RESEARCH ARTICLE

# Real-Time Object Detection Using a Lightweight Two-Stage Detection Network with Efficient Data Representation

**Shaohuang Wang**[1,*]

[1] Cardiff University, Cardiff CF10 3AT, United Kingdom

## Abstract

**In this paper, a novel fast object detection framework is introduced, designed to meet the needs of real-time applications such as autonomous driving and robot navigation. Traditional processing methods often trade off between accuracy and processing speed. To address this issue, a hybrid data representation method is proposed that combines the computational efficiency of voxelization with the detail capture capability of direct data processing to optimize overall performance. The detection framework comprises two main components: a Rapid Region Proposal Network (RPN) and a Refinement Detection Network (RefinerNet). The RPN is used to generate high-quality candidate regions, while the RefinerNet performs detailed analysis on these regions to improve detection accuracy. Additionally, a variety of network optimization strategies are incorporated, including lightweight depthwise separable convolutions and GPU-accelerated parallel inference, to increase processing speed and reduce computational resource consumption. Extensive testing on the KITTI and NEXET datasets has proven the effectiveness of this method in enhancing the accuracy of object detection and real-time processing speed. The experimental results show that, compared to existing technologies, this method performs exceptionally well across multiple evaluation metrics, especially in meeting the stringent requirements of real-time applications in terms of processing speed.**

**Keywords**: object detection, real-time, refinement, network optimization

## 1 Introduction

In fields such as autonomous driving [1] and real-time 3D object recognition for robotic perception [6], 3D object detection technology plays an essential role. To address the demands of such applications, a series of point cloud-based detection methods have been proposed, including voxel-based approaches [5, 9], pillar-based encoders [3], and two-stage proposal-refinement networks [2, 4], each making meaningful strides in balancing accuracy and computational efficiency. Especially in autonomous vehicles using LiDAR-camera fusion [7],

**Submitted:** 05 December 2023
**Accepted:** 16 April 2024
**Published:** 20 April 2024

**Vol.** 1, **No.** 1, 2024.
10.62762/TETAI.2024.320179

***Corresponding author:**
✉Shaohuang Wang
wangs130@cardiff.ac.uk

**Citation**
Wang, S. (2024). Real-Time Object Detection Using a Lightweight Two-Stage Detection Network with Efficient Data Representation. *ICCK Transactions on Emerging Topics in Artificial Intelligence*, 1(1), 17–??.

© 2024 by the Author. Published by Institute of Central Computation and Knowledge. This is an open access article under the CC BY license (https://creativecommons.org/licenses/by/4.0/).

precise 3D object detection is indispensable for safe navigation and obstacle avoidance. Although various technologies and methods have been proposed to handle this task, achieving real-time processing while ensuring high accuracy remains a challenge.

Traditional 3D object detection methods [8, 9] mainly rely on point cloud data obtained from sensors such as LiDAR. These methods include converting point cloud data into a voxel grid (voxelization) or projecting it onto a two-dimensional plane (such as a bird's eye view or front view). Voxelization creates a three-dimensional grid and counts the number of points within each grid cell, transforming unordered point cloud data into structured volumetric data, which facilitates processing using convolutional neural networks (CNNs). However, this approach often involves high-dimensional data, leading to increased computational and storage burdens. Moreover, the process of spatial quantization during voxelization may result in the loss of detailed information, especially at the edges of objects or with smaller objects.

On the other hand, methods that directly process point clouds, such as the renowned PointNet [10] and its subsequent work PointNet++ [11], are capable of learning features directly from the raw point cloud data, avoiding the loss of information in the preprocessing steps. These models utilize global feature descriptors of point clouds, enabling the network to understand the structure and shape of the point clouds holistically. However, these models usually require processing the entire point cloud, demanding substantial computational resources that may not meet the needs of real-time applications. As 3D point cloud processing technology continues to advance, existing methods have seen significant improvements in processing speed and accuracy, but they still face key challenges. For instance, despite the wide attention PointNet and PointNet++ have received in the academic community, their computational and storage requirements remain considerable when processing large-scale data. These methods generally rely on deep neural networks to learn features directly from point clouds, a process that demands a significant amount of computational resources, especially when dealing with large volumes of data or uneven point densities.

To address these challenges, researchers have tried various optimization strategies. For instance, some studies have introduced more efficient neural network architectures, such as Sparse Convolutional Networks. These networks significantly reduce unnecessary computational loads by performing calculations only at the non-empty positions of the data, thus increasing processing speed. Additionally, a stratified processing approach has been proposed, where point clouds are processed at different resolution levels. This method preserves important local details while also capturing global structural information, increasing processing efficiency without compromising accuracy.

Another optimization strategy involves the use of effective data dimensionality reduction and preprocessing techniques to reduce the complexity of the input data. For instance, preprocessing point clouds using methods like Principal Component Analysis (PCA) or autoencoders can extract the most representative features before feeding them into the neural network. This approach can alleviate the network's load to some extent and accelerate the training and inference processes. Moreover, multimodal fusion represents a significant direction for enhancing the performance of 3D point cloud processing. By combining data from different sensors, such as integrating LiDAR point clouds with RGB images, complementary information can be obtained from different perspectives and sensing mechanisms, enhancing the model's understanding of the environment. This fusion not only improves the accuracy of object detection but also strengthens the model's ability to discriminate between different object characteristics in complex environments.

Despite the progress made by existing methods, numerous technical challenges remain, along with room for improvement in processing large-scale 3D point cloud data efficiently and in real time. Consequently, researching and developing more efficient algorithms and technologies to meet the dual demands of speed and accuracy in practical applications is a current hotspot in the field of 3D point cloud processing. This study proposes an efficient convolutional neural network framework for rapid object detection within 3D point clouds. The research significantly enhances the efficiency and accuracy of 3D point cloud object detection through innovative data processing and network design strategies. The main contributions are not only reflected in technical innovations but are also demonstrated through a series of rigorous experiments validating the effectiveness of these methods.

To further strengthen detection robustness, a

comprehensive data augmentation strategy is employed during training, incorporating random flipping, scaling, rotation, and Z-axis perturbation to improve the network's ability to generalize across unseen scenarios and varied environmental conditions. These design choices, combined with the optimized two-stage network architecture, provide a unified solution for real-time 3D object detection that addresses both computational efficiency and localization accuracy. Through extensive testing on the KITTI and NEXET benchmark datasets, the effectiveness and practical applicability of the approach in autonomous driving and robotic navigation are demonstrated. Below is a further expansion of the contributions, including detailed descriptions from an experimental perspective:

1 **Efficient Data Representation:** The proposed data representation method combines the computational efficiency of voxelization with the precision of direct point processing. This hybrid approach not only maintains high efficiency in data handling but also significantly reduces information loss typically associated with traditional voxelization. In the experiments, the performance of pure voxelization, pure point cloud processing, and the hybrid method was compared across various complex scenarios. The results indicate that the hybrid method surpasses traditional approaches in both processing speed and accuracy.

2 **Two-Stage Detection Network:** A two-stage network was designed that initially extracts features through a rapid candidate region proposal network, followed by a refinement network for precise object localization and classification. This staged processing approach allows the use of coarser features for quick filtering in the first stage while focusing on the detailed processing of candidate regions in the second stage. The experiments demonstrated how this method improves detection accuracy compared to single-stage processing, particularly in terms of precision at object boundaries.

3 **Real-Time Processing Capability:** In designing the network, particular emphasis was placed on computational efficiency to ensure that the model could operate within the constraints of limited computational resources. By utilizing GPU acceleration and algorithm optimization, the model achieves real-time processing speeds while maintaining high accuracy. Experiments conducted on the KITTI and NEXET datasets extensively tested the model, achieving leading performance on conventional evaluation metrics and demonstrating superior processing speed across both benchmarks. This demonstrates the model's capability to handle real-time applications effectively, making it highly suitable for scenarios requiring immediate response, such as in autonomous driving systems.

Through these experiments, the high practical value and technological advancement of the proposed method in real-world application scenarios have been demonstrated. The experimental results not only showcase the effectiveness of the approach but also highlight its potential in applications requiring rapid and reliable 3D environmental perception, such as autonomous driving and other similar technologies. This underscores the method's capability to enhance operational efficiency and accuracy in dynamic and complex environments, solidifying its relevance and utility in cutting-edge technological implementations.

## 2 Related Work

The field of fast object detection continues to evolve with the introduction of various innovative network architectures and optimization techniques aimed at further enhancing detection speed and accuracy, particularly on resource-constrained devices. These advancements are detailed from three perspectives: single-step detection networks, multi-scale and feature fusion networks, and networks optimized for mobile devices.

### 2.1 Single-Step Detection Networks

These networks significantly speed up the detection process by directly predicting object classes and locations from the input image, bypassing the traditional step of extracting candidate regions. The YOLO series of networks exemplifies single-step detection. YOLO [12] divides the image into multiple grids, with each grid cell directly predicting bounding boxes and class probabilities. A key advantage of YOLO is its speed, enabling near-real-time object detection, making it particularly well-suited for video stream processing. Over the years, YOLO has seen continuous updates to its architecture, improving both accuracy and processing speed through deeper networks, optimized anchor boxes, and advanced loss functions. YOLO's later versions incorporate features like multi-scale detection, which

allow the network to detect objects of various sizes more effectively, further enhancing its applicability in dynamic and complex environments.Another network, CenterNet [13], offers a simpler yet efficient detection approach by predicting the center points and sizes of objects, rather than traditional bounding box regression. This method significantly reduces network complexity and eliminates the need for region proposal and post-processing steps, thus improving detection speed without sacrificing accuracy. This direct approach to object localization has proven particularly beneficial in real-time applications where efficiency is critical. YOLACT [14] demonstrates that single-step paradigms can be extended beyond pure bounding-box regression to richer per-instance representations without sacrificing real-time throughput. Its architecture decouples prediction into two parallel branches—one generating a small set of prototype features shared across the image, the other predicting instance-specific linear coefficients—and combines them with a single element-wise operation. This decomposition keeps the network shallow and GPU-friendly, achieving high frame rates even under dense, overlapping object layouts. The design principle of separating coarse global features from fine instance-level detail is directly relevant to the two-stage philosophy adopted in this work, where the RPN rapidly produces candidate regions and the RefinerNet subsequently enriches each region with precise spatial features. RefineDet [15], another single-step detection network, introduces a two-stage refinement module, where initial candidate boxes are first generated and then refined for more accurate detection. While this introduces slightly more computation than typical single-stage detectors, the refinement step ensures a higher detection accuracy, balancing speed and precision effectively.

In summary, single-step detection networks like YOLO [12], CenterNet [13], YOLACT [14], and RefineDet [15] have revolutionized the field of object detection by significantly improving speed and efficiency. These networks bypass traditional region proposal methods, making them suitable for real-time applications such as video processing and autonomous systems. YOLO's grid-based approach offers a balance between speed and accuracy, while CenterNet simplifies the detection process by focusing on center points, further reducing complexity. YOLACT demonstrates how decoupling coarse global features from fine instance-level detail within a single-step framework can preserve real-time throughput, a decomposition principle that informs the coarse-to-fine philosophy of the proposed two-stage network. RefineDet introduces a refinement process that enhances accuracy without compromising much on speed.

However, despite their advancements, these models still face challenges, particularly in handling smaller objects or highly dense scenes where multiple overlapping objects can reduce detection accuracy. Furthermore, while speed is a major strength, maintaining high precision in complex environments with varying lighting, occlusion, or extreme object scales remains a challenge. As a result, future research should focus on addressing these limitations by developing more robust architectures that can balance both speed and high accuracy across diverse and challenging scenarios. Building upon the original YOLO framework, subsequent versions have introduced significant architectural improvements. YOLOv4 [30] integrates a series of training tricks, including Mosaic data augmentation, DropBlock regularization, and CIoU loss, achieving optimal speed and accuracy on standard benchmarks. YOLOv7 [31] further advances real-time detection through trainable bag-of-freebies strategies, setting new state-of-the-art results without increasing inference cost. YOLOX [32] departs from anchor-based designs by introducing an anchor-free paradigm combined with a decoupled detection head and the SimOTA label assignment strategy, achieving competitive accuracy at high inference speeds. These three detectors serve as representative baselines in the experimental evaluation of this work.

### 2.2 Multi-Scale and Feature Fusion Networks

These networks leverage image features from different scales, utilizing feature fusion technology to enhance detection accuracy, particularly when dealing with objects of varying sizes. SSD [16] predicts the presence and location of objects across multiple feature maps simultaneously, effectively handling targets of different sizes. SSD employs independent convolutional layers at multiple predetermined scales to predict bounding boxes, thereby significantly improving the detection capability for smaller objects. EfficientDet [17] employs an innovative Bi-directional Feature Pyramid Network (BiFPN) that effectively merges features from different layers. This method optimizes feature utilization and reduces the consumption of computational resources. EfficientDet uses compound scaling technology to balance the network's width,

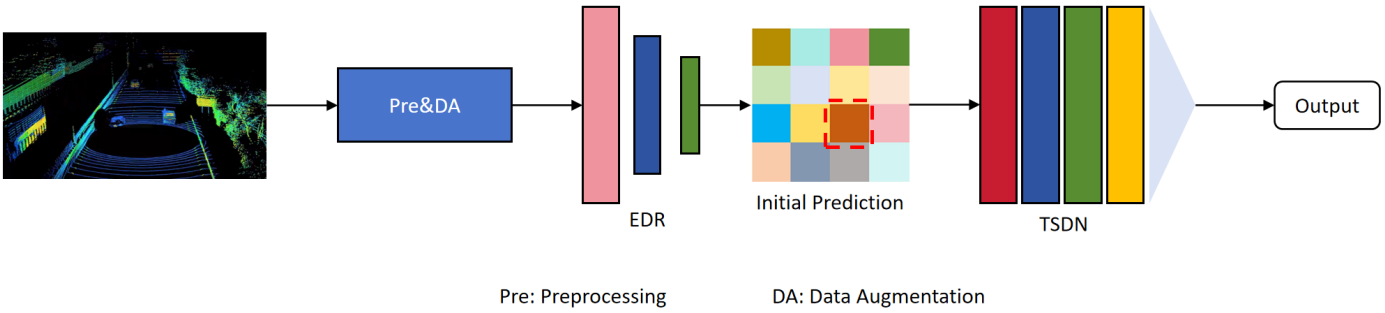


**Figure 1.** Overall structure of the proposed model.

depth, and input resolution, achieving an efficient and precise balance. To manage targets of various sizes and fully leverage multi-scale information in images, multi-scale and feature fusion networks adopt strategies that extract and merge features from different levels. These networks enhance the model's expressive power and adaptability by integrating features of different resolutions. The Feature Pyramid Network (FPN) [18] is a classic architecture for multi-scale feature fusion, performing excellently in natural image object detection tasks. FPN constructs a top-down architecture where semantic information from higher layers gradually percolates to lower layers, enriching each scale with abundant semantic and positional information. This approach is particularly suited for detecting objects that vary significantly in size. Feature fusion across scales can also be substantially enhanced through explicit path augmentation mechanisms. PANet [19] proposes a bottom-up path augmentation strategy that creates additional shortcut connections from lower feature layers to higher ones, directly complementing the top-down pathway of FPN [18]. This design shortens the information propagation path between localization-rich lower layers and semantically-rich upper layers, enabling each scale to benefit from both spatial precision and contextual abstraction. The adaptive feature pooling further allows proposal subnetworks to aggregate features from all pyramid levels simultaneously, yielding more discriminative representations for objects of varying scales. PANet ranked 1st in the COCO 2017 Instance Segmentation Challenge and 2nd in Object Detection, demonstrating the effectiveness of bidirectional feature flow for dense prediction tasks.

### 2.3 Networks Optimized for Mobile Devices

These networks are specifically designed to operate in resource-constrained environments by streamlining network architecture and reducing computational demands to achieve rapid detection. MobileNets [20] utilize depthwise separable convolutions, which significantly reduce the model's size and computational requirements. This makes MobileNets particularly suitable for running on mobile and edge devices while maintaining reasonable detection accuracy. SqueezeDet [21] is a lightweight network designed for low-power devices that combines the rapid detection capabilities of convolutional networks with the need of a compressed network structure. It drastically reduces the model's parameter count and computational needs while maintaining high performance in detection tasks. PeleeNet [22] is another lightweight network designed for mobile devices, employing a densely connected strategy similar to DenseNet but with significant optimizations in computational load and parameter count. The aim of PeleeNet is to provide sufficient detection accuracy with sufficiently low latency, making it suitable for operation on lower-performance devices. ThunderNet [23] continues to advance object detection performance on edge devices through its SNet design and Context Enhancement Module (CEM). SNet serves as the backbone network, focusing on enhancing computational efficiency, while CEM leverages contextual information to improve feature expressiveness, thereby enhancing the accuracy of detecting small objects. These advancements collectively contribute to the viability of deploying advanced object detection technologies in environments where computational resources are limited, such as in embedded systems or mobile applications. This capability is crucial for applications that require real-time processing without access to powerful computing infrastructure, such as autonomous vehicles, drones, or mobile applications that require immediate response and interaction with the environment. As modern devices increasingly operate in dynamic and resource-limited conditions, the importance of these lightweight networks continues to grow. The continuous push for more efficient models ensures that such systems remain

responsive and capable of handling diverse tasks with minimal latency. This is particularly valuable in autonomous systems where timely decisions are crucial for safety, such as avoiding collisions in self-driving cars or enabling quick adjustments in drone flight paths based on real-time sensor input. Moreover, with the growing integration of AI in everyday devices, these lightweight solutions allow for more widespread deployment of advanced object detection technologies, making them accessible in a broader range of scenarios, from edge devices to industrial IoT applications.

## 3 Methodology

The overall architecture of the proposed method is illustrated in Figure 1, which primarily consists of two modules: the Efficient Data Representation (EDR) module and the Two-Stage Detection Network (TSDN) module. The process begins with the raw data being input into a preprocessing and data augmentation module, which serves to enhance the diversity of the data. The enhanced data is then fed into the EDR module for initial refinement. Subsequently, the data processed by the EDR module is input into the TSDN module, where it undergoes further refinement to extract more detailed features. These refined features facilitate the production of the final model results. A detailed description of the method is provided below.

### 3.1 Efficient Data Representation

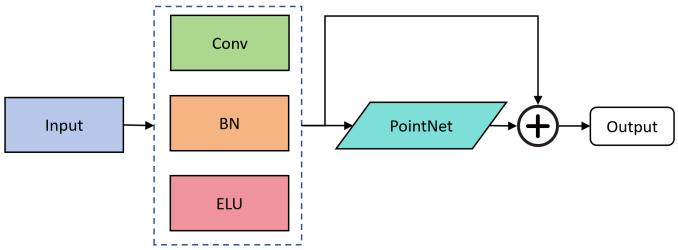


**Figure 2.** Structure of the EDR.

The proposed efficient data representation method comprises two main steps: efficient voxelization and detail-preserving point feature extraction. The purpose of voxelization is to reduce the scale and dimensions of the data, making it more suitable for rapid processing. Point feature extraction, on the other hand, aims to capture the fine structures within each voxel, which is crucial for maintaining the accuracy of object detection. The network model architecture is illustrated in Figure 2.

**Voxelization Process:** Voxelization involves dividing the continuous point cloud space into a regular grid and aggregating point cloud information within each grid cell. The aggregation can be achieved through various statistical measures such as the mean, maximum values, or a combination of features. The voxelization process is defined as follows:

$$V = \frac{1}{|C|} \sum_{p_i \in C} \phi(p_i) \tag{1}$$

$$F_j = \sum_{p_i \in V_j} \omega(p_i) \cdot \phi(p_i) \tag{2}$$

where $C$ represents the set of points within the voxel, $\phi(p_i)$ is the feature extraction function applied to point $p_i$, and $\omega(p_i)$ denotes the learned weights used in the weighted aggregation of Eq. (2). Eq. (1) computes a mean-pooled voxel descriptor $V$, while Eq. (2) produces a weighted feature map $F_j$ that emphasizes more informative points within each voxel. $\phi(p_i)$ can involve simple attributes such as coordinates, intensity, and color, or more complex properties like local curvature or normal vectors.

**Point Feature Extraction:** To further enhance the expressive power of voxelized data, a lightweight PointNet network is employed within each voxel to extract more comprehensive local features. This approach ensures that even if some detail is lost during the voxelization process, these details can still be preserved through the learned features.

$$F'_j = \sum_{p_i \in V_j} \alpha_{p_i} \cdot \mathrm{ELU}(\mathrm{BN}(\mathrm{Conv}(\phi(p_i)))) \tag{3}$$

where $F'_j$ represents the advanced features extracted from voxel (x,y,z). These features include not only geometric information but also deeper contextual information learned by the PointNet model. This combination provides a rich representation that enhances the model's ability to accurately recognize and classify objects within the 3D space.

### 3.2 Two-Stage Detection Network

The two-stage detection network is designed to enhance detection speed while ensuring accuracy, with the RPN structure and the RefinerNet structure as its core module. The model structure is illustrated in Figure 3.

**First Stage:** Rapid Region Proposal Network (RPN). The goal of this stage is to rapidly generate high-quality candidate regions from the hybrid data representation. The RPN utilizes a series of optimized convolutional layers to process voxelized point cloud data, which

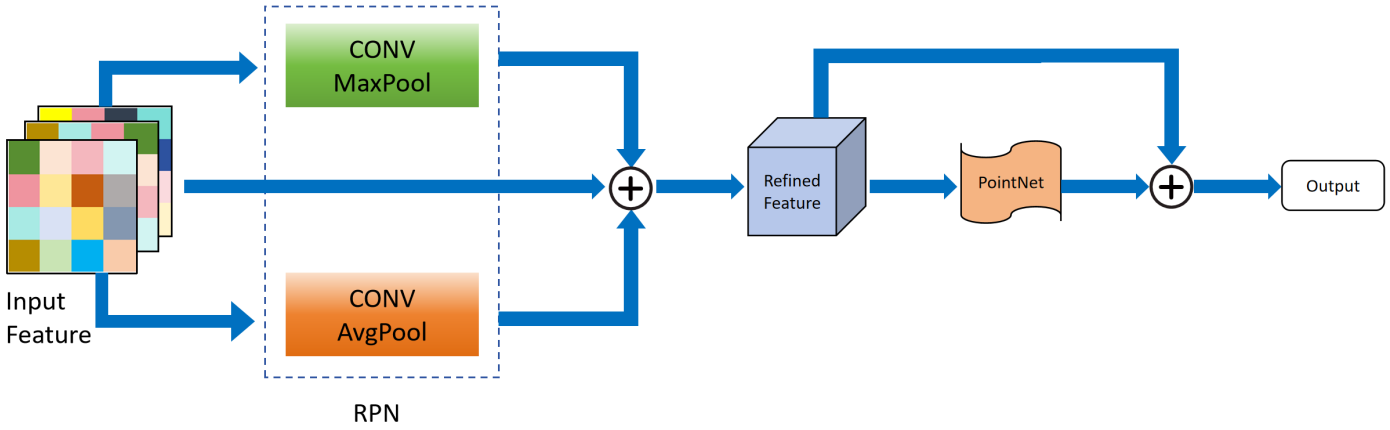


**Figure 3.** Structure of the TSDN.

are specifically designed to capture spatial structures and reduce computational complexity. The proposal of candidate regions is based on a sliding window approach, where the point cloud data within the window is analyzed by a small neural network to assess the presence of potential target objects. The output of the RPN is a set of bounding boxes, each with a score indicating the likelihood of containing an object. This process can be represented by the following formula:

$$Regions = Softmax(DenseConv(Flatten(V))) \quad (4)$$

where $V$ represents the voxelized data that has undergone preliminary processing, Flatten$(\cdot)$ reshapes the voxelized feature volume into a one-dimensional vector, DenseConv$(\cdot)$ denotes a sequence of densely connected convolutional layers that extract spatial features, and Softmax$(\cdot)$ normalizes the output scores to objectness probabilities. Regions denote the collection of candidate areas, enabling the RPN to efficiently filter and prioritize areas for further analysis in the object detection process.

**Second Stage:** Refinement Detection Network (RefinerNet). Once candidate regions are successfully identified, the second stage aims to perform a deeper analysis of these areas to refine the localization and classification of targets. This stage leverages local features extracted from PointNet along with preliminary candidate region information provided by the RPN. RefinerNet applies a more complex neural network structure to each candidate region, which includes deep convolutional networks and attention mechanisms to ensure accurate identification of target details from the refined features.

The refinement stage focuses on enhancing the precision of object detection, particularly in scenarios involving object edges and partial occlusions. This process can be described as follows:

$$F' = \text{ELU}(\text{BN}(\text{Conv}(Regions, F_{rpn}))) \quad (5)$$

$$result = \text{PointNet}(F'_{local}, F') \quad (6)$$

where $F_{rpn}$ denotes the backbone feature map produced by the RPN convolutional layers, $F'$ is the refined regional feature output of the convolutional refinement block, and $F'_{local}$ refers to the voxel-level point features from Eq. (3) aggregated within each candidate region to provide fine-grained geometric detail. These combined features are crucial for refining the detection and classification of objects within the specified areas, ensuring higher accuracy and a more detailed understanding of the scene. Finally, a lightweight PointNet is employed to obtain the detection results.

Through this two-stage detection network, the advantages of rapid detection and precise detection are effectively combined. In the first stage, by swiftly filtering through a large number of spatial areas to identify potential candidate regions, the computational resources required for subsequent processing are significantly reduced. In the second stage, resources are concentrated on in-depth analysis to ensure the accuracy and reliability of detection. This design enables the network to maintain real-time processing capabilities while also meeting the demands for high-precision detection. This approach not only enhances performance but also optimizes operational efficiency, making it particularly suited for applications where both speed and accuracy are critical.

### 3.3 Real-time Processing Capability

To ensure that the 3D point cloud object detection framework operates efficiently in real-time

applications, several measures have been implemented in the design and optimization of its real-time processing capabilities. These measures include optimizing the network architecture and applying computational acceleration techniques.

**Network Architecture Optimization.** In designing the network architecture, the focus was specifically placed on reducing computational load and memory usage while maintaining high detection performance. The strategies adopted include:

- **Lightweight Network Layers:** Depthwise separable convolutions have been opted for instead of traditional convolutions. This type of convolution significantly reduces the model's parameter count and computational demands. Furthermore, depthwise separable convolutions decompose standard convolution operations, reducing the complexity of the model and accelerating data processing speed. This modification not only makes the network lighter and faster but also maintains an acceptable level of accuracy, making it highly suitable for real-time applications where speed is critical.
- **Network Pruning:** By applying network pruning techniques, non-essential neurons and connections are removed, thereby reducing the complexity of the model. This not only enhances the running speed but also decreases the model's storage requirements, making it more suitable for deployment on resource-constrained devices. Network pruning effectively trims redundant elements of the neural network without significantly impacting its predictive performance, optimizing the balance between efficiency and accuracy.

**Computational Acceleration Techniques.** In order to further enhance processing speed, the latest computational acceleration techniques have been employed:

- **GPU Acceleration:** Leveraging the power of Graphics Processing Units (GPUs) to handle parallel computations effectively. GPU acceleration works synergistically with training techniques such as the Adam optimizer [26] and Batch Normalization [27], which together accelerate convergence and stabilize the learning process, further enhancing overall training and inference efficiency. GPUs excel in dealing with matrix and vector operations common in deep learning, which allows for much faster processing than CPU-based computations. This is particularly beneficial for training and deploying deep neural networks where large amounts of data must be processed simultaneously.
- **Model Quantization:** Reducing the precision of the model's parameters from floating-point to lower-bit representations, which can significantly decrease the model size and speed up inference without a substantial loss in accuracy.

## 4 Experiments

To validate the effectiveness and efficiency of the approach, experiments were conducted on 3D object detection using the challenging KITTI benchmark [24] and the large-scale NEXET driving dataset (https://www.getnexar.com/nexet/). As demonstrated in robustness studies of autonomous driving perception systems [25], detection models that perform well on a single benchmark may degrade substantially when deployed across different regions, lighting conditions, or weather patterns, underscoring the need for evaluation on geographically and environmentally diverse data. The NEXET dataset directly addresses this requirement by covering over 77 countries, more than 1,400 cities, and a wide range of lighting, seasonal, and road conditions, making it a more demanding and ecologically valid testbed than region-specific benchmarks for validating the generalization of the proposed detection framework. Additionally, further ablation studies were carried out concerning the method. These experiments were designed to test various aspects of the detection framework under different scenarios and conditions.

### 4.1 Experiment Setup

**Training Details.** Under default conditions, the model was trained using four NVIDIA A30 GPUs, with each GPU handling 2 point clouds, resulting in a total of 8 point clouds processed concurrently. This setup ensures efficient training by distributing the computational load across multiple high-performance GPUs. The Adam optimizer [26] was utilized with a learning rate of 0.001 to optimize the network parameters. This choice of optimizer and learning rate helps in achieving a good balance between fast convergence and training stability. Additionally, Batch Normalization [27] was applied following each parameter layer to help stabilize the learning process by normalizing the inputs of each layer. This method is particularly effective in accelerating

the training phase and improving performance by reducing internal covariate shifts. A weight decay of 0.0001 was implemented in both networks to regularize the model and prevent overfitting. Weight decay works by adding a penalty to the loss function based on the magnitude of the weights, which encourages the model to maintain smaller weights and thus simpler models. The detailed parameters and configurations of the model training are comprehensively documented in Tables 1 and 2, encompassing critical architectural specifications including layer configurations, activation functions, and optimization hyperparameters. This systematic documentation ensures both methodological reproducibility and provides fundamental insights into the model's structural design and training dynamics.

**Data Augmentation:** Data augmentation is a critical technique, especially when the amount of available data is limited, to prevent overfitting during training. For each frame in the dataset, multiple data augmentation methods are employed to diversify the training examples: **Random Flipping:** Left-right random flipping is applied to the frames, which mirrors the point cloud data across the vertical axis. This process helps the model learn to recognize objects regardless of their orientation.

**Random Scaling:** Each frame is randomly scaled by a factor uniformly sampled from the range 0.95 to 1.05. This simulates the effect of objects appearing closer or further away from the sensor and encourages scale invariance in the model.

**Random Rotation:** Random rotation is performed on the entire scene for point clouds around the origin, with the degree sampled uniformly from -30° to 30°. This introduces rotational variance into the dataset, enhancing the model's ability to detect objects at different angles. Additionally, each ground-truth bounding box and its corresponding internal points are randomly perturbed by applying random transformations.

**Random Shift:** The shift for the bounding box is sampled from a predefined range for both the X and Y axes, and a different range for the Z axis, to account for the typical variations in the environment.

**Random Rotation Around the Z-axis:** The rotation angle is uniformly sampled from $-\pi$ to $\pi$, covering the full 360-degree rotation space around the Z-axis.

These data augmentation techniques collectively form a robust strategy to enhance the generalization capability of the 3D object detection framework, preparing it for effective performance in real-world scenarios.

### 4.2 Dataset and Evaluation Metrics

**Dataset:** Two classic datasets were selected to validate the effectiveness of the model under various scenarios: the KITTI and NEXET datasets.

The KITTI dataset contains 7,481 training images and 7,518 testing images for object detection, specifically for 2D/3D object detection tasks. For the evaluation of the test subset and comparison with other methods, results are submitted to the evaluation server. Following the paradigm set in [24], the dataset is divided into a training set (containing 3,712 images and point clouds) with around 14,000 car annotations and a validation set (containing 3,769 images and point clouds). Ablation studies are conducted on this split. For evaluation on the test set, the model is trained on the entire training set consisting of 7,000 point clouds.

The NEXET dataset is a large-scale video dataset extensively used in the field of autonomous driving. It covers over 77 countries, more than 1,400 cities, three lighting conditions (day, night, twilight), four seasons, multiple road conditions (urban, rural, highway, residential, and even desert roads), and various weather conditions (clear, foggy, rainy, snowy). The dataset provides high-fidelity video clips with rich annotation information, making it an ideal choice for assessing the proposed method.

The methods referenced above [8, 9] represent foundational prior work in 3D object detection that directly inform the dataset evaluation protocol adopted in this study. The dataset partitioning and use of full training sets for final test evaluations are standard practices to maximize the learned model's performance and generalizability. NEXET's diverse conditions offer a rigorous testing ground to showcase the robustness of the proposed object detection method across a wide array of real-world driving scenarios.

### 4.3 Evaluation Metrics

In the experiments, the following evaluation methods are adopted to comprehensively assess the model's performance:

**Precision:** Precision is the ratio of correctly predicted positive observations to the total predicted positives. It focuses on the quality of the model's outputs. The

**Table 1.** Experimental environment demonstrated.

| Parameter | Configuration |
|---|---|
| CPU | Intel Core i9-12700KF |
| GPU | NVIDIA Tesla A30 (24 GB) |
| CUDA version | CUDA 11.7 |
| Python version | Python 3.9.13 |
| Deep learning framework | Pytorch 2.0.0 |
| Operating system | Ubuntu 22.04.2 |

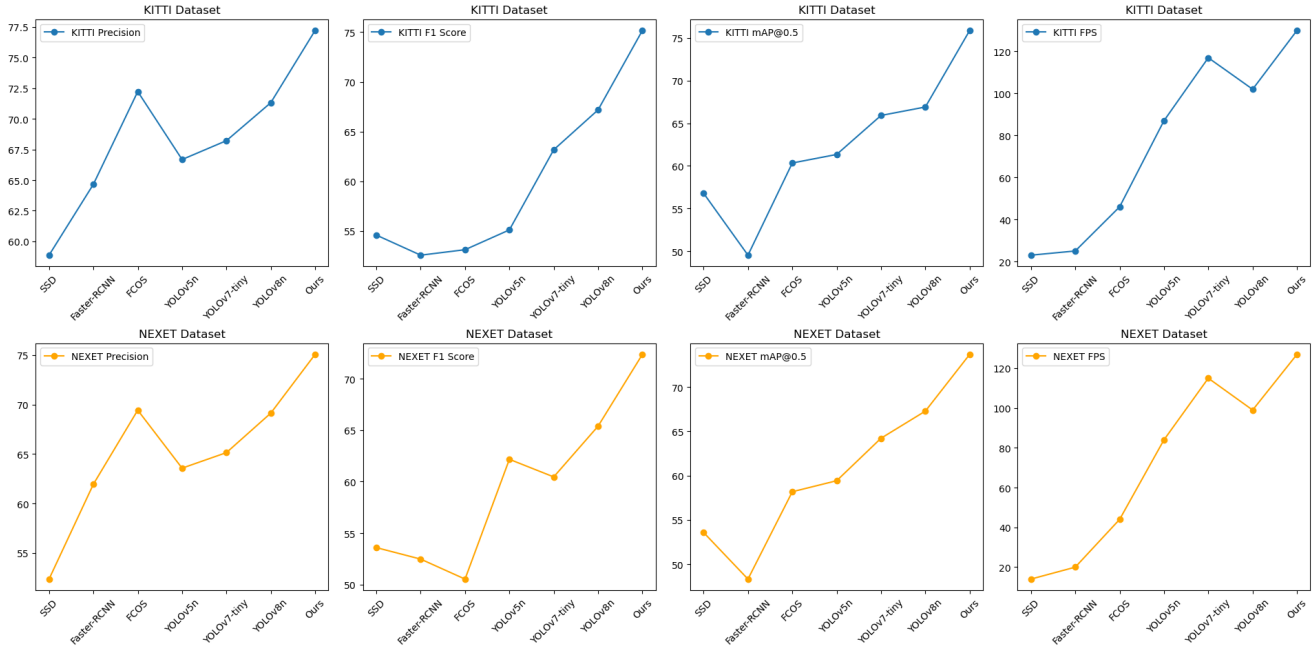


**Figure 4.** Visualization of results.

**Table 2.** Model parameters.

| Parameter | Value |
|---|---|
| Learning Rate | 0.001 |
| Optimizer | Adam |
| Batch Size | 16 |
| Weight Decay | 0.0001 |
| Training Epochs | 350 |
| Activation Function | ELU |
| Number of Layers | 252 |
| Early Stop | True |

formula for calculating precision is:

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives} \quad (7)$$

where $TruePositives$ refer to the number of positive samples that were correctly predicted, while False Positives refer to the number of samples that were incorrectly predicted as positive. Precision is particularly useful when the costs of false positives are high. In the context of object detection, a high precision score indicates that when the model predicts an object is present, it is likely to be correct.

**Recall:** Recall is the proportion of actual positive samples that are correctly identified as such by the model, focusing on the completeness of the model's output. The formula for calculating recall is:

$$Recall = \frac{TruePositives}{TruePositives + FalseNegatives} \quad (8)$$

where $FalseNegatives$ is the number of positive samples that the model incorrectly identified as negative. Recall is an important metric when it is crucial to capture as many positives as possible.

**F1 Score:** The F1 Score is the harmonic mean of precision and recall, used to measure the balance between precision and recall. Its calculation formula is:

$$F1Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (9)$$

This score conveys the balance between precision and recall in a single number, with the best value at 1

**Table 3.** Comparison of different methods on the KITTI and the NEXET datasets.

| Methods | KITTI | | | | NEXET | | | |
|---|---|---|---|---|---|---|---|---|
| | Precision | F1 Score | mAP@0.5 | FPS | Precision | F1 Score | mAP@0.5 | FPS |
| SSD [16] | 58.87 | 54.59 | 56.79 | 23 | 52.36 | 53.59 | 53.58 | 14 |
| Faster-RCNN [28] | 64.66 | 52.57 | 49.54 | 25 | 61.95 | 52.47 | 48.33 | 20 |
| FCOS [29] | 72.23 | 53.12 | 60.34 | 46 | 69.43 | 50.53 | 58.18 | 44 |
| YOLOv4 [30] | 66.68 | 55.11 | 61.34 | 87 | 63.57 | 62.17 | 59.42 | 84 |
| YOLOv7-tiny [31] | 68.22 | 63.17 | 65.91 | 117 | 65.14 | 60.46 | 64.21 | 115 |
| YOLOX [32] | 71.32 | 67.18 | 66.90 | 102 | 69.13 | 65.39 | 67.27 | 99 |
| Proposed method | 77.22 | 75.17 | 75.87 | 130 | 75.07 | 72.37 | 73.67 | 127 |

**Table 4.** Ablation analysis of the proposed method on KITTI and NEXET datasets.

| | Module | | KITTI | | | | NEXET | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | EDR | TSDN | Precision | F1 Score | mAP@0.5 | FPS | Precision | F1 Score | mAP@0.5 | FPS |
| (1) | | | 65.51 | 69.62 | 69.73 | 65 | 70.33 | 67.44 | 68.69 | 54 |
| (2) | ✓ | | 72.45 | 75.21 | 73.71 | 123 | 73.32 | 70.98 | 70.52 | 119 |
| (3) | | ✓ | 73.01 | 76.87 | 74.09 | 127 | 74.83 | 71.92 | 71.94 | 121 |
| (4) | ✓ | ✓ | 77.22 | 75.17 | 75.87 | 130 | 75.07 | 72.37 | 73.67 | 127 |

(perfect precision and recall) and the worst at 0. It is particularly useful when you need to compare two or more models or when the class distribution is imbalanced.

**Mean Average Precision(mAP):** Mean Average Precision is the mean of the Average Precision (AP) scores calculated across multiple classes. It is a crucial metric for evaluating the overall performance of a model in multi-class detection tasks. The calculation is as follows:

$$mAP = \frac{1}{N}\sum_{i=1}^{N} AP_i \tag{10}$$

where $N$ is the number of classes, and $AP_i$ is the Average Precision for the $i^{th}$ class. Each $AP_i$ is calculated by integrating the area under the precision-recall curve for that class across different thresholds. The mAP score provides a single performance measure that incorporates both precision and recall, and it is especially important in scenarios where each class has equal importance.

**mAP@[IoU]:** mAP@[IoU] is the Mean Average Precision calculated at different IoU thresholds. Intersection over Union (IoU) is a metric used to evaluate the overlap between predicted and true bounding boxes. The calculation formula for IoU is as follows:

$$mAP@[IoU] = \frac{1}{N}\sum_{i}^{N} AP@[IoU]_i \tag{11}$$

where $mAP@[IoU]_i$ is used to calculate the Average Precision (AP) for class i at a specific IoU threshold. In object detection tasks, it is common to set an IoU threshold (e.g., 0.5), and a prediction is considered correct only if the Intersection over Union (IoU) between the predicted bounding box and the true bounding box exceeds this threshold. mAP@[IoU] is calculated under this setting, and it is frequently used to evaluate the performance of a model under stricter matching criteria. This metric effectively measures how well the model is able to not just detect the objects but also accurately localize them by closely matching the predicted bounding boxes with the ground truth. The IoU threshold ensures that only predictions that meet a minimum standard of accuracy are considered, which helps to emphasize the quality of the detections. This is particularly important in applications where precision in object localization is critical, such as in autonomous driving, where misjudging the location of an object could lead to incorrect decision-making.

### 4.4 Experimental Results and Analysis

Table 3 and Figure 4 present the comparative performance analysis of state-of-the-art object detection methods across two benchmark datasets, reporting key evaluation metrics including Precision, F1 Score, mAP@0.5, and inference speed (FPS). These quantitative measurements enable a rigorous and comprehensive assessment of detection accuracy and computational efficiency. On the KITTI dataset, the proposed method achieved the best results, reaching

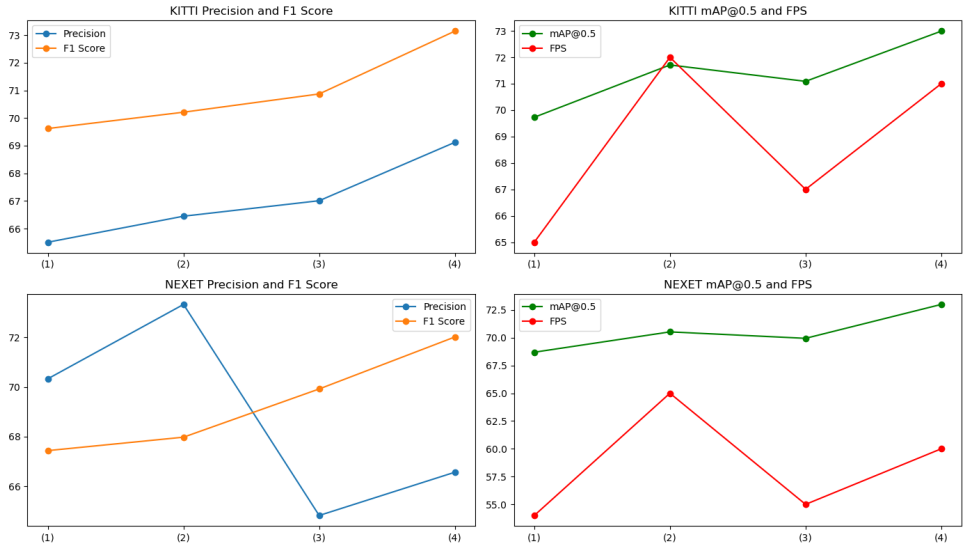


**Figure 5.** Visualization of ablation experiment results. Among them (1) means not to add any module; (2) means to add only the EDR module; (3) means to add only the TSDN module; (4) means to add two modules.

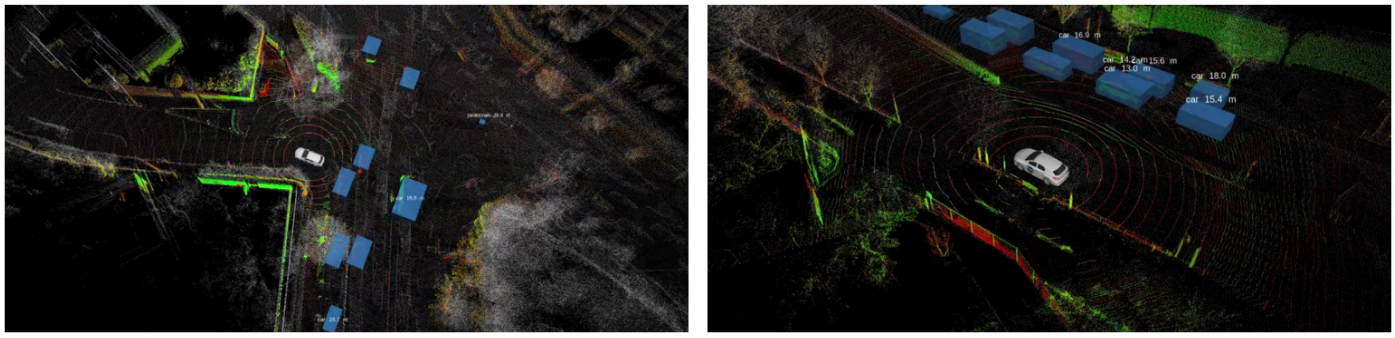

**Figure 6.** Visualization of object detection results.

scores of 77.22% in Precision, 75.17% in F1 Score, 75.87% in mAP@0.5, and 130 FPS. These results not only demonstrate its exceptional performance but also highlight its significant advantage in processing speed. For the NEXET dataset, the proposed method also displayed outstanding performance, specifically achieving 75.07% in Precision, 72.37% in F1 Score, 73.67% in mAP@0.5, and 127 FPS. These results further substantiate the consistency of the proposed method's efficiency and accuracy across different scenarios. The quantification and visualization results are shown in Figures 5 and 6, demonstrating that the proposed method yields favorable outcomes in real-world scenarios. In summary, the adoption of this method has led to significant breakthroughs in both speed and accuracy, while also validating the model's feasibility in practical production environments.

### 4.5 Ablation Study

As shown in Table 4, the results of the ablation studies compare the specific impacts of different module combinations on model performance. Through detailed analysis, the experimental results reveal their individual and combined influences on Precision (Pre), F1 Score, mAP@0.5, and frames per second (FPS). The baseline experiment did not utilize any of the new modules, while subsequent experiments progressively introduced modules to explore the stepwise improvement in model performance. Ultimately, when all modules were integrated, the experimental results achieved the best performance. This series of ablation studies not only verifies the importance of each module in enhancing model performance but also provides strong evidence for designing efficient object detection models, emphasizing the effectiveness of the proposed model in the field of autonomous driving.

## 5 Discussion

This study successfully develops and validates a novel framework for rapid 3D point cloud object detection. Extensive testing on the KITTI and NEXET datasets demonstrates significant advantages

in terms of real-time processing speed and accuracy. These achievements not only advance the technology of 3D point cloud processing but also provide robust technical support for real-time applications such as autonomous driving and robotic navigation. The proposed hybrid data representation method effectively balances processing speed and detection accuracy by combining the benefits of voxelization and direct point feature extraction. This strategy addresses the issue of information loss inherent in traditional voxelization methods while overcoming the high computational costs associated with directly processing point cloud data. Ablation studies further demonstrate the significant contribution of voxelization and point feature extraction to enhancing system performance, offering valuable insights for future research.

Despite the excellent performance of the proposed method in current experiments, there are still some limitations and directions for future improvement. First, while the hybrid data representation method has optimized information retention and computational efficiency, reducing computation time further remains a challenge. Moreover, while the proposed method performs well in vehicle and pedestrian detection tasks, its effectiveness in more complex or dense scenes requires further validation. Future work could explore more efficient voxelization techniques and point feature extraction algorithms to further enhance the system's real-time capabilities and robustness.

## 6 Conclusion

This paper proposes a novel framework for fast 3D point cloud object detection, tailored for complex applications that require real-time responses, such as autonomous driving and robotic navigation. The method is primarily based on an innovative hybrid data representation technique that effectively combines the computational efficiency of voxelization with the high precision of direct point processing, resolving the traditional trade-off between speed and accuracy. The developed hybrid data representation method significantly enhances data processing speed by utilizing both voxelization and point feature extraction, while preserving critical spatial and structural information, thereby improving object detection accuracy. A two-stage architecture is designed, including a rapid region proposal network and a refinement detection network. This design not only enhances detection efficiency but also optimizes the recognition and localization of multiple targets in complex scenes. Extensive experimental results demonstrate that the proposed method significantly outperforms current leading technologies on the KITTI and NEXET datasets in terms of object detection accuracy and processing speed. In particular, in terms of processing speed, the method meets the stringent requirements of real-time applications. Future work will focus on further exploring more efficient voxelization techniques and advanced point feature extraction methods to continuously optimize system performance. Additionally, there are plans to expand the framework to accommodate a wider range of 3D sensory data and more complex application scenarios.

## Data Availability Statement

Data will be made available on request.

## Funding

This work was supported without any funding.

## Conflicts of Interest

The author declares no conflicts of interest.

## Ethical Approval and Consent to Participate

Not applicable.

## References

[1] Arnold, E., Al-Jarrah, O. Y., Dianati, M., Fallah, S., Oxtoby, D., & Mouzakitis, A. (2019). A survey on 3d object detection methods for autonomous driving applications. *IEEE Transactions on Intelligent Transportation Systems, 20*(10), 3782-3795. [CrossRef]

[2] 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).

[3] Lang, A. H., 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).

[4] 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. 10529-10538).

[5] Yan, Y., Mao, Y., & Li, B. (2018). Second: Sparsely embedded convolutional detection. *Sensors, 18*(10), 3337. [CrossRef]

[6] Maturana, D., & Scherer, S. (2015, September). Voxnet: A 3d convolutional neural network for real-time object

recognition. In *2015 IEEE/RSJ international conference on intelligent robots and systems (IROS)* (pp. 922-928). Ieee. [CrossRef]

[7] Wen, L. H., & Jo, K. H. (2021). Fast and accurate 3D object detection for lidar-camera-based autonomous vehicles using one shared voxel-based backbone. *IEEE access, 9*, 22080-22089. [CrossRef]

[8] 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).

[9] 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).

[10] 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).

[11] 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, 30*.

[12] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 779-788).

[13] Duan, K., Bai, S., Xie, L., Qi, H., Huang, Q., & Tian, Q. (2019). Centernet: Keypoint triplets for object detection. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 6569-6578).

[14] Bolya, D., Zhou, C., Xiao, F., & Lee, Y. J. (2019). Yolact: Real-time instance segmentation. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 9157-9166).

[15] Zhang, S., Wen, L., Bian, X., Lei, Z., & Li, S. Z. (2018). Single-shot refinement neural network for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 4203-4212).

[16] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016). Ssd: Single shot multibox detector. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14* (pp. 21-37). Springer International Publishing. [CrossRef]

[17] Tan, M., Pang, R., & Le, Q. V. (2020). Efficientdet: Scalable and efficient object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 10781-10790).

[18] Lin, T. Y., Dollár, P., Girshick, R., He, K., Hariharan, B., & Belongie, S. (2017). Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2117-2125).

[19] Liu, S., Qi, L., Qin, H., Shi, J., & Jia, J. (2018). Path aggregation network for instance segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 8759-8768).

[20] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., ... & Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*. [CrossRef]

[21] Wu, B., Iandola, F., Jin, P. H., & Keutzer, K. (2017). Squeezedet: Unified, small, low power fully convolutional neural networks for real-time object detection for autonomous driving. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops* (pp. 129-137).

[22] Wang, R. J., Li, X., & Ling, C. X. (2018). Pelee: A real-time object detection system on mobile devices. *Advances in neural information processing systems, 31*.

[23] Qin, Z., Li, Z., Zhang, Z., Bao, Y., Yu, G., Peng, Y., & Sun, J. (2019). ThunderNet: Towards real-time generic object detection on mobile devices. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 6718-6727).

[24] Geiger, A., Lenz, P., & Urtasun, R. (2012, June). Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE conference on computer vision and pattern recognition* (pp. 3354-3361). IEEE.

[25] Unal, D., Catak, F. O., Houkan, M. T., Mudassir, M., & Hammoudeh, M. (2023). Towards robust autonomous driving systems through adversarial test set generation. *ISA transactions, 132*, 69-79. [CrossRef]

[26] Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*. [CrossRef]

[27] Ioffe, S., & Szegedy, C. (2015, June). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning* (pp. 448-456). pmlr.

[28] Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems, 28*.

[29] Tian, Z., Shen, C., Chen, H., & He, T. (2019). Fcos: Fully convolutional one-stage object detection. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 9627-9636).

[30] Bochkovskiy, A., Wang, C. Y., & Liao, H. Y. M. (2020). Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*. [CrossRef]

[31] Wang, C. Y., Bochkovskiy, A., & Liao, H. Y. M. (2023). YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 7464-7475).

[32] Ge, Z., Liu, S., Wang, F., Li, Z., & Sun, J. (2021). Yolox: Exceeding yolo series in 2021. *arXiv preprint arXiv:2107.08430*. [CrossRef]