Information Processing and Management 62 (2025) 104155

Contents lists available at ScienceDirect

Information Processing and Management

journal homepage: www.elsevier.com/locate/infoproman

VoxT-GNN: A 3D object detection approach from point cloud based on voxel-level transformer and graph neural network

Qiangwen Zheng [a], Sheng Wu [b,*], Jinghui Wei [a]

[a] The College of Computer and Data Science, Fuzhou University, China
[b] The Academy of Digital China (Fujian), Fuzhou University, China

ARTICLE INFO

Keywords:
3D object detection
Point cloud
Graph Neural Network(GNN)
Transformer

ABSTRACT

Recently, a variety of LiDAR-based methods for the 3D detection of single-class objects, large objects, or in straightforward scenes have exhibited competitive performance. However, their detection performance in complex scenarios with multi - sized and multi - class objects is limited. We observe that the core problem leading to this phenomenon is the insufficient feature learning of small objects in point clouds, making it difficult to obtain more discriminative features. To address this challenge, we propose a 3D object detection framework based on point clouds that takes into account the detection of small objects, termed VoxT-GNN. The framework comprises two core components: a Voxel-Level Transformer (VoxelFormer) for local feature learning and a Graph Neural Network Feed-Forward Network (GnnFFN) for global feature learning. By embedding GnnFFN as an intermediate layer between the encoder and decoder of VoxelFormer, we achieve flexible scaling of the global receptive field while maximally preserving the original point cloud structure. This design enables effective adaptation to objects of varying sizes and categories, providing a viable solution for detection applications across diverse scenarios. Extensive experiments on KITTI and Waymo Open Dataset (WOD) demonstrate the strong competitiveness of our method, particularly showing significant improvements in small object detection. Notably, our approach achieves the second-highest mAP of 65.44% across three categories (car, pedestrian, and cyclist) on KITTI benchmark. The source code is available at https://github.com/yujianxinnian/VoxT-GNN.

1.1. Introduction

Object detection aims to predict the bounding boxes and class labels of key objects in space (Wang & Ye, 2020), playing a vital role in many fields such as fire rescue (Jing et al., 2024), autonomous driving (Mao et al., 2023; Qian et al., 2022; Zamanakos & Tsochatzidis & Amanatiadis & Pratikakis, 2021), text recognition (Jing et al., 2024), robotics (Zhang et al., 2024), and augmented reality (Shreyas & Sheth, 2021). For instance, in autonomous driving (Hawlader et al., 2024; Song et al., 2024), accurately and efficiently detecting objects ahead can assist drivers in making decisions and enhance the driving experience; in text recognition (Nasir & Malik, 2024), detection technology can help identify various word texts, providing technical support for subsequent in-depth analysis. 3D detection technology is a branch of object detection. Currently, LiDAR point clouds are a primary data representation used for 3D object detection (Mao et al., 2023; Qian et al., 2022; Y. et al., 2021), and compared to cameras that generate images, LiDAR sensors can

* Corresponding author.
E-mail address: wusheng@fzu.edu.cn (S. Wu).

https://doi.org/10.1016/j.ipm.2025.104155
Received 26 October 2024; Received in revised form 25 February 2025; Accepted 16 March 2025
Available online 27 March 2025

obtain accurate depth and geometric information in complex and variable scenes, which is beneficial for improving detection performance.

Existing LiDAR-based 3D detection methods have provided effective solutions for various application scenarios. However, these approaches either convert irregular point clouds into regular representations (typically voxel grids) for feature learning at the cost of spatial information loss and weakened feature distinctiveness, or directly process raw point cloud data with computational inefficiency that hinders real-time detection. Notably, numerous methods focus solely on single-category detection (particularly large objects like cars) (Chen et al., 2022; Dong et al., 2023; Feng et al., 2023; He et al., 2022; Kuang et al., 2020; Q. et al., 2023; Tang et al., 2023; Xie et al., 2024; Zheng et al., 2021), while others require extensive modifications or many hyperparameters adjustment to adapt to other object categories (Deng et al., 2021; Lang et al., 2019; Sheng et al., 2021; Shi et al., 2020; Yan & Mao & Li, 2018; Zhou & Tuzel, 2018). Furthermore, several methods (Chen et al., 2020; Wang et al., 2024; Jing et al., 2024; Le et al., 2023; Yin et al., 2021) specifically aim to enhance detection performance for small objects (e.g., pedestrians or cyclists), yet often compromise their competitiveness in large object detection (e.g., vehicles). Our analysis reveals that most existing approaches addressing multi-scale and multi-category detection tend to expand receptive fields or design multi-scale receptive fields, which risks inadequate feature learning for small objects. This limitation results in insufficient discriminative feature acquisition, ultimately restricting these methods to single-category detection or simple scenarios while underperforming in complex environments with diverse object categories and size variations.

To address the aforementioned issue, inspired by the works of VoxelNet (Zhou & Tuzel, 2018), Voxel Set Transformer(He et al., 2022) and Point-GNN (Shi & Rajkumar, 2020), we have voxelized the point cloud and integrated the powerful inference capabilities of Transformer (Vaswani et al., 2017) and GNN(Scarselli et al., 2008) to propose a novel 3D object detection framework named VoxT-GNN. The primary challenge lies in effectively integrating the Transformer and GNN into a cohesive framework while learning high-quality features from point clouds without compromising inference speed. VoxT-GNN is designed not only for single-stage object detection but can also be extended to two-stage detection through integration with a Region of Interest (RoI) module (Chen et al., 2019; Te et al., 2018). It begins by voxelizing the point cloud, followed by constructing local VoxelFormer networks at the voxel-level and global GnnFNN networks to enhance overall 3D detection performance. The principle of our approach is outlined as follows: Firstly, we adopt a partitioning strategy to segment the raw point cloud through voxelization, treating each voxel as an individual region. Within each region, we sample T points (where T is a constant integer), significantly reducing the total number of points while ensuring effective, stable, and well-balanced key points. This process ultimately conserves computational resources for subsequent inference tasks. Secondly, the initial features of the sampled points within each voxel partition are encoded using a Multi-Layer Perceptron (MLP) to embed information regarding reflection intensity, coordinates, and relative coordinates. The VoxelFormer encoder then processes these features to encode the characteristics of the points within each partition and aggregate local features from the corresponding voxels. Subsequently, assuming that each point cloud can be divided into M non-empty voxel regions, we treat each non-empty region as an individual point and utilize aggregated regional features (voxel features) as input for these points. By employing the k-Nearest Neighbors (k-NN) method (Zhao, 2018), we construct a graph representing these M points and implement GnnFNN as a feature exchange network among them. This facilitates learning global features across the entire point cloud. Finally, the decoder component of VoxelFormer maps the exchanged information from M voxels back to T points within each voxel before decoding their respective features. These decoded features are subsequently forwarded to the detection head for object detection purposes. We assert that VoxT-GNN not only enhances discriminative feature learning for precise and fine-grained detection outcomes but also achieves an optimal balance between inference performance and speed—thereby offering distinct advantages in practical applications.

Our contributions can be summarized in several key aspects:

(1) We introduce the VoxT-GNN framework, which effectively harnesses the strengths of feature learning methodologies from both Transformer and Graph Neural Network (GNN) architectures, thereby improving the quality of 3D object detection. Experimental results indicate that our approach not only facilitates real-time detection but also demonstrates superior performance when compared to existing state-of-the-art benchmarks in 3D object detection. This advantage is particularly pronounced in the detection of cyclists and pedestrians.
(2) We designed the scalable receptive field GnnFNN network, which serves as an intermediate layer between the encoder and decoder in VoxelFormer. This is a region-to-region feature learning approach that enables cross-region information transfer, allowing for better acquisition of local and global features, and can significantly improve detection performance.
(3) We propose treating each region (voxel) as a point representing a vertex of a graph and construct this graph using the k-NN method. Extensive experimental results under varying voxel sizes with different K values indicate that these combinations have a substantial impact on detection performance for various objects (cars, pedestrians, cyclists), allowing us to identify optimal configurations.
(4) We extend the core methodologies of VoxT-GNN to two-stage 3D object detection. Experimental findings confirm that this extension achieves superior detection performance.

1.2. Related work

Unlike detection methods based on images (Wang et al., 2024) and videos (Li et al., 2024), LiDAR-only approaches can capture more precise geospatial information and mitigate the impact of extreme weather conditions (Song et al., 2024; Zhang et al., 2024), thereby enhancing detection performance. However, due to the sparse, unordered, irregular, and variable point density characteristics of 3D point clouds, it poses great challenges for feature learning through traditional deep learning models such as MLP and CNN (Bello

et al., 2020; Y. et al., 2021; Zhou & Tuzel, 2018). Generally, LiDAR-only methods can be categorized into the following types (Mao et al., 2023; Qian et al., 2022; Song et al., 2024; Zamanakos et al., 2021):

1.2.1. View-based

First, 3D point clouds are transformed into regular representations such as planar maps or BEV formats. Subsequently, 2D detection algorithms, including multilayer perceptrons (MLPs) and convolutional neural networks (CNNs), are employed for object detection. This approach is intuitive and straightforward; however, it is more vulnerable to the challenges posed by object occlusion and variations in scale. Furthermore, these methods convert 3D point clouds into 2D representations, resulting in a significant loss of geometric spatial information. VeloFCN (Bo et al., 2016) transforms point clouds into a front-view 2D feature map representation for 2D detection. To address the occlusion challenges arising from overlap in the front view, PIXOR (Yang et al., 2018) transforms point clouds into a more compact 2D BEV representation. This representation is subsequently discretized and analyzed using standard 2D object detection algorithms. Based on the PIXOR method, subsequent variants such as (Liang et al., 2021; Yin et al., 2021) have been developed. To address the issues of object occlusion and scale variation, RangeRCNN (Liang et al., 2020) and RangeIOUDet (Liang et al., 2021) introduce a viewpoint view method used before converting to a BEV attempt, achieving point-to-point feature extraction. Such methods are currently less prevalent; however, with the advent of Transformers, the work (Feng et al., 2023) introduced SEFormer. This approach not only preserves effective local structures but also enhances local feature encoding, thereby achieving high-performance detection.

1.2.2. Voxel-based

This methodology transforms sparse and irregular point clouds into regular voxel representations with increased density. Subsequently, it employs 3D CNNs to compute features for non-empty voxels, thereby extracting high-quality features from the point cloud. Although voxel-based methods improve computational efficiency, they also have significant drawbacks (He et al., 2020; Shi et al., 2020): 1) **Loss of spatial information**, as the discretization of point clouds leads to information loss, reducing fine-grained localization accuracy, especially limiting the detection of small objects; 2) **Non-reproducible results**, the non-deterministic nature of point cloud random sampling can lead to irreproducible detection results in 3D object detection models; 3) **Additional computation**, zero-padding for invalid voxels to a certain extent leads to unnecessary computations; 4) **Fixed receptive field**, which limits the detection performance for objects of different sizes and densities. To overcome these challenges, researchers typically focus on two aspects—data representation and model architecture—to design methods that enhance the network's feature capacity and improve object localization accuracy, thereby adapting to more complex scenarios. VoxelNet (Zhou & Tuzel, 2018) introduces the VFE layer within a 3D CNN framework for extracting features from point-wise features within voxel cells. However, due to the cubic computational complexity of 3D CNNs, it significantly increases memory consumption and computational load, leading to low efficiency. Subsequently, SECOND (Yan et al., 2018) greatly reduces memory consumption through sparse convolution operations, then employs the general matrix multiplication (GEMM) algorithm for convolution, which, although increasing computational speed, still involves a large computational load. To further reduce computational cost, SA-SSD (He et al., 2020) treats each point as non-zero and quantifies it into a tensor index, enabling multiple points to share the same index, thereby significantly reducing computational overhead. Additionally, addressing the issues of computational complexity, uneven distribution of pseudo points, and imprecise completion caused by the sparsity of point clouds in many previous methods (Fei et al., 2022; Lin et al., 2023; Xia et al., 2023), SQDNet (Yujian et al., 2024) first achieves a sparse-to-dense matching through grid position indexing, simplifying the pseudo point generation process, and then uses the density of LiDAR points within these grids to mitigate the uneven distribution and noise issues of pseudo points, achieving considerable detection performance. In terms of model architecture, MVF (Zhou et al., 2020) proposed a dynamic voxelization approach to overcome the issue of insufficient computational efficiency in previous methods. To address the problem of a fixed receptive field, Part-A^2 (Shi et al., 2020), HVNet (Ye et al., 2020), and Voxel-FPN (Kuang et al., 2020) improved feature extraction quality through part-aware aggregation or multi-scale voxel feature fusion. Regarding the discrepancy in feature learning across different stages of voxel-based two-stage detection, HRNet (Lu et al., 2024) has developed multi-scale voxel feature detection heads and dynamic sample selection modules for both the first and second stages. This approach aims to achieve a more balanced selection of positive and negative samples, thereby enhancing the performance of 3D car detection within sparse point clouds. Addressing the issue of point cloud sparsity, SphereFormer (Lai et al., 2023) employs radial window self-attention, dividing the space into multiple non-overlapping narrow and elongated windows, thereby expanding the receptive field. This approach aggregates information from dense nearby points to sparse distant points, enhancing detection performance. Subsequently, VoxelNext (Chen et al., 2023) directly utilizes sparse voxel features in place of handcrafted proxies, eliminating the need for sparse-to-dense conversion or post-processing with Non-Maximum Suppression (NMS), thus achieving high-quality 3D detection. HEDNet (Zhang et al., 2024) utilizes an encoder-decoder architecture to model long-range dependencies in point clouds, effectively learning point cloud features and achieving good results in the detection of large and distant objects.

1.2.3. Pillar-based

Essentially, pillar-based methods represent a specialized form of voxelization in which point clouds are discretized into uniformly distributed vertical columns on the x-y plane. PointPillars (Lang et al., 2019) pioneered this approach by encoding point clouds into vertical pillars, significantly reducing the number of voxels and thereby enhancing computational speed. The feature encoder in

PointPillars organizes the point cloud into vertical columns and subsequently redistributes these columns to their original positions to generate pseudo-images from a bird's-eye view (BEV), thus facilitating the application of 2D convolutional neural networks (CNNs) for object detection. Following this innovation, various adaptations of PointPillars (Fan et al., 2022; Feng et al., 2023; Li et al., 2023; Shi et al., 2022; Sun et al., 2022) have been employed for three-dimensional detection tasks. PillarNet (Shi et al., 2022) introduced an "encoder-backend" architecture aimed at improving detection performance. To maintain a global perspective receptive field, SWFormer (Sun et al., 2022) and ESS (Fan et al., 2022) leveraged concepts from Swin Transformer (Liu et al., 2021), implementing a hierarchical window mechanism on pseudo-images that enhances 3D detection efficacy. Moreover, PillarNeXt integrates pillar-based methodologies with advanced 2D detection techniques to maximize the receptive field, resulting in significant improvements in 3D detection performance. Although pillar-based detection methods achieve speed enhancements ranging from two to four times compared to traditional 3D CNN models, it is crucial to acknowledge that the use of pillars for feature extraction may result in a loss of geometric information inherent in the raw 3D point cloud. This loss could potentially affect detection accuracy to some extent. To address the issue of information loss, HANet (Q. et al., 2023)proposed a three-dimensional heatmap to reflect object information, thereby enhancing the feature extraction capability of the backbone network. PillarNeXt (Li et al., 2023) redesigned the local feature aggregator from the perspective of computational resource allocation, expanding the receptive field, which significantly improves detection performance. Subsequently, TransPillars (Luo et al., 2023) approached multi-frame 3D object detection from the perspective of point cloud sequences, utilizing temporal features from consecutive point cloud frames. This method not only enables the fusion of voxel features from different point cloud frames but also achieves multi-scale local feature extraction, greatly enhancing the quality of feature extraction. EFNet (Meng et al., 2024) employs an Enhanced Pillar Feature Module (EPFM) to learn point cloud features from both the interior and exterior spaces of two pillars, and integrates multi-scale information in the detection head to obtain high-quality point cloud features. Additionally, pillar-based methods tend to perform poorly in detecting distant or small objects. To address this, (Li & Lu, 2023) designed a three-branch extended convolutional network that achieves high-quality detection of objects of varying sizes, particularly small objects.

1.2.4. Point-based

Unlike voxel-based methods, point-based approaches emphasize the preservation of the original structure of point clouds, which is beneficial for fine-grained feature extraction. These techniques can achieve a more flexible receptive field by stacking set abstraction layers and are capable of adapting to more complex scenarios; however, they often entail significant computational costs. The PointNet (Qi et al., 2017) method represents a pioneering approach that directly utilizes point clouds to learn 3D features for classification and segmentation tasks. Its subsequent enhancement, PointNet++ (Qi et al., 2017), introduced a density-adaptive layer designed to recursively aggregate point clouds in a hierarchical manner, thereby facilitating the acquisition of adaptively finer-grained information. Subsequent point-based methodologies such as PointRCNN (Shi et al., 2019) and 3DSSD (Yang et al., 2020) are primarily built upon the foundations established by PointNet and PointNet++. To further mitigate computational expenses, later works (Li et al., 2023; Yang et al., 2018) incorporated additional semantic segmentation tasks aimed at filtering out background points that contribute minimally to detection efforts, effectively enhancing computational efficiency. Subsequently, in order to achieve a balance between detection accuracy and inference time, FARP-Net (Xie et al., 2024) proposed a new local global feature aggregation layer that utilizes the complementary correlation between local and global features to achieve adaptive feature fusion and improve inference speed.

1.2.5. Graph-based

Graph neural networks (GNNs) demonstrate robust reasoning capabilities for non-Euclidean data, such as 3D point clouds. This has led to a surge of research employing GNNs for various tasks, including point cloud classification and semantic segmentation. These approaches effectively preserve the geometric information and irregular characteristics inherent in point clouds while fully leveraging contextual information, thereby significantly enhancing detection accuracy. However, the lengthy processes involved in constructing and reasoning through graphs during iterations, coupled with time-consuming model training, present notable challenges. Point-GNN (Shi & Rajkumar, 2020) represents a pioneering effort that treats each point within the point cloud as a graph vertex, connecting points within a specified radius to form graph edges. Subsequently, it utilizes graph neural networks to learn features from the point cloud, showcasing the substantial potential of GNNs for object detection. Subsequently, to improve computational efficiency, a series of GNN-based improvements (Feng et al., 2021; Meraz et al., 2022; Thakur & Peethambaran, 2020; Zhang et al., 2021) have been applied to 3D detection. DEConvGNN (Feng et al., 2021; Thakur & Peethambaran, 2020) introduces attention mechanisms into GNNs, achieving efficient feature aggregation and thus enhancing detection accuracy. DC-GNN (Meraz et al., 2022) adopts a concept of channel-drop and constructs dynamic graphs in each layer to extract more discriminative hierarchical features, thereby improving detection accuracy. Currently, graph-based methods have become a new avenue for 3D point cloud object detection, and research in this direction warrants further exploration. To address the issue of local and global information fusion, DCGNN (Xiong et al., 2023) utilizes density-based clustering queries to replace traditional voxel partitioning and employs graph neural networks to model local and global relationships, achieving high-quality feature fusion.

1.2.6. Hybrid-methods

This approach is typically designed to leverage the strengths of the previously mentioned methods while mitigating their weaknesses. PVConv (Liu et al., 2019) was the first to effectively integrate the advantages of both point-based and voxel-based techniques.

The primary objective of this method is to facilitate feature interaction between voxels and points through point-to-voxel or voxel-to-point transformations, thereby enhancing detection accuracy. To balance detection precision and speed, Fast Point R-CNN (Chen et al., 2019) fully capitalizes on the benefits of both voxels and points, achieving an operational speed of 15 FPS while simultaneously improving detection accuracy. Subsequently, a key innovation in the STD (Yang et al., 2019) framework is its use of spherical anchors, which can attain higher recall rates while also enhancing detection precision. To further improve overall detection performance, PV-RCNN (Shi et al., 2023) deeply integrates 3D sparse convolutional computations from voxels with the flexible receptive field provided by PointNet—a method that has been shown to learn high-quality features from point clouds. Recent studies (Guan et al., 2022; Hu et al., 2022; Shi et al., 2023) have all been developed based on hybrid approaches. Currently, hybrid methods have emerged as a widely adopted design philosophy within the realm of 3D object detection.

1.2.7. 3D self-attention variants in point clouds

Following the remarkable success of Transformer models in natural language processing and their exceptional performance across various computer vision tasks (Dosovitskiy et al., 2020; Liu et al., 2021) a multitude of models leveraging the standard self-attention mechanism have been employed for 3D object detection. These models are often integrated with techniques such as voxel-based, point-based, and graph-based approaches previously mentioned. Prior studies (Guo et al., 2021; Liu et al., 2022; Liu et al., 2021; Mao et al., 2021; Misra et al., 2021; Pan et al., 2021; Sheng et al., 2021; Zhao et al., 2021) that utilized the self-attention mechanism of Transformers for tasks including point cloud semantic segmentation, part segmentation, and object classification have demonstrated competitive performance. Notably, recent research (Dong et al., 2023; He et al., 2022; Hoang et al.,2024; Lu et al.,2023; Pei et al., 2023; Ren et al.,2023; Yang et al., 2023) has further validated the impressive robustness of Transformer-based models, which is anticipated to advance investigations in the domain of safe perception for autonomous driving. VoxSeT (He et al., 2022) replaces one self-attention mechanism with two cross-attention mechanisms, enabling the computation of self-attention for clusters of tokens of arbitrary size and effectively enhancing the performance of 3D car detection. Clusterformer (Pei et al., 2023) treats each object as a cluster composed of non-empty voxels in three-dimensional space and employs Transformers for encoding and decoding processes while directly generating proposals from sparse voxel features. Pvt-ssd (Yang et al., 2023) integrates Transformers within a Point-Voxel module, which efficiently captures long-range context from voxels while accurately determining positions from points. DGT-Det3D (Ren et al., 2023) utilizes dynamic graphs and graph-aware self-attention mechanisms to effectively model long-range contextual relationships in point clouds. VGA (Lu et al., 2023) introduces a graph attention pooling (GAP) module in a two-stage detector, constructing hierarchical local graphs for varying scales and associating information from adjacent voxels to extract Region of Interest (RoI) features. The aforementioned methods typically implement local self-attention by treating objects in 3D space as clusters, which may result in random point loss, or they perform convolutional self-attention on discrete representations of points or voxels, often leading to limited attention ranges. This indicates an ongoing deficiency in the extraction of global features from point clouds. To address this challenge, MVTr (Ai et al., 2024) employs self-attention among multi-feature voxels to capture long-range relationships within point clouds and extract high-quality global features. Moreover, responding to the limitations of traditional detection methods that do not explicitly account for the complete shape of objects, recent studies (Dong et al., 2023; Hoang et al., 2024) have introduced Transformers to develop efficient 3D detectors. These advancements facilitate accurate estimation of the full shapes of objects, thereby achieving high-performance 3D detection for vehicles.

The aforementioned methodologies offer diverse solutions to address challenges in LiDAR-based 3D object detection, with their respective strengths and limitations summarized as follows:

1. View-based Methods (Bo et al., 2016; Chen et al., 2017; Feng et al., 2023; Liang et al., 2019; Liang et al., 2018; Yang et al., 2018). These approaches typically project irregular point clouds into structured representations (e.g., front-view maps or BEV) before processing them via 2D CNNs to enhance feature learning efficiency. While computationally efficient, their inherent 2D-driven 3D detection paradigm inevitably compromises localization accuracy.
2. Voxel-based Methods (He et al., 2020; Kuang et al., 2020; Shi et al., 2020; Yan et al., 2018; Ye et al., 2020; Zhou & Tuzel, 2018). By converting point clouds into regular volumetric grids and employing 3D CNNs for feature extraction, these methods trade computational speed (compared to 2D CNNs) for information loss during voxelization, particularly in sparse regions.
3. Point-based Methods (Qi et al., 2018; Shi et al., 2019; Wang & Jia, 2019; Yang et al., 2020; Yang et al., 2019). Pioneered by PointNet (Qi et al., 2017) and PointNet++ (Qi et al., 2017), these techniques directly process raw point clouds to preserve geometric fidelity, achieving superior detection accuracy at the cost of prohibitive computational overhead due to permutation-invariant operations.
4. Pillar-based Methods (Lang et al., 2019; Luo et al., 2023). A specialized voxel variant that encodes point clouds into vertical pillars (columnar voxels) for compatibility with 2D detection frameworks. While computationally efficient, they inherit voxelization-induced information loss and limited vertical resolution.
5. Graph-based Methods (Meraz et al., 2022; Shi & Rajkumar, 2020; Wang & Solomon, 2021; Xiong et al., 2023; Zarzar et al., 2019). Treating points as graph vertices with radius-connected edges, these methods leverage GNNs to preserve irregular geometries. However, scalability issues arise when processing large-scale scenes due to quadratic edge complexity.
6. Attention-based Methods (Guo et al., 2021; Liu et al., 2022; Liu et al., 2021; Mao et al., 2021; Misra et al., 2021; Pan et al., 2021; Sheng et al., 2021; Zhao et al., 2021). While transformer architectures significantly boost detection performance, most implementations apply self-attention on fixed-size token clusters, constraining receptive field adaptability for multi-scale objects.

Advanced variants enable variable-sized token attention but rely on conventional feed-forward networks or 3D sparse CNNs for cross-voxel communication, limiting hierarchical feature learning.

7. Hybrid Methods (Chen et al., 2019; Guan et al., 2022; Liu et al., 2019; Shi et al., 2023; Tang et al., 2020; Vora et al., 2020; Yang et al., 2019). Strategic integrations of the above paradigms aim to synergize complementary strengths—for instance, combining voxel-based efficiency with point-based geometric fidelity—while mitigating individual weaknesses through architectural innovations.

1.3. Methodology

1.3.1. Preliminary

1.3.1.1. Induced set attention block (ISAB)

Typically, point clouds consist of tens of thousands of points, and applying self-attention mechanisms directly would consume a significant amount of computational resources. To reduce the computational load, (Lee et al., 2019) introduced an Induced Set Attention Block (ISAB), which employs two simplified cross-attention mechanisms driven by a set of latent codes to approximate the complete self-attention within the set. Given an input set $X \in \mathbb{R}^{n\times d}$ and a set $L \in \mathbb{R}^{k\times d}$ containing k latent codes, the output set $O \in \mathbb{R}^{n\times d}$ produced by the ISAB can be formulated as:

$$H = CrossAttention1(L, X) \in \mathbb{R}^{k\times d} \tag{1}$$

$$\widetilde{H} = FFN(H) \in \mathbb{R}^{k\times d} \tag{2}$$

$$O = CrossAttention2(X, \widetilde{H}) \in \mathbb{R}^{n\times d} \tag{3}$$

Where, CrossAttention1 utilizes the latent codes L to transform the input set X into hidden features H, which are then transformed into $\widetilde{H}$ by a point-wise feedforward neural network (FFN). Subsequently, CrossAttention2 attends to the input set X with respect to the hidden features $\widetilde{H}$ to obtain the output set O. It is crucial to highlight that the self-attention mechanism, when applied to induced set attention, can be effectively approximated through low-rank projection. Therefore, performing k-clustering on the input set using the latent codes L as cluster centers presents a viable alternative to full self-attention. Given that $k \ll n$, this method has the potential to significantly alleviate computational burdens.

1.3.1.2. GNNs in the processing of 3D point clouds

To address the challenge of spatial information loss that occurs when converting 3D point clouds into conventional data representations through traditional methods, (Shi & Rajkumar, 2020) proposed a GNN and developed an approach termed Point-GNN. This approach facilitates direct feature learning on irregular point clouds through Graph Construction and GNN Construction, thereby eliminating the necessity for conversion to regular representations.

1.3.1.3. Graph construction

Define a point cloud dataset with n points as the set $P = \{p_1, \ldots, p_n\}$, where each point $p_i = (x_i, v_i)$ consists of 3D coordinates $x_i \in \mathbb{R}^3$ and a state value $v_i \in \mathbb{R}^k$. The state value v_i represents the attribute features of the point in the form of a vector of length k, encompassing features such as reflectance intensity, coordinate differences, and adjacency relationships encoded as features. Given a point cloud P, we construct a graph P by utilizing P as the vertex set. Each point p_i is connected to its neighboring points within a fixed radius r, thereby forming the edge set E. Thus:

$$G = (P, E) \tag{4}$$

$$E = \left\{ \left(p_i, p_j\right) \mid ||x_i - x_j||_2 < r \right\} \tag{5}$$

To preserve the information inherent in the raw point cloud, the dense point cloud is represented as the initial state values v_i of the vertices. This process entails searching for raw points within a radius r surrounding each vertex and utilizing a neural network to extract their features from this collection.

1.3.1.4. GNN construction

Point-GNN(Shi & Rajkumar, 2020) adheres to the traditional graph neural network methodology, which involves refining vertex features through the aggregation of features along edges. The specific formula is presented below:

$$\triangle x_i^t = h^t\left(v_i^t\right) \tag{6}$$

$$v_i^{t+1} = g^t\left(\rho\left(\left\{f^t\left(x_j - x_i + \triangle x_i^t, v_j^t\right)\right\}\right), v_i^t\right) \tag{7}$$

To ensure the translational invariance of the 3D point cloud, Point-GNN employs relative coordinates $x_j - x_i$ as input. In this

context, $\triangle x_i^t$ denotes the variable that captures the coordinate shift of vertex i during each iteration. The function $h^t(.)$ is responsible for calculating the positional shift of vertices, utilizing the state value of the central vertex from the preceding iteration to determine this shift and mitigate variance in translation. At time step t, v_i^t denotes the state of vertex i, while v_j^t indicates the state of vertex j within the neighborhood of vertex i. Furthermore, v_i^{t+1} signifies the state of vertex i at time step t+1. The function $f^t(.)$ serves as an edge state aggregation function in the GNN. The pooling function denoted by $\rho(.)$, which follows edge state aggregation, can employ pooling methods such as max or mean due to the inherent unordered characteristics of point clouds. Additionally, $g^t(.)$ functions as an aggregation mechanism for vertex states within GNN. For these three functions—$f^t(.)$, $g^t(.)$, and $h^t(.)$—they can be modeled using MLP, represented as MLP_f, MLP_g, MLP_h. A residual connection is incorporated into $g^t(.)$.

1.3.2. Voxelization and vectorization

1.3.2.1. Voxelization

Following the voxelization method in VoxelNet (Zhou & Tuzel, 2018), we partition the 3D point cloud space into a regular 3D voxel grid, as shown in Fig. 2.(B). Let the ranges of the 3D voxel grid space along the Z, Y, and X axes be D, H, and W respectively. First, we define the size of the voxels as v_D, v_H, and v_W, and then the dimensions of the generated 3D voxel grid are $D' = D/v_D$, $H' = H/v_H$, and $W' = W/v_W$. Typically, D, H, and W are multiples of v_D, v_H, and v_W.

Specifically, let $\{p_i = (x_i, y_i, z_i) : i = 1, ...n\}$ denote the coordinates of the points, and let $[d_x, d_y, d_z]$ represent the dimensions of the 3D voxels. The voxel coordinates $\mathscr{V}$ can be computed as follows:

$$\mathscr{V} = \left\{ \mathscr{V}_i = \left(\lfloor \frac{x_i}{d_x} \rfloor, \lfloor \frac{y_i}{d_y} \rfloor, \lfloor \frac{z_i}{d_z} \rfloor \right) : i = 1, ..., n \right\} \tag{8}$$

Where $\lfloor \cdot \rfloor$ denotes the floor function. It is essential to acknowledge that the number of points within each voxel can vary considerably due to the inherent irregularity and sparsity of point clouds. Consequently, it is a common practice in practical applications to sample a fixed number of points from each voxel, denoted as T.

1.3.2.2. Vectorization of cross-voxel operations

The features of a voxel can be aggregated from the point features within the voxel. To ensure that the operation of extracting features from each voxel can be parallelized, cross-voxel operations can be vectorized (He et al., 2022). This vectorization can be accomplished using the `scatter_function' (Pytorch, 2023) from the CUDA (Corporation, 2024) kernel library, which facilitates symmetric reductions across various segments of a matrix, including operations such as summation, maximum, and average. The input set is regarded as a single matrix, in which each row corresponds to a point feature. The associated voxel can be accessed through a voxel coordinate table.

Therefore, given the input feature matrix $\{X_i : i = 1, ...n\}$, the simplified feature matrix $\{Y_j : j = 1, ..., m\}$ obtained through reduction by the symmetric function $F(\cdot)$ can be expressed as follows:

$$Y = \{ F(\{X_i : \mathscr{V}_i = j\}) : j = 1, ..., m \} \tag{9}$$

Where the number of non-empty voxel regions is denoted by m, and employing the `scatter_function`, Y can be represented as:

$$Y = F_{scatter}(X, Y) \tag{10}$$

1.3.3. Voxel-Level transformer(voxelformer)

Point clouds exhibit rich structural details in local regions due to their sparse distribution across 3D space. Building upon ISAB(Lee et al., 2019), we design the VoxelFormer network by referencing VSA(He et al., 2022), which directly performs self-attention within local regions. Specifically, we employ the aforementioned voxelization method (Zhou & Tuzel, 2018) to partition the 3D point cloud space into voxel grids (each voxel serving as a region), where each region contains a set of latent codes (equivalent to Q-query vectors in Transformers). The encoder then executes self-attention to learn local features of the point cloud. Meanwhile, we embed a GnnFFN intermediate layer between the encoder and decoder to facilitate inter-voxel information exchange for capturing global features. Subsequently, the decoder broadcasts these global features to points within each voxel region. As illustrated in Fig. 2.(C), we formulate VoxelFormer in matrix multiplication form for clarity. The proposed VoxelFormer module constitutes an instantiation of Transformer architecture: it first transforms input sets into hidden space, then learns finer-grained hidden features through the GnnFFN intermediate layer, and finally decodes them to generate output sets.

It is noteworthy that, owing to the vectorization of cross-voxel operations, points within different voxels can be processed in parallel through linear computation. As a result, by implementing VoxelFormer, it becomes feasible to sample a greater number of points within voxels or even eliminate the need for sampling altogether, thereby circumventing the necessity of point padding as required by conventional voxel-based methods (Zhou & Tuzel, 2018).

1.3.3.1. Encoder

In the encoder design, following the Transformer paradigm, we linearly transform the input features from the preceding module to derive the key $K \in \mathbb{R}^{n \times d}$ and value $V \in \mathbb{R}^{n \times d}$. A cross-attention operation is then performed between the key K and the latent codes

(queries) $L \in \mathbb{R}^{k\times d}$, yielding an attention matrix $A \in \mathbb{R}^{n\times k\times d}$. Subsequently, voxel-wise normalization is applied to A using the voxel coordinate indices of $\mathscr{V}$, resulting in the normalized attention matrix $\widetilde{A}$. This matrix is multiplied with the value V to generate hidden features H. Finally, voxel reduction is performed on the deep features H within the hidden space based on the voxel indices of $\mathscr{V}$, producing locally aggregated voxel features H_r. The computation can be formulated as:

$$\widetilde{A} = Softmax_{scatter}(A, \mathscr{V}), A = KL^T \tag{11}$$

$$H = \widetilde{A}^T V \tag{12}$$

$$H_r = Sum_{scatter}(H, \mathscr{V}) \tag{13}$$

Where $Softmax_{scatter}$ and $Sum_{scatter}$ are the scatter library functions from the aforementioned scatter_function (Pytorch, 2023) library.

1.3.3.2. GNN feed-forward network (GnnFFN)

The core operation of VoxelFormer involves encoding features into a hidden space using latent codes $L \in \mathbb{R}^{k\times d}$ within each voxel region. We implement GnnFFN to achieve more sophisticated and fine-grained feature transformations in this hidden space. Unlike conventional feed-forward networks (FFNs) that perform only point-wise updates, our GnnFFN enables cross-voxel information exchange, thereby capturing long-range contextual dependencies critical for global feature learning in dense point clouds. To this end, we adapt the Point-GNN (Shi & Rajkumar, 2020) operator (abbreviated as PGNN) by taking reduced non-empty voxel features H_r as input (treating voxels as graph vertices) and constructing a voxel-level global graph based on their coordinates C_r. Three iterative message-passing steps are performed to enable cross-region feature interactions in 3D space. Given learnable PGNN weights W_1, W_2, and W_3, the hidden features $\widetilde{H}_r \in \mathbb{R}^{m\times k\times d}$ generated by GnnFFN are computed as:

$$\widetilde{H}_r = PGNN(\mathscr{g}(\sigma(\text{PGNN}(\mathscr{g}(\sigma(\text{PGNN}(\mathscr{g}(H_r, C_r); W_1)), C_r); W_2)), C_r); W_3) \tag{14}$$

Where σ denotes the non-linear activation function, and $\mathscr{g}$ represents the formulation for graph construction. The GnnFFN serves as the core component of the VoxelFormer module due to its ability to capture ideal inductive biases and global contextual information.

It is crucial to emphasize that, unlike Point-GNN (Shi & Rajkumar, 2020), which constructs graphs using neighbors within a fixed radius r (spherical query), we employ the k-NN (Zhao, 2018) method to build global graph across m non-empty voxels. This design stems from treating each non-empty voxel as a graph vertex. Directly applying spherical query between voxel regions would introduce critical issues, as illustrated in Fig. 1: 1) Ambiguity in determining whether a voxel should be included as a graph vertex; 2) Undesirable exclusion of voxels where most raw points reside in overlapping regions, even if the intersection area with the sphere is less than half the voxel size. To circumvent these limitations, our k-NN-based graph construction globally connects each voxel to its K nearest neighbors, ensuring deterministic edge formation. Furthermore, by adjusting the value of K, the global receptive field can be scaled to accommodate objects of different sizes and categories. This is crucial for extracting high-quality global features from point clouds. The impact of K-values in our GnnFFN module will be systematically analyzed in subsequent experiments to identify scenario-optimal configurations.

1.3.3.3. Decoder

In the decoder design, we utilize the enriched hidden features $\widetilde{H}_r$ as input to reconstruct the output set. Specifically, we first broadcast the hidden features of each voxel to all points within its corresponding voxel region using the voxel indices $\mathscr{V}$, generating a feature tensor $\widetilde{H} \in R^{n\times k\times d}$ with the same length as the input set. Subsequently, we employ linear projections to derive query and key-value pairs from both the input set and the hidden features. Given the matrices for queries $\in R^{n\times d}$, keys $K \in R^{n\times k\times d}$, and values $V \in R^{n\times k\times d}$, the output set O of the decoder is as follows:

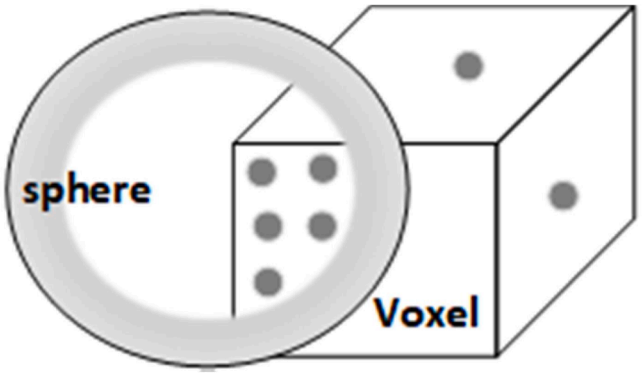

Fig. 1. A form in which a fixed radius sphere intersects with a voxel. Determining whether to exclude the voxel remains a non-trivial challenge, particularly in scenarios requiring robustness to ambiguous spatial or density distributions.

$$A = [\mathscr{A}_1, \ldots, \mathscr{A}_n] = [K_1 Q_1^T, \ldots, K_n Q_n^T], \tag{15}$$

$$\widetilde{\mathscr{A}} = [Softmax(\mathscr{A}), \ldots, Softmax(\mathscr{A}_n)], \tag{16}$$

$$O = [O_1, \ldots, O_n] = [\widetilde{\mathscr{A}}_1^T V_1, \ldots, \widetilde{\mathscr{A}}_n^T V_n] \tag{17}$$

1.3.3.4. Relative position embedding

3D point clouds contain essential coordinate position information, and the positional relationships among individual points are critical components of the local structure within the point cloud. This structural integrity is vital for enhancing overall performance. However, the original self-attention mechanism lacks an inherent representation of the positional order of elements in a sequence. To address this limitation, we introduce a Position Embedding (PE) module for VoxelFormer, which encodes voxel coordinates C_r into high-dimensional features and integrates them with voxel region features H_r. The PE module utilizes Fourier transformation for value retrieval; further details can be found in DETR (Carion et al., 2020).

1.3.4. Voxel-Level transformer and GNN(VoxT-GNN)

The main characteristics of VoxT-GNN are not only its ability to effectively learn local point cloud features but also its capacity to transmit and exchange information across voxel regions to learn high-quality global features. Compared to many existing methods, the critical differences and innovations of VoxT-GNN lie in: 1) The VoxelFormer module transforms the modeling process of point clouds into a local cross-self-attention feature learning process from region-to-region, allowing for the sampling of as many points as possible within voxel regions to better preserve the original structure of the point cloud, thereby learning higher-quality local point cloud features; 2) The GnnFFN intermediate layer treats non-empty voxel regions as points, and all non-empty voxel regions are graphed through the k-NN method, which not only enables long-range contextual information transfer and exchange across regions but also allows dynamic adjustment of the point cloud's global receptive field through different k values to adapt to the 3D detection of objects of different categories and sizes, enabling it to learn more discriminative global point cloud features and thereby improving detection performance; 3) The combined use of VoxelFormer and GnnFFN, along with the decoder of the VoxelFormer module, can better achieve the fusion of local and global point cloud features, making it more suitable for 3D object detection under different categories, sizes, and complex scenarios.

The overall architecture of VoxT-GNN follows the general process of most point cloud-based methods, as shown in Fig. 2.(A), and mainly consists of three components: (1) Feature Learning Backbone Network Module; (2) Points-to-BEV Transformation Module; (3) Detection Head.

1.3.4.1. Feature learning backbone network

Referring to the standard architecture of Transformer network layers, the VoxT-GNN feature learning backbone is composed of interconnected MLPs and VoxelFormer modules. We employ batch normalization as the normalization layer and integrate residual connections into the output of each VoxelFormer module to enhance gradient flow. The semantic level of point cloud features is influenced by the voxel size within the VoxelFormer module, while local feature learning is facilitated by both the encoder and decoder components of this module. In contrast, global feature learning is achieved through the GnnFFN intermediate layers. Unlike methods that progressively down-sample and aggregate point-wise features for context grouping and feature extraction (Qi et al., 2017, 2017, 2019), our proposed VoxelFormer backbone utilizes a region-to-region transformation approach for effective point cloud feature extraction.

1.3.4.2. Points-to-BEV transformation module

Typically, after extracting the 3D high-dimensional depth features of a point cloud, they can be projected into a dense BEV feature model to achieve higher recall rates (Chen et al., 2019; Deng et al., 2021; He et al., 2022; Yan et al., 2018). Therefore, we refer to PointPillars (Lang et al., 2019) and VoxSeT (He et al., 2022) to project the high-dimensional features of the point cloud output by the backbone network onto the BEV. Subsequently, we employ a 2D CNN network with two strides (each stride consisting of three convolutional operations) to perform convolutional processing to increase feature density. The output convolutional connected features are passed to the detection head to achieve bounding-box prediction. We aggregate point features within pillars of size 0.18m $\times$ 0.18m and apply a "soft-pooling" operation to generate BEV features. Given the point-wise output features $X^j \in R^{k \times d}$ within the j-th pillar, the pillar-wise features after pooling using the pooling function F^j can be represented as:

$$F^j = \sum_{m=1}^{k} X_m^j * \mathscr{W}_m^j, \quad \mathscr{W}_m^j = \frac{e^{X_m^j}}{\sum_{m=1}^{k} e^{X_m^j}} \tag{18}$$

1.3.4.3. Detection head and loss

To enhance the feature learning capability of the VoxT-GNN backbone network, we have adopted an approach inspired by PointRCNN (Shi et al., 2019). Specifically, we incorporate the foreground segmentation loss $\mathscr{L}_{seg}$ into the output features, which significantly enhances the accuracy of bounding-box detection. Furthermore, we adhere to established practices in anchor-based methods as outlined in references (Lang et al., 2019; Ren & He & Girshick & Sun, 2017; Yan et al., 2018; Zhou & Tuzel, 2018) for

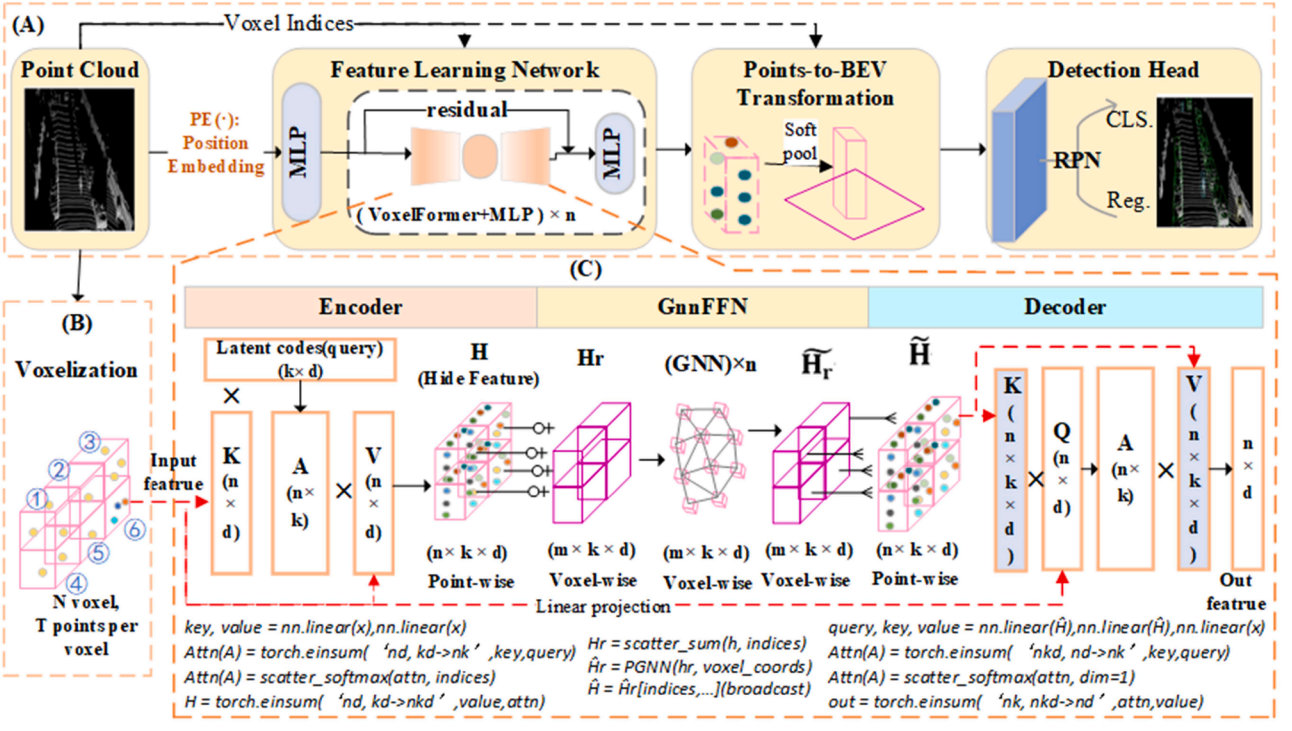

Fig. 2. The comprehensive architecture of the proposed Voxel-Level Transformer and Graph Neural Network (VoxT-GNN). Part (A) uses a standard single-stage detector framework focused on feature learning. Part (B) divides the raw point cloud into uniform voxel grids. Part (C), the VoxelFormer, is the core of the feature network and includes an encoder, a GnnFFN layer, and a decoder. The decoder uses a cross-attention mechanism in each voxel to learn local features. The GnnFFN treats non-empty voxels as graph nodes, constructs a global graph across voxels via k-NN, and learns global features with GNN. These global features are then combined with local voxel points in the decoder to fuse context. The bottom of Part (C) details the matrix operations behind VoxelFormer's workflow.

designing the detection head. The model training loss encompasses several components: the foreground segmentation loss, bounding-box offset regression loss, bounding-box orientation prediction loss, classification loss, and an $L2$ regularization term aimed at mitigating overfitting within the MLP component of the model. Given that the optimization strategy employed in PyTorch (Pytorch, 2023) inherently incorporates $L2$ regularization optimization, we compute the total loss as follows:

$$L = \mathscr{L}_{seg} + \frac{1}{N_p}\left(\mathscr{L}_{cls} + \mathscr{L}_{reg}\right) + \mathscr{L}_{dir}) \tag{19}$$

Where the quantity of positive samples is denoted by N_p, specifically those for which the Intersection over Union (IoU) of the anchors lies within the range $[\sigma 1, \sigma 2]$. $\mathscr{L}_{seg}$ refers to the foreground segmentation loss, while $\mathscr{L}_{cls}$,$\mathscr{L}_{dir}$, and $\mathscr{L}_{reg}$ represent the focal loss for bounding box classification, binary cross-entropy loss for direction prediction, and regression loss for offset (calculated using the Smooth-L1 function), respectively, associated with bounding box detection. For a comprehensive introduction to these concepts, please refer to PointRCNN (Shi et al., 2019) and Second (Yan et al., 2018).

1.3.4.4. Two-stage model

It is important to emphasize that, similar to VFE (Deng et al., 2021; Zhou & Tuzel, 2018), VoxelFormer also encodes features for points within voxel regions. The difference lies in that VFE encodes point-wise features into a single feature vector, whereas VoxelFormer encodes point-wise features into cross-attention feature vector based on latent codes. Consequently, VoxT-GNN can be effectively utilized in two-stage detectors. In this study, we integrate VoxT-GNN with Voxel R-CNN (Deng et al., 2021) to enhance it within a two-stage detection framework. To clearly delineate our contributions, we present the performance of VoxT-GNN across both single-stage and two-stage detectors. Our findings demonstrate that our approach outperforms a variety of existing single-stage and two-stage baselines.

1.4. Experiments

1.4.1. Dataset and metrics

1.4.1.1. Dataset

We trained, validated, and evaluated the proposed VoxT-GNN model utilizing the KITTI dataset (Geiger et al., 2012). The KITTI dataset is one of the most widely employed datasets in the domain of autonomous driving, collaboratively established by the Karlsruhe Institute of Technology (KIT) in Germany and the Toyota Technical Institute (TTI-C) in Chicago, USA. It offers a real-world on-board testing environment for computer vision algorithms, comprising both real image data and corresponding point cloud data collected from urban, rural, and highway scenarios. Each image may contain up to 15 cars and 30 pedestrians, exhibiting varying degrees of occlusion and truncation. For further details, please refer to the official website: https://www.cvlibs.net/datasets/kitti/index.php.

We selected the 3D object detection subset from the KITTI dataset, which comprises 7,481 training samples and 7,518 testing samples. Each sample provides corresponding point cloud and camera image data. Given that the official annotations from KITTI are limited to visible objects within the images, we processed only those point clouds that fall within the field of view of these images. In accordance with standard practices outlined in (Chen et al., 2016; Chen et al., 2017; Xiaozhi et al., 2015; Zhou & Tuzel, 2018), we partitioned the training samples into a training set and a validation set, consisting of 3,712 training samples and 3,769 validation samples respectively. Furthermore, since labels for the test set are not publicly available, predicted results on this set must be submitted to the official KITTI website for evaluation.

In addition to the KITTI(Geiger et al., 2012) dataset, we also conducted generalization validation on version 1.2.0 of the Waymo Open Dataset (WOD)(Sun et al., 2020). The WOD(Sun et al., 2020) comprises 798 training sequences and 202 validation sequences, totaling 1.2TB of data, with 158,361 training samples and 40,077 testing samples respectively. It provides 3D bounding box annotations for three categories: Vehicle, Pedestrian, and Cyclist.

1.4.1.2. Metrics

We follow the official general evaluation protocol of KITTI, which includes 3D detection for three categories: Car, Pedestrian, and Cyclist. Each category consists of three levels: easy, moderate, and hard, and is represented by the average precision (AP). Among them, the IoU threshold for the Car category is 0.7, and the IoU thresholds for the Pedestrian and Cyclist categories are 0.5. The evaluation metric on the test set is to calculate the AP at 40 different recall thresholds (R40). In addition, we follow the general convention of previous studies (He et al., 2022; Shi et al., 2020; Zhou & Tuzel, 2018) and use the AP at 11 recall thresholds on the validation set to evaluate and compare the results of the classic single-stage benchmark networks.

The WOD (Sun et al., 2020) employs the AP and the average precision weighted by heading(APH) metric to comprehensively evaluate algorithm performance, with IoU thresholds set at 0.7 for vehicles, and 0.5 for cyclists and pedestrians, respectively. Unlike the KITTI(Geiger et al., 2012) dataset, which approximates AP calculations using 11-point and 40-point interpolation methods, WOD (Sun et al., 2020) computes AP based on the area under the Precision-Recall (P-R) curve. Furthermore, WOD (Sun et al., 2020) introduces the APH to assess the accuracy of object orientation by incorporating heading errors, thereby enabling a more holistic evaluation of algorithm capabilities. Both metrics consider two difficulty levels: Level 1 (boxes containing more than 5 LiDAR points) and Level 2 (boxes with at least 1 LiDAR point).

1.4.2. Implementation details

In voxel-based 3D detection methods on the KITTI dataset, single-stage detectors typically employ voxel sizes of [0.16m, 0.16m, 4m] and [0.32m, 0.32m, 4m] (He et al., 2022; Lang et al., 2019; Yan et al., 2018). Due to experimental hardware constraints, we modified the former to [0.18m, 0.18m, 4m], while two-stage detection generally adopts a finer resolution of [0.05m, 0.05m, 0.1m]. For the WOD, we set voxel dimensions to [0.32m, 0.32m, 6m]. For clarity, these configurations are abbreviated as V=0.05, V=0.18, and V=0.32 respectively. We systematically investigated the impact of varying k values in k-NN graph construction on VoxT-GNN's performance across these voxel configurations.

For single-stage detection on the KITTI dataset, our experimental findings recommend distinct parameter configurations tailored to specific scenarios. In car-only scenarios, optimal performance is achieved using either V=0.18 with K=10 or V=0.32 with K=3. For environments exclusively containing pedestrians and cyclists, the configuration V=0.18 paired with K=5 significantly enhances small-object detection capabilities. In complex multi-category scenarios involving cars, pedestrians, and cyclists, the unified setting of V=0.18 and K=5 effectively balances detection accuracy across all categories while maintaining priority on automotive targets.

Regarding two-stage detection on KITTI, our analysis suggests differentiated strategies based on scenario complexity. For car-centric applications, K=4 delivers efficient large-object recognition. In mixed-category environments, adopting K=5 preserves robust car detection while substantially improving pedestrian and cyclist identification. For institutions with sufficient computational resources, training specialized models using K=4 for cars and K=5 for vulnerable road users enables task-specific optimization, though this approach requires increased storage and processing capacity.

For single-stage detection on the Waymo Open Dataset (WOD), parameter optimization follows similar scenario-driven principles. Vehicle-only detection achieves peak performance with K=2, while pedestrian/cyclist-focused scenarios benefit most from K=3. In heterogeneous traffic environments containing vehicles, pedestrians, and cyclists, K=2 emerges as the preferred configuration for balanced multi-category detection. Advanced implementations under resource-abundant conditions may further employ dedicated models with K=2 for vehicles and K=3 for pedestrians/cyclists, thereby maximizing category-specific detection fidelity at the cost of doubled computational overhead.

1.4.2.1. Model setup

For the KITTI dataset, our preprocessing pipeline selects LiDAR point clouds within specific spatial bounds: points along the X-axis are constrained to [0m, 69.12m], the Y-axis to [-39.68m, 39.68m] (adjusted to [-39.60m, 39.60m] for the V=0.18 configuration), and the Z-axis to [-3m, 1m], with further segmentation using KITTI foreground image boundaries. For the WOD, we retain points within the X/Y-axis range of [-74.24m, 74.24m] and Z-axis limits of [-2m, 4m]. The architecture employs a four-layer VoxelFormer structure with iterative processing, where voxel dimensions along the X/Y-axes double progressively across layers, accompanied by feature channel expansions from 16 to 128 dimensions. Each VoxelFormer block integrates three core components: an encoder, a GnnFFN intermediate layer with three iterations of Point-GNN operators, and a decoder. Following established practices in Set Transformer (Lee et al., 2019) and VoxSeT (He et al., 2022), we assign each voxel region a set of 8-dimensional latent codes and implement PE modules with 64-channel bandwidth. Initial point states are represented by 7-dimensional features (x, y, z, r, δx, δy, δz), where (x, y, z) denote 3D coordinates, r reflects intensity values, and (δx, δy, δz) represent deviations from the voxel centroid. These raw features undergo transformation via a MLP to generate 16-dimensional embeddings for subsequent processing.

1.4.2.2. Training and inference

In single-stage detection implementations, our method underwent end-to-end training for 120 epochs on an RTX 4090 GPU using the Adam optimizer. For the WOD, the training protocol was reduced to 65 epochs. The learning rate was initialized at 0.015 with one-cycle policy decay, while momentum parameters were regulated within a range of [0.85, 0.95]. Additional hyperparameters included a batch size of 2 and weight decay coefficient of 0.01. For two-stage detection, we extended VoxT-GNN by integrating it with Voxel R-CNN (Deng et al., 2021), where each voxel sampled 8 points during feature aggregation. The training regimen employed 115 epochs, with remaining parameters inherited from the baseline Voxel R-CNN configuration.

Our implementation inherits anchor configuration strategies from established frameworks while maintaining category-specific parameterization. For single-stage detection, we adopt the anchor settings from SECOND (Yan et al., 2018), whereas the two-stage detection framework utilizes the anchor configuration of Voxel R-CNN (Deng et al., 2021). Crucially, both detection paradigms employ identical anchor specifications and matching criteria across all object categories (cars, pedestrians, and cyclists), with parameterization details outlined as follows:

Car Detection. The dimensions of the anchor are defined as follows: length $l^a = 3.9$ meters, width $w^a = 1.6$ meters, and height $h^a = 1.56$ meters. The criteria for matching anchors are established as follows: An anchor is classified as a positive sample if it exhibits the highest IoU with any ground truth box, or if its IoU exceeds 0.6. Conversely, an anchor is deemed a negative sample if its IoU with all ground truths falls below 0.45. Anchors that have an IoU in the range of $0.45 \leq IoU \leq 0.6$ are disregarded.

Pedestrian Detection. The dimensions of the anchor for pedestrian detection are specified as follows: length $l^a = 0.8$ meters, width $w^a = 0.6$ meters, and height $h^a = 1.73$ meters.

Cyclist Detection. For cyclist detection, the dimensions of the anchor are set to: length $l^a = 1.76$ meters, width $w^a = 0.6$ meters, and height $h^a = 1.73$ meters.

The matching criteria for anchors in both pedestrian and cyclist detection remain consistent: An anchor is classified as a positive sample if it exhibits the highest IoU with any ground truth box, or if its IoU exceeds 0.5; conversely, it is classified as a negative sample when its IoU with every ground truth is less than 0.35; anchors exhibiting an IoU within the range of $0.35 \leq IoU \leq 0.5$ will be ignored.

For the WOD, the anchor dimensions and matching protocols for cars, pedestrians, and cyclists strictly adhere to the official standardized criteria. All remaining hyperparameters inherit default configurations from the OpenPCDet (Openpcdet, 2020) toolbox.

1.4.2.3. Data augmentations

To mitigate overfitting risks, we implement a suite of conventional data augmentation techniques, including global scaling, random global rotation, horizontal flipping, and coordinate perturbation, following established practices in prior works (Chen et al., 2019; Shi & Rajkumar, 2020; Yan et al., 2018; Zhou & Tuzel, 2018).

1.4.3. KITTI dataset results

1.4.3.1. Single-stage 3D detection results

Currently, many 3D object detection algorithms either rely on multi-modal data (which incurs high costs), focus solely on car detection, or employ inconsistent recall thresholds (some use 11 while others adopt 40). To enable more objective comparisons, we evaluated the Average Precision (AP) of VoxT-GNN's single-stage detection framework at 11 recall thresholds on the KITTI validation set. We conducted comprehensive comparisons with classic single-stage baseline algorithms that exclusively utilize point cloud data while simultaneously detecting three categories: Car, Pedestrian, and Cyclist, as demonstrated in Table 1. Although Point-GNN (He et al., 2022) does not report evaluation data for pedestrian and cyclist detection, we incorporate it into the comparison of car detection results due to our adoption of its operator. The experimental results from the table show that:

To highlight the detection performance for pedestrians and cyclists, we set the number of neighbors K in the k-NN algorithm to 5 and the voxel size V to 0.18, training a model accordingly. This model was compared with other baseline methods, and the results demonstrated that it achieved a comprehensive lead with significant advantages in cyclist and pedestrian detection: in pedestrian detection across easy, moderate, and hard levels, our method outperformed by 6.61, 5.89, and 5.01 AP, respectively; in cyclist detection across the same difficulty levels, our method led by 6.68, 2.02, and 0.56 AP, respectively. In car detection, although slightly inferior to the high-performing baseline SECOND (Yan et al., 2018), the gap was minimal, and the detection performance was essentially on par with the leading methods.

To enhance the detection performance for cars, we set the number of neighbors K in the k-NN algorithm to 10 and the voxel size V to 0.18, and trained a model. When comparing this model with other baseline methods, the results showed that in car detection, our method achieved the best performance in all but the moderate-level car detection, where it was slightly inferior to SECOND (Yan et al., 2018). Additionally, our method also demonstrated strong competitiveness in pedestrian and cyclist detection, securing leading positions in pedestrian detection at the easy level and in cyclist detection at both the easy and moderate levels.

In summary, for simple scenarios primarily involving cars, such as highways and expressways in China, a model can be trained with K=10 and V=0.18 to highlight high-quality car detection. For complex scenarios, such as urban traffic, a model can be trained with K=5 and V=0.18 to adapt to the detection of objects of various sizes and categories. This demonstrates that our method, by adjusting the value of K, achieves a controllable global receptive field that enhances the ability to learn features of small objects while ensuring the detection performance of large objects. Moreover, it effectively captures more discriminative features for small objects.

1.4.3.2. Two-stage 3D detection results

To further validate the effectiveness of 3D feature learning in VoxT-GNN, we extended it to a two-stage detection framework and conducted extensive experiments. The results were submitted to the KITTI official server, achieving performance on the 3D object detection benchmark with 40 recall thresholds in the test set. We compared our method with state-of-the-art approaches that use only LiDAR point clouds and cover at least two detection categories, as shown in Table 2. We extended VoxT-GNN by integrating it with the RoI head of Voxel R-CNN (Deng et al., 2021) to form a two-stage 3D detector. The hyperparameter K was set to 4 and 5 respectively to train different models.

When K=5, our method demonstrates strong competitiveness in car and cyclist detection, achieving leading performance in cyclist detection with improvements of 0.86 and 1.09 AP over state-of-the-art methods under easy and hard difficulty levels, respectively. For pedestrian detection, although there remains a performance gap compared to several strong baselines such as STD (Yang et al., 2019), PointPillars (Lang et al., 2019), and Part-A^2 (Shi et al., 2020), our method attains comparable performance to most existing

Table 1
The Average Precision (AP) with 11 recall thresholds for 3D object detection is compared with the classic single-stage baseline on the KITTI validation set. The top 2 methods are highlighted in bold and underlined respectively.

Method	Reference	Modality	Car			Pedestrian			Cyclist		
			Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard
VoxelNet(Zhou & Tuzel, 2018)	CVPR	LiDAR	81.97	65.46	62.85	57.86	53.42	48.87	67.17	47.65	45.11
SECOND(Yan et al., 2018)	Sensors		88.61	**78.62**	77.22	56.55	52.98	47.73	80.58	67.15	63.1
PointPillars(Lang et al., 2019)	CVPR		86.46	77.28	74.65	57.75	52.29	47.9	80.04	62.61	59.52
Point-gnn(Shi & Rajkumar, 2020)	CVPR		87.89	78.34	77.38	-	-	-	-	-	-
VoxSeT(He et al., 2022)	CVPR		88.45	78.48	77.07	60.62	54.74	50.39	84.07	68.11	65.14
VoxT-GNN (K=5, V=0.18)	-	LiDAR	88.32	77.73	76.74	**67.23**	**60.63**	**55.40**	**90.75**	**70.13**	**65.70**
VoxT-GNN (K=10, V=0.18)			**88.88**	78.54	**77.60**	61.31	54.22	49.58	86.25	69.09	64.90

Table 2
The AP with 40 recall thresholds for 3D object detection is compared with state-of-the-art LiDAR-only methods on the KITTI test set. All methods cover at least 2 categories of detection. The top three methods are emphasized in bold, underline, and italic respectively. The mAP is mean Average Precision.

Method	Reference	Modality	Car			Pedestrian			Cyclist			mAP
			Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard	
VoxelNet(Zhou & Tuzel, 2018)	CVPR	LiDAR	77.47	65.11	57.73	39.48	33.69	33.69	61.22	48.36	44.37	51.24
SECOND(Yan et al., 2018)	Sensors		83.13	73.66	66.20	51.07	42.56	37.29	70.51	53.85	46.90	58.35
PointPillars(Lang et al., 2019)	CVPR		79.05	74.99	68.30	*52.08*	*43.53*	**41.49**	75.78	59.07	52.92	60.80
PointRCNN(Shi et al., 2019)	CVPR		86.96	75.64	70.70	47.98	39.37	36.01	74.96	58.82	52.53	60.33
STD(Yang et al., 2019)	ICCV		87.95	79.71	75.09	**53.29**	42.47	38.35	78.69	61.59	55.30	63.60
Part-A^2(Shi et al., 2020)	TPAMI		87.81	78.49	73.51	53.10	43.35	40.06	78.69	61.59	55.30	63.54
Voxel-RCNN(Deng et al., 2021)	AAAI		88.09	80.99	76.50	47.91	40.57	38.21	76.42	62.01	55.94	62.96
PV-RCNN(Shi et al., 2020)	CVPR		*90.25*	81.43	76.82	*52.17*	43.29	*40.29*	78.60	63.71	57.65	64.91
Point-GNN(Shi & Rajkumar, 2020)	CVPR		88.33	79.47	72.29	51.92	**43.77**	40.14	78.60	63.48	57.08	63.90
PDV(Hu et al., 2022)	CVPR		90.43	*81.86*	**77.36**	47.80	40.56	38.46	*83.04*	67.81	60.46	*65.31*
IA-SSD(Zhang et al., 2022)	CVPR		88.87	80.32	75.10	49.01	41.20	38.03	80.78	66.01	58.12	64.16
SVGA-Net(He et al., 2022)	AAAI		87.33	80.47	75.91	48.48	40.39	37.92	78.58	62.28	54.88	62.92
PV-RCNN++(Shi et al., 2023)	IJCV		90.14	81.88	*77.15*	-	-	-	82.22	67.33	60.04	-
DFAF3D(Tang et al., 2023)	IVC		88.59	79.37	*72.21*	47.58	40.99	37.65	82.09	65.86	59.02	63.71
BSAODet(W. et al., 2023)	TCSVT		88.89	81.74	77.14	51.71	43.63	41.09	82.65	67.79	60.26	**66.10**
PG-RCNN(Koo et al., 2023)	ICCV		89.38	**82.13**	77.33	-	-	-	82.77	*67.82*	61.25	-
PV-GNN_Cyc&Ped(Fei et al., 2024)	Research Square (PREPRINT)					48.78	42.00	36.91	78.58	62.54	55.28	
PASS-PV-RCNN++(Chen & Zhang & Zheng, 2024)	arXiv		87.65	81.28	76.79	47.66	41.95	38.9	80.43	68.45	60.93	64.89
VoxT-GNN (combined with Voxel R-CNN, K=5)	-	LiDAR	90.14	81.35	76.70	47.56	40.51	38.29	83.90	**68.50**	**62.02**	65.44
VoxT-GNN (combined with Voxel R-CNN, K=4)			**90.67**	81.70	77.07	44.00	38.57	36.52	**83.93**	*68.40*	*61.21*	64.67

approaches. Compared with Voxel R-CNN (Deng et al., 2021), while no significant improvement is observed in pedestrian detection, our 3D feature learning approach substantially enhances car and cyclist detection performance, particularly achieving AP gains of 7.48, 6.49, and 6.08 in cyclist detection under easy, moderate, and hard difficulty levels, respectively. Notably, our method achieves the second-highest overall mean average precision(mAP) of 65.44 across all categories. When K=4, compared to K=5, the model exhibits better car detection performance and comparable cyclist detection capability but inferior pedestrian detection results. Therefore, we recommend employing K=4 for training in single-category scenarios dominated by cars, while setting K=5 is preferable for multi-scale and multi-category complex detection scenarios.

The comparative analysis of experimental results demonstrates that, regardless of whether K=5 or K=4 is employed, our proposed feature learning mechanism with an adaptively variable global receptive field not only ensures high-quality detection for large-scale objects like cars, but also maintains competitive performance in comprehensive detection across objects of varying sizes and categories. Notably, this approach demonstrates superior capability in capturing stronger discriminative features for small-scale cyclist objects.

Furthermore, our LiDAR-only approach is comprehensively compared with state-of-the-art multi-modal methods across car, pedestrian, and cyclist detection categories, as detailed in Table 3. Experimental results demonstrate that our method outperforms numerous two-stage detection approaches in car and cyclist detection, achieving third place in car detection at the easy difficulty level. This indicates that our LiDAR-exclusive method exhibits competitive advantages compared to many multi-modal counterparts. Through subsequent improvements, our approach holds substantial potential to further narrow the performance gap with existing multi-modal state-of-the-art methods and may even surpass them in specific detection scenarios.

1.4.3.3. Comparison of 3D detection results among similar models

Research on 3D object detection using Graph Neural Network (GNN) concepts is one of the significant directions in the field. To differentiate from these methods, we selected PV-GNN(Fei et al., 2024), GVFNet(Li et al., 2023), and SVGA-Net(He et al., 2022), which are also relevant to this study and use the KITTI dataset, for a comprehensive comparison, as shown in Table 4. Since PV-GNN trains separate models for car detection and for pedestrian and cyclist detection, we present their results separately. The principle of PV-GNN involves first-stage feature learning using conventional VoxelNet(Zhou & Tuzel, 2018) and PointNet(Qi et al., 2017, 2017), followed by local graph construction at keypoints within each region proposal and GNN-based point cloud feature refinement in the second stage. Compared with PV-GNN: our method shows comparable performance (slightly inferior but within the same detection tier) for car detection; achieves comparable results with minor mutual advantages in pedestrian detection; and demonstrates significant superiority in cyclist detection, outperforming by 5.32, 5.96, and 6.74 AP in easy, moderate, and hard levels respectively. This substantial advantage in cyclist detection may stem from our enhanced feature learning approach: our first-stage 3D feature learning integrates cross-attention mechanisms for local feature extraction and voxel-level GNN for global feature learning, surpassing PV-GNN's feature learning strategy. GVFNet and SVGA-Net both adopt pseudo-voxel concepts, selecting M points as cloud centers and constructing spherical voxels by gathering fixed-radius neighbors, then designing corresponding graphs for GNN-based feature learning. Compared with GVFNet: our method shows marked superiority in car detection, leading by 4.71, 4.23, and 4.50 AP across three difficulty levels (comparisons unavailable for pedestrian/cyclist detection due to missing GVFNet results). Compared with SVGA-Net: our approach leads by 2.81, 0.88, and 0.79 AP in car detection across difficulty levels; shows comparable pedestrian detection performance; and achieves significant cyclist detection advantages of 5.32, 6.22, and 7.14 AP respectively. The overall superiority over both methods originates from fundamental limitations in pseudo-voxel construction: the inherent sparsity, irregularity, and variable density of point clouds make optimal determination of pseudo-voxel centers (M) and sampling counts challenging, degrading feature quality. In contrast, our method preserves traditional voxel partitioning while implementing parallelizable cross-attention mechanisms within voxels to maximize point retention and maintain original cloud structure for superior local feature learning. Concurrently, our global graph construction between voxel regions enables efficient cross-voxel information exchange through GNNs, effectively capturing long-range contextual relationships in point clouds.

1.4.4. WOD results

Due to the massive scale of the WOD, which contains over 20 times more data volume than the KITTI dataset and exhibits approximately 6 times larger spatial coverage per-frame point cloud compared to KITTI, our study faces computational constraints with limited hardware resources (only one RTX 4090 GPU). Following the common practice established in OpenPCDet (Openpcdet, 2020), we adopt approximately 20% of the training samples for model training. The evaluation is conducted using official Waymo evaluation metrics, where both AP and APH are calculated across the entire validation set at two difficulty levels.

To validate the generalization capability of our method on the WOD, we compared it with classical single-stage 3D detection baselines evaluated on 20% of the training set. As shown in Table 5, except for our evaluation results, other methods' data are sourced from the official OpenPCDet (Openpcdet, 2020) repository. The experimental results demonstrate that our VoxT-GNN achieves competitive performance across all 3D detection categories. Notably, it outperforms existing methods in vehicle and pedestrian detection, exhibiting significant advantages in pedestrian AP metrics. Specifically, it surpasses CenterPoint (Yin et al., 2021) by 2.85 and 3.04 in Level 1 and Level 2 AP metrics for pedestrian detection, respectively. For cyclist detection, VoxT-GNN ranks second, closely following CenterPoint (Yin et al., 2021), while outperforming SECOND (Yan et al., 2018) and PointPillars (Lang et al., 2019) baselines. These comparative experiments on the WOD validation set highlight the strong generalization capability of VoxT-GNN in complex 3D detection scenarios involving diverse object sizes and categories. Its superior performance in vehicle and pedestrian detection can be attributed to two key designs:

Table 3
The AP with 40 recall thresholds for 3D object detection is compared with state-of-the-art multi-modal methods on the KITTI test set. All methods cover at least 2 categories of detection. The top three methods are emphasized in bold, underline, and italic respectively.

Method	Reference	Modality	Car			Pedestrian			Cyclist			mAP
			Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard	
F-PointNet (Qi et al., 2018)	CVPR	RGB + LiDAR	82.19	69.79	60.59	50.53	42.15	38.08	72.27	56.12	49.01	57.86
F-ConvNet (Wang & Jia, 2019)	IROS		87.36	76.39	66.69	52.16	43.38	38.80	*81.98*	*65.07*	56.54	63.15
AVOD-FPN (KuKu et al., 2018)	IROS		83.07	71.76	65.73	50.46	42.27	39.04	63.76	50.55	44.93	56.84
CLOCs PVCas (Pang et al., 2020)	IROS		88.94	80.67	77.15	47.30	39.42	36.97	77.33	62.02	55.52	62.81
Fast-CLOCs-PV (Pang et al., 2022)	WACV		89.11	80.34	76.98	52.10	42.72	39.08	82.83	65.31	57.43	65.1
HMFI (Li et al., 2022)	ECCV		88.90	81.93	77.30	50.88	42.65	39.78	*84.02*	70.37	62.57	66.49
CAT-DET (Zhang et al., 2022)	CVPR		89.87	81.32	76.68	54.26	45.44	41.94	83.68	68.81	61.65	67.07
UPIDet (Zhang et al., 2024)	NIPS		89.13	82.97	80.05	55.59	48.77	46.12	86.74	**74.32**	67.45	70.13
MLF-DET (Lin et al., 2023)	ICANN		91.18	*82.89*	*77.89*	50.86	45.29	42.05	83.31	*70.71*	*63.71*	*67.54*
TED-M (Wu et al., 2023)	AAAI		**91.61**	**85.28**	**80.68**	**55.85**	**49.21**	**46.52**	**88.82**	74.12	**66.84**	**71.00**
VPFNet (Wang et al., 2024)	IT-ITS		88.51	76.39	76.74	*54.65*	*48.36*	*44.98*	77.64	64.10	58.00	65.49
VoxT-GNN (combined with Voxel R-CNN, K=5)	-	LiDAR	*90.14*	81.35	76.70	47.56	40.51	38.29	83.90	68.50	62.02	65.44

Table 4
The AP with 40 recall thresholds for two-stage 3D object detection is compared with methods that have strong relevance on the KITTI test set.

Method	Car			Pedestrian			Cyclist			mAP
	Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard	
PV-GNN_Car (Fei et al., 2024)	**91.64**	**82.49**	**77.28**	-	-	-	-	-	-	-
PV-GNN_Ped&Cyc (Fei et al., 2024)	-	-	-	**48.78**	**42.00**	36.91	78.58	62.54	55.28	-
GVFNet (Li et al., 2023)	85.43	77.12	72.20	-	-	-	-	-	-	-
SVGA-Net (He et al., 2022)	87.33	80.47	75.91	48.48	40.39	37.92	78.58	62.28	54.88	62.92
Ours(combined with Voxel R-CNN, K=5)	90.14	81.35	76.70	47.56	40.51	**38.29**	**83.90**	**68.50**	**62.02**	65.44

Table 5
A comparative evaluation of VoxT-GNN against classical baseline methods for single-stage 3D detection on the WOD validation set. The top two methods are emphasized in bold and underline respectively.

Method	Vec_L1		Vec_L2		Ped_L1		Ped_L2		Cyc_L1		Cyc_L2	
	AP/APH		AP/APH		AP/APH		AP/APH		AP/APH		AP/APH	
SECOND (Yan et al., 2018)	70.96	70.34	62.58	62.02	65.23	54.24	57.22	47.49	57.13	55.62	54.97	53.53
PointPillars (Lang et al., 2019)	70.43	69.83	62.18	61.64	66.21	46.32	58.18	40.64	55.26	51.75	53.18	49.80
CenterPoint (Yin et al., 2021)	71.33	70.76	63.16	62.65	72.09	65.49	64.27	**58.23**	**68.68**	**67.39**	**66.11**	**64.87**
VoxT-GNN (Ours)	**72.21**	**71.67**	**64.02**	**63.53**	**75.75**	**65.89**	**67.31**	**58.35**	68.22	66.91	65.74	64.47

1. Local Fine-Grained Feature Capture: The reduced cross-attention mechanism within voxel regions effectively extracts fine-grained local information from point clouds.
2. Global Feature Learning: GNNs enable cross-voxel information propagation and interaction, facilitating the learning of deeply embedded global 3D features.

This combination validates the method's generalization potential for 3D perception tasks, particularly excelling in scenarios requiring both local detail preservation and global context modeling.

Furthermore, we compared our single-stage VoxT-GNN detector with several classical two-stage detectors under identical training settings (approximately 20% of the WOD training set). As shown in Table 6, the evaluation results of two-stage baselines are sourced from the official OpenPCDet (Openpcdet, 2020) repository. Experimental results demonstrate that our single-stage VoxT-GNN achieves second-ranked performance in pedestrian and cyclist detection, significantly outperforming Part-A2-Anchor(Shi et al., 2020) and PV-RCNN(Shi et al., 2020). This highlights not only the strong generalization capability of VoxT-GNN but also its superior performance in detecting small-to-medium-sized objects. These improvements validate that the adaptive receptive field design effectively identifies optimal K-values to balance detection accuracy across varying object scales, enabling robust performance in multi-scale, multi-category scenarios. However, a performance gap remains in vehicle detection compared to two-stage methods, primarily due to two factors:

1. Inherent Single-Stage Limitations: While single-stage detectors generally underperform two-stage counterparts in complex tasks, the latter involve higher computational complexity. Due to experimental constraints, we were unable to implement full-scale two-stage training on WOD.
2. Parameter Sensitivity: Key parameters (latent code length, K-value selection, and voxel size) critically influence the receptive field adaptability. Limited by computational resources, we could not exhaustively test all parameter combinations to optimize vehicle detection.

Extensive experiments on WOD validate the effectiveness and versatility of VoxT-GNN in 3D detection tasks. Its ability to handle

Table 6
A comparative analysis between the single-stage detection results of VoxT-GNN and classical two-stage baseline methods on the WOD validation set. The top two methods are emphasized in bold and underline respectively.

Method	Scheme	Vec_L1		Vec_L2		Ped_L1		Ped_L2		Cyc_L1		Cyc_L2	
		AP/APH		AP/APH		AP/APH		AP/APH		AP/APH		AP/APH	
Part-A2-Anchor (Shi et al., 2020)	two-stage	74.66	74.12	65.82	65.32	71.71	62.24	62.46	54.06	66.53	65.18	64.05	62.75
PV-RCNN (Shi et al., 2020)	two-stage	75.41	74.74	67.44	66.80	71.98	61.24	63.70	53.95	65.88	64.25	63.39	61.82
PV-RCNN++(Shi et al., 2023)	two-stage	**77.82**	**77.32**	**69.07**	**68.62**	**77.99**	**71.36**	**69.92**	**63.74**	**71.80**	**70.71**	**69.31**	**68.26**
VoxT-GNN	single-stage	72.21	71.67	64.02	63.53	75.75	65.89	67.31	58.35	68.22	66.91	65.74	64.47

small/medium objects while maintaining competitive performance across diverse scenarios underscores its practical value for real-world applications requiring multi-scale perception. Future work will focus on refining parameter optimization strategies and expanding the framework to bridge the performance gap in vehicle detection.

1.4.5. Ablation study

We conducted a series of ablation studies primarily on the KITTI dataset (supplemented by partial experiments on WOD) to analyze the role of distinct components in VoxT-GNN. These experiments systematically dissect the contributions of key architectural elements, providing insights into how each module enhances the model's overall detection performance.

1.4.5.1. The impact of varying k values on the construction of k-NN graphs in single-stage detection

In single-stage 3D detection experiments on the KITTI dataset, we evaluated the impact of varying K-values under two voxel sizes (V=0.18 and V=0.32), with results summarized in Tables 7 and 8. The analysis reveals the following insights:

When V = 0.18: For car detection, the range and mean absolute deviation (MAD) values indicate that varying K-values have minimal impact on detection performance, allowing greater flexibility in parameter selection, though K=10 yields relatively optimal results; For pedestrian detection, both range and MA values reveal moderate sensitivity to K-values, with K=2, 5, and 7 achieving better performance, where K=5 consistently excels across all difficulty levels; Regarding cyclist detection, while K=1 produces abnormally high range values due to insufficient detection accuracy (indicating outlier behavior), other K-values exhibit comparable performance, with K=2 and 5 demonstrating superiority, particularly K=5 showing the best results across all difficulty levels.

Based on this experimental analysis under V=0.18, we recommend: 1) For car-only scenarios, K=10 is preferred for balanced performance; 2) For complex multi-category environments containing cars, pedestrians, and cyclists, K=5 optimizes pedestrian/cyclist detection while maintaining car detection quality; 3) With sufficient computational and storage resources, dedicated models using K=10 (car detection) and K=5 (pedestrian and cyclist detection) could be trained separately. These recommendations emphasize scenario-specific K-value optimization to address heterogeneous detection requirements.

When V=0.32: For car detection, the range and MA values suggest minimal sensitivity to K-values, allowing flexible parameter selection, though K=3 or 6 yields relatively superior performance; For pedestrian detection, both range and MA values demonstrate significant sensitivity to K-values, with K=10 achieving optimal results across all difficulty levels; Similarly, cyclist detection exhibits strong dependence on K-values, particularly under moderate difficulty levels, where K=4 consistently outperforms others.

Based on experimental analysis under V=0.32, we propose: 1) For single-category scenarios (cars, pedestrians, or cyclists individually), adopt K=3/6 (vehicles), K=10 (pedestrians), and K=4 (cyclists) for category-specific optimization; 2) For complex multi-category environments containing all three object types, prioritize K=10 to maximize pedestrian detection performance while maintaining acceptable car and cyclist detection accuracy; 3) With sufficient computational and storage resources, train specialized models with K=3/6 (car), K=10 (pedestrian), and K=4 (cyclist) to address heterogeneous detection requirements. These guidelines highlight the necessity of adaptive K-value configuration aligned with scenario complexity and resource availability.

These experiments validate that adaptive K-value selection enables flexible adjustment of the receptive field, effectively addressing diverse 3D detection demands in both simple and complex scenarios. By tuning K-values alongside voxel sizes, the method achieves robust performance across varying object scales and categories, demonstrating its adaptability to real-world applications requiring multi-scale perception.

Additionally, we conducted a comparative analysis of single-stage detection performance under different K values on the WOD, as

Table 7
The AP with 11 recall thresholds for the comparison of 3D object detection with VoxT-GNN across different K values, utilizing only point clouds from the KITTI validation set. V=0.18m. The best value is enhanced with bold.

Voxel size	K	Car			Pedestrian			Cyclist		
		Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard
V=0.18	K=1	86.38	77.09	76.11	63.28	56.54	51.62	59.93	49.76	47.05
	K=2	88.02	77.83	76.93	*63.9*	*57.79*	52.72	*87.21*	*69.37*	65.57
	K=3	88.12	77.70	76.62	*8* 63.57	56.65	51.93	86.85	69.06	*65.69*
	K=4	88.63	78.50	77.51	60.99	56.12	51.59	85.37	65.73	63.34
	K=5	88.32	77.73	76.74	**67.23**	**60.63**	**55.40**	**90.75**	**70.13**	**65.70**
	K=6	88.69	78.29	77.08	61.36	56.04	51.46	86.40	68.21	64.41
	K=7	*88.77*	78.36	77.59	63.97	57.61	*52.96*	86.24	68.04	63.80
	K=8	88.67	*78.57*	77.50	62.48	57.33	52.67	87.03	68.75	64.60
	K=9	88.67	**78.60**	**77.62**	61.62	56.07	50.78	85.39	68.37	64.29
	K=10	**88.88**	78.54	*77.60*	61.31	54.22	49.58	86.25	69.09	64.90
	Range	2.5	1.51	1.51	6.24	6.41	5.82	30.82	20.37	18.65
	Avg	88.32	78.12	77.13	62.98	56.90	52.07	84.14	66.65	62.94
	Mean Absolute Deviation (MAD)	0.48	0.43	0.43	1.43	1.15	1.09	4.84	3.56	3.18
	Best K=10(Car) K=5(Pedestrian, Cyclist)	88.88	78.54	77.60	67.23	60.63	55.40	90.75	70.13	65.70

Table 8
The AP with 11 recall thresholds for the comparison of 3D object detection with VoxT-GNN across different K values, utilizing only point clouds from the KITTI validation set. V=0.32m.The best value is enhanced with bold.

Voxel size	K	Car			Pedestrian			Cyclist		
		Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard
V=0.32	K=1	*88.68*	78.38	77.25	*61.0*	*56.53*	*51.82*	82.88	66.62	63.49
	K=2	88.40	78.06	77.05	*8* 57.49	53.33	48.35	84.13	65.64	62.52
	K=3	**88.87**	**78.54**	77.23	60.38	56.14	51.34	83.16	67.07	63.11
	K=4	87.91	78.02	77.09	60.43	56.32	51.44	**85.61**	**70.66**	**66.11**
	K=5	88.14	78.36	*77.3*	60.08	55.36	50.23	85.08	65.62	63.02
	K=6	88.53	**78.54**	*6* **77.49**	59.27	54.05	49.03	82.59	67.03	63.13
	K=7	88.22	78.16	77.11	60.08	56.11	50.31	85.21	69.00	*65.49*
	K=8	88.49	*78.51*	77.32	60.73	56.42	50.92	84.49	69.86	65.43
	K=9	88.51	78.36	77.34	58.32	54.33	49.13	84.78	*70.24*	65.05
	K=10	88.35	78.34	77.21	**61.28**	**56.76**	**51.95**	*85.32*	69.75	65.57
	Range	0.96	0.52	0.44	3.79	3.43	3.60	3.02	5.04	3.59
	Avg	88.41	78.33	77.25	59.91	55.54	50.45	84.33	68.15	64.30
	Mean Absolute Deviation (MAD)	0.26	0.15	0.11	0.93	1.01	1.04	0.91	1.75	1.24
	Best K=3(Car) K=10(Pedestrian) K=4(Cyclist)	88.87	78.54	77.23	61.28	56.76	51.95	85.61	70.66	66.11

detailed in Tables 9 and 10. Due to the large size of single-frame data in WOD and hardware constraints (a single RTX 4090 GPU), the maximum K value was limited to 4. Experimental results demonstrate that for vehicle, pedestrian, and cyclist detection: 1) Performance at K=4 is significantly inferior to other K values; 2) Detection metrics for K=1, 2, and 3 are comparable, belonging to the same performance tier; 3) K=2 and K=3 show superior performance with complementary advantages, though K=2 achieves the best average performance of 67.01 AP and 65.14 APH. These findings lead to the following conclusions: In WOD, excessively large K values (e.g., K=4) expand the global receptive field excessively, degrading 3D feature quality for objects of different categories and sizes. Appropriately smaller K values (neither too large nor too small) enable high-quality 3D feature extraction across diverse object types and scales. Based on empirical analysis, we recommend for WOD single-stage 3D detection: 1) K=2 for vehicle-only scenarios; 2) K=3 for pedestrian/cyclist-exclusive scenarios; 3) K=2 for complex multi-category environments; 4) With sufficient computational resources, dedicated models with K=2 (vehicle detection) and K=3 (pedestrian and cyclist detection) could be trained separately. These conclusions further validate that adaptive receptive field adjustment through K-value optimization effectively addresses diverse 3D detection requirements across scenarios.

1.4.5.2. The impact of varying k values on the construction of k-NN graphs in two-stage detection

In two-stage detection, we also conducted experiments on different K values using the KITTI test set, as shown in Table 11. Due to experimental constraints, the maximum K value that could be set was 6. We analyzed the impact of different K values on detection performance. For car detection, K values of 3, 4, and 5 achieved relatively better performance for the easy level, while changes in K had minimal impact on the moderate and difficult levels. For pedestrian detection, K = 5 demonstrated the best performance, with a noticeable gap compared to other K values. For cyclist detection, larger K values were preferred, with K = 5 achieving relatively better results. In terms of overall performance, K = 5 provided the best results, outperforming the second-best by 0.61 mAP. Based on these results, we recommend the following for two-stage detection: 1) For simple scenarios containing only cars, K = 4 is suitable; 2) For complex scenarios with cars, pedestrians, and cyclists, K = 5 is optimal, as it ensures good car detection while achieving better performance for pedestrians and cyclists; 3) If sufficient computational and storage resources are available, separate models with K = 4 for cars and K = 5 for pedestrians and cyclists. The experimental evidence confirms our method's capability to address heterogeneous object scale challenges through intelligent parameter adaptation, effectively serving both simple and complex 3D detection scenarios.

1.4.5.3. The impact of voxel size

The comparison between the optimal detection schemes and average values at V=0.18 and V=0.32 is shown in Table 12. For car detection, the voxel size shows minimal impact on detection performance, whether evaluated through average values or optimal

Table 9
The AP for single-stage 3D object detection is compared with different K values of VoxT-GNN only from point clouds on the WOD validation set. The top two methods are emphasized in bold and underline respectively.

K	Vec_L1(AP)	Vec_L2(AP)	Ped_L1(AP)	Ped_L2(AP)	Cyc_L1(AP)	Cyc_L2(AP)	AVG
K=4	63.89	55.88	70.01	61.91	65.09	62.71	63.25
K=3	<u>71.90</u>	<u>63.79</u>	<u>75.40</u>	<u>66.98</u>	**68.23**	<u>65.72</u>	<u>68.67</u>
K=2	**72.21**	**64.02**	**75.75**	**67.31**	<u>68.22</u>	**65.74**	**68.88**
K=1	71.22	62.77	74.76	66.18	67.68	65.22	67.97

Table 10
The APH for single-stage 3D object detection is compared with different K values of VoxT-GNN only from point clouds on the WOD validation set. The top two methods are emphasized in bold and underline respectively.

K	Vec_L1(APH)	Vec_L2(APH)	Ped_L1(APH)	Ped_L2(APH)	Cyc_L1(APH)	Cyc_L2(APH)	AVG
K=4	63.30	55.36	56.47	49.76	63.42	61.09	58.23
K=3	71.39	63.32	**65.95**	**58.41**	**66.96**	**64.50**	65.09
K=2	**71.67**	**63.53**	65.89	58.35	66.91	64.47	**65.14**
K=1	70.69	62.29	64.95	57.31	66.30	63.89	64.24

Table 11
The AP with 40 recall thresholds for the comparison of two-stage 3D object detection with VoxT-GNN across different K values on the KITTI test set.

K	Car			Pedestrian			Cyclist			mAP
	Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard	
K=6	88.54	81.63	**77.08**	*45.33*	39.99	37.91	*83.25*	**68.61**	*61.16*	64.83
K=5	*90.14*	*81.35*	76.70	**47.56**	**40.51**	**38.29**	83.90	68.50	**62.02**	**65.44**
K=4	**90.67**	**81.70**	77.07	44.00	*38.57*	36.52	**83.93**	*68.40*	61.21	*64.67*
K=3	90.22	81.22	76.75	45.30	38.46	*37.29*	81.15	67.11	60.60	64.23
K=2	88.05	81.23	*76.78*	45.72	38.26	37.24	81.96	67.54	61.06	64.20
K=1	88.12	80.83	76.44	43.25	37.90	35.61	80.58	66.30	59.55	63.18

detection schemes. Regarding pedestrian detection, the V=0.18 configuration demonstrates superior performance across all three difficulty levels. For cyclist detection, while the V=0.32 configuration achieves marginally better results in the moderate and hard levels under optimal detection schemes, the V=0.18 configuration generally exhibits better detection effectiveness across all three difficulty levels. Therefore, we recommend: For car detection, either V=0.18 with K=10 or V=0.32 with K=3 should be adopted; For pedestrian and cyclist detection, V=0.18 with K=5 is preferred; In complex scenarios containing cars, pedestrians, and cyclists, the configuration of V=0.18 with K=5 is recommended, and this setting maintains satisfactory car detection performance while achieving optimal detection results for pedestrians and cyclists.

1.4.5.4. The impact of voxelformer and GnnFFN components

In practical applications, since GnnFFN is an integrated module within VoxelFormer, this study omits separate ablation experiments for GnnFFN. We conducted single-stage detection ablation experiments on the KITTI validation set to validate the effectiveness of our designed 3D feature learning components, with results presented in Table 13. Our analysis reveals: (1) Regardless of voxel size (V=0.18 or V=0.32), the combined use of VoxelFormer and GnnFFN significantly improves detection accuracy, particularly demonstrating more pronounced enhancements in pedestrian and cyclist detection. This verifies the effectiveness of our 3D feature learning framework: the VoxelFormer encoder employs a reduced cross-attention mechanism for local feature learning within voxel regions, the GnnFFN middleware facilitates global feature interaction across-voxel regions, and the decoder achieves effective fusion of local-global features. (2) Using VoxelFormer without GnnFFN results in a marked decline in detection performance, especially more substantial degradation for pedestrians and cyclists. For example, under the voxel size V = 0.18, the detection performance for pedestrians and cyclists at the moderate difficulty level decreased by 4.60% and 6.30%, respectively. This underscores the critical role of cross-voxel information transfer implemented by GnnFFN, particularly for enhancing detection of medium/small-sized objects. (3) The absence of both components leads to severe performance deterioration, with particularly significant drops in pedestrian and cyclist detection. For example, at the voxel size of V = 0.18, the detection performance for pedestrians and cyclists at the moderate difficulty level decreased by 12.37% and 11.51%, respectively. These findings collectively demonstrate that our method achieves stronger feature for small objects while maintaining high-quality car detection.

1.4.5.5. Latency and runtime memory

We conducted comparative experiments on inference latency and runtime memory consumption between single-stage and two-stage detection frameworks, as summarized in Table 14. Compared to other methods, our VoxT-GNN exhibits slightly higher latency but remains within 60ms, ensuring real-time detection capabilities. However, it requires more runtime memory due to its integration of computationally intensive Transformer and GNN modules. This trade-off is justified by its superior detection accuracy, as demonstrated in Table 2. Overall, our method achieves a favorable balance between real-time inference and competitive detection performance. While the increased memory overhead reflects the complexity of the architecture, the enhancements in detection quality validate the design choices for scenarios prioritizing accuracy without sacrificing real-time requirements.

1.5. Conclusion

We present VoxT-GNN, a LiDAR-based 3D object detection method that effectively addresses small objects while maintaining detection performance. This novel voxel-domain framework integrates Transformer and GNN architectures, with its core concept revolving around modeling point cloud processing as region-to-region transformations to better preserve the structural information of

Table 12
The AP with 11 recall thresholds of 3D object detection comparison with Best and Avg value in different voxel size only from point clouds on the KITTI validation set. The avg of cyclist detection at voxel V=0.18 excluded the outlier when K=1.

Voxel Size	Item	Car			Pedestrian			Cyclist		
		Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard
V=0.18	Best	**88.88**	**78.54**	**77.60**	**67.23**	**60.63**	**55.40**	**90.75**	70.13	65.70
V=0.32	Best	88.87	**78.54**	77.23	61.28	56.76	51.95	85.61	**70.66**	**66.11**
Improvement		-0.01	0	-0.37	-5.95	-3.87	-3.45	-5.14	+0.53	+0.41
V=0.18	Avg	88.32	78.12	77.13	**62.98**	**56.90**	**52.07**	**86.83**	**68.53**	**64.70**
V=0.32	Avg	**88.41**	**78.33**	**77.25**	59.91	55.54	50.45	84.33	68.15	64.30
Improvement		+0.09	+0.21	+0.12	-3.07	-1.36	-1.62	-2.5	-0.38	-0.4

Table 13
The ablation results are reported using the AP with 11 recall thresholds for each component of the proposed VoxT-GNN on the KITTI validation set.

Voxel Size	Components		Car-3D			Pedestrian-3D			Cyclist-3D		
	VoxelFormer	GnnFNN	Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard
V=0.18	✓	✓	**88.88**	**78.54**	**77.60**	**67.23**	**60.63**	**55.40**	**90.75**	**70.13**	**65.70**
	✓		87.82	77.61	76.26	65.16	57.84	51.90	85.09	65.71	63.33
	×	×	87.79	77.30	71.42	59.07	53.13	48.40	81.55	62.06	58.78
V=0.32	✓	✓	**88.87**	**78.54**	**77.23**	**61.28**	**56.76**	**51.95**	**85.61**	**70.66**	**66.11**
	✓		87.59	77.79	76.57	59.89	54.57	48.82	82.50	63.62	62.28
	×	×	87.04	76.77	73.30	53.76	46.69	42.81	81.25	63.99	60.21

Table 14
The latency and runtime memory performance on the KITTI dataset were evaluated using an NVIDIA 4090 GPU.

Stage	Models	Latency	Memory (runtime)
Single-stage	SECOND (Yan et al., 2018)	23ms	637MB
	SECOND_IOU (Yan et al., 2018)	26ms	653MB
	PointPillars (Lang et al., 2019)	18ms	787MB
	VoxT-GNN(Ours)	48ms	1489MB
Two-stage	PointRCNN (Shi et al., 2019)	36ms	896MB
	PointRCNN_IOU (Shi et al., 2019)	36ms	901MB
	Part-A^2 (Shi et al., 2020)	44ms	929MB
	Part-A^2-Free (Shi et al., 2020)	47ms	673MB
	PV-RCNN (Shi et al., 2020)	58ms	1122MB
	Voxel R-CNN (Deng et al., 2021)	40ms	544MB
	VoxT-GNN combined with Voxel R-CNN(Ours)	59ms	2872MB

raw point clouds. The designed 3D feature learning module dynamically adjusts global receptive fields through the K-value configuration and identifies scene-adaptable K-values. Specifically, VoxelFormer enhances local feature extraction by sampling a larger number of points to maximally preserve point cloud geometry and capture fine-grained patterns, while GnnFNN employs k-NN graph construction to scale global receptive fields, thereby generating higher-quality global features for objects of varying sizes and categories. Extensive experiments on the KITTI dataset demonstrate that VoxT-GNN achieves competitive performance in both single-stage detection and extended two-stage frameworks, establishing itself as a promising alternative for point cloud modeling. In future work, we will optimize the Transformer and GNN architecture to further enhance detection capabilities.

Author statement

This manuscript is the authors' original work and has not been published nor has it been submitted simultaneously elsewhere. All authors have checked the manuscript and have agreed to the submission.

CRediT authorship contribution statement

Qiangwen Zheng: Writing – review & editing, Writing – original draft, Software, Methodology, Investigation, Formal analysis, Conceptualization. **Sheng Wu:** Writing – review & editing, Supervision, Methodology, Funding acquisition, Formal analysis, Conceptualization. **Jinghui Wei:** Visualization, Validation, Supervision, Data curation.

Data availability

Data will be made available on request.

References

Ai, L., Xie, Z., Yao, R., & Yang, M. (2024). MVTr: multi-feature voxel transformer for 3d object detection. *The Visual Computer, 40*(3), 1453–1466. https://doi.org/10.1007/s00371-023-02860-8

Bello, S. A., Yu, S., Wang, C., Adam, J. M., & Li, J. (2020). Review: deep learning on 3d point clouds. *Remote Sensing, 12*(11), 1729. https://doi.org/10.3390/rs12111729

Bo, L., Zhang, T., & Tian, X. (2016). Vehicle detection from 3d lidar using fully convolutional network. *Arxiv Preprint Arxiv:1608.07916*.

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., & Zagoruyko, S. (2020). End-to-end object detection with transformers. *European conference on computer vision* (pp. 213–229). Cham: Springer International Publishing.

Chen, C., Chen, Z., Zhang, J., & Tao, D. (2022). SASA: semantics-augmented set abstraction for point-based 3d object detection. *Proceedings of the AAAI Conference On Artificial Intelligence, 36*(1), 221–229. https://doi.org/10.1609/aaai.v36i1.19897

Chen, Q., Sun, L., Wang, Z., Jia, K., & Yuille, A. (2020). *Object as hotspots: an anchor-free 3d object detection approach via firing of hotspots*. Cornell University Library. ReportarXiv.org https://go.exlibris.link/XcsP2v8t.

Chen, S., Zhang, H., & Zheng, N. (2024). Leveraging anchor-based LiDAR 3d object detection via point assisted sample selection. *Arxiv Preprint Arxiv:2403.01978*.

Chen, X., Kundu, K., Zhang, Z., Ma, H., Fidler, S., & Urtasun, R. (2016). Monocular 3d object detection for autonomous driving. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2147–2156).

Chen, X., Ma, H., Wan, J., Li, B., & Xia, T. (2017). Multi-view 3d object detection network for autonomous driving. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition* (pp. 1907–1915).

Chen, Y., Liu, J., Zhang, X., Qi, X., & Jia, J. (2023). Voxelnext: fully sparse voxelnet for 3d object detection and tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 21674–21683).

Chen, Y., Liu, S., Shen, X., & Jia, J. (2019). Fast point r-CNN. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 9774–9783).

Corporation, N. (9-15). *NVIDIA documentation hub*. https://docs.nvidia.com/#all-documents.

Deng, J., Shi, S., Li, P., Zhou, W., Zhang, Y., & Li, H. (2021). *Voxel r-CNN: towards high performance voxel-based 3d object detection (Report No. 2374-3468)*. Cornell University Library. arXiv.org.

Dong, Y., Kang, C., Zhang, J., Zhu, Z., Wang, Y., Yang, X., … Zhu, J. (2023). Benchmarking robustness of 3d object detection to common corruptions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 1022–1032).

Dong, Z., Ji, H., Huang, X., Zhang, W., Zhan, X., & Chen, J. (2023). PeP: a point enhanced painting method for unified point cloud tasks. *Arxiv Preprint Arxiv: 2310.07591*. doi:10.48550/arxiv.2310.07591.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T.,... Gelly, S. (2020). An image is worth 16x16 words: transformers for image recognition at scale. *Arxiv Preprint Arxiv:2010.11929*.

Fan, L., Pang, Z., Zhang, T., Wang, Y., Zhao, H., Wang, F., … Zhang, Z. (2022). Embracing single stride 3d object detector with sparse transformer. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 8458–8468).

Fei, B., Yang, W., Chen, W., Li, Z., Li, Y., Ma, T., … Ma, L. (2022). Comprehensive review of deep learning-based 3d point cloud completion processing and analysis. *Ieee Transactions On Intelligent Transportation Systems, 23*(12), 22862–22883. https://doi.org/10.1109/TITS.2022.3195555

Fei, H., Zhao, J., Zhang, Z., Wang, H., & Huang, X. (2024). PV-GNN: point-voxel 3d object detection based on graph neural network. *PREPRINT (Version 1) Available at Research Square*. https://doi.org/10.21203/rs.3.rs-4598182/v1

Feng, M., Gilani, S. Z., Wang, Y., Zhang, L., & Mian, A. (2021). Relation graph network for 3d object detection in point clouds. *Ieee Transactions On Image Processing, 30*, 92–107. https://doi.org/10.1109/TIP.2020.3031371

Feng, X., Du, H., Fan, H., Duan, Y., & Liu, Y. (2023). Seformer: structure embedding transformer for 3d object detection. In *Proceedings of the AAAI Conference on Artificial Intelligence* (pp. 632–640).

Geiger, A., Lenz, P., & Urtasun, R. (2012). Are we ready for autonomous driving? The KITTI vision benchmark suite. In *2012 IEEE conference on computer vision and pattern recognition* (pp. 3354–3361).

Guan, T., Wang, J., Lan, S., Chandra, R., Wu, Z., Davis, L., & Manocha, D. (2022). M3DETR: multi-representation, multi-scale, mutual-relation 3d object detection with transformers. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision* (pp. 2293–2303).

Guo, M., Cai, J., Liu, Z., Mu, T., Martin, R. R., & Hu, S. (2021). Pct: point cloud transformer. *Computational Visual Media, 7*(2), 187–199.

Hawlader, F., Robinet, F., & Frank, R. (2024). Leveraging the edge and cloud for v2x-based real-time object detection in autonomous driving. *Computer Communications, 213*, 372–381. https://doi.org/10.1016/j.comcom.2023.11.025

He, C., Li, R., Li, S., & Zhang, L. (2022). Voxel set transformer: a set-to-set approach to 3d object detection from point clouds. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 8417–8427).

He, C., Zeng, H., Huang, J., Hua, X., & Zhang, L. (2020). Structure aware single-stage 3d object detection from point cloud. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 11873–11882).

He, Q., Wang, Z., Zeng, H., Zeng, Y., & Liu, Y. (2022). SVGA-net: sparse voxel-graph attention network for 3d object detection from point clouds. In *Proceedings of the AAAI Conference on Artificial Intelligence* (pp. 870–878).

Hoang, H. A., Bui, D. C., & Yoo, M. (2024). TSSTDet: transformation-based 3-d object detection via a spatial shape transformer. *Ieee Sensors Journal, 24*(5), 7126–7139. https://doi.org/10.1109/JSEN.2024.3350770

Hu, J. S., Kuai, T., & Waslander, S. L. (2022). Point density-aware voxels for lidar 3d object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 8469–8478).

Jing, R., Zhang, W., Li, Y., Li, W., & Liu, Y. (2024). Dynamic feature focusing network for small object detection. *Information Processing & Management, 61*(6), Article 103858.

Jing, R., Zhang, W., Liu, Y., Li, W., Li, Y., & Liu, C. (2024). An effective method for small object detection in low-resolution images. *Engineering Applications of Artificial Intelligence, 127*, Article 107206. https://doi.org/10.1016/j.engappai.2023.107206

Koo, I., Lee, I., Kim, S., Kim, H., Jeon, W., & Kim, C. (2023). PG-RCNN: semantic surface point generation for 3d object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (pp. 18142–18151).

Ku, J., Mozifian, M., Lee, J., Harakeh, A., & Waslander, S. L. (2018). Joint 3d proposal generation and object detection from view aggregation. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 1–8).

Kuang, H., Wang, B., An, J., Zhang, M., & Zhang, Z. (2020). Voxel-FPN: multi-scale voxel feature aggregation for 3d object detection from LIDAR point clouds. *Sensors, 20*(3), 704. https://doi.org/10.3390/s20030704

Lai, X., Chen, Y., Lu, F., Liu, J., & Jia, J. (2023). Spherical transformer for lidar-based 3d recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognitio* (pp. 17545–17555).

Lang, A., Vora, S., Caesar, H., Zhou, L., Yang, J., & Beijbom, O. (2019). PointPillars: fast encoders for object detection from point clouds. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 12697–12705).

Le, D. T., Shi, H., Rezatofighi, H., & Cai, J. (2023). Accurate and real-time 3d pedestrian detection using an efficient attentive pillar network. *IEEE Robotics and Automation Letters, 8*(2), 1159–1166. https://doi.org/10.1109/LRA.2022.3233234

Lee, J., Lee, Y., Kim, J., Kosiorek, A., Choi, S., & Teh, Y. W. (2019). Set transformer: a framework for attention-based permutation-invariant neural networks. In *International conference on machine learning* (pp. 3744–3753).
Li, H., & Lu, Y. (2023). 3d object detection based on point cloud in automatic driving scene. *Multimedia Tools and Applications, 83*(5), 13029–13044. https://doi.org/10.1007/s11042-023-15963-0
Li, J., Luo, C., & Yang, X. (2023). PillarNeXt: rethinking network designs for 3d object detection in LiDAR point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 17567–17576).
Li, P., Zhang, Y., Yuan, L., & Xu, X. (2024). Fully transformer-equipped architecture for end-to-end referring video object segmentation. *Information Processing & Management, 61*(1), Article 103566. https://doi.org/10.1016/j.ipm.2023.103566
Li, X., Shi, B., Hou, Y., Wu, X., Ma, T., Li, Y., & He, L. (2022). Homogeneous multi-modal feature fusion and interaction for 3d object detection. In *European Conference on Computer Vision* (pp. 691–707).
2023-1-1 Li, Y., Wang, A., Jing, H., & Bu, D. (2023). GVFNet: graph-voxel fusion network for 3d object detection. In K. Kondo, M. Horng, J. Pan, & P. Hu (Eds.), *International Conference on Intelligent Information Hiding and Multimedia Signal Processing* (pp. 195–204). Singapore.
Liang, M., Yang, B., Chen, Y., Hu, R., & Urtasun, R. (2019). Multi-task multi-sensor fusion for 3d object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 7345–7353).
Liang, M., Yang, B., Wang, S., & R, U. (2018). Deep continuous fusion for multi-sensor 3d object detection. In *Proceedings of the European conference on computer vision (ECCV)* (pp. 641–656).
Liang, Z., Zhang, M., Zhang, Z., Zhao, X., & Pu, S. (2020). Rangercnn: towards fast and accurate 3d object detection with range image representation. *Arxiv Preprint Arxiv:2009.00206*. doi:10.48550/arXiv.2009.00206.
Liang, Z., Zhang, Z., Zhang, M., Zhao, X., & Pu, S. (2021). RangeIoUDet: range image based real-time 3d object detector optimized by intersection over union. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 7136–7145).
Lin, F., Yue, Y., Hou, S., Yu, X., Xu, Y., Yamada, K. D., & Zhang, Z. (2023). Hyperbolic chamfer distance for point cloud completion. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 14595–14606).
Lin, Z., Shen, Y., Zhou, S., Chen, S., & Zheng, N. (2023). Mlf-det: multi-level fusion for cross-modal 3d object detection. In *International Conference on Artificial Neural Networks* (pp. 136–149).
Liu, J., He, T., Yang, H., Su, R., Tian, J., Wu, J.,... Ouyang, W. (2022). 3d-queryis: a query-based framework for 3d instance segmentation. *Arxiv Preprint Arxiv: 2211.09375*. doi:10.48550/arXiv.2211.09375.
Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., … Guo, B. (2021). Swin transformer: hierarchical vision transformer using shifted windows. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)* (pp. 10012–10022).
Liu, Z., Tang, H., Lin, Y., & Han, S. (2019). Point-voxel CNN for efficient 3d deep learning. *Advances in Neural Information Processing Systems* (p. 32). https://doi.org/10.48550/arXiv.1907.03739
Liu, Z., Zhang, Z., Cao, Y., Hu, H., & Tong, X. (2021). Group-free 3d object detection via transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Visio* (pp. 2949–2958).
Lu, B., Sun, Y., Yang, Z., Song, R., Jiang, H., & Liu, Y. (2024). HRNet: 3d object detection network for point cloud with hierarchical refinement. *Pattern Recognition, 149*, Article 110254.
Lu, B., Sun, Y., & Yang, Z. (2023). Voxel graph attention for 3-d object detection from point clouds. *Ieee Transactions On Instrumentation and Measurement, 72*, 1–12. https://doi.org/10.1109/TIM.2023.3301907
Luo, Z., Zhang, G., Zhou, C., Liu, T., Lu, S., & Pan, L. (2023). TransPillars: coarse-to-fine aggregation for multi-frame 3d object detection. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision* (pp. 4219–4228).
Mao, J., Shi, S., Wang, X., & Li, H. (2023). 3d object detection for autonomous driving: a comprehensive survey. *International Journal of Computer Vision, 131*(8), 1909–1963. https://doi.org/10.1007/s11263-023-01790-1
Mao, J., Xue, Y., Niu, M., Bai, H., Feng, J., Liang, X., … Xu, C. (2021). Voxel transformer for 3d object detection. In *Proceedings of the IEEE/CVF international conference on computer visio* (pp. 3164–3173).
Meng, X., Zhou, Y., Du, K., Ma, J., Meng, J., Kumar, A., … Wang, S. (2024). EFNet: enhancing feature information for 3d object detection in LiDAR point clouds. *Journal of the Optical Society of America A, 41*(4), 739–748. https://doi.org/10.1364/JOSAA.511948
Meraz, M., Ansari, M. A., Javed, M., & Chakraborty, P. (2022). DC-GNN: drop channel graph neural network for object classification and part segmentation in the point cloud. *International Journal of Multimedia Information Retrieval, 11*(2), 123–133. https://doi.org/10.1007/s13735-022-00236-7
Misra, I., Girdhar, R., & Joulin, A. (2021, 2021-1-1). An end-to-end transformer model for 3d object detection pp. 2906-2917).
Nasir, T., & Malik, M. K. (2024). Efficient CRNN: towards end-to-end low resource urdu text recognition using depthwise separable convolutions and gated recurrent units. *Information Processing & Management, 61*(1), Article 103544.
Openpcdet, D. T. O. (2020). (10.1). *OpenPCDet: an open-source toolbox for 3d object detection from point clouds.* https://github.com/open-mmlab/OpenPCDet.
Pan, X., Xia, Z., Song, S., Li, L. E., & Huang, G. (2021). 3d object detection with pointformer. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 7463–7472).
Pang, S., Morris, D., & Radha, H. (2020). CLOCs: camera-LiDAR object candidates fusion for 3d object detection. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 10386–10393).
Pang, S., Morris, D., & Radha, H. (2022). Fast-CLOCs: fast camera-LiDAR object candidates fusion for 3d object detection. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision* (pp. 3747–3756).
Pei, Y., Zhao, X., Li, H., Ma, J., Zhang, J., & Pu, S. (2023). Clusterformer: cluster-based transformer for 3d object detection in point clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (pp. 6664–6673).
Pytorch (2023). (5-24). Torch.scatter. https://pytorch.org/docs/2.3/generated/torch.scatter.html#torch.scatter.
Q, X., Y, C., G, C., G, C., D, X., J, S., & Z, W. (2023). 3-d HANet: a flexible 3-d heatmap auxiliary network for object detection. *Ieee Transactions On Geoscience and Remote Sensing, 61*, 1–13. https://doi.org/10.1109/TGRS.2023.3250229
Qi, C. R., Litany, O., He, K., & Guibas, L. J. (2019). Deep hough voting for 3d object detection in point clouds. In *proceedings of the IEEE/CVF International Conference on Computer Vision* (pp. 9277–9286).
Qi, C. R., Liu, W., Wu, C., Su, H., & Guibas, L. (2018). Frustum PointNets for 3d object detection from RGB-d data. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 918–927).
Qi, C. R., Su, H., Mo, K., & Guibas, L. J. (2017). Pointnet: deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 652–660).
Qi, C. R., Yi, L., Su, H., & Guibas, L. J. (2017). Pointnet++: deep hierarchical feature learning on point sets in a metric space. *Advances in Neural Information Processing Systems* (p. 30). https://doi.org/10.48550/arXiv.1706.02413
Qian, R., Lai, X., & Li, X. (2022). 3d object detection for autonomous driving: a survey. *Pattern Recognition, 130*, Article 108796. https://doi.org/10.1016/j.patcog.2022.108796
Ren, S., He, K., Girshick, R., & Sun, J. (2017). Faster r-CNN: towards real-time object detection with region proposal networks. *Ieee Transactions On Pattern Analysis and Machine Intelligence, 39*(6), 1137–1149. https://doi.org/10.1109/TPAMI.2016.2577031
Ren, S., Pan, X., Zhao, W., Nie, B., & Han, B. (2023). Dynamic graph transformer for 3d object detection. *Knowledge-Based Systems, 259*, Article 110085. https://doi.org/10.1016/j.knosys.2022.110085
Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., & Monfardini, G. (2008). The graph neural network model. *IEEE Transactions On Neural Networks, 20*(1), 61–80.
Sheng, H., Cai, S., Liu, Y., Deng, B., Huang, J., Hua, X., & Zhao, M. (2021). Improving 3d object detection with channel-wise transformer. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 2743–2752).
Shi, G., Li, R., & Ma, C. (2022). *PillarNet: real-time and high-performance pillar-based 3d object detection.* Cornell University Library. ReportarXiv.org.

Shi, S., Guo, C., Jiang, L., Wang, Z., Shi, J., Wang, X., & Li, H. (2020). PV-RCNN: point-voxel feature set abstraction for 3d object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 10526–10535).
Shi, S., Jiang, L., Deng, J., Wang, Z., Guo, C., Shi, J., … Li, H. (2023). PV-RCNN++: point-voxel feature set abstraction with local vector representation for 3d object detection. *International Journal of Computer Vision, 131*(2), 531–551. https://doi.org/10.1007/s11263-022-01710-9
Shi, S., Wang, X., & Li, H. (2019). Pointrcnn: 3d object proposal generation and detection from point cloud. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 770–779).
Shi, S., Wang, Z., Shi, J., Wang, X., & Li, H. (2020). From points to parts: 3d object detection from point cloud with part-aware and part-aggregation network. *Ieee Transactions On Pattern Analysis and Machine Intelligence, 43*(8), 2647–2664. https://doi.org/10.1109/TPAMI.2020.2977026
Shi, W., & Rajkumar, R. (2020). Point-GNN: graph neural network for 3d object detection in a point cloud. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 1708–1716).
Shreyas, E., & Sheth, M. H. (2021). 3d object detection and tracking methods using deep learning for computer vision applications. In *2021 International Conference on Recent Trends on Electronics, Information, Communication & Technology (RTEICT)* (pp. 735–738).
Song, Z., Liu, L., Jia, F., Luo, Y., Jia, C., Zhang, G., … Wang, L. (2024). Robustness-aware 3d object detection in autonomous driving: a review and outlook. *Ieee Transactions On Intelligent Transportation Systems, 25*, 15407–15436. https://doi.org/10.1109/TITS.2024.3439557
Sun, P., Kretzschmar, H., Dotiwalla, X., Chouard, A., & Anguelov, D. (2020), 2020-1-1 *Scalability in perception for autonomous driving: waymo open dataset*.
Sun, P., Tan, M., Wang, W., Liu, C., Xia, F., Leng, Z., & Anguelov, D. (2022). *SWFormer: sparse window transformer for 3d object detection in point clouds*. Cornell University Library. ReportarXiv.org.
Tang, H., Liu, Z., Zhao, S., Lin, Y., Lin, J., Wang, H., & Han, S. (2020). Searching efficient 3d architectures with sparse point-voxel convolution. In *European conference on computer vision* (pp. 685–702). Cham: Springer International Publishing.
Tang, Q., Bai, X., Guo, J., Pan, B., & Jiang, W. (2023). DFAF3d:a dual-feature-aware anchor-free single-stage 3d detector for point clouds. *Image and Vision Computing, 129*(C), Article 104594. https://doi.org/10.1016/j.imavis.2022.104594
Te, G., Hu, W., Zheng, A., & Guo, Z. (2018). RGCNN: regularized graph CNN for point cloud segmentation. In *Proceedings of the 26th ACM international conference on Multimedia* (pp. 746–754). New York, NY, USA.
Thakur, S., & Peethambaran, J. (2020). Dynamic edge weights in graph neural networks for 3d object detection. *Arxiv Preprint Arxiv:2009.08253*.
Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., … Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems* (p. 30).
Vora, S., Lang, A. H., Helou, B., & Beijbom, O. (2020). PointPainting: sequential fusion for 3d object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 4603–4611).
W, X., Y, P., C, L., J, G., Y, W., & X, L. (2023). Balanced sample assignment and objective for single-model multi-class 3d object detection. *Ieee Transactions On Circuits and Systems for Video Technology, 33*(9), 5036–5048. https://doi.org/10.1109/TCSVT.2023.3248656
Wang, C., Chen, H., Chen, Y., Hsiao, P., & Fu, L. (2024). VoPiFNet: voxel-pixel fusion network for multi-class 3d object detection. *Ieee Transactions On Intelligent Transportation Systems, 25*(8), 8527–8537. https://doi.org/10.1109/TITS.2024.3392783
Wang, J., Kong, X., Nishikawa, H., Lian, Q., & Tomiyama, H. (2024). Dynamic point - pixel feature alignment for multimodal 3 - d object detection. *IEEE Internet of Things Journal, 11*(7), 11327–11340. https://doi.org/10.1109/JIOT.2023.3329884
Wang, X., Li, K., & Chehri, A. (2024). Multi-sensor fusion technology for 3d object detection in autonomous driving: a review. *Ieee Transactions On Intelligent Transportation Systems, 25*(2), 1148–1165. https://doi.org/10.1109/TITS.2023.3317372
Wang, Y., & Solomon, J. (2021). *Object DGCNN: 3d object detection using dynamic graphs*. Cornell University Library. ReportarXiv.org https://go.exlibris.link/gDq0Vwrg.
Wang, Y., & Ye, J. (2020). An overview of 3d object detection. *Arxiv Preprint Arxiv:2010.15614*.
Wang, Z., & Jia, K. (2019). Frustum convnet: sliding frustums to aggregate local point-wise features for amodal 3d object detection. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 1742–1749).
Wu, H., Wen, C., Li, W., Li, X., Yang, R., & Wang, C. (2023). Transformation-equivariant 3d object detection for autonomous driving. *Proceedings of the AAAI Conference On Artificial Intelligence, 37*(3), 2795–2802. https://doi.org/10.1609/aaai.v37i3.25380
Xia, Z., Liu, Y., Li, X., Zhu, X., Ma, Y., Li, Y., … Qiao, Y. (2023). Scpnet: semantic scene completion on point cloud. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 17642–17651).
Xiaozhi, C., Kaustav, K., & Zhu, Y. (2015). 3d object proposals for accurate object class detection. *Advances in Neural Information Processing Systems* (p. 28).
Xie, T., Wang, L., Wang, K., Li, R., Zhang, X., Zhang, H., … Li, J. (2024). FARP-net: local-global feature aggregation and relation-aware proposals for 3d object detection. *Ieee Transactions On Multimedia, 26*, 1027–1040. https://doi.org/10.1109/TMM.2023.3275366
Xiong, S., Li, B., & Zhu, S. (2023). DCGNN: a single-stage 3d object detection network based on density clustering and graph neural network. *Complex & Intelligent Systems, 9*(3), 3399–3408. https://doi.org/10.1007/s40747-022-00926-z
Y, G., H, W., Q, H., H, L., L, L., & M, B. (2021). Deep learning for 3d point clouds: a survey. *Ieee Transactions On Pattern Analysis and Machine Intelligence, 43*(12), 4338–4364. https://doi.org/10.1109/TPAMI.2020.3005434
Yan, Y., Mao, Y., & Li, B. (2018). SECOND: sparsely embedded convolutional detection. *Sensors, 18*(10), 3337. https://doi.org/10.3390/s18103337
Yang, B., Luo, W., & R, U. (2018). PIXOR: real-time 3d object detection from point clouds. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition* (pp. 7652–7660).
Yang, H., Wang, W., Chen, M., Lin, B., He, T., Chen, H., … Ouyang, W. (2023). Pvt-ssd: single-stage 3d object detector with point-voxel transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 13476–13487).
Yang, Z., Sun, Y., Liu, S., Shen, X., & Jia, J. (2018). Ipod: intensive point-based object detector for point cloud. *Arxiv Preprint Arxiv:1812.05276*. https://doi.org/10.48550/arXiv.1812.05276
Yang, Z., Sun, Y., Liu, S., Shen, X., & Jia, J. (2019). Std: sparse-to-dense 3d object detector for point cloud. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 1951–1960).
Yang, Z., Sun, Y., Liu, S., & Jia, J. (2020). 3DSSD: point-based 3d single stage object detector. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 11037–11045).
Ye, M., Xu, S., & Cao, T. (2020). HVNet: hybrid voxel network for LiDAR based 3d object detection. *Computer Vision and Pattern Recognition* (pp. 1628–1637). https://doi.org/10.1109/CVPR42600.2020.00170
Yin, T., Zhou, X., & Krahenbuhl, P. (2021). Center-based 3d object detection and tracking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 11784–11793).
Yujian, M., Wu, Y., Zhao, J., Yinghao, H., Wang, J., & Yan, J. (2024). Sparse query dense: enhancing 3d object detection with pseudo points. *ACM Multimedia*.
Zamanakos, G., Tsochatzidis, L., Amanatiadis, A., & Pratikakis, I. (2021). A comprehensive survey of LIDAR-based 3d object detection methods with deep learning for autonomous driving. *Computers & Graphics, 99*, 153–181. https://doi.org/10.1016/j.cag.2021.07.003
Zarzar, J., Giancola, S., & Ghanem, B. (2019). PointRGCN: graph convolution networks for 3d vehicles detection refinement. *Arxiv Preprint Arxiv:1911.12236*. doi:10.48550/arXiv.1911.12236.
Zhang, A., Eranki, C., Zhang, C., Park, J., Hong, R., Kalyani, P., … Esteva, M. (2024). Toward robust robot 3-d perception in urban environments: the UT campus object dataset. *IEEE Transactions On Robotics, 40*, 3322–3340. https://doi.org/10.1109/TRO.2024.3400831
Zhang, G., Junnan, C., Gao, G., Li, J., & Hu, X. (2024). HEDNet: a hierarchical encoder-decoder network for 3d object detection in point clouds. *Advances in Neural Information Processing Systems* (p. 36).
Zhang, Y., Chen, J., & Huang, D. (2022). CAT-det: contrastively augmented transformer for multi-modal 3d object detection. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 908–917).

Zhang, Y., Hou, J., & Yuan, Y. (2024). A comprehensive study of the robustness for lidar-based 3d object detectors against adversarial attacks. *International Journal of Computer Vision, 132*(5), 1592–1624. https://doi.org/10.1007/s11263-023-01934-3

Zhang, Y., Hu, Q., Xu, G., Ma, Y., Wan, J., & Guo, Y. (2022). Not all points are equal: learning highly efficient point-based detectors for 3d lidar point clouds. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 18953–18962).

Zhang, Y., Huang, D., & Wang, Y. (2021). PC-RGNN: point cloud completion and graph neural network for 3d object detection. In *Proceedings of the AAAI Conference on Artificial Intelligence* (pp. 3430–3437).

Zhang, Y., Zhang, Q., Hou, J., Yuan, Y., & Xing, G. (2024). Unleash the potential of image branch for cross-modal 3d object detection. *Advances in Neural Information Processing Systems* (p. 36).

Zhao, H., Jiang, L., Jia, J., Torr, P. H., & Koltun, V. (2021). Point transformer. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 16259–16268).

Zhao, W. L. (2018). Approximate k-NN graph construction: a generic online approach. *Ieee Transactions On Multimedia, 24*, 1909–1921.

Zheng, W., Tang, W., Jiang, L., & Fu, C. (2021). SE-SSD: self-ensembling single-stage object detector from point cloud. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 14489–14498).

Zhou, Y., Sun, P., Zhang, Y., Anguelov, D., Gao, J., Ouyang, T., … Vasudevan, V. (2020). End-to-end multi-view fusion for 3d object detection in LiDAR point clouds. In *Conference on Robot Learning* (pp. 923–932).

Zhou, Y., & Tuzel, O. (2018). Voxelnet: end-to-end learning for point cloud based 3d object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 4490–4499).