The 3D Object Detection Algorithm Based on Transformer Voxel Encoding for Autonomous Vehicles

Zhibing Duan, Jinju Shao, Zhipeng Zhai, Jinlei Zhang and Lei Wang

***Abstract*—In point cloud-based 3D object detection, PointPillars encodes features into a pseudo-image by partitioning the point cloud into voxels with unrestricted height and subsequently extracting features using a 2D convolutional architecture. This approach significantly enhances computational efficiency. However, feature extraction occurs on a per-voxel basis, which limits the receptive field and introduces a bottleneck in detection accuracy. Furthermore, the Feature Pyramid Network (FPN) employed in PointPillars results in information loss during downsampling, particularly affecting the detection of small objects.To address these limitations, we propose a novel 3D object detection framework based on the Transformer architecture. Specifically, we introduce a Transformer Feature Encoding (TFE) module to encode voxel features, thereby expanding the receptive field. Additionally, we incorporate an FPN-based Dual Path Convolutional Network (DPCN) as the backbone for feature extraction, effectively mitigating detail loss during downsampling. Extensive experimental evaluations demonstrate that our proposed approach outperforms PointPillars and other voxel-based methods, particularly in detecting small objects.**

***Index Terms*—Autonomous vehicles, Point-cloud, 3D object detection, Transformer, 2D convolution**

I. Introduction

3D object detection [1]−[3] enables autonomous vehicles [4] to recognize and characterize obstacles, making it a key technology in driverless driving and a focal point of research. Among existing approaches, point cloud-based 3D object detection has garnered significant attention due to the unique advantages of LiDAR sensors. Consequently, improving detection accuracy and computational efficiency has become a major research focus.

Point-based methods [5]−[7] perform object detection by clustering or segmenting the point cloud, extracting features from the identified clusters or foreground points, and mapping them back to 3D detection boxes. By directly processing point cloud data, these methods effectively preserve geometric details, leading to high detection accuracy. However, as feature extraction is performed on a per-point basis, these approaches struggle to meet real-time requirements when processing large-scale point clouds.

Voxel-based methods [8]−[10] partition the 3D space into structured grids, extracting feature representations at the voxel level. These features are then processed using 3D convolution [11], as seen in [8]−[9], or handcrafted feature extraction techniques, as in [10], followed by a 2D convolutional detection framework [12]. This transformation of unordered, sparse point cloud data into a structured format facilitates efficient feature extraction via 2D convolution. However, the reliance on 3D convolution significantly increases computational complexity, making real-time processing challenging.

PointPillars [13], proposed by Alex H. Lang et al., addresses this issue by adopting a more efficient voxelization strategy. Unlike traditional voxel-based methods, PointPillars partitions the space into pillars (columns with unrestricted height) and employs PointNet [14] to extract features within each pillar. This approach eliminates the need for computationally expensive 3D convolutions, significantly improving inference speed while maintaining competitive detection accuracy.

PointPillars strikes a balance between detection accuracy and real-time performance, making it a significant step toward the practical application of 3D object detection. However, it still faces several limitations. First, since features are extracted at the voxel level, the model struggles to capture contextual and global information, restricting its receptive field and thereby limiting detection accuracy. Second, the Feature Pyramid Network (FPN) [15] utilized in PointPillars leads to information loss during downsampling. Moreover, its simplistic downsampling strategy fails to fully leverage pseudo-image features, negatively impacting object detection, particularly for small objects.

To address these challenges, we propose a Transformer-based 3D object detection algorithm for point cloud processing. First, the point cloud is partitioned into 3D non-overlapping voxels along the height dimension. Features are then extracted from the point cloud within these voxels to generate voxel-level feature representations. The Transformer enhancing the model's ability to capture global contextual dependencies and expanding the receptive field. Subsequently, the features undergo further processing via a

Manuscript received January 11, 2025; revised April 18, 2025.

This work was supported in part by the Shandong Province Major Science and Technology Innovation Project under Grant 2023CXGC010111, and in part by the Small and Medium-sized Enterprise Innovation Capability Improvement Project under Grant 2022TSGC2277.

Zhibing Duan is a postgraduate student at the Shandong University of Technology, Zibo, 255000 China (e-mail: zhibingduan6@gmail.com).

Jinju Shao is a professor at the Shandong University of Technology, Zibo, 255000 China (corresponding author to provide phone: 13589573050; e-mail: sjjgbh@163.com).

Zhipeng Zhai is a postgraduate student at the Shandong University of Technology, Zibo, 255000 China (e-mail: zhipengzhai31@gmail.com).

Jinlei Zhang is a senior engineer at the Shandong University of Technology, Zibo, 255000 China (e-mail: hucuijuan@jaran.com.cn).

Lei Wang is a manager at Cestera Motor Co., Ltd., Zibo, 255000 China (e-mail: wanglei@suntae.cn).

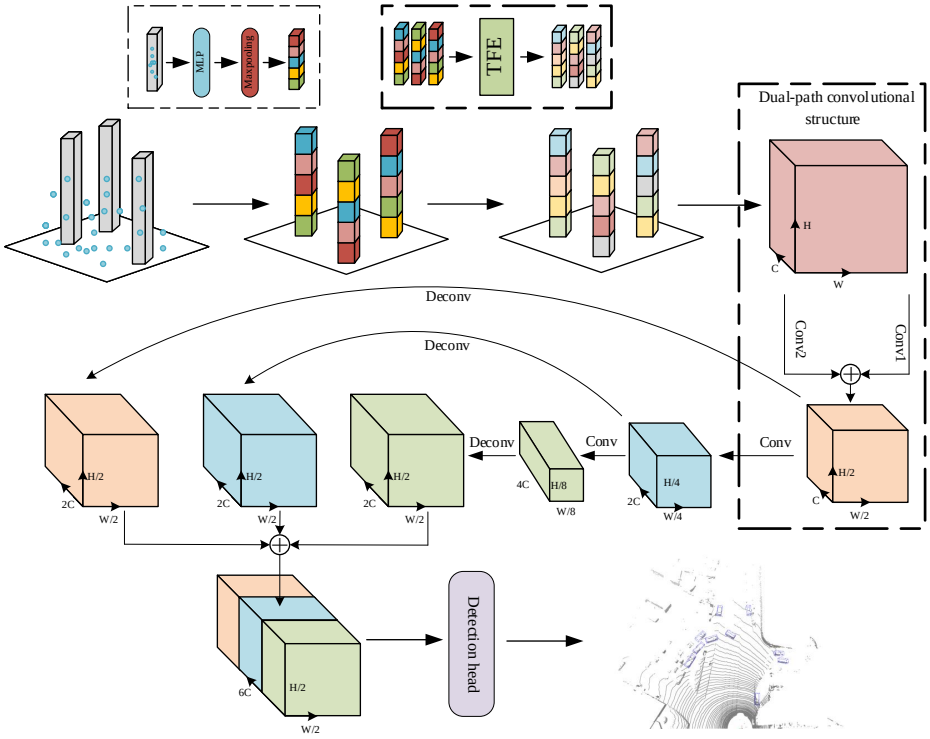


Fig. 1. Overall pipeline of our method

2D convolutional architecture, where a Dual-Path Convolutional Network (DPCN) based on FPN is designed to capture multi-scale features and reduce information loss, thereby improving detection accuracy, particularly for small objects. Finally, the processed feature map is passed to the detection head to complete object detection. The main workflow of the proposed algorithm is illustrated in Fig. 1.

The proposed method is evaluated on the KITTI benchmark dataset, and experimental results demonstrate its superior performance compared to existing 3D object detection algorithms. Notably, our approach achieves a significant improvement in small object detection accuracy, highlighting its effectiveness in addressing the limitations of PointPillars and other voxel-based methods.

To summarize, our contribution is as follows:

1) We introduce a novel approach that integrates the Transformer Feature Encoding (TFE) module into point cloud-based 3D object detection. This enhances the model's receptive field and representation capability, offering valuable insights into the application of Transformers for point cloud processing.
2) We propose the Dual-Path Convolutional Network (DPCN) module, built upon the Feature Pyramid Network (FPN), which effectively captures multi-scale features while minimizing information loss during downsampling, thereby improving the detection accuracy of small objects.
3) Extensive experiments demonstrate that our Transformer-based 3D object detection algorithm surpasses existing methods in accuracy while maintaining real-time processing efficiency.

The remainder of this paper is organized as follows: Section 2 reviews related work, including the application of Transformers in point cloud processing and recent advancements in 2D convolutional neural networks. Section 3 provides a detailed description of the proposed object detection algorithm, covering the Transformer Feature Encoding (TFE) module and the Dual-Path Convolutional Network (DPCN) architecture. Section 4 presents the experimental setup, outlining the hardware, software, experimental parameters, and datasets, followed by quantitative comparisons and qualitative visual analyses to demonstrate the superiority of our approach. Finally, Section 5 concludes the paper with key remarks and future research directions.

II. Related Work

A. Transformer in Point Cloud Processing

The Transformer is a deep learning model based on the attention mechanism, originally introduced for natural language processing [17]. In recent years, it has been successfully adapted for computer vision tasks, particularly 2D object detection, sparking considerable interest in its application to point cloud object detection. However, due to the large-scale nature of point cloud data, directly computing self-attention on individual points incurs significant computational overhead, hindering real-time deployment. To mitigate this challenge, various approaches have been proposed.

Pointformer [18], inspired by PointNet++ [5], employs farthest point sampling (FPS) to select key points, followed by a ball query to define local regions around them, within which the Transformer is applied to local features. While Pointformer achieves strong detection performance, its high model complexity leads to slow inference times. Swformer [19], inspired by the Swin Transformer [20], voxelizes the point cloud in a bird's-eye view and partitions it into windows, computing self-attention within each window. However, variations in the number of non-empty voxels per window result in feature sequences of inconsistent lengths, reducing training efficiency. The Single-Step Sparse Transformer [21] applies self-attention to predefined regions

after voxelization, but this limits the receptive field.

In summary, directly computing self-attention for each point incurs excessive computational costs, while applying it to localized voxel regions constrains the receptive field, and using point clusters increases model complexity. To address these limitations, we propose a novel approach where self-attention is computed interactively across all voxels after voxelization. This expands the model's receptive field, enabling better capture of global contextual information, while also reducing computational overhead compared to point-wise self-attention, thereby improving efficiency.

B. 2D Convolutional Neural Networks

In point cloud-based 3D object detection tasks, feature extraction is typically performed using multi-layer perceptrons (MLPs), 3D convolution, or 2D convolution. MLPs are well-suited for point-based detection methods; however, their computational cost makes them impractical for real-time processing of large-scale point clouds. 3D convolution can be applied in two ways: first, by extracting voxel-based features using 3D convolution, and second, by performing direct convolution on the original point cloud with irregular 3D kernels. While the former suffers from high computational complexity, the latter faces efficiency challenges. In contrast, 2D convolution is computationally efficient and well-suited for real-time applications, but its feature extraction capability is relatively limited.

To enhance the feature extraction capacity of 2D convolutional networks, He et al. [22] proposed the Residual Convolutional Network (ResNet), which introduces residual connections to allow gradient flow across layers. This design enables the network to bypass less informative layers, facilitating deeper architectures and improving feature extraction capabilities. As a result, ResNet has been widely adopted in various domains [23]-[24]. However, increasing network depth inevitably leads to higher model complexity and longer inference times. To address this, DenseNet [25] was introduced, incorporating dense connectivity where each layer receives inputs from all preceding layers. This structure enhances feature reuse and improves representation learning but also increases the risk of overfitting.

Liu et al. [26] proposed the Adaptive Spatial Feature Fusion (ASFF) module, based on the Feature Pyramid Network (FPN) [15], which dynamically selects and fuses multi-layer feature maps. This approach enhances the network's ability to capture multi-scale information, improving detection accuracy. However, it introduces additional trainable parameters for learning fusion weights, leading to increased computational overhead and potential overfitting.

Another strategy to enhance the expressive power of 2D convolutional networks is the integration of attention mechanisms, such as the spatial attention mechanism (SAM) and channel attention mechanism (CAM) proposed by Hu et al. [27]. These mechanisms assign weights to different spatial locations or channels to emphasize crucial features, thereby improving performance in various computer vision tasks. However, since the weight assignment process lacks interpretability, the performance gains from these modules can be unpredictable.

To address these limitations, we propose a Dual-Path Convolutional Network (DPCN) architecture based on FPN to enhance the feature extraction capability of 2D convolutional networks. Our approach employs two parallel convolutional paths with different kernel sizes to capture multi-scale point cloud features. The extracted features are then fused to mitigate information loss, thereby improving the detection accuracy of small objects. Notably, our design achieves this enhancement while introducing minimal additional parameters, ensuring an optimal balance between accuracy and computational efficiency.

III. Methodology

A. Transformer feature encoding

The Transformer model captures the importance of each part of the input data and the relationships between these parts by directly comparing them. Specifically, it assigns weights based on the correlations between the input components, with the final output integrating all parts of the input features. This ability to model global relationships makes the Transformer highly suitable for point cloud processing, as it effectively expands the receptive field. Moreover, the self-attention mechanism in Transformers is invariant to the order and quantity of the input data, further reinforcing its suitability for processing point clouds.

Given the large number of points in a point cloud, directly computing self-attention on the individual points can be computationally expensive. Drawing inspiration from PointPillars, we propose encoding the point cloud using voxels and applying self-attention directly to the voxel representations. This method not only enhances the model's ability to expand its receptive field and improve its expressive power but also reduces the computational burden typically associated with processing individual points.

Upon inputting the raw point cloud, it is first divided into highly unconstrained voxels. Each voxel's positional encoding is recorded, followed by feature extraction using PointNet and pooling [12] to reduce the dimensionality. This process yields the feature representation of each voxel. For clarity, the processed point cloud data is represented as a (P, C) tensor, where P denotes the voxel and C represents the feature vector dimension. Subsequently, the Transformer Feature Encoding (TFE) module is employed to encode the voxel features, as illustrated in Fig. 2.

First, the voxel feature vectors is represented as a set: $X = (x_1, x_2, \cdots, x_n)$, where n represents the number of non-empty voxels. Then the search weight parameter matrix $W^q \in R^{C \times d_k}$, the key weight parameter matrix $W^k \in R^{C \times d_k}$, and the value weight parameter matrix $W^v \in R^{C \times C}$ are initialized, where d_k is the dimensionality of the key of the feature vector. Then the queries $Q = (q_1, q_2, \cdots, q_n)$, keys $K = (k_1, k_2, \cdots, k_n)$ and values $V = (v_1, v_2, \cdots, v_n)$ of the feature vector are computed using equations (1), (2), and (3), respectively:

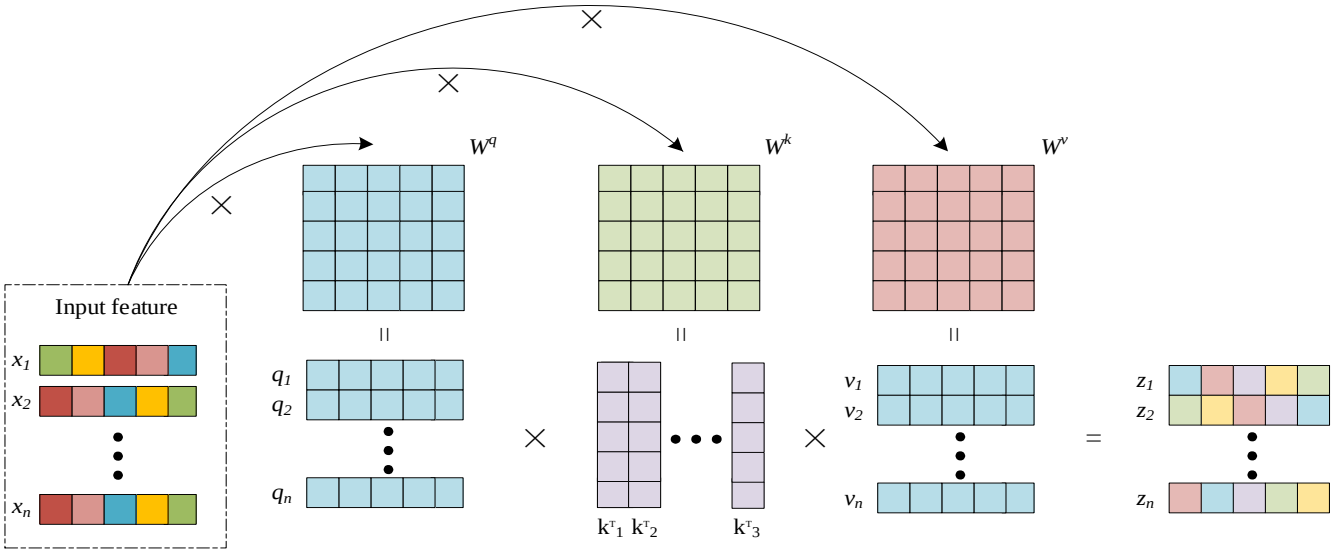


Fig. 2. Schematic diagram of the TFE module

$$Q = XW^q \tag{1}$$

$$K = XW^k \tag{2}$$

$$V = XW^v \tag{3}$$

Then, the query q_i of each feature vector is multiplied by the transpose of the key k_i of all the other feature vectors, respectively, and then by $1/\sqrt{d_k}$. $1/\sqrt{d_k}$ is a scaling factor to prevent the inner product result from being too large and causing the gradient to vanish. Then the softmax function [28] is applied to obtain the weights of the values, and finally the weights are dot-producted with the values v_i of the corresponding feature vectors, respectively. The attention score vector for each feature vector is obtained. The above steps can be expressed in equation (4):

$$Attention(Q,K,V) = softmax(QK^T / \sqrt{d_k})V \tag{4}$$

Next, the attention score vectors of all feature vectors are summed up, then the self-attention of the feature vectors $Z = (z_1, z_2, \cdots, z_n)$ is obtained. The self-attention of the voxel features is used as the new feature vector and this completes the encoding of the voxel features. Finally, the position of each voxel P is reduced according to the voxel encoding to obtain the pseudo-image data of size (H,W,C) (where H denotes the height and W denotes the width), which is easy to process using 2D convolutional architecture.

B. Dual-path convolutional feature extraction

After the voxel features are encoded by the Transformer, they undergo additional feature extraction through a 2D convolutional architecture. Typically, voxel-based methods employ the 2D convolutional architecture of the Feature Pyramid Network (FPN) for this task. However, the downsampling process inherent in such architectures leads to a partial loss of detailed information, which can hinder the detection of small objects. To address this challenge, we propose the Dual-Path Convolutional Network (DPCN) on the FPN, designed to preserve more detailed information in the feature map. Moreover, DPCN utilizes convolutions at multiple scales, further enhancing the model's expressive capability.

While the introduction of DPCN adds additional parameters to the model, it also increases the computational load and introduces the potential risk of overfitting. To mitigate these challenges, we incorporate the DPCN architecture only during the first downsampling stage, striking a careful balance between performance improvement and computational efficiency.

After the point cloud data of size (H,W,C) is fed into the proposed 2D convolutional architecture, as shown in **Fig. 3**. In the first path, the point cloud data is firstly subjected to convolution operation by the convolution kernel of size 3×3, step size 2, and padding 1, to obtain the feature map of size $(H/2, W/2, C/2)$, which is a step to downsampling the point cloud data to reduce the computational amount. Then it is sequentially processed by two convolution kernels of size 3×3, step size 1, and padding 1. The size of the feature map is constant to $(H/2, W/2, C/2)$ after processing. This step expands the receptive field of the model, allowing the model to more fully understand the contextual information.

The second path is the opposite of the first path, where the point cloud data is first processed by two convolution kernels of size 3×3, step 1, and fill 1, with a constant feature map size of (H,W,C) . Then it is downsampled again by a convolution kernel processed with a size of 3×3, a step of 2, and a fill of 1 to obtain a feature map with size $(H/2, W/2, C/2)$. It can be seen that the second path first expands the receptive field of the original data before downsampling, which increases part of the computation, but well preserves the detail information of the feature map, which is more conducive to the detection of small objects.

The feature maps of the same size obtained on the two convolutional paths are spliced and fused along the direction of the feature channel to obtain the fused feature maps under different scales of feature extraction with the size of $(H/2, W/2, C)$. Then, through a series of convolutional down-sampling, up-sampling and feature map splicing operations, a feature map of size $(H/2, W/2, 6C)$ is obtained, and finally fed into the single-stage detector (SSD) [29] to complete the classification of the objects in the point cloud and the fitting of the detection frame.

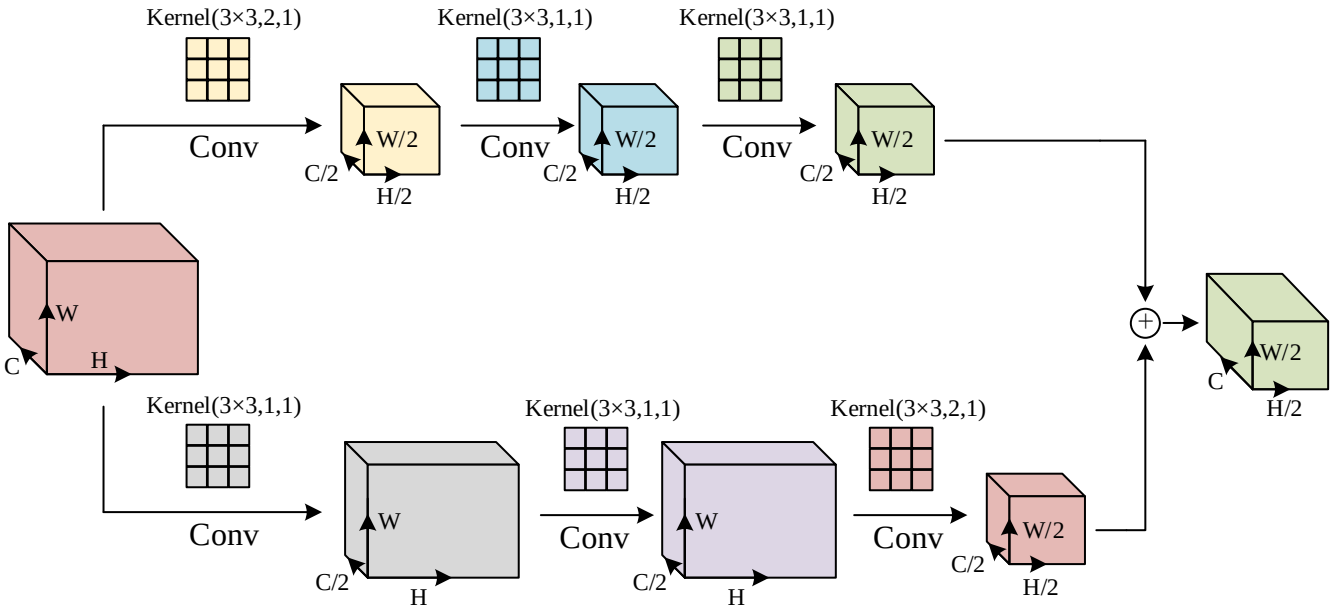


Fig. 3. Schematic diagram of the DPCN architecture

C. loss function

The loss function is set to optimize the parameters of the model in the training iterations so that the predicted values can be closer to the real values to achieve the best detection results. In this paper, the real 3D object frame and anchor frame are defined by the following variables $(x, y, z, w, l, h, \theta)$, where x, y, z denotes the coordinate position of the 3D frame, w, l, h denotes the width, length and height of the 3D frame, and θ denotes the heading angle. The residuals of the localization regression of the real 3D object box and anchor box are defined as:

$$\Delta x = \frac{x^{gt} - x^a}{d^a}, \Delta y = \frac{y^{gt} - y^a}{d^a}, \Delta z = \frac{z^{gt} - z^a}{d^a}$$
$$\Delta w = \log\frac{w^{gt}}{w^a}, \Delta l = \log\frac{l^{gt}}{l^a}, \Delta h = \log\frac{h^{gt}}{h^a} \quad (5)$$
$$\Delta\theta = \sin(\theta^{gt} - \theta^a)$$

where x^{gt} denotes the real 3D frame, x^a denotes the anchor frame, and $d^a = \sqrt{(w^a)^2 + (l^a)^2}$. The total localization loss function L_{loc} is then defined as:

$$L_{loc} = \sum_{b\in(x,y,z,w,l,h,\theta)} SmoothL1(\Delta b) \quad (6)$$

Since the heading angle localization loss cannot distinguish the flipped box, the heading angle is additionally learned in the discrete direction using softmax to define the facing loss function L_{dir} .

The target object classification loss function L_{cls} is defined using the focal loss function:

$$L_{cls} = -\alpha_a (1 - p^a)^\gamma \log p^a \quad (7)$$

where p^a denotes the probability that the anchor frame is of a particular category. α and γ are hyperparameters, following the values of the original paper $\alpha = 0.25$ and $\gamma = 2$. Thus, the total loss L_{loc} is defined as:

$$L_{loc} = \frac{1}{N_{pos}}(\beta_{loc} L_{loc} + \beta_{cls} L_{cls} + \beta_{dir} L_{dir}) \quad (8)$$

where N_{pos} is the number of positive samples in the anchor frame, $\beta_{loc} = 2$ is the weight of the localization loss function, $\beta_{cls} = 1$ is the weight of the classification loss function, and $\beta_{dir} = 0.2$ is the weight of the orientation loss function.

IV. Experiments

A. Experimental Setup

Experimental Environment Construction

The object detection algorithm proposed in this paper was implemented in Python, utilizing several Python libraries, including PyTorch, NumPy, and OpenCV. The system was deployed on an Ubuntu 18.04 operating system, based on the Linux kernel. The hardware configuration consisted of an Intel i7-9750H CPU, an NVIDIA GeForce RTX 2060 GPU, and 15.5 GB of RAM.

Data Preparation

The data used in the experiments were sourced from the KITTI object detection benchmark dataset [30]. Only the point cloud training data were used to train the model, while both the image and point cloud test data were employed to visualize the detection performance. To improve model training, the official training set was further split into 3,712 training samples and 3,769 validation samples.

Hyperparameter Settings

We set the size of a single voxel to be 0.16m× 0.16m, and set the maximum number of voxels in a single frame during training to be 16000, and the maximum number of points within a voxel to be 32. Consider the point cloud range to be $(x_{min}, y_{min}, z_{min}, x_{max}, y_{max}, z_{max}) = (0\,\text{m}, -39.68\,\text{m}, -3\,\text{m}, 69.12\,\text{m}, 39.68\,\text{m}, 1\text{m})$. In the Transformer framework, we set the

dimension of the feature vector key d_k to 64. The anchor frame dimensions were $(0.6\text{m}, 0.8\text{m}, 1.73\text{m})$ for pedestrians, $(0.6\text{m}, 1.76\text{m}, 1.73\text{m})$ for cyclists, and $(1.6m, 3.9m, 1.56m)$ for bicyclists. In removing the redundant anchor frames using the Non-maximal Suppression (NMS), the IOU threshold was set to 0.5 . In addition, the hyperparameter settings during model training were shown in Table I.

TABLE I
HYPERPARAMETER SETTINGS FOR MODEL TRAINING

Hyperparameters	Value
Number of Training Epochs	160
Batch Size	6
Weight Decay	0.01
Optimizer	Adam
Initial Learning Rate	0.00025
Maximum Learning Rate	0.0025
Learning Strategy	Cosine Annealing

B. Experimental Results and Analysis

Training losses

We plot the relationship between the training loss and the number of steps in Fig. 4. In Fig. 4, the horizontal axis represents the number of training steps and the vertical axis represents the training loss. As shown, the training loss decreases gradually with an increase in the number of training steps, indicating that the model's prediction accuracy improves as training progresses.

Quantitative Analysis

After training the model, the official KITTI evaluation metrics were used to assess the algorithm's detection performance, specifically the Average Precision (AP) at different Intersection over Union (IoU) thresholds.

The algorithms were evaluated in four modes on the KITTI dataset: Bird's Eye View (BEV), 3D, 2D, and Average Orientation Similarity (AOS) modes. The BEV mode represents the object's projection onto the ground from a top-down view and is primarily used to evaluate the accuracy of the object's horizontal position and the algorithm's perception of lane boundaries. The 3D mode assesses the accuracy of the object's dimensions and pose, including its length, width, height, and orientation. The AOS mode measures the discrepancy between the predicted and actual object orientation. The 2D mode evaluates the algorithm's accuracy in the image plane, although we do not focus on this mode in our evaluation.

The KITTI dataset is divided into three difficulty levels: easy, medium, and hard. As the difficulty level increases, object occlusion becomes more pronounced, and the point cloud becomes sparser. We quantitatively compare the detection accuracy of our algorithm with MV3D [31], AVOD-FPN [32], VoxelNet [8], SECOND [9], SECOND-V1.5, F-ConvNet [33], and PointPillars [13], covering almost all voxel-based 3D object detection methods. This comparison demonstrates the superiority of our method among voxel-based detection approaches.

Table II presents a comparison of detection accuracy for cars, pedestrians, and cyclists at the three difficulty levels in BEV mode. The mAP (Mean Average Precision) in the table represents the average accuracy of car, pedestrian, and cyclist detection under medium difficulty. This metric is used by the official KITTI dataset to rank the performance of each algorithm.

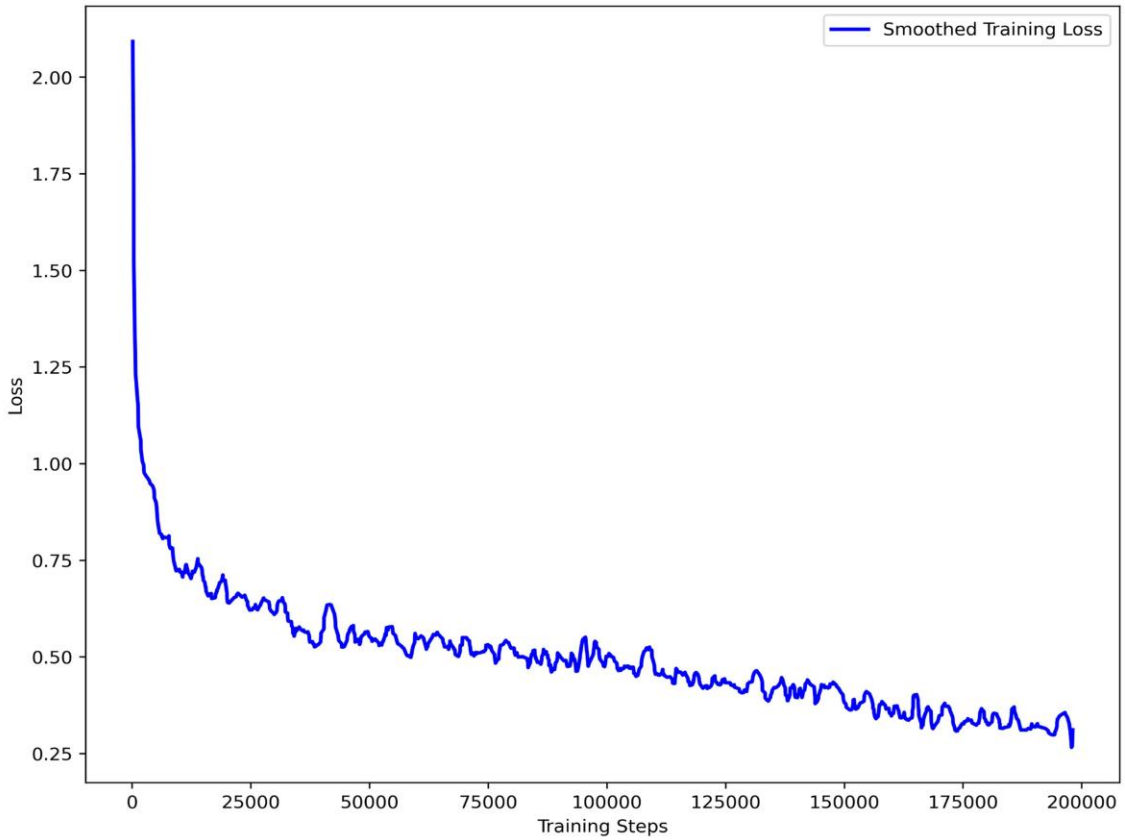


Fig. 4. Plot of the training loss

TABLE II
COMPARISON OF OBJECT DETECTION ACCURACY AMONG ALGORITHMS IN BEV MODE

Algorithms	mAP	Car			Pedestrian			Cyclist		
		Easy	Mod	Hard	Easy	Mod	Hard	Easy	Mod	Hard
VoxelNet	58.25	89.35	79.26	77.39	46.13	40.74	38.11	66.70	54.76	50.55
MV3D	-	86.62	78.93	69.80	-	-	-	-	-	-
AVOD-FPN	63.86	88.53	83.79	77.90	58.49	50.32	46.98	68.03	57.48	50.77
SECOND	60.56	88.07	79.37	77.95	55.10	46.27	44.76	73.67	56.04	48.78
SECOND-V1.5	-	89.27	86.37	81.04	-	-	-	-	-	-
F-ConvNet	67.23	89.51	85.84	76.11	57.04	48.96	44.33	84.16	66.88	60.05
Pointpillars	69.67	89.62	87.55	85.32	59.11	54.32	50.50	84.41	67.14	63.74
Ours	71.27	89.81	87.65	85.48	63.90	58.35	54.74	85.38	67.80	64.23

TABLE III
COMPARISON OF OBJECT DETECTION ACCURACY OF ALGORITHMS IN 3D MODE

Algorithms	mAP	Car			Pedestrian			Cyclist		
		Easy	Mod	Hard	Easy	Mod	Hard	Easy	Mod	Hard
VoxelNet	49.05	77.47	65.11	57.73	39.48	33.69	31.5	61.22	48.36	44.37
MV3D	-	74.97	63.63	54.00	-	-	-	-	-	-
AVOD-FPN	55.44	81.94	71.88	66.38	50.46	42.27	38.08	64.00	52.18	46.61
SECOND	56.69	83.13	73.66	66.20	51.07	42.56	37.29	70.51	53.85	46.90
SECOND-V1.5	-	84.65	75.96	68.71	-	-	-	-	-	-
F-ConvNet	61.61	86.36	76.39	66.69	52.16	43.38	38.80	81.98	65.07	56.54
Pointpillars	62.78	86.42	76.71	74.17	51.35	47.98	43.80	81.75	63.66	60.91
Ours	64.57	86.55	77.23	74.30	54.69	51.03	46.67	83.87	65.46	62.57

TABLE IV
COMPARISON OF OBJECT DETECTION ACCURACY OF EACH ALGORITHM IN AOS MODE

Algorithms	mAP	Car			Pedestrian			Cyclist		
		Easy	Mod	Hard	Easy	Mod	Hard	Easy	Mod	Hard
SECOND	54.53	87.84	81.31	71.95	51.56	43.51	38.78	80.97	57.20	55.14
Pointpillars	68.17	90.49	88.68	85.73	49.36	46.73	43.84	85.02	69.10	66.28
Ours	71.83	90.68	89.13	87.60	56.14	52.45	49.29	87.44	73.90	71.01

As shown in Table II, the average mAP of our algorithm in BEV mode was the highest, with a 1.46% improvement compared to PointPillars, which ranked second. Notably, our algorithm excelled in pedestrian detection and also demonstrated strong performance in car and cyclist detection.

Table III presents a comparison of detection accuracies for cars, pedestrians, and cyclists at three difficulty levels in 3D mode. From Table III, it is evident that our algorithm achieved the highest average accuracy in 3D mode, with a 1.86% improvement over PointPillars, the second-best performer. Our algorithm outperformed all other methods in detecting all object types across all difficulty levels, with particularly strong results in pedestrian and cyclist detection.

Table IV shows the comparison of detection accuracies for SECOND, PointPillars, and our algorithm in AOS mode for cars, pedestrians, and cyclists. As shown in Table IV, our algorithm's average mAP was the highest in AOS mode, surpassing PointPillars by 3.66%. Our algorithm led in detection accuracy for all object types, with notable improvements in pedestrian and cyclist detection.

In summary, the quantitative experimental results demonstrate that our algorithm consistently outperforms similar algorithms in all three modes, achieving the highest average mAP. Apart from a modest improvement in cyclist detection accuracy in BEV mode, our algorithm showed significant gains in pedestrian and cyclist detection, especially in 3D and AOS modes. The overall improvement in pedestrian detection accuracy in BEV mode further highlights the effectiveness of our algorithm in detecting small objects like pedestrians and cyclists.

We tested the average running time of our algorithm using Python's time function. Table V presents a comparison of the running times of our algorithm with other similar object detection algorithms. As shown in Table V, our algorithm has an average processing time of 16.95 ms per frame of point cloud data, operating at approximately 59 Hz. This performance meets the real-time requirements for object detection, as algorithms are typically considered real-time if they process point clouds at a frequency greater than 20 Hz. Furthermore, our algorithm runs faster than most of its counterparts. Although the running time of our algorithm is slightly longer compared to PointPillars, which runs at 62 Hz, as mentioned in the previous section, the detection accuracy of our algorithm is significantly higher than that of PointPillars.

TABLE V
COMPARISON OF RUNNING TIME OF EACH ALGORITHM

Algorithms	Time(s)
VoxelNet	0.22
MV3D	0.24
AVOD-FPN	0.1
SENCOND	0.05
Pointpillars	0.016
Ours	0.017

Qualitative Inorganic Analysis

To offer a more intuitive comparison of object detection performance, we visualized the detection results of both the PointPillars algorithm and the proposed algorithm in various scenarios from the KITTI and nuScenes datasets. This visualization enables a direct comparison of the detection performance between the two algorithms.

1) On the KITTIi dataset

Fig. 5 and Fig. 6 present a comparison of the detection performance of the two algorithms in scenes with a higher number of vehicles. Figure (a) showed the visualization of the detection effect of the algorithm proposed in this paper,

and Figure (b) showed the visualization of the detection effect of Pointpillars. The gray points in the figure were point clouds and the blue box indicates the car's 3D detection box. In Fig. 5(b), PointPillars failed to detect the fourth car on the right side of the road, while the proposed algorithm successfully detected it, as shown in Fig. 5(a). Similarly, in Fig. 6(b), PointPillars missed the car in the right lane ahead, but our algorithm detected it successfully, as shown in Fig. 6(a). These visual results clearly demonstrate that the proposed algorithm outperforms PointPillars in detecting vehicles.

Fig. 7 and Fig. 8 show the comparison of detection performance in scenes with more cyclists and pedestrians. Figure (a) showed the visualization of the detection effect of the algorithm proposed in this paper, and Figure (b) showed the visualization of the detection effect of Pointpillars. The gray points in the figure were point clouds, the red box is the pedestrian detection box, the green box was the cyclist detection box, and the blue box is the car detection box. In Fig. 7(b), PointPillars failed to detect the pedestrians on the left side of the scene, whereas the proposed algorithm successfully detected them, as seen in Fig. 7(a). In Fig. 8(b), where five cyclists are present in the middle of the road, PointPillars detected only two, with the others either missed or misdetected. In contrast, our algorithm successfully detected all five cyclists, as shown in Fig. 8(a).

2) On the nuScenes dataset

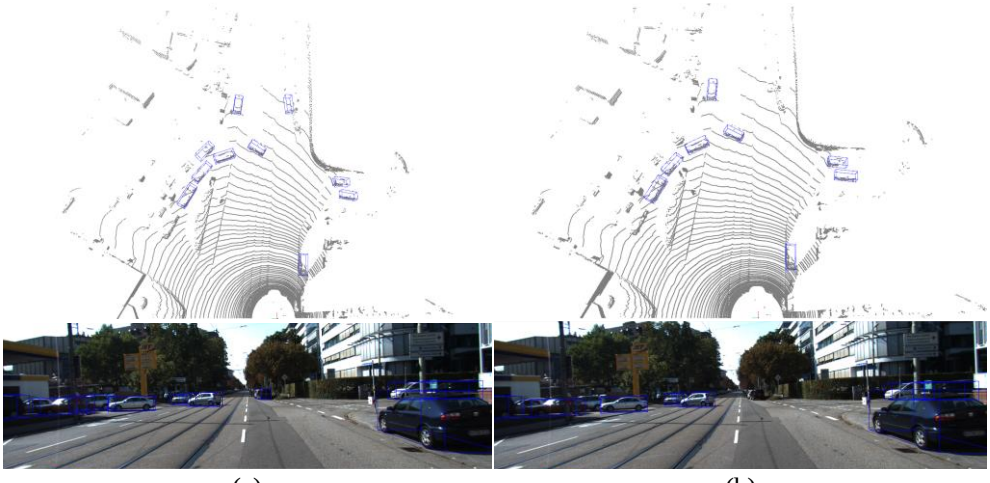
Fig. 9 and Fig. 10 illustrate the comparison of target detection performance between our algorithm and PointPillars. Figure (a) shows the visualization of the detection effect of the algorithm proposed in this paper, and Figure (b) shows the visualization of the detection effect of Pointpillars. The gray points in the figure are point clouds, the red box is the pedestrian detection box, and the blue box is the car detection box.

As shown in these figures, our algorithm outperforms PointPillars in detecting both cars and pedestrians. These results, derived from the nuScenes dataset, further highlight the strong generalization ability of our algorithm. Missed detections of distant targets are mainly attributed to sparse point clouds, where insufficient data is available at greater distances. Overall, our algorithm excels at detecting nearby targets.

In conclusion, although the proposed algorithm still missed some detections, its overall detection accuracy, particularly for pedestrians and cyclists, is superior to that of PointPillars, with significant improvements in detecting small objects like pedestrians and cyclists.

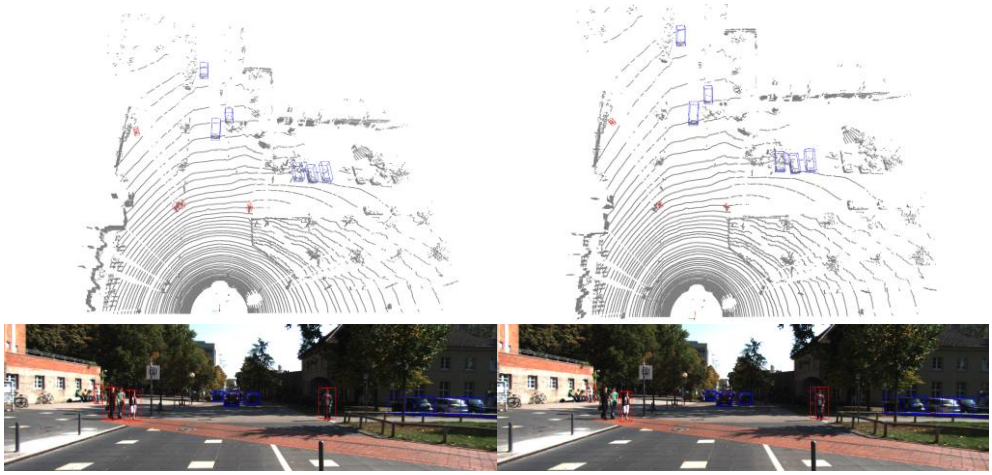
(a) (b)

Fig. 5. Comparison of the two algorithms' detection effect visualization results in the first scenario of the KITTI dataset



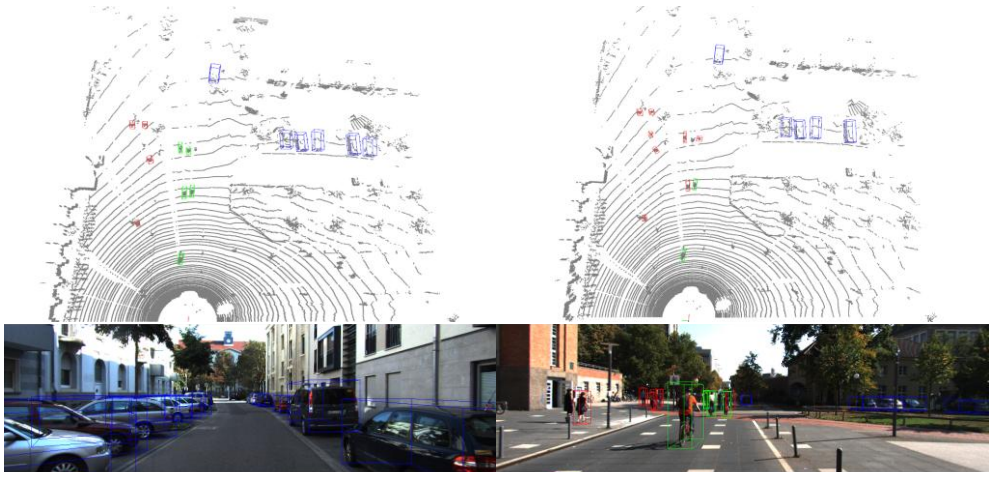
(a) (b)

Fig. 6. Comparison of the two algorithms' detection effect visualization results in the second scenario of the KITTI dataset



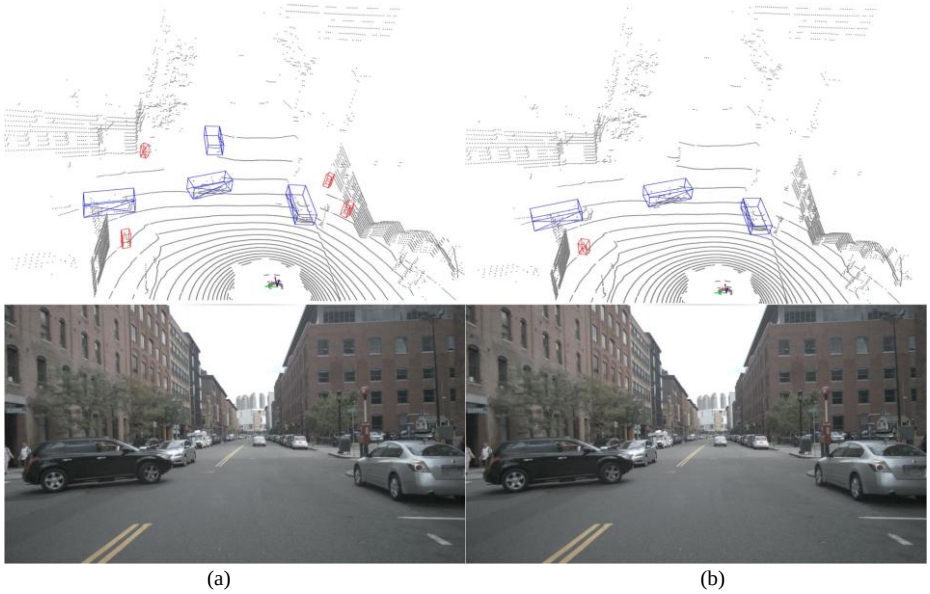
(a) (b)

Fig. 7. Comparison of the two algorithms' detection effect visualization results in the third scenario of the KITTI dataset



(a) (b)

Fig. 8. Comparison of the two algorithms' detection effect visualization results in the fourth scenario of the KITTI dataset



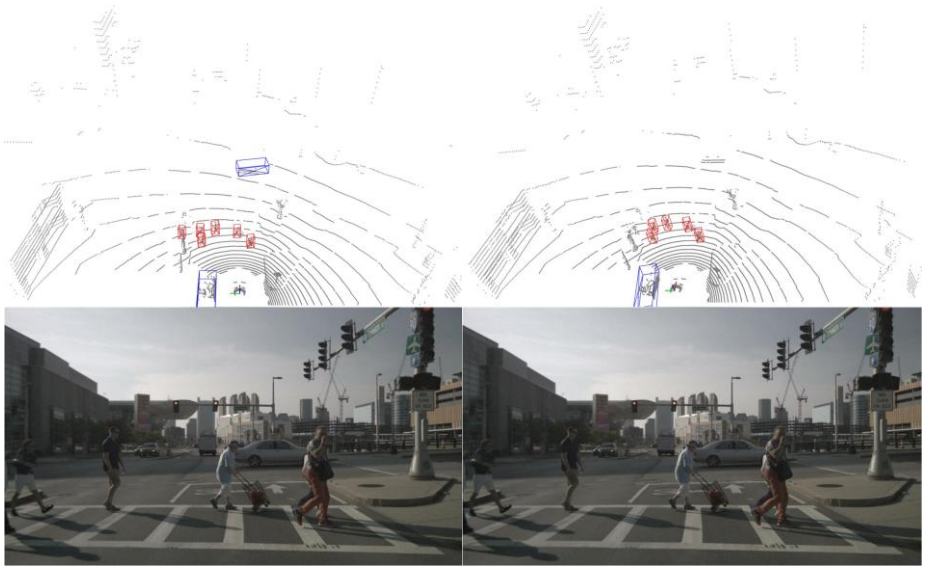
(a) (b)

Fig. 9. Comparison of the results of the visualization of the detection effect of the two algorithms in the first scene of the nuScenes dataset

C. Ablation Experiment

To validate the effectiveness of the TFE module and the dual-path convolutional architecture proposed in this paper, we conducted ablation experiments. We compared the detection accuracies of PointPillars, PointPillars + TFE module, and PointPillars + TFE module + DPCN architecture for cars, pedestrians, and cyclists across three difficulty levels in 3D mode. The specific results are shown in Table VI, where "+" indicates PointPillars + TFE module and "++" indicates PointPillars + TFE module + DPCN architecture.

As shown in Table VI, adding the Transformer Feature Encoding (TFE) module to PointPillars resulted in a slight



(a) (b)

Fig. 10. Comparison of the results of the visualization of the detection effect of the two algorithms in the second scene of the nuScenes dataset

TABLE VI
COMPARISON OF ACCURACY OF OBJECT DETECTION BY POINTPILLARS IN 3D MODE WITH OR WITHOUT THIS PAPER MODULE

Algorithms	mAP	Car			Pedestrian			Cyclist		
		Easy	Mod	Hard	Easy	Mod	Hard	Easy	Mod	Hard
Pointpillars	62.78	86.63	76.71	74.17	51.35	47.98	43.80	81.75	63.66	60.91
+	62.86	86.81	76.92	74.66	53.10	47.51	44.12	81.36	64.16	62.29
++	64.64	86.55	77.23	74.30	54.69	51.03	46.67	83.87	65.46	62.57

increase in mAP (0.08%). However, notable improvements were observed in pedestrian detection accuracy at the easy level (1.75%) and cyclist detection accuracy at the hard level (1.38%). After incorporating

the dual-path convolutional architecture (DPCN), the mAP showed significant improvement, with substantial increases in detection accuracy for pedestrians and cyclists. These ablation experiment results effectively demonstrate the contributions of the TFE module and DPCN architecture in enhancing object detection accuracy.

V. CONCLUSION

In the field of 3D object detection based on LiDAR point clouds, challenges arise from the limited receptive field in voxel-based methods and the information loss caused by downsampling in 2D convolutional architectures of feature pyramids. To address these issues, we propose a novel 3D object detection algorithm based on the Transformer model.

First, we introduce a voxel-based Transformer Feature Encoding (TFE) module that effectively expands the model's receptive field. This improvement enhances the model's ability to capture global contextual information, which is crucial for detecting objects in complex environments.

Additionally, we integrate a dual-path convolutional architecture based on the Feature Pyramid Network (FPN). This architecture reduces information loss during the downsampling process while improving the model's expressive power by utilizing convolutions at different scales for feature extraction. This enables the model to handle multi-scale objects more effectively.

Through quantitative experiments, we show that our proposed algorithm outperforms other voxel-based methods in terms of detection accuracy, particularly for small objects. The results demonstrate significant improvements in detecting challenging objects such as vehicles, pedestrians, and cyclists in 3D point cloud data.

Finally, qualitative experiments and visualizations confirm that our algorithm effectively detects various objects in real-world scenarios. Ablation experiments further validate that both the Transformer-based feature encoding module and the dual-path convolution architecture significantly enhance the model's detection accuracy.

REFERENCES

[1] Yukang Chen, Jianhui Liu, Xiangyu Zhang, Xiaojuan Qi, and Jiaya Jia, "VoxelNeXt: Fully Sparse VoxelNet for 3D Object Detection and Tracking," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, June 2023, pp. 21674–21683.

[2] Hai Wu, Chenglu Wen, Shaoshuai Shi, Xin Li, and Cheng Wang, "Virtual Sparse Convolution for Multimodal 3D Object Detection," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, June 2023, pp. 21653–21662.

[3] Yilun Chen, Zhiding Yu, Yukang Chen, Shiyi Lan, Animashree Anandkumar, Jiaya Jia, and Jose M. Alvarez, "FocalFormer3D: Focusing on Hard Instances for 3D Object Detection," in Proceedings of the IEEE/CVF International Conference on Computer Vision, October 2023, pp. 8394–8405.

[4] Md Mashrur Rana and Kazi Iftekharul Hossain, "Connected and Autonomous Vehicles and Infrastructures: A Literature Review," International Journal of Pavement Research and Technology, vol. 16, no. 2, pp. 264–284, March 2023.

[5] Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas, "PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space," in Advances in Neural Information Processing Systems, vol. 30, 2017.

[6] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li, "PointRCNN: 3D Object Proposal Generation and Detection from Point Cloud," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, June 2019, pp. 770–779.

[7] Zetong Yang, Yanan Sun, Shu Liu, and Jiaya Jia, "3DSSD: Point-Based 3D Single-Stage Object Detector," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, June 2020, pp. 11040–11048.

[8] Yin Zhou and Oncel Tuzel, "VoxelNet: End-to-End Learning for Point Cloud-Based 3D Object Detection," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, June 2018, pp. 4490–4499.
[9] Yan Yan, Yuxing Mao, and Bo Li, "SECOND: Sparsely Embedded Convolutional Detection," Sensors, vol. 18, no. 10, p. 3337, October 2018.
[10] Bin Yang, Wenjie Luo, and Raquel Urtasun, "PIXOR: Real-Time 3D Object Detection from Point Clouds," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, June 2018, pp. 7652–7660.
[11] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller, "Multi-View Convolutional Neural Networks for 3D Shape Recognition," in Proceedings of the IEEE International Conference on Computer Vision, December 2015, pp. 945–953.
[12] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner, "Gradient-Based Learning Applied to Document Recognition," Proceedings of the IEEE, vol. 86, no. 11, pp. 2278–2324, November 1998.
[13] Alexander H. Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom, "PointPillars: Fast Encoders for Object Detection from Point Clouds," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, June 2019, pp. 12697–12705.
[14] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas, "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, July 2017, pp. 652–660.
[15] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie, "Feature Pyramid Networks for Object Detection," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, July 2017, pp. 2117–2125.
[16] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin, "Attention Is All You Need," in Advances in Neural Information Processing Systems, vol. 30, 2017.
[17] Junhong Chen, Hong Dai, Shuang Wang, and Chengrui Liu, "Improving Accuracy and Efficiency in Time Series Forecasting with an Optimized Transformer Model," Engineering Letters, vol. 32, no. 1, pp. 1-11, 2024.
[18] Xinwei Pan, Zhichao Zhou, Shuran Song, Li Erran Li, and Gao Huang, "3D Object Detection with PointFormer," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, June 2021, pp. 7463–7472.
[19] Peng Sun, Ming Tan, Wenhai Wang, Changqian Liu, Fangyun Zhao, Zhiliang Leng, and Dragomir Anguelov, "SWFormer: Sparse Window Transformer for 3D Object Detection in Point Clouds," in European Conference on Computer Vision, October 2022, pp. 426–442, Cham: Springer Nature Switzerland.
[20] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo, "Swin Transformer: Hierarchical Vision Transformer Using Shifted Windows," in Proceedings of the IEEE/CVF International Conference on Computer Vision, October 2021, pp. 10012–10022.
[21] Lincheng Fan, Zihan Pang, Tian Zhang, Yuxuan Wang, Hang Zhao, Feixiang Wang, Nan Wang, and Zhenguo Zhang, "Embracing Single Stride 3D Object Detector with Sparse Transformer," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 8458–8468.
[22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep Residual Learning for Image Recognition," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.
[23] Aji Setiawan, Kusworo Adi, and Catur Edi Widodo, "Comparative Analysis of Deep Convolutional Neural Network for Accurate Identification of Foreign Objects in Rice Grains," Engineering Letters, vol. 32, no. 2, pp. 315-324, 2024.
[24] Zhengpeng Li, Jun Hu, Zhuang Liang, and Jiansheng Wu, "MRAUnet++: A Novel Multi-Scale Residual Attention Network for Enhanced Rectal Cancer Segmentation," Engineering Letters, vol. 32, no. 4, pp. 880-888, 2024.
[25] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger, "Densely Connected Convolutional Networks," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 4700–4708.
[26] Songtao Liu, Di Huang, Yunhong Wang, "Learning Spatial Fusion for Single-Shot Object Detection," arXiv preprint, arXiv:1911.09516, Nov. 2019.
[27] Jie Hu, Li Shen, Gang Sun, "Squeeze-and-Excitation Networks," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 7132-7141.
[28] John S. Bridle, "Probabilistic Interpretation of Feedforward Classification Network Outputs, with Relationships to Statistical Pattern Recognition," in Neurocomputing: Algorithms, Architectures, and Applications, May 1990, pp. 227-236, Berlin, Heidelberg: Springer Berlin Heidelberg.
[29] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, Alexander C. Berg, "SSD: Single Shot Multibox Detector," in Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, Oct. 11-14, 2016, pp. 21-37, Springer International Publishing.
[30] Andreas Geiger, Philip Lenz, Raquel Urtasun, "Are We Ready for Autonomous Driving? The KITTI Vision Benchmark Suite," in 2012 IEEE Conference on Computer Vision and Pattern Recognition, Jun. 16, 2012, pp. 3354-3361, IEEE.
[31] Shuang Li, Kai Geng, Guoyuan Yin, Zhenyu Wang, Minghui Qian, "MVMM: Multiview Multimodal 3D Object Detection for Autonomous Driving," IEEE Transactions on Industrial Informatics, vol. 20, no. 1, pp. 845-853, 2023.
[32] Jason Ku, Melissa Mozifian, Jungwook Lee, Ali Harakeh, Steven L. Waslander, "Joint 3D Proposal Generation and Object Detection from View Aggregation," in 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Oct. 2018, pp. 1-8, IEEE.
[33] Zhixin Wang, Kui Jia, "Frustum ConvNet: Sliding Frustums to Aggregate Local Point-Wise Features for Amodal 3D Object Detection," in 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Nov. 2019, pp. 1742-1749, IEEE.