112]	75.7	79.7
BGA-PN++ [112]	77.5	80.2
DRNet [90]	78.0	80.3
GBNet [91]	77.8	80.5
SimpleView [38]	-	80.5±0.3
PRANet [20]	79.1	82.1
MVTN [42]	-	82.8
PointBERT [142]	-	83.1
DeltaConv [119]	-	84.7
PointMAE [80]	-	85.2
PointMLP [78]	83.9±0.5	85.4±0.3
RepSurf [94]	83.1	86.0
PointNext [86]	85.8±0.6	87.7±0.4
CDFormer (ours)	**87.2±0.3**	**88.4±0.2**

4.4.3 Point Cloud Part Segmentation

Dataset and Metrics. ShapeNetPart [139] is a popular dataset for object part segmentation, which is composed of 16,880 3D models from 16 different shape categories (*e.g.*, *"airplane"* and *"chair"*), where 14,006 models for training and 2,874 for testing. For each category, its number of parts is between 2 and 6, and there are total 50 different parts. For evaluation metrics, we report the instance mIoU along with the throughput speed ($instance/second$).

Results. Table 4.3 demonstrates the results of CDFormer compared to previous approaches. Our CDFormer outperforms congeneric transformer-based methods, *e.g.*, Point Transformer [151] and Stratified Transformer [58], and other representative approaches like KPConv [108], while it is comparable to the best results of PointNext [86] (87.0% *vs.* 87.0%). As we will see from Table 4.4 that the

TABLE 4.3. Results on ShapeNetPart.

Method	Ins. mIoU	Throughput
PointNet [83]	83.7	**1184**
PointNet++ [84]	85.1	708
DGCNN [118]	85.2	147
ASSANet-L [85]	86.1	640
PointMLP [78]	86.1	270
PVCNN [73]	86.2	-
PCT [41]	86.4	-
KPConv [108]	86.4	44
Point Transformer [151]	86.6	297
Stratified Transformer [58]	86.6	398
CurveNet [126]	86.8	97
PointNeXt [86]	**87.0**	76
CDFormer (ours)	**87.0**	84

CDFormer defeats PointNext on S3DIS (76.0% *vs.* 74.9%), we thus conjecture that the collect-and-distribute mechanism in CDFormer can not fully exploit the advantage of capturing long-range dependencies on the small size of point cloud considering the huge difference on the size of samples in ShapeNetPart (2,048) and S3DIS (up to 80,000 during training). The part segmentation results of multiple objects are visualized in Figure 4.5. As observed, the predictions from CDFormer are semantically reasonable and close to the ground truth, further validating its effectiveness.

4.4.4 Point Cloud Scene Segmentation

Dataset and Metrics. S3DIS [6] is a widely used benchmark for point cloud scene segmentation, containing 271 rooms in 6 areas collected from three buildings. The point is annotated with 13 semantic categories such as *"ceiling"* and *"bookcase"*. Following [84, 151], we evaluate the proposed method on the Area 5 and also perform the standard 6-fold cross-validation. ScanNetV2 [27] is another challenging dataset that consists of 1,513 room scenes. Among these scenes, 1,201 are used for training and 312 for validation purposes. Each sampled point

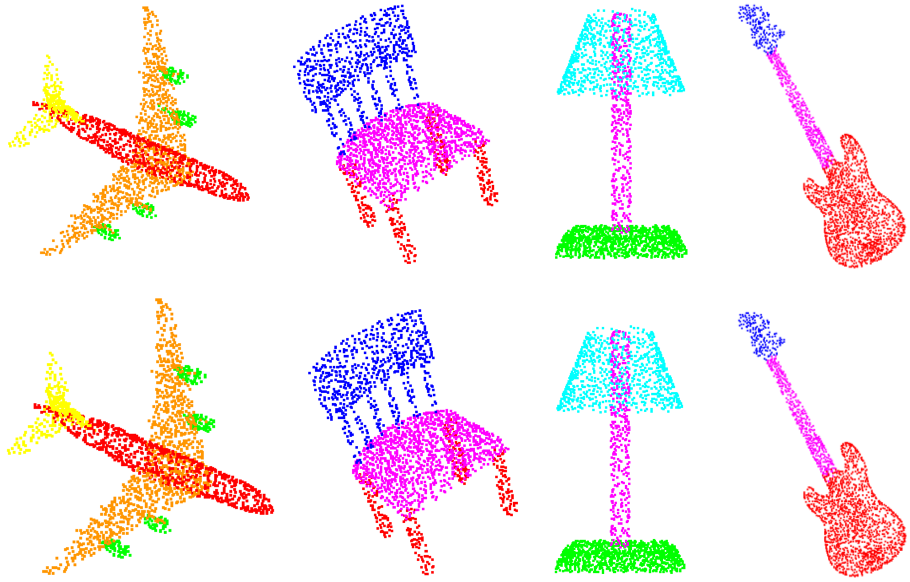


FIGURE 4.5. Visualizations of object part segmentation on multiple different categories from the ShapeNetPart dataset. The top row represents ground truth and the predictions from the proposed CDFormer are in the second row.

is assigned a semantic label from one of the available 20 categories, including *"floor"* and *"table"*. Similar to previous methods [43, 72, 124, 137], we make evaluations on the validation and testing sets. Regarding the metrics, we report performance using mean class-wise intersection over union (mIoU), mean of class-wise accuracy (mAcc), and overall point-wise accuracy (OA) as in [86, 124].

Results. In Table 4.4 and 4.5, the proposed CDFormer achieves new state-of-the-art performances on both Area 5 and standard 6-fold cross-validation. Notably, the recent approach PointNext [86] has 41.6M parameters for its best results, while our CDFormer with 25.7M parameters achieves a clear improvement of 1.7% and 1.1% mIoU under Area 5 and 6-fold cross-validation. Additionally, with and without using 2D images, DeepViewAgg [97] obtains 69.5% and 74.7% respectively, which nevertheless is also inferior to CDFormer (76.0%). Additionally, we provide visualizations in Figure 4.6. The proposed CDFormer exhibit exceptional ability in capturing short- and long-range contexts, and accurately

TABLE 4.4. Results on S3DIS using 6-fold cross-validation. The **bold** and underline denote the first and second best performances. Note that * indicates additional 2D images are used and † represents the reproduced results.

Method	**mIoU**	**mACC**	**OA**	ceiling	floor	wall	beam	column	window	door	table	chair	sofa	bookcase	board	clutter
PointNet [83]	47.6	66.2	78.6	88.0	88.7	69.3	42.4	23.1	47.5	51.6	54.1	42.0	9.6	38.2	29.4	35.2
RSNet [50]	56.5	66.5	-	92.5	92.8	78.6	32.8	34.4	51.6	68.1	59.7	60.1	16.4	50.2	44.9	52.0
SPG [59]	62.1	73.0	86.4	89.9	95.1	76.4	62.8	47.1	55.3	68.4	73.5	69.2	63.2	45.9	8.7	52.9
PointCNN [66]	65.4	75.6	88.1	94.8	97.3	75.8	63.3	51.7	58.4	57.2	71.6	69.1	39.1	61.2	52.2	58.6
PointWeb [150]	66.7	76.2	87.3	93.5	94.2	80.8	52.4	41.3	64.9	68.1	71.4	67.1	50.3	62.7	62.2	58.5
ShellNet [149]	66.8	-	87.1	90.2	93.6	79.9	60.4	44.1	64.9	52.9	71.6	84.7	53.8	64.6	48.6	59.4
RandLA-Net [48]	70.0	82.0	88.0	93.1	96.1	80.6	62.4	48.0	64.4	69.4	69.4	76.4	60.0	64.2	65.9	60.1
KPConv [108]	70.6	79.1	-	93.6	92.4	83.1	63.9	54.3	66.1	76.6	57.8	64.0	69.3	74.9	61.3	60.3
SCF-Net [33]	71.6	82.7	88.4	93.3	96.4	80.9	64.9	47.4	64.5	70.1	71.4	81.6	67.2	64.4	67.5	60.9
BAAF [92]	72.2	83.1	88.9	93.3	96.8	81.6	61.9	49.5	65.4	73.3	72.0	83.7	67.5	64.3	67.0	62.4
CBL [105]	73.1	79.4	89.6	94.1	94.2	85.5	50.4	58.8	70.3	78.3	75.7	75.0	71.8	74.0	60.0	62.4
PT [151]	73.5	81.9	90.2	-	-	-	-	-	-	-	-	-	-	-	-	-
DeepViewAgg [97]*	74.7	-	-	90.0	96.1	85.1	66.9	56.3	71.9	78.9	79.7	73.9	69.4	61.1	75.0	65.9
PointNext-XL [86]	74.9	83.0	90.3	-	-	-	-	-	-	-	-	-	-	-	-	-
PointNext-XL [86]†	74.9	83.0	90.3	94.1	96.8	85.0	61.4	64.2	68.5	78.7	76.9	70.2	74.3	70.7	69.9	63.2
CDFormer (ours)	**76.0**	**84.6**	**90.7**	94.4	97.8	86.7	70.8	66.7	69.1	78.9	77.8	64.9	75.3	71.4	71.1	63.6

segmenting complicated scenes. We also present the results on ScanNetV2 in Table 4.6. As observed, our CDFormer achieves the best performance of 76.2% on the validation set and 76.6% on the testing set, further demonstrating its superiority.

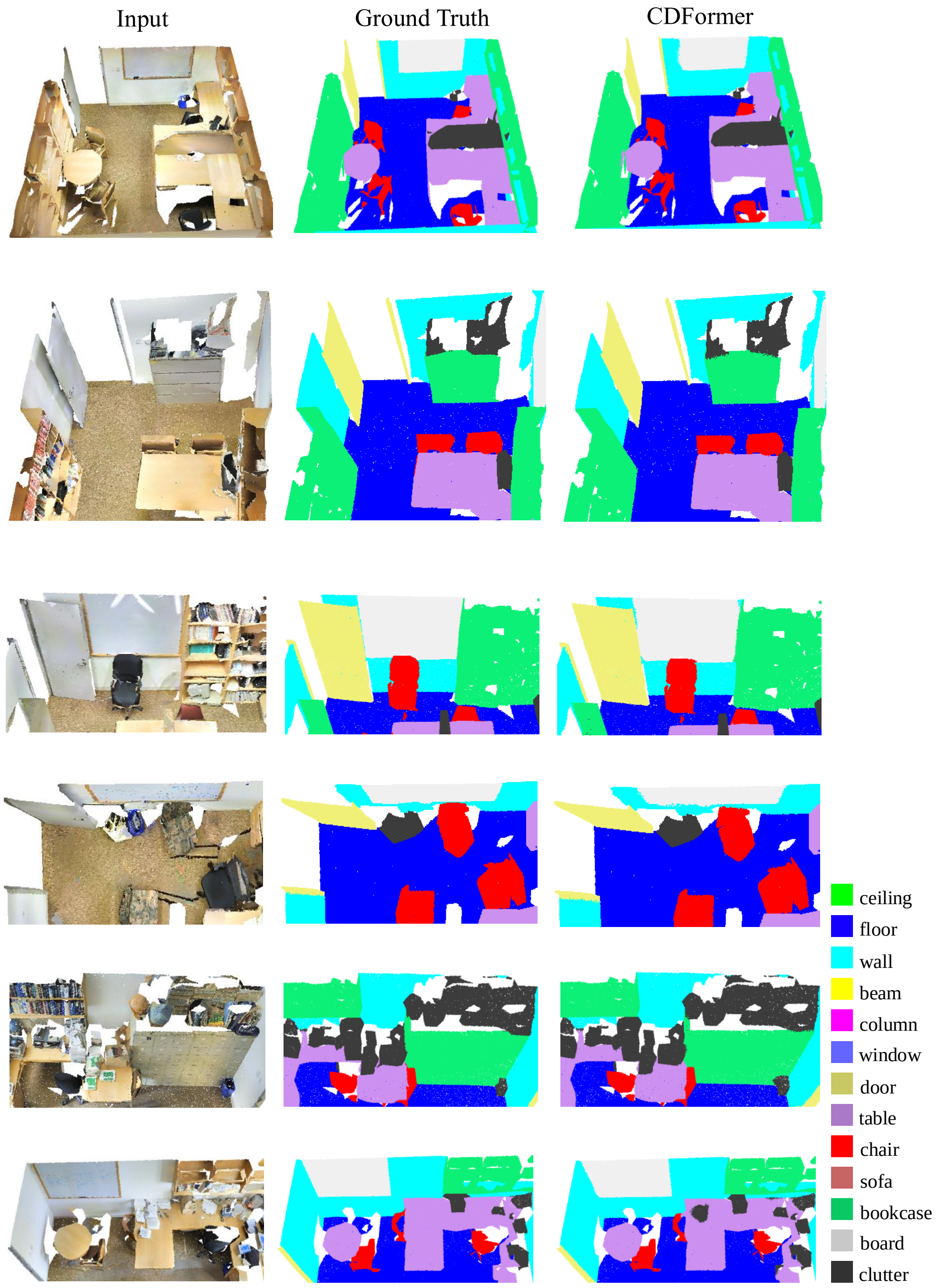


FIGURE 4.6. Visualizations of semantic predictions on Area 5 of S3DIS by comparing CDFormer with the ground truth.

4.4.5 Ablation Studies

We perform all the ablation studies on Area 5 of S3DIS.

TABLE 4.5. Results of the proposed CDFormer and recent state-of-the-arts on Area 5 of S3DIS. The **bold** and underline denote the first and second best performances.

Method	**mIoU**	**mACC**	**OA**	ceiling	floor	wall	beam	column	window	door	table	chair	sofa	bookcase	board	clutter
PointNet [83]	41.1	49.0	-	88.8	97.3	69.8	0.1	3.9	46.3	10.8	59.0	52.6	5.9	40.3	26.4	33.2
SegCloud [107]	48.9	57.4	-	90.1	96.1	69.9	0.0	18.4	38.4	23.1	70.4	75.9	40.9	58.4	13.0	41.6
PointCNN [66]	57.3	63.9	85.9	92.3	98.2	79.4	0.0	17.6	22.8	62.1	74.4	80.6	31.7	66.7	62.1	56.7
SPG [59]	58.0	66.5	86.4	89.4	96.9	78.1	0.0	42.8	48.9	61.6	84.7	75.4	69.8	52.6	2.1	52.2
PCT[41]	61.3	67.7	-	92.5	98.4	80.6	0.0	19.4	61.6	48.0	76.6	85.2	46.2	67.7	67.9	52.3
HPEIN [54]	61.9	68.3	87.2	91.5	98.2	81.4	0.0	23.3	65.3	40.0	75.5	87.7	58.5	67.8	65.6	49.4
MinkowskiNet [22]	65.4	71.7	-	91.8	98.7	86.2	0.0	34.1	48.9	62.4	81.6	89.8	47.2	74.9	74.4	58.6
KPConv [108]	67.1	72.8	-	92.8	97.3	82.4	0.0	23.9	58.0	69.0	81.5	91.0	75.4	75.3	66.7	58.9
CGA-Net[77]	68.6	-	-	94.5	98.3	83.0	0.0	25.3	59.6	71.0	92.2	82.6	76.4	77.7	69.5	61.5
CBL [105]	69.4	75.2	90.6	93.9	98.4	84.2	0.0	37.0	57.7	71.9	91.7	81.8	77.8	75.6	69.1	62.9
PT [151]	70.4	76.5	90.8	94.0	98.5	86.3	0.0	38.0	63.4	74.3	89.1	82.4	74.3	80.2	76.0	59.3
PointNext-XL [86]	70.5	76.8	90.6	94.2	98.5	84.4	0.0	37.7	59.3	74.0	83.1	91.6	77.4	77.2	78.8	60.6
StratifiedFormer [58]	72.0	78.1	**91.5**	96.2	98.7	85.6	0.0	46.1	60.0	76.8	92.6	84.5	77.8	75.2	78.1	64.0
CDFormer (ours)	**72.2**	**78.5**	91.2	95.1	98.8	86.3	0.0	49.3	62.4	72.1	83.5	92.3	83.9	76.1	75.9	62.7

Collect-and-Distribute Mechanism. To explore the effectiveness of the proposed collect-and-distribute (CD) mechanism, we use the counterpart vanilla parameter-free (or tiny) operations as baselines. Specifically, we only keep the max operation to aggregate local features while discarding neighbor self-attention when no *collect*. Also, we adopt an interpolation layer followed by a simple MLP to add back to local points if no *distribute*. In Table 4.7, we find that without the collect-and-distribute mechanism, it achieves a relatively low mIoU 67.6%, which is then improved by *collect* for catching the long-range dependencies to 70.1%. Furthermore, when further distributing the long-range contexts back to

TABLE 4.6. Results of mIoU (%) on both validation and testing set (including per-category IoU) of ScanNetV2.

Method	**Val**	**Test**	bathtub	bed	bookshelf	cabinet	chair	counter	curtain	desk	door
PointNet++ [84]	53.5	55.7	73.5	66.1	68.6	49.1	74.4	39.2	53.9	45.1	37.5
RandLA-Net [48]	-	64.5	77.8	73.1	69.9	57.7	82.9	44.6	73.6	47.7	52.3
PointConv [123]	61.0	66.6	78.1	75.9	69.9	64.4	82.2	47.5	77.9	56.4	50.4
PointASNL [134]	63.5	66.6	70.3	78.1	75.1	65.5	83.0	47.1	76.9	47.4	53.7
KPConv [108]	69.2	68.4	84.7	75.8	78.4	64.7	81.4	47.3	77.2	60.5	59.4
CBL [105]	-	70.5	76.9	77.5	80.9	68.7	82.0	43.9	81.2	66.1	59.1
PointNext [86]	71.5	71.2	-	-	-	-	-	-	-	-	-
PointMeta [68]	71.4	83.5	78.5	82.1	68.4	84.6	53.1	86.5	61.4	59.6	
SparseCNN [39]	72.8	72.5	64.7	82.1	84.6	72.1	86.9	53.3	75.4	60.3	61.4
MinkUNet [22]	72.2	73.6	85.9	81.8	83.2	70.9	84.0	52.1	85.3	66.0	64.3
StratifiedFormer [58]	74.3	74.7	90.1	80.3	84.5	75.7	84.6	51.2	82.5	69.6	64.5
BPNet [49]	73.9	74.9	90.9	81.8	81.1	75.2	83.9	48.5	84.2	67.3	64.4
PTv2 [124]	75.4	75.2	74.2	80.9	87.2	75.8	86.0	55.2	89.1	61.0	68.7
CDFormer (ours)	**76.2**	**76.6**	77.9	79.2	86.1	75.3	88.4	59.1	88.7	62.6	71.3

Method	floor	other.	picture	refrig.	shower.	sink	sofa	table	toilet	wall	window
PointNet++ [84]	94.6	37.6	20.5	40.3	35.6	55.3	64.3	49.7	82.4	75.6	51.5
RandLA-Net [48]	94.5	45.4	26.9	48.4	74.9	61.8	73.8	59.9	82.7	79.2	62.1
PointConv [123]	95.3	42.8	20.3	58.6	75.4	66.1	75.3	58.8	90.2	81.3	64.2
PointASNL [134]	95.1	47.5	27.9	63.5	69.8	67.5	75.1	55.3	81.6	80.6	70.3
KPConv [108]	93.5	45.0	18.1	58.7	80.5	69.0	78.5	61.4	88.2	81.9	63.2
CBL [105]	94.5	51.5	17.1	63.3	85.6	72.0	79.6	66.8	88.9	84.7	68.9
PointNext [86]	-	-	-	-	-	-	-	-	-	-	-
PointMeta [68]	95.3	50.0	24.6	67.4	88.8	69.2	76.4	62.4	84.9	84.4	67.5
SparseCNN [39]	95.5	57.2	32.5	71.0	87.0	72.4	82.3	62.8	93.4	86.5	68.3
MinkUNet [22]	95.1	54.4	28.6	73.1	89.3	67.5	77.2	68.3	87.4	85.2	72.7
StratifiedFormer [58]	95.6	57.6	26.2	74.4	86.1	74.2	77.0	70.5	89.9	86.0	73.4
BPNet [49]	95.7	52.8	30.5	77.3	85.9	78.8	81.8	69.3	91.6	85.6	72.3
PTv2 [124]	96.0	55.9	30.4	76.6	92.6	76.7	79.7	64.4	94.2	87.6	72.2
CDFormer (ours)	97.6	57.7	33.4	69.8	95.1	82.0	82.8	68.3	95.4	89.3	72.2

TABLE 4.7. Ablation study on the CD mechanism.

Configuration	LSA	Collect	Distribute	mIoU (%)	mAcc (%)	OA (%)
i	✓			67.6	74.1	89.7
ii	✓	✓		70.1	76.6	90.7
iii	✓		✓	70.3	77.0	90.7
iv	✓	✓	✓	**72.2**	**78.5**	**91.2**

the local points, we achieve the best performances on all metrics, *i.e.*, 72.2% mIoU, 78.5% mAcc, and 91.2% OA. Besides, if without *collect* to explicitly and broadly explore long-range dependencies, using the max aggregation with *Distribute* (ses iii) significantly drops the performance from 72.2% to 70.3% on mIoU.

Context-Aware Position Encoding. To evaluate the influence of position encoding, we compare the proposed method with and without using the position

TABLE 4.8. Ablation study on CAPE.

Configuration	QKV	CA	Plain	mIoU (%)	mAcc (%)	OA (%)
i				68.9	76.6	89.7
ii	✓			69.7	76.3	91.0
iii	✓	✓		**72.2**	**78.5**	**91.2**
iv	✓		✓	70.8	77.5	90.9

encoding in the query, key, and value (**QKV**), respectively. We also compare the context-aware (**CA**) format of position encoding with a vanilla one (*i.e.*, **Plain**) as in [30]. As shown in Table 4.8, without the position encoding, we have dissatisfied results that are consistent with the observations in [58, 151]. Next, we employ a normal encoding without context-aware format (see ii) on QKV, *i.e.*, no interaction with the contexts Q, K. This time, it does work but with only marginal improvement. Lastly, when using the proposed CAPE (see iii), it can achieve the state-of-the-art performance 72.2%. Through the dynamic adaptation based on input contexts, the position clues are further enhanced in the proposed CDFormer to effectively learn local-global relations. Additionally, we also show the results of using a vanilla version of position encoding [30] (see iv) for a fair comparison.

Visualization Comparisons. To further demonstrate the improvement of CDFormer against its baselines, we provide visualizations in Figure 4.7 by comparing to **w/o CD** (see i in Table 4.7) and **w/o CAPE** (see ii in Table 4.8). Taking the first row as an example, without the collect-and-distribute mechanism, the model fails to utilize the semantic-correct long-range contexts and thus can not precisely segment local points; without CAPE, it loses the enhanced position clues for better communication within points and thus obtains dissatisfied results; the CDFormer achieves the convincing predictions as expected.

Model Scalability. We evaluate the model scalability under four stages with $[2, 2, 6, 2]$ blocks via the following three configurations:

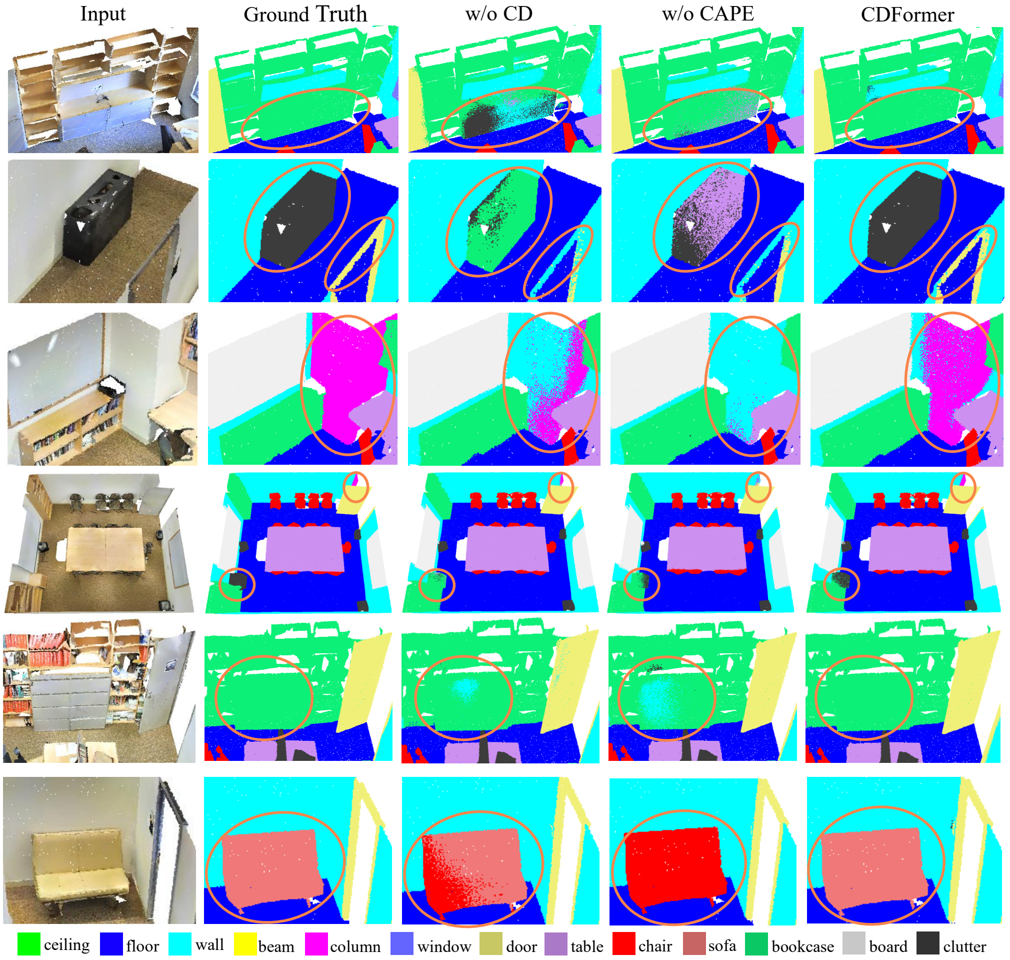


FIGURE 4.7. Visualizations of scene semantic segmentation on Area 5 of S3DIS by comparing CDFormer with its baselines and ground truth. The improved areas against baselines are highlighted by the orange circles.

- CDFormer-S: $\mathcal{C} = [16, 32, 64, 128], \mathcal{H} = [1, 2, 4, 8]$
- CDFormer-B: $\mathcal{C} = [32, 64, 128, 256], \mathcal{H} = [2, 4, 8, 16]$
- CDFormer-L: $\mathcal{C} = [48, 96, 192, 384], \mathcal{H} = [3, 6, 12, 24]$

As shown in Table 4.9, there are clear performance improvements when using a larger model.For example, the improvements are very impressive in terms of both mIoU and mAcc, *i.e.*, mIoU: $67.6\% \rightarrow 69.7\% \rightarrow 72.2\%$; mAcc: $74.9\% \rightarrow 77.1\% \rightarrow 78.5\%$. Notably, the consistent improvements by increasing the model

TABLE 4.9. Ablation study on model scalability.

Model	#Params	mIoU (%)	mAcc (%)	OA (%)
CDFormer-S	3.1M	67.6	74.9	89.8
CDFormer-B	11.7M	69.7	77.1	90.4
CDFormer-L	25.7M	**72.2**	**78.5**	**91.2**

TABLE 4.10. Ablation study on number of neighbors.

Configuration	K	mIoU (%)	mAcc (%)	OA (%)
i	$[4, 4, 4] \times 4$	69.2	75.7	90.3
ii	$[8, 8, 8] \times 4$	71.1	77.9	**91.4**
iii	$[16, 16, 16] \times 4$	**72.2**	**78.5**	91.2
iv	$[24, 24, 24] \times 4$	70.6	77.5	91.0
v	$[8, 16, 16] \times 4$	71.0	77.4	91.1
vi	$[16, 8, 16] \times 4$	70.7	77.6	90.9
vii	$[16, 16, 8] \times 4$	70.2	76.8	91.1
viii	$[8, 8, 8] \times 2 + [16, 16, 16] \times 2$	71.3	78.4	91.4
ix	$[16, 16, 16] \times 2 + [8, 8, 8] \times 2$	71.1	77.5	91.2

size illustrate the extraordinary scalability of the proposed CDFormer for point cloud analysis.

Number of Neighbors. We show the influences for using the different number of neighbors in CD Blocks for patch division, NSA, and NCA. For simplicity, we use the same K for all modules and stages. As show in Table 4.10, when increasing the number of neighbors from $K = 4$ to $K = 8$, the improvement ($69.2\% \rightarrow 71.1\%$) is significant. This is possibly because a relatively large number of neighbors can bring better long-range contexts, which help the proposed CDFormer learn the local-global structures. Notably, we also find that the advancement from $K = 8$ to $K = 16$ is modest. This kind of saturated phenomenon is expected since too much contexts may introduce different categories of semantics that degrade the local features. Similar observations have also been reported in [151]. Therefore, we use $K = 16$ as default.

Point Coverage. In Section 4.3.2, we employ the patch division on the point cloud to generate M patches with K points in each patch. Specifically, we first

TABLE 4.11. Point coverage on different stages.

Stage	Uncovered Ratio	Maximum Times Covered
1	0.28%	6.00
2	0.28%	5.26
3	0.38%	4.62
4	0.24%	4.18

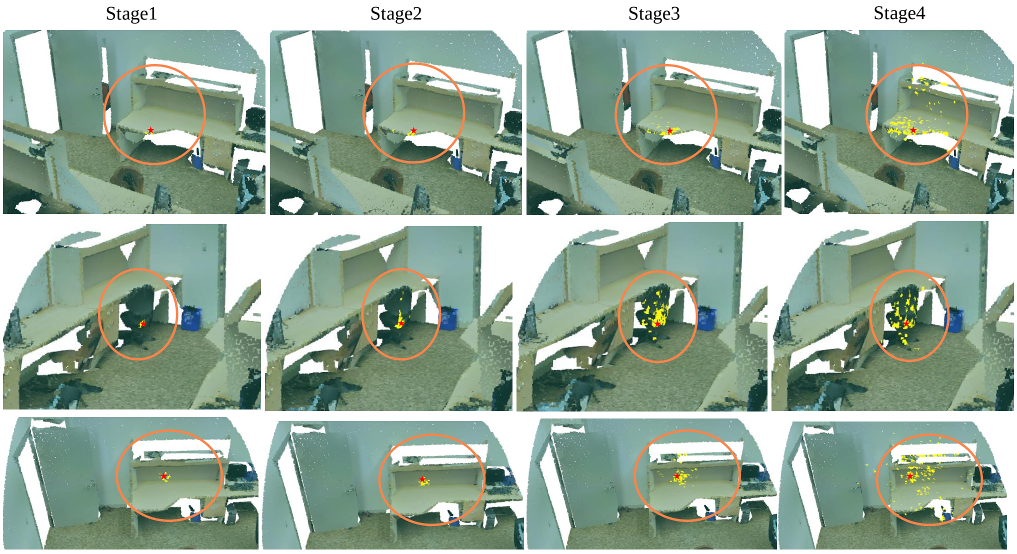


FIGURE 4.8. Visualization of activation maps where region of interest are highlighted by orange circles at different stages of CDFormer given the feature of interest (marked as red star).

utilize the FPS [31, 84] to downsampling the point features $\boldsymbol{X} \in \mathbb{R}^{N\times C}$ to local patch centers $\bar{\boldsymbol{X}} \in \mathbb{R}^{M\times C}$ where $M = N/S$ and S is the scale factor. After that, we group the K nearest neighbors around each patch center and reformulate M patches as $\hat{\boldsymbol{X}} \in \mathbb{R}^{M\times K\times C}$. However, this process may result in some points being left uncovered while others are covered multiple times. Here we present a quantitative analysis. We show the statistics of points covered multiple times or uncovered at different stages of CDFormer on the S3DIS dataset in Table 4.11. As we can see, no more than 0.4% of all points are uncovered across all stages. In addition, each point is covered by two times on average due to $M = N/8$ and $MK = 2N$, and maximally covered by six times. As a result, their impact should be negligible.

Activation Maps. We illustrate the activation maps of different stages of CDFormer to show the effectiveness of collect-and-distribute mechanism in Figure 4.8. Specifically, we backtrack the feature of interest from each stage and calculate the gradients for each input point. A larger gradient indicates that the current point contributes more to the feature of interest. In rows one to three, we select points of interest from classes *"table"*, *"chair"*, and *"bookcase"* respectively. As observed, the number of perceptive points increases inside the same category as the stage progresses due to the collect-and-distribute mechanism. For example, in the case of a point on the edge of a table, it can first interact with more points along that edge before reaching points inside the table itself. Similarly, for a point on a chair, it gradually captures all relevant points distributed across its surface while avoiding irrelevant or nearby points from other categories.

4.5 Summary

In this chapter, we introduce the innovative CDFormer for 3D point cloud analysis. With our proposed collect-and-distribute mechanism, it can capture both short- and long-range contexts simultaneously, effectively learning local-global representations of point clouds. Furthermore, we enhance position clues dynamically via context-aware position encoding to facilitate communication among point features. We conducted comprehensive experiments on S3DIS, ScanNetV2, ShapeNetPart, ModelNet40, and ScanObjectNN for segmentation and classification tasks to demonstrate the superiority of CDFormer and the advantages of each design choice.

CHAPTER 5

Multi-Scale Composite Representation Learning

This chapter delves into the investigation of multi-scale composite representation learning, aiming to effectively capture objects at different scales from the perspective of the model's overall framework. The proposed PointHR framework maintains multiple resolutions concurrently and enables frequent communication between these resolutions within each stage, thus enabling the capture of multi-scale objects.

Previous point cloud segmentation methods usually utilize an encoder-decoder framework, which initially encodes point clouds into low-resolution representations and subsequently decodes high-resolution predictions. Inspired by the success of high-resolution architectures in image dense prediction, in this chapter, we explore high-resolution architectures for 3D point cloud segmentation. Specifically, we generalize high-resolution architectures using a unified pipeline named PointHR, which includes a knn-based sequence operator for feature extraction and a differential resampling operator to efficiently communicate different resolutions. Additionally, we propose to avoid numerous on-the-fly computations of high-resolution architectures by *pre-computing* the indices for both sequence and resampling operators. By doing so, we deliver highly competitive high-resolution architectures while capitalizing on the benefits of well-designed point cloud blocks without additional effort. To evaluate these architectures for dense point cloud analysis, we conduct thorough experiments using S3DIS and

ScanNetV2 datasets, where the proposed PointHR outperforms recent state-of-the-art methods without any bells and whistles. The source code is available at https://github.com/haibo-qiu/PointHR.

5.1 Introduction

In recent years, 3D point clouds have gained extensive attention due to their crucial role in many real-world applications, such as autonomous driving [2, 67], robotics [64, 136], and AR/VR [3, 18]. Unlike 2D images, each point cloud consists of a set of 3D points characterized by their Cartesian coordinates (x, y, z), providing a view-invariant and geometry-accurate representation of the real-world 3D scene. 3D point cloud segmentation aims to predict semantic labels for all points of the point cloud, which requires a learned representation that is both spatially accurate (for individual points) and semantically rich (for category prediction). However, developing effective deep architectures for 3D point cloud representation learning is non-trivial.

3D scenes represented by point clouds often contain objects of different scales, such as a small cup on a large table in an office room, which requires the deep model to capture multi-scale contexts within point clouds. In typical deep neural networks [13, 46, 74], we tend to believe that shallow features (high resolution) contain more accurate spatial information while deep features (low resolution) include more semantic clues. Therefore, previous point cloud segmentation methods [86, 151] mainly explore multi-scale information by downsampling and upsampling features in series using an encoder-decoder paradigm: they first encode the input point clouds by progressively downsampling the point features and then decode back to the original scale using upsampling on lower scale features to generate dense predictions. Specifically, feature interactions only occur in adjacent scale representations, limiting the learning of rich multi-scale semantics. Additionally, the final largest resolution representation is recovered step-by-step from low resolution, which compromises spatial accuracy. These designs make such a vanilla encoder-decoder framework insufficient to capture rich multi-scale contexts. Inspired by high-resolution architectures for 2D visual recognition [114], we introduce PointHR, which explicitly maintains high-low resolution features in parallel during the entire network for point cloud segmentation. Unlike previous hierarchical methods [86, 151] that use only one resolution feature in each stage, PointHR keeps multiple resolutions (from high to low) simultaneously and facilitates frequent communication between all resolutions within each stage.

Different from structurally-located pixels in 2D images, point clouds consist of a set of irregular and unordered points, making it non-trivial to employ high-resolution architectures. Therefore, we approach point clouds for high-resolution architectures as follows. Firstly, the input point clouds are considered as a sequence of (x, y, z) points, allowing for a general sequence operator to be used for feature extraction. For example, a popular sequence operator could be self-attention [113], which was originally designed to model relationships between a sequence of text tokens. However, since the self-attention operator has quadratic time complexity $\mathcal{O}(N^2)$ for a sequence of length N, it is computationally infeasible to directly apply this operator to a point cloud with tens of thousands of points. One solution is to constrain the self-attention operator to perform only on each point and its K nearest neighboring points, thus reducing the complexity to be linear with respect to length N as demonstrated in Wu et al. [124] and Zhao et al. [151]. Another sequence operator could be pure MLPs, which can process unordered sequences when feeding permuted training data. To aggregate local information, K nearest neighbor features are retrieved, followed by MLPs to fuse and update the current point feature as shown in Lin et al. [68]. Thus, we formulate the basic block as a knn-based sequence operator, allowing for the use of well-designed point cloud blocks [68, 124, 151] in high-resolution architectures without additional effort.

In addition to the sequence operator, another important aspect of high-resolution architectures is the resampling operator, which can efficiently communicate different scale features in high frequency with upsampling and downsampling. A common resampling strategy in point clouds is a combination of farthest point sampling [31, 84] with knn features aggregation/interpolation. Recently, an efficient grid-based pooling and unpooling strategy [124] has been introduced, which first splits a point cloud into non-overlapping grids and then maps each grid of points to a new one and vice versa. With the unified aforementioned formulation for sequence operators, these resampling methods can be easily adopted in PointHR. However, all of them require calculating the indices for knn collection and resampling in each operation. This incurs a significant computational cost, particularly when considering the numerous resampling operations in high-resolution architectures. Fortunately, we have found that the calculations of these indices only depend on scale, specifically the corresponding point coordinates. These coordinates remain unchanged throughout the entire network. Therefore, we propose to *pre-compute* the indices for knn collection and resampling operation, which are saved to the cache so that indices retrieval is only needed instead of on-the-fly re-computation.

Our main contributions are summarized as follows:

- We present a new framework for point cloud segmentation, PointHR, which aims to maintain high resolutions for learning both semantically-rich and spatially-precise point cloud representations.
- We explore high-resolution architectures using unified sequence and resampling operators, allowing off-the-shelf point cloud blocks and layers to be employed in PointHR without additional efforts. Besides, we *pre-compute* the indices for knn collection and resampling operation to avoid on-the-fly re-computation.
- We conduct comprehensive experiments on two popular point cloud segmentation datasets, namely S3DIS [6] and ScanNetV2 [27], where

the proposed PointHR demonstrates new state-of-the-art performance, suggesting the effectiveness of exploring high-resolution architectures for point cloud segmentation.

5.2 Related Work

High-Resolution Architectures. High-Resolution Network (HRNet), originally proposed by Sun et al. [102] for human pose estimation, maintains multiple branches for multi-scale features, particularly high-resolution representations, which facilitate learning spatially more precise heatmaps for pose estimation. With its repeated cross-scale feature interactions in deep layers, HRNet also learns rich semantic features. Consequently, Wang et al. [114] extended HRNet to general dense prediction tasks such as semantic segmentation and object detection, achieving impressive performances. HRFormer [143] employs emerging attention blocks [113] to replace vanilla convolution layers in the high-resolution framework, thereby expanding its modeling capacity. Additionally, Zhang et al. [145] adopts HRNet for person re-identification to address the issue of multiple resolutions of input images. The aforementioned studies have all focused on 2D images, and we further explore high-resolution architectures for 3D point clouds.

Point Cloud Segmentation. Here we mainly introduce methods that directly take raw points as input without extra transformations (*e.g.*, voxelization and projection). These point-based methods usually develop novel modules or frameworks, such as MLPs [78, 86, 83, 84], point convolutions [108, 123], and attentions [41, 58, 81, 88, 151], to directly learn from raw points. Particularly, PointNet [83] was the first to process point clouds using multi-layer perceptrons (MLPs). PointNet++ [84] improved upon this by introducing a hierarchical neural network that learns local features. PointNext [86], on the other hand, proposes advanced training strategies to significantly improve the performance of PointNet++. Additionally, it introduces InvResMLP blocks and formulates the PointNext architecture for further improvement. Meanwhile, KPConv [108] presents kernel point convolution as a new point convolution operator that takes neighboring points as input and processes them with spatially located weights. Recently, the popular transformer architecture has been introduced to point cloud tasks [58, 81, 124, 151]. PTv1 [151] proposes vector attention to aggregate neighbor features. PTv2 [124] further introduces grouped vector attention to more efficiently learn discriminative representations while avoiding overfitting. Stratified Transformer [58] employs local window-based self-attention and captures long-range contexts by sampling nearby points densely and distant points sparsely. Differently, in this chapter, we mainly explore how to form an effective backbone framework, and the block designs of the above methods can be trivially integrated in our framework.

5.3 Method

First, we present an overview of high-resolution architectures for 3D point clouds. Next, we delve into the unified format of sequence operator with various instantiations. Finally, we investigate efficient multi-scale fusion in conjunction with the resampling operator using the *pre-compute* strategy.

5.3.1 High-Resolution Architectures

A point cloud can be represented as a sequence of N points $\boldsymbol{P} \in \mathbb{R}^{N \times C_{raw}}$, where C_{raw} is the feature dimension of each point that typically includes xyz coordinates as well as other attributes such as its normal vector. The goal of point cloud segmentation is to predict a semantic category label for each point $(x, y, z) \in \boldsymbol{P}$. Specifically, the final predictions share the same spatial dimension with the raw points, *i.e.*, $\boldsymbol{Y} \in R^{N,cls}$, where cls is the total number of semantic categories.

The overall PointHR pipeline for point cloud segmentation is illustrated in Figure 5.1, where the raw input point clouds $N \times C_{raw}$ is first embedded to $N \times C_0$, and then downsampled to $N/S \times C_1$. Next, it starts with a high-resolution branch as the first stage by taking the point feature $N/S \times C_1$ as input. Subsequently, additional high-to-low resolution branches calculated by spatially dividing the factor S and doubling the channel of feature dimension are incrementally integrated into the architecture as new stages. For $i = 1, 2, 3, 4$, the i_{th} stage consisting of i branches outputs i different scales point features, and each branch has M_i blocks that is building by stacking B_i sequence operators. After each stage, the learned multi-resolution features are fused as the input of the next stage in a per-branch manner. Notably, the resolution corresponds to the number of points when applying high-resolution architectures for point clouds. The entire framework can be formulated as follows:

$$\begin{tikzcd}
\boldsymbol{F}_{11} \arrow[r] \arrow[dr] & \boldsymbol{F}_{21} \arrow[r] & \boldsymbol{F}_{31} \arrow[r] & \boldsymbol{F}_{41} \\
& \boldsymbol{F}_{22} \arrow[r] \arrow[dr] & \boldsymbol{F}_{32} \arrow[r] & \boldsymbol{F}_{42} \\
& & \boldsymbol{F}_{33} \arrow[r] \arrow[dr] & \boldsymbol{F}_{43} \\
& & & \boldsymbol{F}_{44},
\end{tikzcd} \tag{5.1}$$

where point feature $\boldsymbol{F}_{ij} \in \mathbb{R}^{\frac{N}{S^i} \times 2^{j-1} C_i}$ where $i \in \{1, 2, 3, 4\}$ and $j \in \{1, \cdots, i\}$. It should be also noted that previous hierarchical methods [86, 124] only contain $\boldsymbol{F}_{11} \rightarrow \boldsymbol{F}_{22} \rightarrow \boldsymbol{F}_{33} \rightarrow \boldsymbol{F}_{44}$, which is corresponding to the red-arrow flow in Figure 5.1.

These outputs from the final stage of PointHR join forces with the original resolution feature in the decoder, enabling the propagation of information from low-resolution to high-resolution. Finally, a segmentation head that consists of linear layers generates the final prediction of $N \times C_{cls}$, where C_{cls} represents the number of semantic categories. We summarize the typical configurations of PointHR via the number of modules M_i, the number of blocks B_i, and the number of channels C_i, *e.g.*, $(M_1, M_2, M_3, M_4) = (1, 1, 2, 1)$ and $(B_1, B_2, B_3, B_4) =$

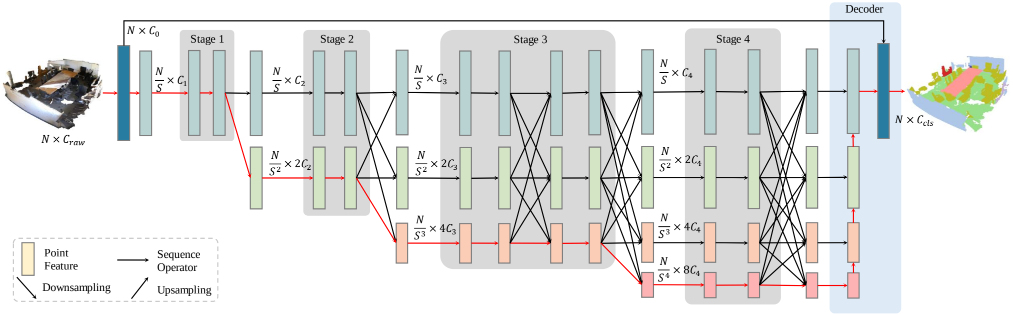


FIGURE 5.1. The overall PointHR pipeline for point cloud segmentation.

$(2, 2, 2, 2)$ corresponding to Figure 5.1. A detailed formulation of PointHR configurations is shown in Appendix 5.6.8.

5.3.2 Sequence Operator

For 2D images, high-resolution architectures [114] usually adopt convolutional layers as the basic blocks to extract local information, while it is not straightforward to employ such an operation on point clouds due to the irregular and unordered characteristics. Therefore, to explore high-resolution architectures for point clouds, we first formulate the aforementioned basic block via a unified sequence operator, *i.e.*, regarding point clouds as a sequence of points and treating it as a sequence processing task. By doing this, we hope that most existing point cloud blocks can be directly used in high-resolution architectures. Specifically, when taking the point clouds $\boldsymbol{P} \in \mathbb{R}^{N \times C}$ as input, the sequence operator Θ will first embed each point $\boldsymbol{p}_i$ where $i \in \{1, \cdots, N\}$ into new feature space $\bar{\boldsymbol{p}}_i$, then K nearest neighbors (KNN) for each point are fetched as $\bar{\boldsymbol{p}}_j$ where $j \in \{1, \cdots, K\}$ to collect local clues. After that, the sequence operator aggregates those local information to the current point $\hat{\boldsymbol{p}}_i$. Lastly, the final representation $\tilde{\boldsymbol{p}}_i$ is obtained by incorporating original features with updated current features. The whole process is illustrated in the left part of Figure 5.2 and can be

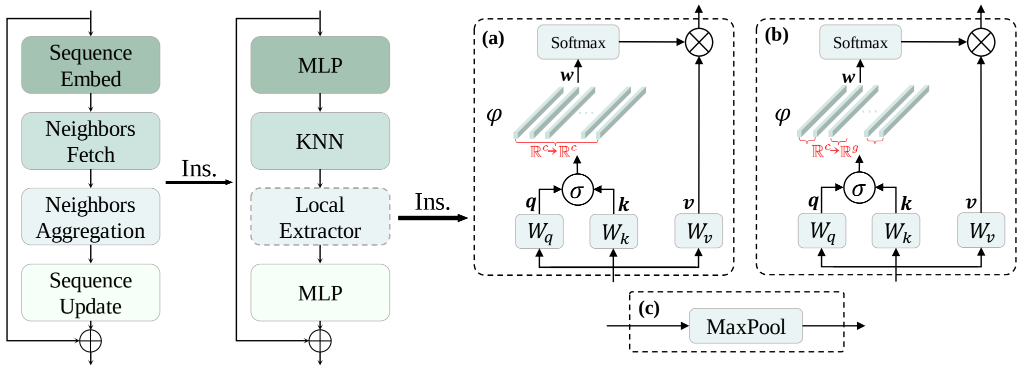


FIGURE 5.2. An illustration of the general sequence operator Θ along with its instantialization where the “Ins.” represents the instantialization. Note that (a) vector attention [151], (b) grouped vector attention [124], and (c) pure MLPs [68] are three instantializations of the Local Extractor defined by Equation 5.4, 5.5, and 5.6 respectively.

mathematically formulated as follows:

$$\bar{\boldsymbol{p}}_i = \Theta_{se}(\boldsymbol{p}_i) \quad \Rightarrow \quad \bar{\boldsymbol{p}}_j = \Theta_{nf}(\bar{\boldsymbol{p}}_i) \tag{5.2}$$

$$\hat{\boldsymbol{p}}_i = \Theta_{na}(\bar{\boldsymbol{p}}_j) \quad \Rightarrow \quad \tilde{\boldsymbol{p}}_i = \Theta_{su}(\hat{\boldsymbol{p}}_i) + \boldsymbol{p}_i. \tag{5.3}$$

where Θ_{se}, Θ_{nf}, Θ_{na}, and Θ_{su} indicate *Sequence Embed, Neighbors Fetch, Neighbors Aggregation*, and *Sequence Update*, respectively, in Figure 5.2.

To show the effectiveness of the aforementioned sequence operator formulation, we discuss several popular point cloud blocks as the possible instantialization in the following. For example, self-attention [113] proposed for handling a sequence of words is capable of capturing the relationships of all elements in the sequence. Previous approaches [124, 151] employ the attention only on each point with its K neighboring points to reduce time complexity from $\mathcal{O}(N^2)$ to $\mathcal{O}(NK^2)$, rather than utilizing a global attention on all sequence elements. In particular, PTv1 [151] has shown that vector attention is more effective in handling point clouds than the original scalar attention [113]. It employs attention weights that are vectors calculated based on the relation operation between query and key, which can effectively modulate individual feature channels. Specifically, given a point $\boldsymbol{p}_i$ and its neighbors $\mathcal{N}(\boldsymbol{p}_i) = \{\boldsymbol{p}_j | \boldsymbol{p}_j \in \text{KNN}(\boldsymbol{p}_i)\}$, Multilayer perceptrons (MLPs) are employed to map the point feature to query $\boldsymbol{q}_i$, key $\boldsymbol{k}_i$, and value $\boldsymbol{v}_i$, and then the vector attention can be formulated as follows:

$$\boldsymbol{w}_{ij} = \varphi(\sigma(\boldsymbol{q}_i, \boldsymbol{k}_i)), \qquad \boldsymbol{p}_i^{attn} = \sum_{\boldsymbol{p}_j \in \mathcal{N}(\boldsymbol{p}_i)} \text{Softmax}(\boldsymbol{W}_i)_j \odot \boldsymbol{v}_j, \tag{5.4}$$

where $\odot$ represents the Hadamard product. σ denotes a relation operation such as subtraction. $\varphi : \mathbb{R}^c \rightarrow \mathbb{R}^c$ is the MLPs which calculates attention vectors to re-weight $\boldsymbol{v}_j$ by channels before performing aggregation. PTv2 [124] further introduces a grouped vector attention to improve the model efficiency. Specifically, it is achieved by dividing channels of the value $\boldsymbol{v}_i \in \mathbb{R}^c$ evenly into g groups ($1 \leq g \leq c$). Then MLPs $\varphi : \mathbb{R}^c \rightarrow \mathbb{R}^g$ outputs a grouped attention vector with g channels instead of c. Channels within the same group share the same scalar attention weight from the grouped attention vector. As a result, Equation 5.4 is modified as follows:

$$\boldsymbol{w}_{ij} = \varphi(\sigma(\boldsymbol{q}_i, \boldsymbol{k}_i)), \qquad \boldsymbol{p}_i^{attn} = \sum_{\boldsymbol{p}_j \in \mathcal{N}(\boldsymbol{p}_i)} \sum_{l=1}^{g} \sum_{m=1}^{c/g} \text{Softmax}(\boldsymbol{W}_i)_{jl} \cdot \boldsymbol{v}_j^{lc/g+m}. \tag{5.5}$$

The sequence operator can be pure MLPs as well, which can handle unordered sequences when augmenting the training data by all kinds of permutations. Specifically, in Lin et al. [68], the features of K nearest neighbours are used to collect local clues, which is followed by maxpooling and MLPs to fuse current point features, which can be formulated as follows:

$$\boldsymbol{p}_i^{aggre} = \text{MaxPool}(\phi(\boldsymbol{p}_j)), \qquad \boldsymbol{p}_j \in \mathcal{N}(\boldsymbol{p}_i), \tag{5.6}$$

where ϕ is the MLPs that embeds the input point feature $\boldsymbol{p}_i$, and $\mathcal{N}(\boldsymbol{p}_i)$ represents the neighbors of $\boldsymbol{p}_i$. From an united perspective, we can consider the instantialization of Equation 5.2 and 5.3 as follows:

$$\Theta_{se} = \text{MLP}, \quad \Theta_{nf} = \text{KNN}, \quad \Theta_{su} = \text{MLP}. \tag{5.7}$$

Notably, the only difference lies in Θ_{na}, which acts as the local extractor to capture neighbor information as shown in the right part of Figure 5.2, defined by Equation 5.4, 5.5, and 5.6.

5.3.3 Resampling Operator and Multi-Scale Fusion

Besides the sequence operator, a resampling operator is required to perform cross-resolution interactions between different branches. Specifically, for j_{th} branch where $j \in \{1, \cdots, i\}$ in the i_{th} stage, there are three possible fusion situations: 1) when fusing features from the same branch, a simple identity connection is applied; 2) when features come from the above a_{th} branch, *i.e.*, higher resolution than current one, $j_{th} - a_{th}$ number of successive downsampling modules are employed; 3) if features from the below b_{th} branch, *i.e.*, lower resolution than current one, an upsampling module is adopted. Finally, those processed features are summed as the updated features for j_{th} branch. For example, we can calculate the fused features of 2_{nd} branch in the 3_{rd} stage as follows:

$$\bar{\boldsymbol{F}}_{32} = \boldsymbol{F}_{32} + \text{downsampling}(\boldsymbol{F}_{31}) + \text{upsampling}(\boldsymbol{F}_{33}). \tag{5.8}$$

To effectively communicate different scales, the above fusion process are frequently made between and inside stages. However, those resampling operations, including both downsampling and upsampling, are extremely time-consuming in point clouds due to the unordered characteristics. For example, when provided with an input point cloud containing N points, the classical farthest point sampling (FPS) [31, 84] requires the calculation of downsampling and upsampling neighboring indices. This process has a time complexity of $\mathcal{O}(N^2)$ as mentioned in Hu et al. [48]. Another option could be the simpler grid pooling and unpooling method proposed in Wu et al. [124], which divides a point cloud into non-overlapping grids. The pooling is to sample each grid of points as a new points, and the unpooling is simply back-projecting the point to the original higher resolution grid of points.

However, both of them require calculating the indices for knn collection and resampling in each operation. This leads to a high computational cost, particularly because of the numerous resampling operations in PointHR. Fortunately, we identify those indices only depend on current point scale, which stays unchanged throughout the entire network. Specifically, we find that those knn indices solely rely on point coordinates so that each branch with the same resolution shares the same knn indices across all stages. As for resampling indices, they can be also shared when operating two specific resolution branches across different stages. Hence, we propose to *pre-compute* the indices for knn collection and resampling operation, which are saved to the cache to avoid on-the-fly computations, thus making it possible to efficiently employ high-resolution architectures for 3D dense point cloud analysis. The detailed description can be found in Appendix 5.6.5.

5.4 Experiments

5.4.1 Datasets and Metrics

We evaluate the proposed PointHR on two widely-used benchmarks, *i.e.*, S3DIS [6] and ScanNetV2 [27], for point cloud semantic segmentation. S3DIS contains 271 rooms in 6 areas collected from three different buildings. Each point in the room is annotated with one of 13 semantic categories such as *“ceiling"* and *“bookcase"*. Following previous methods [124, 151], we keep Area 5 as the testing set and use remains for training PointHR. ScanNetV2 is another larger dataset, which consists of 1,513 room scenes, where 1,201 scenes for training and 312 for validation. Point clouds are created by sampling vertices from meshes that are reconstructed from RGB-D frames. Each sampled point is then assigned a semantic label from one of 20 categories, such as *“floor"* and *“table"*. Regarding the evaluation metrics, similar to Qian et al. [86], Wu et al. [124] and Zhao et al. [151], we adopt the mean class-wise intersection over union (mIoU), mean of class-wise accuracy (mAcc), and overall point-wise accuracy (OA). We further present evaluations on ShapeNetPart [139] and ModelNet40 [125] in Appendix 5.6.1 and 5.6.2.

5.4.2 Implementation Details

For the default configurations of PointHR, we employ $(M_1, M_2, M_3, M_4) = (1, 1, 5, 4)$ and $(C_1, C_2, C_3, C_4) = (64, 32, 32, 32)$, $B_i = 2$ and $K_i = 16$ for all $i \in \{0, 1, 2, 3\}$, the grouped vector attention defined by Equation 5.5 as the sequence operator, as well as the grid pooling and unpooling discussed in Section 5.3.3 as the sampling strategy.

For S3DIS [6], following the practice in Wu et al. [124] and Zhao et al. [151], the grid sampling with size 0.04m is first employed on the raw input points. During the training process, we apply popular data augmentations such flip, scale, jitter, random drop, and we also use the sphere crop on the entire scene and constrain the maximum number of input points to 100,000. Considering the training set of S3DIS is relatively small (*i.e.*, only 204 samples), we follow Qian et al. [86], Wu et al. [124] and Zhao et al. [151] to enlarge the size by repeating $30\times$ to obtain 6,120 samples. We train PointHR using four V100 GPUs for 100 epochs with batch size 12, and set the learning rate to 0.006 and drop it by $1/10$ at 60 and 80 epochs. AdamW optimizer [75] with the weight decay 0.05 and cross-entropy loss are applied. The pooling size are set to $(0.1, 0.2, 0.4, 0.8)$ for gird pooling to achieve approximately $6\times$ downsampling scale. For ScanNetV2 [27], we follow Wu et al. [124] to employ grid sampling with size 0.02m on the raw point clouds. We also repeat the its training set (1,201 scenes) by $9\times$ to get 10,809 samples. Considering its larger scale than S3DIS, AdamW optimizer [75] with a smaller weight decay 0.02 are applied. OneCycleLR [100] scheduler is employed, where

TABLE 5.1. Quantitative results under mIoU (%), mAcc (%), and OA (%) metrics including per-category IoU are reported on Area 5 of S3DIS [6]. The **bold** denotes the best performance.

Method	**mIoU**	**mAcc**	**OA**	ceiling	floor	wall	beam	column
PointNet [83]	41.1	49.0	-	88.8	97.3	69.8	0.1	3.9
SegCloud [107]	48.9	57.4	-	90.1	96.1	69.9	0.0	18.4
PointCNN [66]	57.3	63.9	85.9	92.3	98.2	79.4	0.0	17.6
SPGraph [59]	58.0	66.5	86.4	89.4	96.9	78.1	0.0	42.8
PCT [41]	61.3	67.7	-	92.5	98.4	80.6	0.0	19.4
HPEIN [54]	61.9	68.3	87.2	91.5	98.2	81.4	0.0	23.3
MinkUNet [22]	65.4	71.7	-	91.8	98.7	86.2	0.0	34.1
KPConv [108]	67.1	72.8	-	92.8	97.3	82.4	0.0	23.9
CGA-Net [77]	68.6	-	-	94.5	98.3	83.0	0.0	25.3
CBL [105]	69.4	75.2	90.6	93.9	98.4	84.2	0.0	37.0
PTv1 [151]	70.4	76.5	90.8	94.0	98.5	86.3	0.0	38.0
PointNext [86]	70.5	76.8	90.6	94.2	98.5	84.4	0.0	37.7
PointMeta [68]	71.3	-	90.8	-	-	-	-	-
PointMixer [21]	71.4	77.4	-	94.2	98.2	86.0	0.0	43.8
PTv2 [124]	71.6	77.9	91.1	-	-	-	-	-
StraFormer [58]	72.0	78.1	91.5	96.2	98.7	85.6	0.0	46.1
PointHR (ours)	**73.2**	**78.7**	**91.8**	94.0	98.5	87.5	0.0	53.7

Method	window	door	table	chair	sofa	bookcase	board	clutter
PointNet [83]	46.3	10.8	59.0	52.6	5.9	40.3	26.4	33.2
SegCloud [107]	38.4	23.1	70.4	75.9	40.9	58.4	13.0	41.6
PointCNN [66]	22.8	62.1	74.4	80.6	31.7	66.7	62.1	56.7
SPGraph [59]	48.9	61.6	84.7	75.4	69.8	52.6	2.1	52.2
PCT [41]	61.6	48.0	76.6	85.2	46.2	67.7	67.9	52.3
HPEIN [54]	65.3	40.0	75.5	87.7	58.5	67.8	65.6	49.4
MinkUNet [22]	48.9	62.4	81.6	89.8	47.2	74.9	74.4	58.6
KPConv [108]	58.0	69.0	81.5	91.0	75.4	75.3	66.7	58.9
CGA-Net [77]	59.6	71.0	92.2	82.6	76.4	77.7	69.5	61.5
CBL [105]	57.7	71.9	91.7	81.8	77.8	75.6	69.1	62.9
PTv1 [151]	63.4	74.3	89.1	82.4	74.3	80.2	76.0	59.3
PointNext [86]	59.3	74.0	83.1	91.6	77.4	77.2	78.8	60.6
PointMeta [68]	-	-	-	-	-	-	-	-
PointMixer [21]	62.1	78.5	90.6	82.2	73.9	79.8	78.5	59.4
PTv2 [124]	-	-	-	-	-	-	-	-
StraFormer [58]	60.0	76.8	92.6	84.5	77.8	75.2	78.1	64.0
PointHR (ours)	62.9	80.2	84.2	92.5	75.4	76.5	84.8	61.8

the learning raises from 0.0005 to 0.005 in the first 5 epochs and cosine annealing to 0 in the remaining 95 epochs. We use $(0.06, 0.15, 0.375, 0.9375)$ for gird pooling to approximate $6\times$ downsampling scale. Other settings are kept the same as in S3DIS.

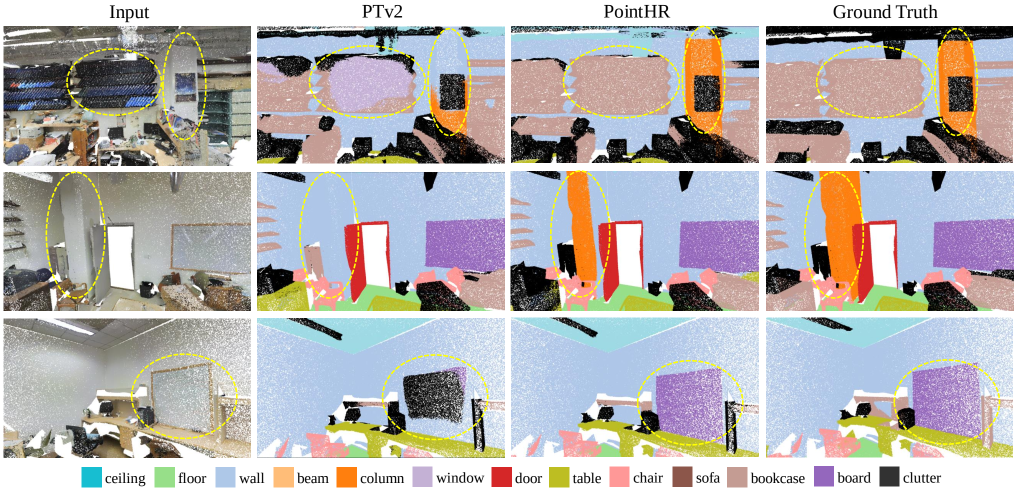


FIGURE 5.3. Visualization of point cloud segmentation results on the Area 5 of S3DIS. Note that yellow circles highlight the improvements made by PointHR over PTv2.

5.4.3 Results on S3DIS

Quantitative Results: Table 5.1 demonstrates the results of recent state-of-the-art methods and the proposed PointHR on the Area 5 of S3DIS. Apparently, our PointHR achieves best performances on all three metrics, *i.e.*, mIoU (%), mAcc (%), and OA (%). It surpasses different kinds of methods including transformer-based [58, 124, 151], mlp-based [86, 68, 21], and graph-based [108, 66] approaches. For example, PointHR outperforms the current state-of-the-art model, StraFormer [58], with a clear margin in terms of mIoU, 73.2% *vs.* 72.0%. It is also worthy noting that PointHR uses the same block (grouped vector attention) as in PTv2 [124], but PointHR takes the lead on all three metrics, *i.e.*, 73.2% *vs.* 71.6%, 78.7% *vs.* 77.9%, and 91.8% *vs.* 91.1%. We believe that this is partially because of the high-resolution architectures, which explicitly maintain multi-scale features in parallel and allow for interactions across scales. Besides, as for per-category IoU, we find that PointHR achieves better performances on those large flat objects such as *“column"*, *“door"*, and *“board"*. We thus conjecture that PointHR has a very good ability to efficiently capture information at different scales to accurately segment both boundary and inner points.

TABLE 5.2. Quantitative results on ScanNetV2.

Method	#Params	FLOPs	Val (%)	Test (%)
PointNet++ [84]	1.0M	7.2G	53.5	55.7
RandLA-Net [48]	1.3M	5.8G	-	64.5
PointConv [123]	-	-	61.0	66.6
PointASNL [134]	-	-	63.5	66.6
KPConv [108]	15M	-	69.2	68.6
CBL [105]	18.6M	-	-	70.5
PTv1 [151]	7.8M	5.7G	70.6	-
PointNext [86]	41.6M	84.8G	71.5	71.2
PointMeta [68]	19.7M	11.0	72.8	71.4
SparseCNN [39]	-	-	69.3	72.5
MinkUNet [22]	-	-	72.2	73.6
StraFormer [58]	8.02M	12.4G	74.3	74.7
BPNet [49]	-	-	73.9	74.9
PTv2 [124]	11.3M	14.3G	75.4	75.2
PointHR (ours)	7.1M	10.3G	75.4	**76.6**

Qualitative Results: We visually compare the predictions made by PTv2 [124] and PointHR, as well as ground truths on Area 5 of S3DIS in Figure 5.3. As we see that the *"column"* area is challenging since it usually looks very similar to the *"wall"* area, but different only in shape. However, while both the *"column"* and *"wall"* have a flat shape in short-range views, they exhibit distinct characteristics in long-range perspectives. Specifically, the *"column"* takes on a cube-like shape, which serves as a key feature for differentiation from the *"wall"*. Our PointHR can effectively maintain high-resolution features and incorporate cross-scale fusion within the network architecture. This enables better capture long-long range contexts, which are crucial for accurate recognition on both internal and boundary points of *"columns"*. Similar situations are observed on the *"board"*.

5.4.4 Results on ScanNetV2

Quantitative Results: Next we evaluate PointHR on the more challenging benchmark ScanNetV2. Similar to Han et al. [43], Liu et al. [72], Wu et al. [124] and

Yang et al. [137], except for the results on validation set, we have further submited our predictions on testing set to the official testing server. All the performances using mIoU metric are reported in Table 5.2. Not surprisingly, PointHR delivers another state-of-the-art performance 76.6%, which surpasses the current best method, *i.e.*, 75.2% from PTv2 [124] even with about 40$\%$ fewer parameters and FLOPs (7.1M *vs.* 11.3M and 10.3G *vs.* 14.3G). In addition, PointHR directly taking points as input also outperforms those voxel-input methods [49, 22] that can conveniently use 3D convolutions. Note that BPNet [49] achieves 74.9% using both point clouds and corresponding 2D images as input, which is still inferior to PointHR that only works on points.

Qualitative Results: Figure 5.4 demonstrates the semantic segmentation results obtained by PointHR on ScanNetV2. Our PointHR performs well on different kinds of scenes such as office, bathroom, and bedroom. It is also worth noting that PointHR can handle numerous objects in a scene well, as observed with the many chairs surrounding the table being precisely segmented.

5.4.5 Ablation Studies

All ablation studies are conducted on the validation set of ScanNetV2.

Model Scalability: We investigate the scalability of PointHR by fixing $B_i = 2$, $K_i = 16$ for all $i \in \{0, 1, 2, 3\}$ and increasing modules $M = (M_1, M_2, M_3, M_4)$ and channels $C = (C_1, C_2, C_3, C_4)$. Four different scales of PointHR are obtained as demonstrated in Table 5.3. As we can observe, the overall trend is that the metric mIoU increase as the model size grows. PointHR-S only marginally outperforms PointHR-T with deeper depth, which we suspect is because only 16 channels for $\{C_i | i \in \{1, 2, 3\}\}$ significantly limit the capacity of model. When doubling the channels to 32 as PointHR-B, the performance is remarkably boosted from 73.0% to 74.9$\%$. We further increase the depth to get PointHR-L with the best performance 75.4$\%$.

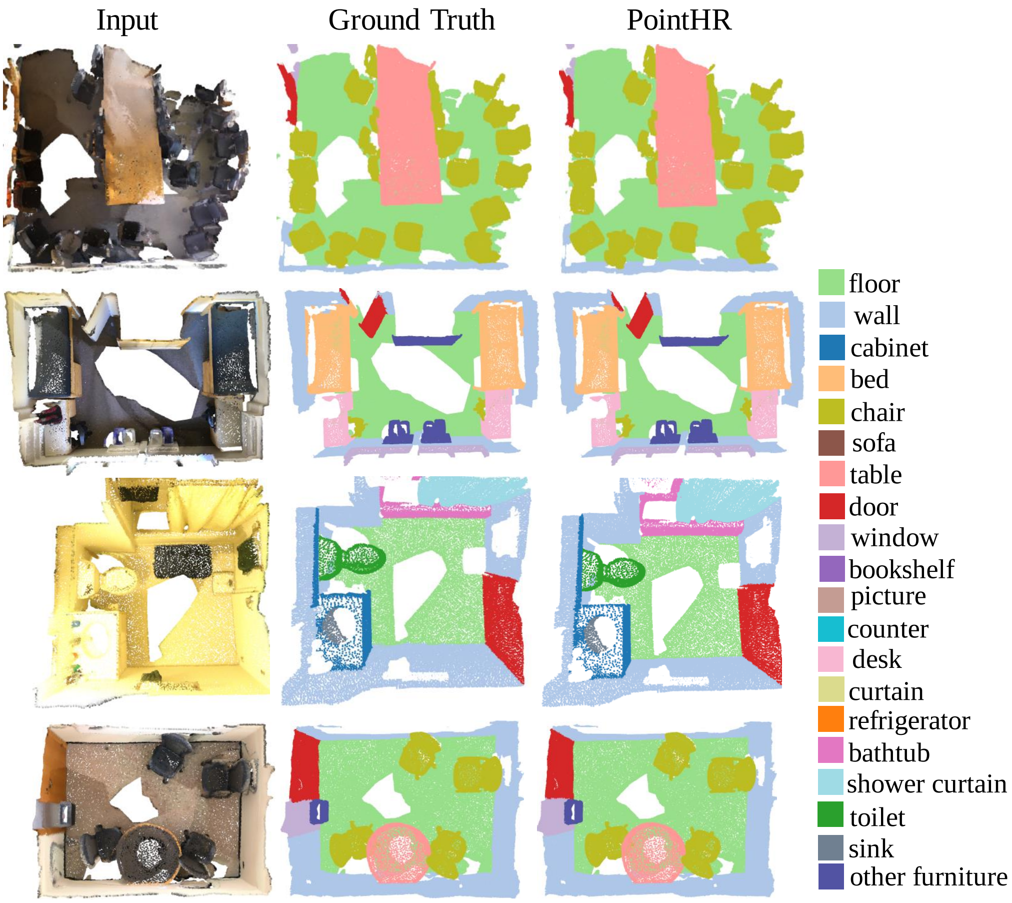


FIGURE 5.4. Visualization of point cloud segmentation by the proposed PointHR on ScanNetV2. More illustrations are available in Appendix 5.6.9.

TABLE 5.3. Ablation studies on model scalability.

Model	$(M_1, \cdots, M_4)$	$(C_1, \cdots C_4)$	#Params	FLOPs	mIoU (%)
PointHR-T	$(1, 1, 2, 1)$	$(64, 16, 16, 16)$	0.6M	2.5G	72.7
PointHR-S	$(1, 1, 3, 2)$	$(64, 16, 16, 16)$	1.0M	2.8G	73.0
PointHR-B	$(1, 1, 3, 2)$	$(64, 32, 32, 32)$	3.9M	6.8G	74.9
PointHR-L	$(1, 1, 5, 4)$	$(64, 32, 32, 32)$	7.1M	10.3G	75.4

Sequence Operator: We explore three different sequence operators discussed in Section 5.3.2, *i.e.*, pure MLPs [68], vector attention [151], and grouped vector attention [124] defined by Equation 5.6, 5.4, and 5.5 respectively. The results are presented in Table 5.4 by comparing PointHR with the specific sequence operator

TABLE 5.4. Ablation studies on different sequence operators (SO). *mlps*: pure MLPs [68], *va*: vector attention [151], *gva*: grouped vector attention [124]. HR denotes the high-resolution architecture.

Method	SO	HR	#Params	FLOPs	Val (%)	Test (%)
PointMeta [68]	*mlps*	✗	19.7M	11.0G	72.8	71.4
PointHR-B	*mlps*	✓	2.3M	1.8G	**74.2**	-
PTv1 [151]	*va*	✗	7.8M	5.7G	70.6	-
PointHR-B	*va*	✓	3.4M	3.0G	**74.6**	-
PTv2 [124]	*gva*	✗	11.3M	14.3G	**75.4**	75.2
PointHR-B	*gva*	✓	3.9M	6.8G	74.9	-
PointHR-L	*gva*	✓	7.1M	10.3G	**75.4**	**76.6**

with its original method. As we can see from the first group, PointHR-B integrated with *mlps* surpasses PointMeta [68] by 1.4% mIoU but with significantly smaller parameters and FLOPs (2.3M *vs.*19.7M and 1.8G *vs.*11.0G) . Meanwhile, PointHR-B with *va* improves the performance of PTv1 [151] by a large margin 4% with approximately half parameters and FLOPs. The above observations confirm the effectiveness and generalization ability of PointHR.

Resampling Strategy: We compare the furthest point sampling plus KNN to grid pooling under the model configuration PointHR-B. The FPS version named as PointHR-B-FPS is also *pre-computing* the downsampling and upsampling index for later fetching to make a fair comparison. It achieves 73.9% mIoU, which is clearly inferior to 74.9% made by PointHR-B. Besides, the speed of PointHR is about 40% faster than PointHR-B-FPS as it requires 244 V100 GPU hours while PointHR-B-FPS needs 412 GPU hours. If the model becomes deeper and increases the frequency of cross scale interactions, the gap on speed will further grow.

5.5 Summary

In this chapter, we explore high-resolution architectures for 3D point cloud segmentation. To achieve this, we formulate the proposed PointHR in a unified way using a sequence operator and a resampling operator, enabling use of those off-the-shelf point cloud blocks and modules without additional efforts. Besides, we propose to *pre-compute* the indices for knn collection and resampling operation to avoid on-the-fly computations, thus efficiently employing high-resolution architectures. Comprehensive experiments on popular point cloud segmentaion datasets, S3DIS [6] and ScanNetV2 [27], demonstrate the effectiveness of high-resolution architectures for 3D dense point cloud analysis and also yield new state-of-the-art performances.

5.6 Appendix

In this appendix, we begin by evaluating the proposed PointHR on ShapeNetPart [139] and ModelNet40 [125], respectively. Then we conduct ablation studies on the decoder design. Next, the detailed per-category IoUs on the testing split of ScanNetV2 are provided. Subsequently, we perform a memory analysis on PointHR. After that, we demonstrate the details of *pre-compute* indices as outlined in Section 5.3.3. Finally, we presenting the typical configurations table of PointHR and deliver more additional visualizations.

5.6.1 Evaluation on ShapeNetPart

We have further evaluated the proposed PointHR on ShapeNetPart [139], which stands as a widely recognized dataset employed for the task of point cloud part segmentation. This dataset comprises 16,880 3D models, categorized into 16 distinct shape categories, such as *“car"* and *“table"*. In the context of data splitting, 14,006 models are designated for the training set, while the remaining

TABLE 5.5. Results on ShapeNetPart.

Method	#Params	mIoU (%)
PointNet [83]	3.6M	83.7
PointNet++ [84]	1.5M	85.1
DGCNN [118]	1.3M	85.2
PointConv [123]	-	85.7
ASSANet-L [85]	-	86.1
PointMLP [78]	13.2M	86.1
PVCNN [73]	-	86.2
PCT [41]	-	86.4
KPConv [108]	14.3M	86.4
PTv1 [151]	7.8M	86.6
StraFormer [58]	8.0M	86.6
CurveNet [126]	-	86.8
PointNeXt [86]	22.5M	87.0
PointHR (ours)	7.4M	**87.2**

2,874 models are reserved for the testing set. Notably, each category exhibits a varying number of constituent parts, ranging from 2 to 6, yielding a total of 50 distinct parts across all categories. The reported evaluation metric is the instance mean Intersection over Union (mIoU). Table 5.5 presents the results of PointHR compared to previous methods. As observed, our PointHR achieves state-of-the-art performance 87.2% with reasonable parameters 7.4M.

5.6.2 Evaluation on ModelNet40

Although PointHR is specifically designed for point cloud segmentation, it can also be used for classification tasks with slight modifications. Specifically, we modify the feature propagation direction in the decoder by changing it from low-to-high to high-to-low, and then applying a global maxpooling. We evaluate PointHR on ModelNet40 dataset [125], which serves as a canonical dataset widely utilized for point cloud classification. This dataset comprises a total of 9,843 CAD models allocated to the training set, with an additional 2,468

TABLE 5.6. Results on ModelNet40.

Method	Inputs	#Points	OA(%)
PointNet [83]	xyz	1024	89.2
PointNet++ [84]	xyz	1024	90.7
PointNet++ [84]	xyz+norm	5000	91.9
PointCNN [66]	xyz	1024	92.5
PointConv [123]	xyz+norm	1024	92.5
KPConv [108]	xyz	7000	92.9
DGCNN [118]	xyz	1024	92.9
PointASNL [134]	xyz	1024	92.9
PointNext [86]	xyz	1024	93.2
PosPool [70]	xyz	5000	93.2
PCT [41]	xyz	1024	93.2
SO-Net [60]	xyz	5000	93.4
PTv1 [151]	xyz	1024	93.7
PointMLP [78]	xyz	1024	**94.1**
PointHR (ours)	xyz	1024	93.9

CAD models designated for the testing set. These models span across 40 distinct object categories. The primary evaluation metric employed for assessing model performance is the overall accuracy (OA). We report the performance of PointHR and other previous approaches in Table 5.6. Finally, PointHR achieves comparable accuracy to previous state-of-the-art methods.

5.6.3 Decoder Design

For the decoder design, we conduct ablation studies including 1) directly sum of different resolution features; 2) progressively fusion of adjacent features; and 3) progressively fusion of adjacent features with the sequence operator for refinement, which are corresponding to the Sum, PG and PGR of Table 5.7, respectively. We choose PointHR-T as the baseline and perform corresponding experiments on ScanNet under mIoU (%). As shown in Table 5.7, we find that progressively

TABLE 5.7. Strategies.

	PointHR-T
Sum	71.6%
PG	71.9%
PGR	72.7%

TABLE 5.8. Results of per-category IoU (%) on the testing set from ScanNetV2 corresponding to Table 5.2.

Method	**mIoU**	bathtub	bed	bookshelf	cabinet	chair	counter	curtain	desk	door	floor	other.	picture	refrig.	shower.	sink	sofa	table	toilet	wall	window
PointNet++ [84]	55.7	73.5	66.1	68.6	49.1	74.4	39.2	53.9	45.1	37.5	94.6	37.6	20.5	40.3	35.6	55.3	64.3	49.7	82.4	75.6	51.5
RandLA-Net [48]	64.5	77.8	73.1	69.9	57.7	82.9	44.6	73.6	47.7	52.3	94.5	45.4	26.9	48.4	74.9	61.8	73.8	59.9	82.7	79.2	62.1
PointConv [123]	66.6	78.1	75.9	69.9	64.4	82.2	47.5	77.9	56.4	50.4	95.3	42.8	20.3	58.6	75.4	66.1	75.3	58.8	90.2	81.3	64.2
PointASNL [134]	66.6	70.3	78.1	75.1	65.5	83.0	47.1	76.9	47.4	53.7	95.1	47.5	27.9	63.5	69.8	67.5	75.1	55.3	81.6	80.6	70.3
KPConv [108]	68.4	84.7	75.8	78.4	64.7	81.4	47.3	77.2	60.5	59.4	93.5	45.0	18.1	58.7	80.5	69.0	78.5	61.4	88.2	81.9	63.2
CBL [105]	70.5	76.9	77.5	80.9	68.7	82.0	43.9	81.2	66.1	59.1	94.5	51.5	17.1	63.3	85.6	72.0	79.6	66.8	88.9	84.7	68.9
PointNext [86]	71.2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
PointMeta [68]	71.4	83.5	78.5	82.1	68.4	84.6	53.1	86.5	61.4	59.6	95.3	50.0	24.6	67.4	88.8	69.2	76.4	62.4	84.9	84.4	67.5
SparseCNN [39]	72.5	64.7	82.1	84.6	72.1	86.9	53.3	75.4	60.3	61.4	95.5	57.2	32.5	71.0	87.0	72.4	82.3	62.8	93.4	86.5	68.3
MinkUNet [22]	73.6	85.9	81.8	83.2	70.9	84.0	52.1	85.3	66.0	64.3	95.1	54.4	28.6	73.1	89.3	67.5	77.2	68.3	87.4	85.2	72.7
StraFormer [58]	74.7	90.1	80.3	84.5	75.7	84.6	51.2	82.5	69.6	64.5	95.6	57.6	26.2	74.4	86.1	74.2	77.0	70.5	89.9	86.0	73.4
BPNet [49]	74.9	90.9	81.8	81.1	75.2	83.9	48.5	84.2	67.3	64.4	95.7	52.8	30.5	77.3	85.9	78.8	81.8	69.3	91.6	85.6	72.3
PTv2 [124]	75.2	74.2	80.9	87.2	75.8	86.0	55.2	89.1	61.0	68.7	96.0	55.9	30.4	76.6	92.6	76.7	79.7	64.4	94.2	87.6	72.2
PointHR (ours)	76.6	79.0	82.3	88.1	74.9	87.1	58.7	91.8	65.5	68.5	97.3	56.0	36.3	58.2	93.3	81.6	82.7	69.8	97.4	89.7	73.9

fusing adjacent features brings high performance, which can be further enhanced by a following sequence operator for refinement. Hence, we opt for the final decoder design as the default.

5.6.4 Detailed Results on ScanNetV2

We additionally provide the results of per-category IoU (%) on the testing split of ScanNetV2 to complement the previous Table 5.2. All the results are obtained from the official testing leaderboard[1] and demonstrated in Table 5.8.

[1] https://kaldir.vc.in.tum.de/scannet_benchmark/semantic_label_3d

5.6.5 Pre-Computation

As discussed in Section 5.3.3, we propose to *pre-compute* the indices for knn collection and resampling operation, which are saved to the cache to avoid on-the-fly computations. Here, we provide an example to illustrate how to *pre-compute* the resampling indices.

Taking the FPS with KNN resampling strategy as an example, we need to compute the index mapping high-resolution to low-resolution for downsampling and the index of K neighbors that interpolate low-resolution to high-resolution for upsampling. Recall that these down- and upsampling operations are abundant in PointHR. However, we found that the resampling operations in the later stages includes those in the previous stages. For example, the index for downsampling from the 2_{nd} to the 3_{rd} branch in stage 3 is the same as the 2_{nd} to 3_{rd} branch in stage 4 because they share the same resolutions. As such, we can compute all the mapping index, including both downsampling and upsampling index, in stage 4, which covers resampling relationships of all the stages. Specifically, in each iteration, we first downsample the input point clouds into four different resolutions and calculate all the downsampling and upsampling indices between these four resolutions. Thereafter, we feed the point cloud features, along with the index, to the network. This enables resampling operations to fetch the corresponding index to obtain down- and up-scale features, thereby speeding up the whole process. We compared the training latency of one iteration with 3 batch size in one V100 GPU for PointHR model with and without the *precomputed* neighbor index. The results showed a latency of 1.86s and 2.18s, respectively. Note that 100 epochs consist of approximately 360,000 iterations, indicating that the *precomputed* neighbor index can save about 32 GPU hours for one training process. A similar situation arises for employing grid pooling and unpooling.

TABLE 5.9. Results of throughput

Method	#Params (M)	FLOPS (G)	Val (%)	Test (%)	Thp. (ins./sec.)
PointMeta	19.7	11.0	72.8	71.4	71
PTv1 [151]	7.8	5.7	70.6	-	38
PTv2 [124]	11.3	14.3	75.4	75.2	39
StraFormer [58]	8.03	12.4	74.3	74.7	22
PointHR-T	0.6	2.5	72.7	-	32
PointHR-S	1.0	2.8	73.0	-	25
PointHR-B	3.9	6.8	74.9	-	21
PointHR-L	7.1	10.3	75.4	**76.6**	17

5.6.6 Speed Analysis

We have also compared the inference speed of PointHR with recent SOTA methods in terms of Throughput (instance / second) in Table 5.9. The results are shown in the following table, where we follow the same strategy as in PointNext [86] using a V100 32GB GPU for all methods. As we can see, our PointHR have a comparable Throughput with Stratified Transformer [58], which is slower than PTv1/PTv2. As discussed in the original HRNet paper, different branches in the same stage are currently evaluated in a sequential way (actually a for-loop), since current deep learning packages usually lack the support of multi-branch parallel computation. This can be a possible solution to further improve the computational efficiency of high-resolution architectures.

5.6.7 Memory Analysis

TABLE 5.10. Comparisons on memory usage (G).

Method	Memory(G)	Method	Memory(G)
PointNext [86]	43.14	PointHR-T	9.40
PointMeta [68]	20.38	PointHR-S	10.84
PTv1 [151]	15.13	PointHR-B	17.53
PTv2 [124]	21.50	PointHR-L	23.58

Since our PointHR maintains high-resolution branch throughout all the stage, the memory usage would be a concern. However, two designs of PointHR significantly reduce the heavy memory cost. The first one is that the highest resolution branch across all stage is N/S instead of N as illustrated in Figure 5.1, where $S = 6$ in our experiments. The second strategy is that we employ a small beginning feature dimension, *e.g.*, the channel of the highest resolution is only 32 for PointHR-L as shown in Table 5.3. Here we provide a quantitative comparison on memory usage between PointHR and recent state-of-the-art methods [86, 68, 151, 124] using 300K points as an example for input on a single GPU (because both PTv2 [124] and PointHR use a batch of three point clouds with 100K points per GPU). As shown in the Table 5.10, all the different configurations of PointHR achieve the reasonable memory usage, *e.g.*, the largest variant PointHR-L can be trained on the 24GB GPU.

5.6.8 Configurations

The typical configurations of PointHR are defined by the number of modules M_i, blocks B_i, neighbors K_i, and channels C_i as shown in Table 5.11. These correspond to the pipeline illustrated in Figure 5.1 and Equation 5.1.

TABLE 5.11. The typical PointHR configuration. **SO**: sequence operator; S: scale factor; M_i: the number of modules; B_i: the number of blocks; K_i: the number of neighbors; C_i: the number of channels.

Res.	Stage 1	Stage 2	Stage 3	Stage 4
$S\times$	$[\ \mathbf{SO},K_1,C_1\ ]\times B_1\times M_1$	$[\ \mathbf{SO},K_2,C_2\ ]\times B_2\times M_2$	$[\ \mathbf{SO},K_3,C_3\ ]\times B_3\times M_3$	$[\ \mathbf{SO},K_4,C_4\ ]\times B_4\times M_4$
$S^2\times$		$[\ \mathbf{SO},K_2,2C_2\ ]\times B_2\times M_2$	$[\ \mathbf{SO},K_3,2C_3\ ]\times B_3\times M_3$	$[\ \mathbf{SO},K_4,2C_4\ ]\times B_4\times M_4$
$S^3\times$			$[\ \mathbf{SO},K_3,4C_3\ ]\times B_3\times M_3$	$[\ \mathbf{SO},K_4,4C_4\ ]\times B_4\times M_4$
$S^4\times$				$[\ \mathbf{SO},K_4,8C_4\ ]\times B_4\times M_4$

5.6.9 Visualizations

More visualizations on S3DIS and ScanNetV2 are illustrated in Figure 5.5 and 5.6. Thanks to its multi-scale characteristic, the proposed PointHR is capable of predicting accurate semantic categories for point clouds even on challenging scenes. For example, PointHR segments the difficult boundaries such as legs of chairs and tables precisely, and also makes smooth predictions on the inner area of large objects like board.

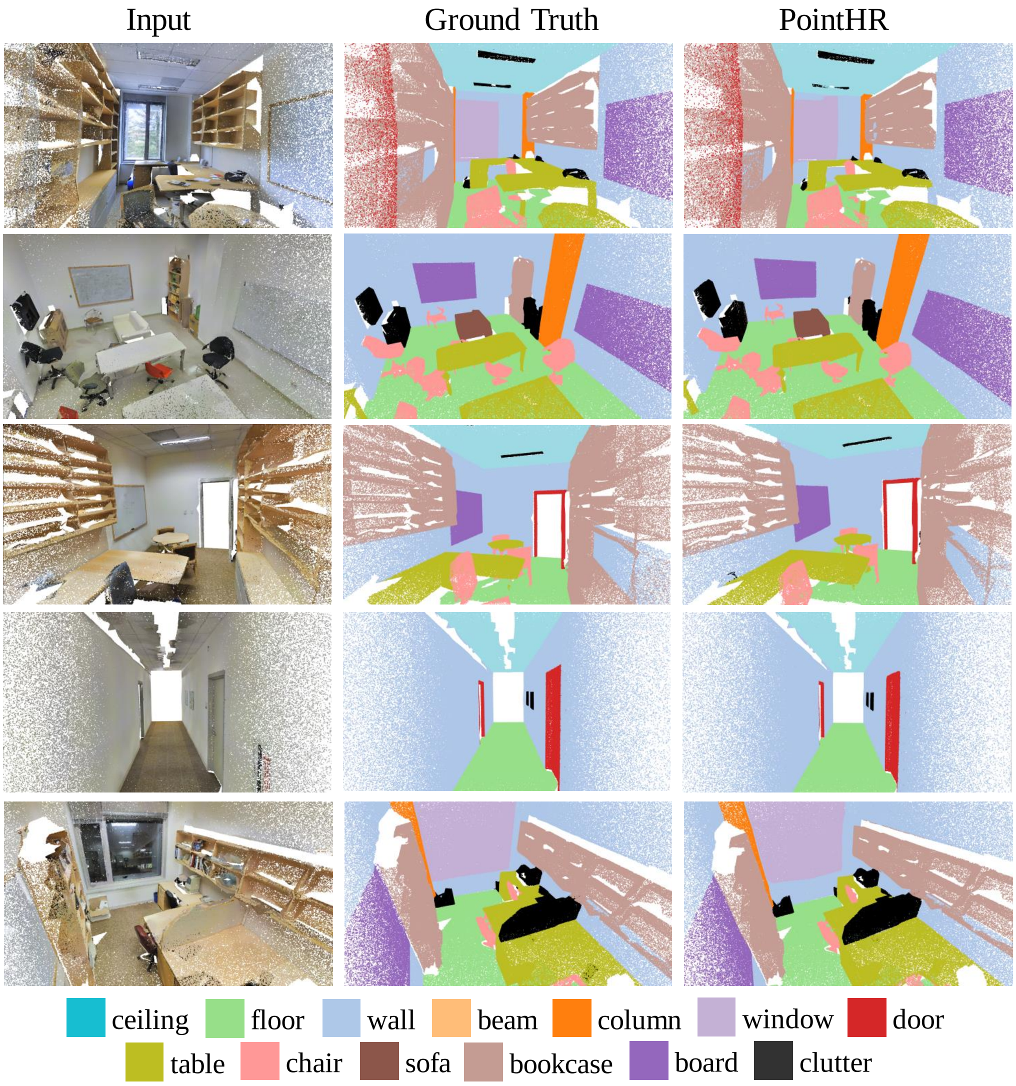


FIGURE 5.5. Visualizations of point cloud segmentation by PointHR on S3DIS. From left to right: input, ground truth, and predictions made by PointHR.

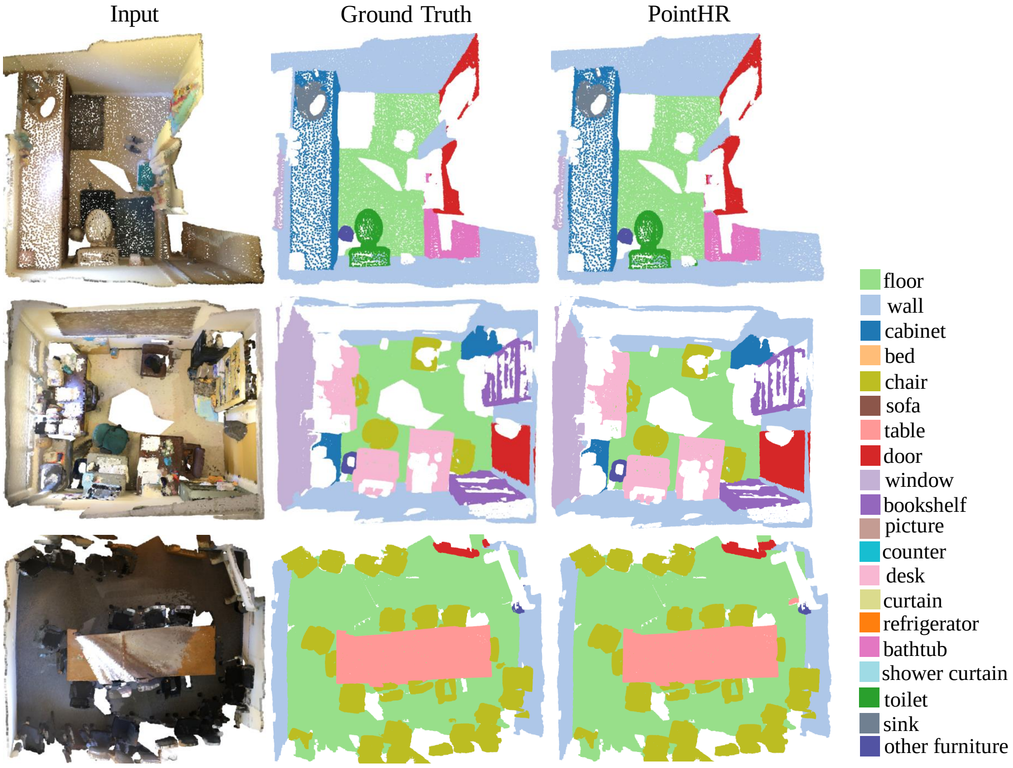


FIGURE 5.6. Visualizations of point cloud segmentation by PointHR on ScanNetV2

CHAPTER 6

Conclusions

This chapter presents a comprehensive summary of the contributions discussed in the preceding chapters, while also proposing promising directions for future research.

6.1 Contributions

This thesis centers on the research of effective representation learning for comprehensive point cloud scene understanding. It addresses three key challenges: interpreting the complex space structure, capturing sufficient contextual information, and perceiving multi-scale objects. Previous methods have either focused on learning plain representations that only emphasize local details or a certain scale of the scene [83, 151], or have not adequately explored fused representations for learning [84, 58]. Consequently, these limitations have resulted in suboptimal solutions for scene understanding. To tackle these issues, the thesis introduces three complementary approaches that involve learning composite representations using different schemes, namely multi-view fusion, short-long range integration, and multi-scale communication.

A novel Geometric Flow Network (GFNet) has been developed for multi-view composite representation learning, enabling a comprehensive interpretation of the complex space structure of 3D point clouds. To facilitate the flow of geometric correspondence information across different views, the GFNet incorporates a geometric flow module. Additionally, grid sampling and KPConv have been integrated into the network to eliminate time-consuming and non-differentiable post-processing. This enables end-to-end training and testing of the GFNet, enhancing both efficiency and effectiveness.

To capture sufficient contextual information, a collect-and-distribute mechanism has been proposed, integrating short-range and long-range clues. This mechanism builds the CD Block module, which enables the simultaneous capture of both short-range and long-range contexts. Consequently, it allows for the learning of comprehensive local-global contextual composite representations of point clouds. Moreover, the thesis enhances position clues dynamically through context-aware position encoding, improving the communication among point features.

In order to fully perceive multi-scale objects, a new high-resolution framework named PointHR has been introduced. PointHR maintains multiple resolutions simultaneously and facilitates frequent communication between all resolutions within each stage. This enables the learning of multiple scales of information. The proposed PointHR framework is formulated in a unified manner, utilizing a sequence operator and a resampling operator. This formulation allows for the utilization of off-the-shelf point cloud blocks and modules without additional effort. Furthermore, the thesis proposes the pre-computation of indices for knn collection and resampling operations to efficiently employ high-resolution architectures, avoiding on-the-fly computations.

In summary, the research works described address the challenges of interpreting complex space structures, capturing sufficient contextual information, and perceiving multi-scale objects in point cloud scene understanding. These contributions collectively advance the field of learning effective representations for comprehensive point cloud scene understanding.

6.2 Future Research

The current research on composite representation learning for 3D scene understanding primarily focuses on point clouds as a single modality. However, there is potential to further enrich the fused representations by incorporating other modalities such as images and texts.

In terms of the image modality, 2D and 3D data offer complementary information. 2D images provide detailed texture and color information, while 3D point clouds provide valuable shape and geometry knowledge. To effectively learn composite representations involving both images and point clouds, it is necessary to establish geometric correspondences and alignments to learn effective representations.

With regard to the text modality, text descriptions can provide additional semantic clues to facilitate comprehensive scene understanding of 3D point cloud scenes. Inspired by the achievements of large language models (LLMs) [9, 109, 110] and large multimodal models (LMMs) [61, 93, 154], pretrained text representations exhibit excellent robustness and generalization abilities. Therefore, incorporating text modality can enhance the capability of vanilla point cloud representations to handle unseen scenarios.

Bibliography

[1] Martin Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu and Xiaoqiang Zheng. ‘TensorFlow: A system for large-scale machine learning’. In: *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*. 2016, pp. 265–283. URL: https://www.usenix.org/system/files/conference/osdi16/osdi16-abadi.pdf.

[2] Eren Erdal Aksoy, Saimir Baci and Selcuk Cavdar. ‘Salsanet: Fast road and vehicle segmentation in lidar point clouds for autonomous driving’. In: *IEEE Intelligent Vehicles Symposium*. 2020, pp. 926–932.

[3] Evangelos Alexiou, Nanyang Yang and Touradj Ebrahimi. ‘PointXR: A toolbox for visualization and subjective evaluation of point clouds in virtual reality’. In: *International Conference on Quality of Multimedia Experience (QoMEX)*. IEEE. 2020, pp. 1–6.

[4] Yara Ali Alnaggar, Mohamed Afifi, Karim Amer and Mohamed ElHelw. ‘Multi Projection Fusion for Real-time Semantic Segmentation of 3D LiDAR Point Clouds’. In: *IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. 2021, pp. 1800–1809.

[5] Inigo Alonso, Luis Riazuelo, Luis Montesano and Ana C Murillo. ‘3d-mininet: Learning a 2d representation from point clouds for fast and efficient 3d lidar semantic segmentation’. In: *IEEE Robotics and Automation Letters* 5.4 (2020), pp. 5432–5439.

[6] Iro Armeni, Ozan Sener, Amir R Zamir, Helen Jiang, Ioannis Brilakis, Martin Fischer and Silvio Savarese. ‘3d semantic parsing of large-scale indoor spaces’.

In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 1534–1543.

[7] Jens Behley, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Cyrill Stachniss and Jurgen Gall. 'Semantickitti: A dataset for semantic scene understanding of lidar sequences'. In: *IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019, pp. 9297–9307.

[8] Maxim Berman, Amal Rannen Triki and Matthew B Blaschko. 'The lovász-softmax loss: A tractable surrogate for the optimization of the intersection-over-union measure in neural networks'. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 4413–4421.

[9] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell et al. 'Language models are few-shot learners'. In: *Advances in Neural Information Processing Systems (NeurIPS)* 33 (2020), pp. 1877–1901.

[10] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan and Oscar Beijbom. 'nuscenes: A multimodal dataset for autonomous driving'. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 11621–11631.

[11] Yue Cao, Jiarui Xu, Stephen Lin, Fangyun Wei and Han Hu. 'Gcnet: Non-local networks meet squeeze-excitation networks and beyond'. In: *IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*. 2019, pp. 0–0.

[12] Ke Chen, Ryan Oldja, Nikolai Smolyanskiy, Stan Birchfield, Alexander Popov, David Wehr, Ibrahim Eden and Joachim Pehserl. 'Mvlidarnet: Real-time multi-class scene understanding for autonomous driving using multiple views'. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020, pp. 2288–2294.

[13] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy and Alan L Yuille. 'Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 40.4 (2017), pp. 834–848.

[14] Liang-Chieh Chen, George Papandreou, Florian Schroff and Hartwig Adam. 'Rethinking atrous convolution for semantic image segmentation'. In: *arXiv preprint arXiv:1706.05587* (2017).

[15] Liang-Chieh Chen, Yi Yang, Jiang Wang, Wei Xu and Alan L Yuille. 'Attention to scale: Scale-aware semantic image segmentation'. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 3640–3649.

[16] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff and Hartwig Adam. 'Encoder-decoder with atrous separable convolution for semantic image segmentation'. In: *European Conference on Computer Vision (ECCV)*. 2018, pp. 801–818.

[17] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li and Tian Xia. 'Multi-view 3d object detection network for autonomous driving'. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 1907–1915.

[18] Yunqiang Chen, Qing Wang, Hong Chen, Xiaoyu Song, Hui Tang and Mengxiao Tian. 'An overview of augmented reality technology'. In: *Journal of Physics: Conference Series*. Vol. 1237. IOP Publishing. 2019, p. 022082.

[19] Ran Cheng, Ryan Razani, Ehsan Taghavi, Enxu Li and Bingbing Liu. '2-S3Net: Attentive feature fusion with adaptive feature selection for sparse semantic segmentation network'. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, pp. 12547–12556.

[20] Silin Cheng, Xiwu Chen, Xinwei He, Zhe Liu and Xiang Bai. 'Pra-net: Point relation-aware network for 3d point cloud analysis'. In: *IEEE Transactions on Image Processing (TIP)* 30 (2021), pp. 4436–4448.

[21] Jaesung Choe, Chunghyun Park, Francois Rameau, Jaesik Park and In So Kweon. 'Pointmixer: Mlp-mixer for point cloud understanding'. In: *European Conference on Computer Vision (ECCV)*. Springer. 2022, pp. 620–640.

[22] Christopher Choy, JunYoung Gwak and Silvio Savarese. '4d spatio-temporal convnets: Minkowski convolutional neural networks'. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 3075–3084.

[23] Christopher Choy, Junha Lee, Rene Ranftl, Jaesik Park and Vladlen Koltun. 'High-dimensional Convolutional Networks for Geometric Pattern Recognition'.

In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[24] Xiangxiang Chu, Zhi Tian, Yuqing Wang, Bo Zhang, Haibing Ren, Xiaolin Wei, Huaxia Xia and Chunhua Shen. 'Twins: Revisiting the design of spatial attention in vision transformers'. In: *Advances in Neural Information Processing Systems (NeurIPS)* 34 (2021).

[25] Spconv Contributors. *Spconv: Spatially Sparse Convolution Library*. https://github.com/traveller59/spconv. 2022.

[26] Tiago Cortinhal, George Tzelepis and Eren Erdal Aksoy. 'SalsaNext: Fast, uncertainty-aware semantic segmentation of LiDAR point clouds'. In: *International Symposium on Visual Computing (ISVC)*. 2020, pp. 207–222.

[27] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser and Matthias Nießner. 'Scannet: Richly-annotated 3d reconstructions of indoor scenes'. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 5828–5839.

[28] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li and Li Fei-Fei. 'Imagenet: A large-scale hierarchical image database'. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2009, pp. 248–255.

[29] Jacob Devlin, Ming-Wei Chang, Kenton Lee and Kristina Toutanova. 'Bert: Pre-training of deep bidirectional transformers for language understanding'. In: *arXiv preprint arXiv:1810.04805* (2018).

[30] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly et al. 'An image is worth 16x16 words: Transformers for image recognition at scale'. In: *arXiv preprint arXiv:2010.11929* (2020).

[31] Yuval Eldar, Michael Lindenbaum, Moshe Porat and Yehoshua Y Zeevi. 'The farthest point strategy for progressive image sampling'. In: *IEEE Transactions on Image Processing (TIP)* 6.9 (1997), pp. 1305–1315.

[32] Haoqi Fan, Bo Xiong, Karttikeya Mangalam, Yanghao Li, Zhicheng Yan, Jitendra Malik and Christoph Feichtenhofer. 'Multiscale vision transformers'. In: *IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021, pp. 6824–6835.

[33] Siqi Fan, Qiulei Dong, Fenghua Zhu, Yisheng Lv, Peijun Ye and Fei-Yue Wang. 'SCF-Net: Learning spatial contextual features for large-scale point cloud segmentation'. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, pp. 14504–14513.

[34] Jun Fu, Jing Liu, Haijie Tian, Yong Li, Yongjun Bao, Zhiwei Fang and Hanqing Lu. 'Dual attention network for scene segmentation'. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 3146–3154.

[35] Andreas Geiger, Philip Lenz and Raquel Urtasun. 'Are we ready for autonomous driving? the kitti vision benchmark suite'. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2012, pp. 3354–3361.

[36] Martin Gerdzhev, Ryan Razani, Ehsan Taghavi and Liu Bingbing. 'Tornadonet: multiview total variation semantic segmentation with diamond inception module'. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2021, pp. 9543–9549.

[37] Ross Girshick. 'Fast r-cnn'. In: *IEEE/CVF International Conference on Computer Vision (ICCV)*. 2015, pp. 1440–1448.

[38] Ankit Goyal, Hei Law, Bowei Liu, Alejandro Newell and Jia Deng. 'Revisiting point cloud shape classification with a simple and effective baseline'. In: *International Conference on Machine Learning (ICML)*. PMLR. 2021, pp. 3809–3820.

[39] Benjamin Graham, Martin Engelcke and Laurens Van Der Maaten. '3d semantic segmentation with submanifold sparse convolutional networks'. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 9224–9232.

[40] Jiaqi Gu, Hyoukjun Kwon, Dilin Wang, Wei Ye, Meng Li, Yu-Hsin Chen, Liangzhen Lai, Vikas Chandra and David Z Pan. 'Multi-scale high-resolution vision transformer for semantic segmentation'. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, pp. 12094–12103.

[41] Meng-Hao Guo, Jun-Xiong Cai, Zheng-Ning Liu, Tai-Jiang Mu, Ralph R Martin and Shi-Min Hu. 'Pct: Point cloud transformer'. In: *Computational Visual Media* 7.2 (2021), pp. 187–199.

[42] Abdullah Hamdi, Silvio Giancola and Bernard Ghanem. ‘Mvtn: Multi-view transformation network for 3d shape recognition’. In: *IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021, pp. 1–11.

[43] Lei Han, Tian Zheng, Lan Xu and Lu Fang. ‘Occuseg: Occupancy-aware 3d instance segmentation’. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 2940–2949.

[44] Junjun He, Zhongying Deng and Yu Qiao. ‘Dynamic multi-scale filters for semantic segmentation’. In: *IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019, pp. 3562–3572.

[45] Junjun He, Zhongying Deng, Lei Zhou, Yali Wang and Yu Qiao. ‘Adaptive pyramid context network for semantic segmentation’. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 7519–7528.

[46] Kaiming He, Xiangyu Zhang, Shaoqing Ren and Jian Sun. ‘Deep residual learning for image recognition’. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778.

[47] Pedro Hermosilla, Tobias Ritschel, Pere-Pau Vázquez, Àlvar Vinacua and Timo Ropinski. ‘Monte carlo convolution for learning on non-uniformly sampled point clouds’. In: *ACM Transactions On Graphics (TOG)* 37.6 (2018), pp. 1–12.

[48] Qingyong Hu, Bo Yang, Linhai Xie, Stefano Rosa, Yulan Guo, Zhihua Wang, Niki Trigoni and Andrew Markham. ‘Randla-net: Efficient semantic segmentation of large-scale point clouds’. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 11108–11117.

[49] Wenbo Hu, Hengshuang Zhao, Li Jiang, Jiaya Jia and Tien-Tsin Wong. ‘Bi-directional projection network for cross dimension scene understanding’. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, pp. 14373–14382.

[50] Qiangui Huang, Weiyue Wang and Ulrich Neumann. ‘Recurrent slice networks for 3d segmentation of point clouds’. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 2626–2635.

[51] Zilong Huang, Xinggang Wang, Lichao Huang, Chang Huang, Yunchao Wei and Wenyu Liu. ‘Ccnet: Criss-cross attention for semantic segmentation’. In:

IEEE/CVF International Conference on Computer Vision (ICCV). 2019, pp. 603–612.

[52] Matterport Inc. *Matterport 3d models of interior spaces.* http://matterport.com/. Accessed: 2015-06-01. 2015.

[53] Maximilian Jaritz, Jiayuan Gu and Hao Su. 'Multi-view pointnet for 3d scene understanding'. In: *IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*. 2019, pp. 0–0.

[54] Li Jiang, Hengshuang Zhao, Shu Liu, Xiaoyong Shen, Chi-Wing Fu and Jiaya Jia. 'Hierarchical point-edge interaction network for point cloud semantic segmentation'. In: *IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019, pp. 10433–10441.

[55] Deyvid Kochanov, Fatemeh Karimi Nejadasl and Olaf Booij. 'Kprnet: Improving projection-based lidar semantic segmentation'. In: *arXiv preprint arXiv:2007.12668* (2020).

[56] Artem Komarichev, Zichun Zhong and Jing Hua. 'A-cnn: Annularly convolutional neural networks on point clouds'. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 7421–7430.

[57] Alex Krizhevsky, Ilya Sutskever and Geoffrey E Hinton. 'Imagenet classification with deep convolutional neural networks'. In: *Advances in Neural Information Processing Systems (NeurIPS)* 25 (2012).

[58] Xin Lai, Jianhui Liu, Li Jiang, Liwei Wang, Hengshuang Zhao, Shu Liu, Xiaojuan Qi and Jiaya Jia. 'Stratified Transformer for 3D Point Cloud Segmentation'. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2022).

[59] Loic Landrieu and Martin Simonovsky. 'Large-scale point cloud semantic segmentation with superpoint graphs'. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 4558–4567.

[60] Jiaxin Li, Ben M Chen and Gim Hee Lee. 'So-net: Self-organizing network for point cloud analysis'. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 9397–9406.

[61] Junnan Li, Dongxu Li, Silvio Savarese and Steven Hoi. 'Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models'. In: *arXiv preprint arXiv:2301.12597* (2023).

[62] Wei Li, Xing Wang, Xin Xia, Jie Wu, Xuefeng Xiao, Min Zheng and Shiping Wen. 'SepViT: Separable Vision Transformer'. In: *arXiv preprint arXiv:2203.15380* (2022).

[63] Xia Li, Zhisheng Zhong, Jianlong Wu, Yibo Yang, Zhouchen Lin and Hong Liu. 'Expectation-maximization attention networks for semantic segmentation'. In: *IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019, pp. 9167–9176.

[64] Xuyou Li, Shitong Du, Guangchun Li and Haoyu Li. 'Integrate point-cloud segmentation with 3D LiDAR scan-matching for mobile robot localization and mapping'. In: *Sensors* 20.1 (2019), p. 237.

[65] Yanghao Li, Chao-Yuan Wu, Haoqi Fan, Karttikeya Mangalam, Bo Xiong, Jitendra Malik and Christoph Feichtenhofer. 'MViTv2: Improved multiscale vision transformers for classification and detection'. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022.

[66] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di and Baoquan Chen. 'Pointcnn: Convolution on x-transformed points'. In: *Advances in Neural Information Processing Systems (NeurIPS)* 31 (2018).

[67] Ying Li, Lingfei Ma, Zilong Zhong, Fei Liu, Michael A Chapman, Dongpu Cao and Jonathan Li. 'Deep learning for lidar point clouds in autonomous driving: A review'. In: *IEEE Transactions on Neural Networks and Learning Systems (TNNLS)* 32.8 (2020), pp. 3412–3432.

[68] Haojia Lin, Xiawu Zheng, Lijiang Li, Fei Chao, Shanshan Wang, Yan Wang, Yonghong Tian and Rongrong Ji. 'Meta Architecure for Point Cloud Analysis'. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2023).

[69] Venice Erin Liong, Thi Ngoc Tho Nguyen, Sergi Widjaja, Dhananjai Sharma and Zhuang Jie Chong. 'Amvnet: Assertion-based multi-view fusion network for lidar semantic segmentation'. In: *arXiv preprint arXiv:2012.04934* (2020).

[70] Ze Liu, Han Hu, Yue Cao, Zheng Zhang and Xin Tong. 'A closer look at local aggregation operators in point cloud analysis'. In: *European Conference on Computer Vision (ECCV)*. Springer. 2020, pp. 326–342.

[71] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin and Baining Guo. 'Swin transformer: Hierarchical vision transformer using shifted windows'. In: *IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021, pp. 10012–10022.

[72] Zhengzhe Liu, Xiaojuan Qi and Chi-Wing Fu. 'One thing one click: A self-training approach for weakly supervised 3d semantic segmentation'. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, pp. 1726–1736.

[73] Zhijian Liu, Haotian Tang, Yujun Lin and Song Han. 'Point-voxel cnn for efficient 3d deep learning'. In: *Advances in Neural Information Processing Systems (NeurIPS)* 32 (2019).

[74] Jonathan Long, Evan Shelhamer and Trevor Darrell. 'Fully convolutional networks for semantic segmentation'. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 3431–3440.

[75] Ilya Loshchilov and Frank Hutter. 'Decoupled weight decay regularization'. In: *arXiv preprint arXiv:1711.05101* (2017).

[76] Ilya Loshchilov and Frank Hutter. 'Sgdr: Stochastic gradient descent with warm restarts'. In: *arXiv preprint arXiv:1608.03983* (2016).

[77] Tao Lu, Limin Wang and Gangshan Wu. 'Cga-net: Category guided aggregation for point cloud semantic segmentation'. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 11693–11702.

[78] Xu Ma, Can Qin, Haoxuan You, Haoxi Ran and Yun Fu. 'Rethinking network design and local geometry in point cloud: A simple residual mlp framework'. In: *International Conference on Learning Representations (ICLR)* (2022).

[79] Andres Milioto, Ignacio Vizzo, Jens Behley and Cyrill Stachniss. 'Rangenet++: Fast and accurate lidar semantic segmentation'. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2019, pp. 4213–4220.

[80] Yatian Pang, Wenxiao Wang, Francis EH Tay, Wei Liu, Yonghong Tian and Li Yuan. 'Masked autoencoders for point cloud self-supervised learning'. In: *European conference on computer vision (ECCV)*. Springer. 2022, pp. 604–621.

[81] Chunghyun Park, Yoonwoo Jeong, Minsu Cho and Jaesik Park. 'Fast Point Transformer'. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2022).

[82] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga et al. 'Pytorch: An imperative style, high-performance deep learning library'. In: *Advances in Neural Information Processing Systems (NeurIPS)* 32 (2019).

[83] Charles R Qi, Hao Su, Kaichun Mo and Leonidas J Guibas. 'Pointnet: Deep learning on point sets for 3d classification and segmentation'. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 652–660.

[84] Charles Ruizhongtai Qi, Li Yi, Hao Su and Leonidas J Guibas. 'Pointnet++: Deep hierarchical feature learning on point sets in a metric space'. In: *Advances in Neural Information Processing Systems (NeurIPS)* 30 (2017).

[85] Guocheng Qian, Hasan Hammoud, Guohao Li, Ali Thabet and Bernard Ghanem. 'ASSANet: An Anisotropic Separable Set Abstraction for Efficient Point Cloud Representation Learning'. In: *Advances in Neural Information Processing Systems (NeurIPS)* 34 (2021), pp. 28119–28130.

[86] Guocheng Qian, Yuchen Li, Houwen Peng, Jinjie Mai, Hasan Abed Al Kader Hammoud, Mohamed Elhoseiny and Bernard Ghanem. 'PointNeXt: Revisiting PointNet++ with Improved Training and Scaling Strategies'. In: *Advances in Neural Information Processing Systems (NeurIPS)* (2022).

[87] Haibo Qiu, Baosheng Yu, Yixin Chen and Dacheng Tao. 'PointHR: Exploring High-Resolution Architectures for 3D Point Cloud Segmentation'. In: *arXiv preprint arXiv:2310.07743* (2023).

[88] Haibo Qiu, Baosheng Yu and Dacheng Tao. 'Collect-and-Distribute Transformer for 3D Point Cloud Analysis'. In: *arXiv preprint arXiv:2306.01257* (2023).

[89] Haibo Qiu, Baosheng Yu and Dacheng Tao. 'GFNet: Geometric Flow Network for 3D Point Cloud Semantic Segmentation'. In: *Transactions on Machine*

Learning Research (2022). URL: https://openreview.net/forum?id=LSAAlS7Yts.

[90] Shi Qiu, Saeed Anwar and Nick Barnes. 'Dense-resolution network for point cloud classification and segmentation'. In: *IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. 2021, pp. 3813–3822.

[91] Shi Qiu, Saeed Anwar and Nick Barnes. 'Geometric back-projection network for point cloud classification'. In: *IEEE Transactions on Multimedia* 24 (2021), pp. 1943–1955.

[92] Shi Qiu, Saeed Anwar and Nick Barnes. 'Semantic segmentation for real point cloud scenes via bilateral augmentation and adaptive fusion'. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, pp. 1757–1767.

[93] Alec Radford, Jong Wook Kim, Chris Hallacy, A. Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger and Ilya Sutskever. 'Learning Transferable Visual Models From Natural Language Supervision'. In: *International Conference on Machine Learning (ICML)* (2021).

[94] Haoxi Ran, Jun Liu and Chengjie Wang. 'Surface representation for point clouds'. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, pp. 18942–18952.

[95] Joseph Redmon and Ali Farhadi. 'Yolov3: An incremental improvement'. In: *arXiv preprint arXiv:1804.02767* (2018).

[96] Shaoqing Ren, Kaiming He, Ross Girshick and Jian Sun. 'Faster r-cnn: Towards real-time object detection with region proposal networks'. In: *Advances in Neural Information Processing Systems (NeurIPS)* 28 (2015).

[97] Damien Robert, Bruno Vallet and Loic Landrieu. 'Learning Multi-View Aggregation In the Wild for Large-Scale 3D Semantic Segmentation'. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, pp. 5575–5584.

[98] Olaf Ronneberger, Philipp Fischer and Thomas Brox. 'U-net: Convolutional networks for biomedical image segmentation'. In: *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*. Springer. 2015, pp. 234–241.

[99] Karen Simonyan and Andrew Zisserman. ‘Very deep convolutional networks for large-scale image recognition’. In: *arXiv preprint arXiv:1409.1556* (2014).

[100] Leslie N Smith and Nicholay Topin. ‘Super-convergence: Very fast training of neural networks using large learning rates’. In: *Artificial intelligence and machine learning for multi-domain operations applications*. Vol. 11006. SPIE. 2019, pp. 369–386.

[101] Hang Su, Subhransu Maji, Evangelos Kalogerakis and Erik Learned-Miller. ‘Multi-view convolutional neural networks for 3d shape recognition’. In: *IEEE/CVF International Conference on Computer Vision (ICCV)*. 2015, pp. 945–953.

[102] Ke Sun, Bin Xiao, Dong Liu and Jingdong Wang. ‘Deep high-resolution representation learning for human pose estimation’. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 5693–5703.

[103] Haotian Tang, Zhijian Liu, Shengyu Zhao, Yujun Lin, Ji Lin, Hanrui Wang and Song Han. ‘Searching efficient 3d architectures with sparse point-voxel convolution’. In: *European Conference on Computer Vision (ECCV)*. 2020, pp. 685–702.

[104] Liyao Tang, Zhe Chen, Shanshan Zhao, Chaoyue Wang and Dacheng Tao. ‘All Points Matter: Entropy-Regularized Distribution Alignment for Weakly-supervised 3D Segmentation’. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2023. URL: https://openreview.net/forum?id=utQms7PPx5.

[105] Liyao Tang, Yibing Zhan, Zhe Chen, Baosheng Yu and Dacheng Tao. ‘Contrastive Boundary Learning for Point Cloud Segmentation’. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, pp. 8489–8499.

[106] Maxim Tatarchenko, Jaesik Park, Vladlen Koltun and Qian-Yi Zhou. ‘Tangent convolutions for dense prediction in 3d’. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 3887–3896.

[107] Lyne Tchapmi, Christopher Choy, Iro Armeni, JunYoung Gwak and Silvio Savarese. ‘Segcloud: Semantic segmentation of 3d point clouds’. In: *International Conference on 3D Vision (3DV)*. IEEE. 2017, pp. 537–547.

[108] Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette and Leonidas J Guibas. ‘Kpconv: Flexible and deformable convolution for point clouds’. In: *IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019, pp. 6411–6420.

[109] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar et al. ‘Llama: Open and efficient foundation language models’. In: *arXiv preprint arXiv:2302.13971* (2023).

[110] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov and Thomas Scialom. ‘Llama 2: Open Foundation and Fine-Tuned Chat Models’. In: *arXiv preprint arXiv: 2307.09288* (2023).

[111] Larissa T Triess, David Peter, Christoph B Rist and J Marius Zöllner. ‘Scan-based semantic segmentation of lidar point clouds: An experimental study’. In: *IEEE Intelligent Vehicles Symposium*. 2020, pp. 1116–1121.

[112] Mikaela Angelina Uy, Quang-Hieu Pham, Binh-Son Hua, Thanh Nguyen and Sai-Kit Yeung. ‘Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data’. In: *IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019, pp. 1588–1597.

[113] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser and Illia Polosukhin. 'Attention is all you need'. In: *Advances in Neural Information Processing Systems (NeurIPS)* 30 (2017).

[114] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang et al. 'Deep high-resolution representation learning for visual recognition'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 43.10 (2020), pp. 3349–3364.

[115] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo and Ling Shao. 'Pvt v2: Improved baselines with pyramid vision transformer'. In: *Computational Visual Media* 8.3 (2022), pp. 415–424.

[116] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo and Ling Shao. 'Pyramid vision transformer: A versatile backbone for dense prediction without convolutions'. In: *IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021, pp. 568–578.

[117] Xiaolong Wang, Ross Girshick, Abhinav Gupta and Kaiming He. 'Non-local neural networks'. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 7794–7803.

[118] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein and Justin M Solomon. 'Dynamic graph cnn for learning on point clouds'. In: *ACM Transactions On Graphics (TOG)* 38.5 (2019), pp. 1–12.

[119] Ruben Wiersma, Ahmad Nasikun, Elmar Eisemann and Klaus Hildebrandt. 'Deltaconv: anisotropic operators for geometric deep learning on point clouds'. In: *ACM Transactions on Graphics (TOG)* 41.4 (2022), pp. 1–10.

[120] Bichen Wu, Alvin Wan, Xiangyu Yue and Kurt Keutzer. 'Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar point cloud'. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2018, pp. 1887–1893.

[121] Bichen Wu, Xuanyu Zhou, Sicheng Zhao, Xiangyu Yue and Kurt Keutzer. 'Squeezesegv2: Improved model structure and unsupervised domain adaptation for road-object segmentation from a lidar point cloud'. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2019, pp. 4376–4382.

[122] Kan Wu, Houwen Peng, Minghao Chen, Jianlong Fu and Hongyang Chao. ‘Rethinking and improving relative position encoding for vision transformer’. In: *IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021, pp. 10033–10041.

[123] Wenxuan Wu, Zhongang Qi and Li Fuxin. ‘Pointconv: Deep convolutional networks on 3d point clouds’. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 9621–9630.

[124] Xiaoyang Wu, Yixing Lao, Li Jiang, Xihui Liu and Hengshuang Zhao. ‘Point transformer V2: Grouped Vector Attention and Partition-based Pooling’. In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2022.

[125] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang and Jianxiong Xiao. ‘3d shapenets: A deep representation for volumetric shapes’. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 1912–1920.

[126] Tiange Xiang, Chaoyi Zhang, Yang Song, Jianhui Yu and Weidong Cai. ‘Walk in the cloud: Learning curves for point clouds shape analysis’. In: *IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021, pp. 915–924.

[127] Chenfeng Xu, Bichen Wu, Zining Wang, Wei Zhan, Peter Vajda, Kurt Keutzer and Masayoshi Tomizuka. ‘Squeezesegv3: Spatially-adaptive convolution for efficient point-cloud segmentation’. In: *European Conference on Computer Vision (ECCV)*. 2020, pp. 1–19.

[128] Chenfeng Xu, Bohan Zhai, Bichen Wu, Tian Li, Wei Zhan, Peter Vajda, Kurt Keutzer and Masayoshi Tomizuka. ‘You only group once: Efficient point-cloud processing with token representation and relation inference module’. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2021, pp. 4589–4596.

[129] Jianyun Xu, Ruixiang Zhang, Jian Dou, Yushi Zhu, Jie Sun and Shiliang Pu. ‘Rpvnet: A deep and efficient range-point-voxel fusion network for lidar point cloud segmentation’. In: *IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021, pp. 16024–16033.

[130] Mutian Xu, Runyu Ding, Hengshuang Zhao and Xiaojuan Qi. ‘Paconv: Position adaptive convolution with dynamic kernel assembling on point clouds’. In:

IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2021, pp. 3173–3182.

[131] Yifan Xu, Tianqi Fan, Mingye Xu, Long Zeng and Yu Qiao. ‘Spidercnn: Deep learning on point sets with parameterized convolutional filters’. In: *European Conference on Computer Vision (ECCV)*. 2018, pp. 87–102.

[132] Xu Yan, Jiantao Gao, Jie Li, Ruimao Zhang, Zhen Li, Rui Huang and Shuguang Cui. ‘Sparse Single Sweep LiDAR Point Cloud Segmentation via Learning Contextual Shape Priors from Scene Completion’. In: *arXiv preprint arXiv:2012.03762* (2020).

[133] Xu Yan, Jiantao Gao, Chaoda Zheng, Chao Zheng, Ruimao Zhang, Shuguang Cui and Zhen Li. ‘2DPASS: 2D Priors Assisted Semantic Segmentation on LiDAR Point Clouds’. In: *European Conference on Computer Vision (ECCV)*. 2022.

[134] Xu Yan, Chaoda Zheng, Zhen Li, Sheng Wang and Shuguang Cui. ‘Pointasnl: Robust point clouds processing using nonlocal neural networks with adaptive sampling’. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 5589–5598.

[135] Jianwei Yang, Chunyuan Li, Pengchuan Zhang, Xiyang Dai, Bin Xiao, Lu Yuan and Jianfeng Gao. ‘Focal self-attention for local-global interactions in vision transformers’. In: *Advances in Neural Information Processing Systems (NeurIPS)* (2021).

[136] Lei Yang, Yanhong Liu, Jinzhu Peng and Zize Liang. ‘A novel system for off-line 3D seam extraction and path planning based on point cloud segmentation for arc welding robot’. In: *Robotics and Computer-Integrated Manufacturing* 64 (2020), p. 101929.

[137] Yu-Qi Yang, Yu-Xiao Guo, Jian-Yu Xiong, Yang Liu, Hao Pan, Peng-Shuai Wang, Xin Tong and Baining Guo. ‘Swin3D: A Pretrained Transformer Backbone for 3D Indoor Scene Understanding’. In: *arXiv preprint arXiv:2304.06906* (2023).

[138] Maosheng Ye, Rui Wan, Shuangjie Xu, Tongyi Cao and Qifeng Chen. ‘DRINet++: Efficient Voxel-as-point Point Cloud Segmentation’. In: *arXiv preprint arXiv:2111.08318* (2021).

[139] Li Yi, Vladimir G. Kim, Duygu Ceylan, I-Chao Shen, Mengyan Yan, Hao Su, Cewu Lu, Qixing Huang, Alla Sheffer and Leonidas Guibas. ‘A Scalable Active Framework for Region Annotation in 3D Shape Collections’. In: *SIGGRAPH Asia* (2016).

[140] Minghao Yin, Zhuliang Yao, Yue Cao, Xiu Li, Zheng Zhang, Stephen Lin and Han Hu. ‘Disentangled non-local neural networks’. In: *European Conference on Computer Vision (ECCV)*. Springer. 2020, pp. 191–207.

[141] Xumin Yu, Lulu Tang, Yongming Rao, Tiejun Huang, Jie Zhou and Jiwen Lu. ‘Point-BERT: Pre-training 3D Point Cloud Transformers with Masked Point Modeling’. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2022).

[142] Xumin Yu, Lulu Tang, Yongming Rao, Tiejun Huang, Jie Zhou and Jiwen Lu. ‘Point-bert: Pre-training 3d point cloud transformers with masked point modeling’. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, pp. 19313–19322.

[143] Yuhui Yuan, Rao Fu, Lang Huang, Weihong Lin, Chao Zhang, Xilin Chen and Jingdong Wang. ‘Hrformer: High-resolution transformer for dense prediction’. In: *Advances in Neural Information Processing Systems (NeurIPS)* (2021).

[144] Feihu Zhang, Jin Fang, Benjamin Wah and Philip Torr. ‘Deep fusionnet for point cloud semantic segmentation’. In: *European Conference on Computer Vision (ECCV)*. 2020, pp. 644–663.

[145] Guoqing Zhang, Yu Ge, Zhicheng Dong, Hao Wang, Yuhui Zheng and Shengyong Chen. ‘Deep high-resolution representation learning for cross-resolution person re-identification’. In: *IEEE Transactions on Image Processing (TIP)* 30 (2021), pp. 8913–8925.

[146] Hang Zhang, Kristin Dana, Jianping Shi, Zhongyue Zhang, Xiaogang Wang, Ambrish Tyagi and Amit Agrawal. ‘Context encoding for semantic segmentation’. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 7151–7160.

[147] Jinlai Zhang, Lyujie Chen, Bo Ouyang, Binbin Liu, Jihong Zhu, Yujin Chen, Yanmei Meng and Danfeng Wu. ‘Pointcutmix: Regularization strategy for point cloud classification’. In: *Neurocomputing* 505 (2022), pp. 58–67.

[148] Yang Zhang, Zixiang Zhou, Philip David, Xiangyu Yue, Zerong Xi, Boqing Gong and Hassan Foroosh. ‘Polarnet: An improved grid representation for online lidar point clouds semantic segmentation’. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 9601–9610.

[149] Zhiyuan Zhang, Binh-Son Hua and Sai-Kit Yeung. ‘ShellNet: Efficient Point Cloud Convolutional Neural Networks Using Concentric Shells Statistics’. In: *IEEE/CVF International Conference on Computer Vision (ICCV)* (2019), pp. 1607–1616.

[150] Hengshuang Zhao, Li Jiang, Chi-Wing Fu and Jiaya Jia. ‘Pointweb: Enhancing local neighborhood features for point cloud processing’. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 5565–5573.

[151] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr and Vladlen Koltun. ‘Point transformer’. In: *IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021, pp. 16259–16268.

[152] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang and Jiaya Jia. ‘Pyramid scene parsing network’. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 2881–2890.

[153] Hengshuang Zhao, Yi Zhang, Shu Liu, Jianping Shi, Chen Change Loy, Dahua Lin and Jiaya Jia. ‘Psanet: Point-wise spatial attention network for scene parsing’. In: *European Conference on Computer Vision (ECCV)*. 2018, pp. 267–283.

[154] Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li and Mohamed Elhoseiny. ‘MiniGPT-4: Enhancing Vision-Language Understanding with Advanced Large Language Models’. In: *arXiv preprint arXiv: 2304.10592* (2023).

[155] Xinge Zhu, Hui Zhou, Tai Wang, Fangzhou Hong, Yuexin Ma, Wei Li, Hongsheng Li and Dahua Lin. ‘Cylindrical and asymmetrical 3d convolution networks for lidar segmentation’. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, pp. 9939–9948.

[156] Zhen Zhu, Mengde Xu, Song Bai, Tengteng Huang and Xiang Bai. ‘Asymmetric non-local neural networks for semantic segmentation’. In: *IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019, pp. 593–602.