frontiers | Frontiers in Neurorobotics

TYPE Original Research
PUBLISHED 20 January 2025
DOI 10.3389/fnbot.2024.1484276



OPEN ACCESS

EDITED BY
Fei Wang,
Dalian Maritime University, China

REVIEWED BY
Zhensheng Shi,
Ocean University of China, China
Rui Ma,
Dalian Maritime University, China
He Shuqian,
Hainan Normal University, China

*CORRESPONDENCE
Guojing Li
li_guojing@126.com

RECEIVED 21 August 2024
ACCEPTED 30 December 2024
PUBLISHED 20 January 2025

CITATION
Zhao H, Zhang S, Peng X, Lu Z and Li G (2025) Improved object detection method for autonomous driving based on DETR. *Front. Neurorobot.* 18:1484276. doi: 10.3389/fnbot.2024.1484276



# Improved object detection method for autonomous driving based on DETR

Huaqi Zhao[1], Songnan Zhang[1], Xiang Peng[1], Zhengguang Lu[1] and Guojing Li[2]*

[1]The Heilongjiang Provincial Key Laboratory of Autonomous Intelligence and Information Processing, School of Information and Electronic Technology, Jiamusi University, Jiamusi, China, [2]School of Materials Science and Engineering, Jiamusi University, Jiamusi, China

Object detection is a critical component in the development of autonomous driving technology and has demonstrated significant growth potential. To address the limitations of current techniques, this paper presents an improved object detection method for autonomous driving based on a detection transformer (DETR). First, we introduce a multi-scale feature and location information extraction method, which solves the inadequacy of the model for multi-scale object localization and detection. In addition, we developed a transformer encoder based on the group axial attention mechanism. This allows for efficient attention range control in the horizontal and vertical directions while reducing computation, ultimately enhancing the inference speed. Furthermore, we propose a novel dynamic hyperparameter tuning training method based on Pareto efficiency, which coordinates the training state of the loss functions through dynamic weights, overcoming issues associated with manually setting fixed weights and enhancing model convergence speed and accuracy. Experimental results demonstrate that the proposed method surpasses others, with improvements of 3.3%, 4.5%, and 3% in average precision on the COCO, PASCAL VOC, and KITTI datasets, respectively, and an 84% increase in FPS.

KEYWORDS

**object detection, feature extraction, transformer encoder, loss function, parameter tuning**

## 1 Introduction

Autonomous driving technology utilizes a combination of sensor technology, artificial intelligence, big data analysis and processing, and computer vision to enable computers to safely drive vehicles with partial or unmanned intervention. Object detection plays a crucial role in recognizing targets during autonomous driving and assists the central control system in providing necessary driving commands.

In autonomous driving scenarios, objects such as vehicles, pedestrians, and traffic signs are distributed across multiple scales depending on their distance. Distant pedestrians and traffic signs often appear as small targets, while nearby vehicles dominate the frame as large targets. Object size and appearance vary significantly due to differences in distance and angle. Multi-scale feature extraction addresses this challenge by capturing multi-level features simultaneously, enhancing the robustness of detecting objects across various scales (Lin et al., 2017). Additionally, complex backgrounds such as buildings and trees often interfere with object detection, while the dynamic nature of targets' such as moving pedestrians and vehicles' further complicates the task. Transformer-based

architectures excel in modeling global contextual relationships, enabling them to adapt effectively to dynamic target variations, particularly in scenarios involving occlusions. Attention mechanisms offer a practical solution for resource-constrained embedded devices by efficiently allocating computational resources. By focusing computational power on key object regions, attention mechanisms significantly enhance real-time performance. Moreover, autonomous driving requires addressing multiple optimization objectives, such as the simultaneous classification and localization of objects. Adjusting and designing loss function weights can further optimize the model's multi-objective learning capabilities (Ou et al., 2023).

The integration of these technologies—multi-scale feature extraction, attention mechanisms, and optimized loss functions—provides greater accuracy, robustness, and efficiency for object detection in autonomous driving. These advancements establish a foundation for developing intelligent autonomous driving systems capable of reliable performance in complex real-world scenarios.

Object detection involves both traditional and deep-learning-based methods. Traditional methods typically include generating candidate frames, extracting features, and performing classification (Dalal and Triggs, 2005). Non-maximum suppression (NMS) (Neubeck and Van Gool, 2006) is then used to remove redundant candidate boxes. However, traditional methods rely heavily on manual design and feature selection, exhibit low efficiency and poor robustness, and are not capable of handling real-time autonomous driving. Deep learning architectures have emerged as two-stage and one-stage methods. Two-stage methods are based on convolutional neural networks for classification, of which Fast R-CNN is representative, with a high detection accuracy; however, it is still unable to eliminate the NMS process and cannot realize end-to-end detection (Ren et al., 2015). One-stage methods are based on convolutional neural networks for regression and perform better in terms of inference speed. Redmon et al. (2016) first proposed YOLO, and this series of algorithms (Ge et al., 2021; Li et al., 2023; Yung et al., 2022) occupies a dominant position among one-stage algorithms, with a wide range of industrial applications.

Transformer-based object detection methods have demonstrated significant application potential in autonomous driving technology. As the demands for adaptability to complex scenarios, real-time performance, and multimodal data processing in autonomous driving continue to increase, transformers, with their exceptional global modeling capabilities and end-to-end optimization framework, have become a key driving force in advancing perception technology for autonomous driving. In autonomous driving scenarios, challenges such as occlusion, lighting variations, and complex backgrounds are common. The global self-attention mechanism of transformers can accurately capture the global contextual information of the input data, thereby effectively separating objects from the background.

The architecture of the transformer, which is extensively used in natural language processing (NLP), has recently attracted interest in the field of computer vision (Vaswani et al., 2017). Carion et al. (2020) introduced the detection transformer (DETR), which reframes object detection as an ensemble prediction task, eliminating the need for NMS operations. This approach enables end-to-end object detection with enhanced global modeling capabilities, outperforming Fast R-CNN. In autonomous driving, objects are often occluded by buildings or other objects, DETR can infer the targets in occluded areas using global contextual information. Subsequently, Zhu et al. (2020) proposed Deformable DETR, which incorporates a deformable attention module to focus attention selectively on specific sampling points within the feature map, which reduces the computational overhead and accelerates training. Wang et al. (2022) presented the anchor DETR, integrating an anchor point mechanism into query vectors to address the issue of poor interpretability. DINO-DETR leverages a comparative training denoising method and hybrid query selection strategy for anchor point initialization (Zhang et al., 2022). Zong et al. (2023) introduced the H-DETR algorithm, which employs a hybrid matching approach during the Hungarian matching phase and incorporates one-to-many matching branches, offering a novel avenue for enhancement. BEVFormer (Li et al., 2024) achieves a more precise bird's-eye-view (BEV) environment modeling in complex scenarios, providing robust support for path planning and decision-making in autonomous driving systems. Mushtaq et al. (2024) proposed PLC-Fusion, which leverages transformer architectures to extract features from both images and point clouds, significantly enhancing the accuracy and efficiency of multimodal object detection.

Despite the diverse improvement perspectives provided by previous studies, in the field of autonomous driving, DETR-like detectors still suffer from the following problems:

The limited capability to detect objects across different scales, as well as lack of precision in determining the exact positions of objects, results in suboptimal detection accuracy in autonomous driving situations.

The performance of the model is hindered by the attention mechanism layer within the encoder, particularly when dealing with higher-resolution images. This results in a considerable increase in computational cost and memory complexity, which, in turn, affects both model accuracy and inference speed.

During the training phase, the hyperparameters, including the weight of each loss function, are manually set. However, this manual approach incurs a high cost for tuning the parameters and overlooks the dynamic balancing issue between the loss functions. Consequently, the model experiences slow convergence and lacks convergence accuracy.

Therefore, we propose an improved autonomous driving object detection method based on DETR, which contains three improvements:

- We propose a multi-scale feature and location information extraction method. A network that incorporates multi-scale residual partition units with a coordinate attention module was designed to improve the multi-scale detection and position sensing capabilities.
- We designed a transformer encoder based on an efficient attention mechanism. A grouped attention mechanism layer is deployed in the encoder, which computes the attention region in parallel in the horizontal and vertical groups, fully learns the image features from different directions, and maintains a balance between local and global information by controlling the range of attention

computation. This effectively reduces the computational overhead and improves average precision (AP) and inference speed.

- We propose a novel dynamic hyperparameter tuning method based on Pareto efficiency, which involves automatically updating the weights of various loss functions during the training process, ensuring continuous coordination of their training states. The aim of this approach was to accelerate the convergence process and improve the accuracy of the final convergence of the detector.

## 1.1 Feature extraction in object detection

The feature extraction network utilizes multi-layered techniques to capture semantic information from images, focusing on two main approaches: convolutional neural network-based feature extraction network and transformer-based feature extraction network. Obtaining diverse object characteristics, particularly multi-scale features, not only speed up the convergence of the model but also greatly enhances the detection capability. ViT (Dosovitskiy et al., 2020), as a pioneering work to implement the transformer architecture in computer vision, has a larger receptive field and modeling capability than convolutional neural networks. When ViT is used as a feature extractor, it imposes a significant burden on model training. If multi-scale feature extraction is performed in such a case, this obviously expands on such drawbacks. DN-Deformable-DETR employs the Swin Transformer (Li et al., 2022), which has less computational overhead but still fails to capture multi-scale features at the corresponding stages.

In contrast, convolutional neural networks progressively deepen the extraction of features through convolution operators. This inherent property of extracting multi-scale features is advantageous for solving multi-scale feature problems. VGGNet uses stacked convolutional layers to solve multi-scale problems (Simonyan and Zisserman, 2014); however, the number of model parameters is large and inefficient. Lin et al. (2017) successfully deployed an FPN structure for object detection tasks; however, this approach seriously affects the inference speed and is even more inapplicable to the higher computational complexity of DETR-like models. DINO-DETR (Zhang et al., 2022) deploys convolution in the encoding and decoding phases; however, this increases the design difficulty of the model and significantly increases the amount of computation, which is not conducive to the development of a lightweight model. Recently, more efficient multi-scale feature extraction networks have emerged (Huang et al., 2017; Yu et al., 2018; Gao et al., 2019; Hou et al., 2021). Based on the characteristics of the object detection model, applying these networks in the backbone is a practical choice. In addition, current methods lack the ability to sense target location information (Hou et al., 2021), which is critical for improving the precision of object detection tasks. Building on previous research, our proposed network architecture focuses on extracting multi-scale features and precise location information to improve accuracy.

## 1.2 Transformer encoder

The transformer encoder is an essential part of the detector, and experiments have demonstrated that the encoder contributes approximately 11% to the AP but accounts for approximately 85% of the model's computational effort (Lin et al., 2022). The attention layer is the core of the encoder. It is more difficult for inefficient encoders to cope with autonomous driving object detection scenarios, which directly affects the inference speed.

DETR was the first to use the ViT module as an encoder, incorporating the transformer into the object detection framework by employing a multi-head attention (MSA) mechanism in the attention layer. MSA is a form of self-attention mechanism that converts a feature vector into a sequence, enabling the model to detect relationships between various components of the entire input through the representation of all possible interactions between elements within a sequence. In DETR, the global attention mechanism is computationally intensive, which leads to difficulties in model training. The Swin Transformer employs the idea of local attention, which limits attention computation to a fixed window and reduces the computational overhead (Liu Z. et al., 2021). On this basis, Shuffle Transformer further enhances the information exchange between windows over long distances by spatial shuffling (Huang et al., 2021), and CS Transformer employs "cross-shaped window attention" to improve computational efficiency (Dong et al., 2022). MobileVit employs a hybrid architecture that combines VIT and CNN for initial deployment on mobile devices (Mehta and Rastegari, 2021). ElasticViT first trains a high-quality VIT super-network and subsequently determines the best sub-network to be deployed to further reduce latency (Tang et al., 2023). In summary, this study focused on designing an efficient attention layer for encoders.

## 1.3 Optimization of model training parameters

During the training phase of an end-to-end network, multiple loss functions that handle both regression and classification tasks are commonly employed. However, it is often overlooked that these loss functions can interact with one another, significantly affecting model performance based on their relative weightings. Kendall et al. (2018) introduced a method of uncertainty to weigh losses by employing a Bayesian framework that emphasizes prediction uncertainty to automatically set weights for these loss functions. Mahapatra and Rajan (2020) enhanced the gradient-based multi-objective optimization algorithm by considering the loss function as multiple targets, assigning upper bounds to them, and successfully applying this optimization across different deep learning tasks. Lin et al. (2019) developed an algorithmic framework to ensure Pareto efficiency, thereby guaranteeing compliance with the Pareto condition and successful application of the optimization algorithm. Liu X. et al. (2021) introduced a novel gradient optimization algorithm using Stein variational gradient descent (SVGD) to analyze the Pareto frontier, which effectively solves high-dimensional problems and yields more uniformly distributed and diversified solutions on the Pareto front, thus optimizing the model. Lin et al. (2021) also proposed a random weighting approach, which includes both random loss and random gradient weighting and demonstrated improved generalization in experimental settings. These contributions present optimization strategies for model parameters from two major perspectives:

the dynamic weighting of loss functions and gradients. However, these approaches are broad, and there has been limited research specifically targeting object detection models. Building on these advancements, this study focused on an algorithm designed to adjust the weights of the loss functions.

## 2 Improved object detection method for autonomous driving based on DETR

The method is composed of three parts, with the framework illustrated in Figure 1, including the multi-scale feature and location information extraction method, Transformer encoder based on the group axial attention mechanism, and dynamic hyperparameter tuning training method based on Pareto efficiency. To perform object detection for autonomous vehicle driving, the backbone network captures features from the image intended for detection and subsequently passes the feature map to the encoder. Following the encoding process of the feature map, vectors K and V are generated by the encoder and fed into the decoder in conjunction with the query vector Q. Finally, following the dynamic hyperparameter tuning training method based on Pareto efficiency, the prediction head captures the output of the decoder and ultimately derives the desired target information.

### 2.1 Multi-scale feature and location information extraction method

Object detection is a crucial task that relies on both multi-scale features and precise location information of the target. Our research introduces a multi-scale feature and location information extraction method designed to acquire detailed features at multiple scales and enhance the target location information by improving the backbone network structure, as illustrated in Figure 2. The initial step involves passing the image input through a detection network consisting of four stages. Each stage comprises two types of modules: one for extracting multi-scale features and one for coordinating attention. At the beginning of each stage, the feature map is input into the multi-scale feature extraction module. This process integrates various residual units into the convolutional structure to extract features of different scales from the image. Subsequently, the feature map proceeds to the coordinate attention module, which captures the positional details from the extracted multi-scale features, further enhancing the detection capability.

#### 2.1.1 Multi-scale feature extraction module

The input feature map $X \in R^{H\times W\times C}$ is partitioned into $n$ groups after $1 \times 1$ convolution, and each group of data is represented by $X_i$ , where $i \in \{1, 2, ...n\}$, and the number of channels shrinks $n$ minus a multiple into the residual unit branch, as shown in Figure 2. In each branch, except for $X_1$, each group of data undergoes a $3 \times 3$ convolution operation, denoted as $K_i$, and the output is denoted as $Y_i$. $Y_i$ is derived from the $i$-th group of data and the output $Y_{i-1}$ of group $i-1$ after. $Y_i$ is defined as follows:

$$Y_i = \begin{cases} X_i & i = 1; \\ K_i\left(X_i\right) & i = 2; \\ K_i\left(X_i + Y_{i-1}\right) & 2 < i \leq \text{n}. \end{cases} \tag{1}$$

where the previously processed features $Y_{i-1}$ are included in the convolution operation $Y_{i-1}$ of the $i$-th group when $i > 2$. Segmentation is processed in a multi-scale manner, and each time the segmented feature $X_i$ undergoes a $1 \times 1$ convolution, it expands the receptive field of $X_i$. The outputs of all the different scales $Y_i$, which are spliced in the channel dimension, are subjected to a $1 \times 1$ convolution operation to obtain a fused image, $Z$, which facilitates the extraction of the previous procedure, as shown in Equation 2:

$$Z = Conv(Concat(Y_1, ..., Y_n)) \tag{2}$$

where $Y_1$ is directly output without convolution, which serves to reuse the features. The methods described above enable the extraction of characteristics at a more precise level of granularity, featuring varied receptive fields and multiple scales. The above process achieves the progressive fusion of multi-scale features, where each set of convolutions relies on the output of the previous set. This step-by-step stacking mechanism captures features at different scales. The outputs of the $s$ groups are aggregated through concatenation, forming a more powerful representation. Mathematically, this feature extraction process is similar to recursive convolution, ensuring that features are adequately expressed across different scales.

#### 2.1.2 Coordinate attention module

The module for coordinates exploits the location information in feature maps across channels, which not only aids in model recognition and localization of specific areas but also improves the detection of distant relationships in visual tasks. Figure 2 illustrates the use of an attention module that employs a network with branches to compute attention weights. These weights are subsequently multiplied with the initial feature map to generate the ultimate output. To process feature map $X \in R^{H\times W\times C}$, adaptive mean pooling is conducted in the height and width directions. Combining features across spatial dimensions, these two operations produce a collection of direction-sensitive feature maps with dimensions $C \times H \times 1$ and $C \times 1 \times W$, respectively. Through these transformations, remote relationships are captured along one spatial dimension while maintaining precise position information along the other, as demonstrated in Equations 3, 4:

$$z_c^h(h) = \frac{1}{W} \sum_{0\leq i\leq W} X_c(h, i) \tag{3}$$

$$z_c^w(w) = \frac{1}{H} \sum_{0\leq i\leq H} X_c(j, w) \tag{4}$$

where $z_c^h(h)$ and $z_c^w(w)$ denote the outputs of the $C$-th channel for height $h$ and width $w$, respectively. After the above operations, the position information is encoded, and the feature map is generated.

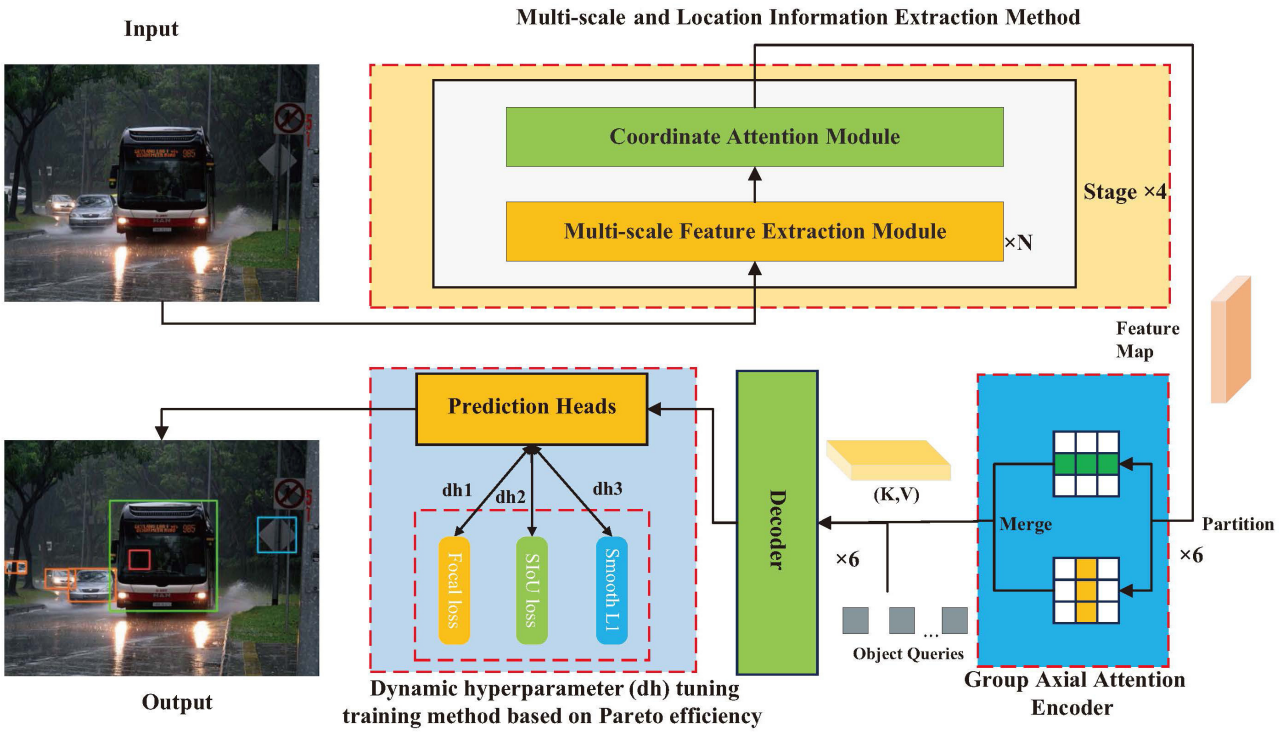


FIGURE 1
Overview the proposed method.

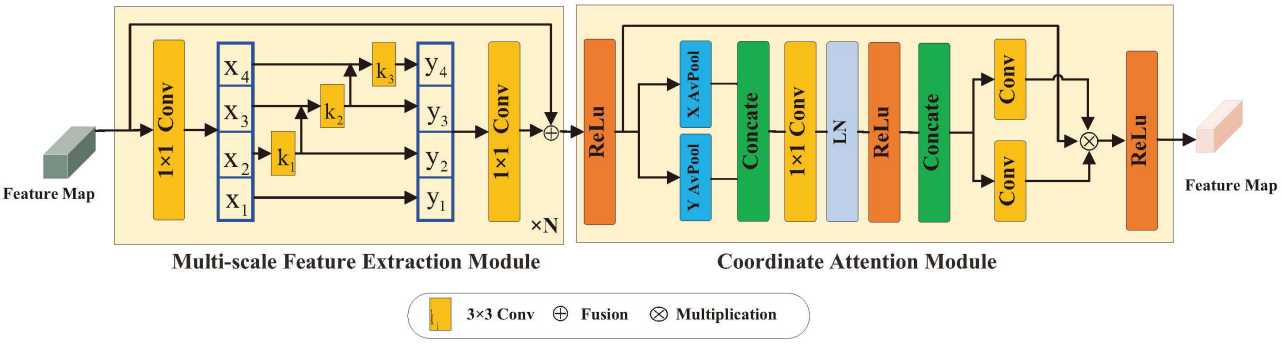


FIGURE 2
Multi-scale feature and location information extraction method.

First, the outputs of Equations 3, 4 are spliced and fed into a 1 × 1 convolution $F_1$, as shown in Equation 5:

$$f = \delta(F_1[Z^h, Z^w]) \tag{5}$$

where [,] denotes splicing by the spatial dimension, $\delta$ is the non-linear activation function, $f \in R^{C/r\times(H+W)}$ denote the feature maps generated in the horizontal and vertical directions, $f^h \in R^{C/r\times H}$ and $r$ denotes the reduction multiplier. Subsequently, $f \in R^{C/r\times(H+W)}$ is divided into $f^h \in R^{C/r\times H}$ and $f^w \in R^{C/r\times W}$ by spatial dimension, and $f^h$ and $f^w$ are adjusted into tensors with the same number of channels using two 1 × 1 convolutions, as shown in Equations 6, 7:

$$g^h = \sigma(F_h(f^h)) \tag{6}$$

$$g^w = \sigma(F_w(f^w)) \tag{7}$$

where $\sigma$ is the sigmoid function, $g^h$ and $g^w$ are the attention weights, and the output coordinate attention feature map is given by

$$Y_c(i,j) = X_c(i,j) \times g^h(i) \times g^w(j) \tag{8}$$

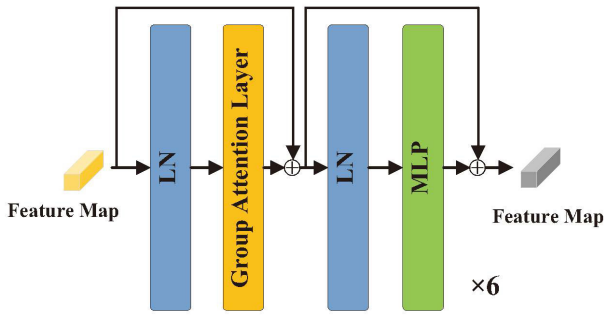


FIGURE 3
Structure of encoder.

## 2.2 Transformer encoder based on group axial attention mechanism

Although global attention mechanisms are effective in modeling long-range dependencies, they can result in high computational costs in downstream tasks, such as object detection, particularly with high-resolution images. Some approaches restrict attention to a window, which can hinder the information exchange between windows and limit the receptive field. Therefore, we introduce a transformer encoder that utilizes a group axial attention layer. Unlike in traditional transformer encoders, the input features in the group axial attention layer are divided into horizontal and vertical groups based on their dimensions. Subsequently, self-attention is calculated separately within each group before merging and mapping to the output. This approach allows for the comprehensive learning of image features from various orientations while maintaining a balance between local and global information by controlling the attention range. As a result, it effectively reduces computational costs and enhances inference speed and accuracy.

### 2.2.1 Structure of the encoder

As illustrated in Figure 3, the encoder consists of six identical layers for the input $T \in R^{H\times W\times C}$, which is transformed into a matrix $T \in R^{N\times C}$. Before entering the encoder, the structure is represented by

$$\begin{cases} Z'_L = T_{L-1} + GL\left(T_{L-1}\right) \\ Z_L = LN\left(Z'_L\right) + FFN\left(Z'_L\right), L = 1, 2, \ldots, 6 \\ Y = Z_6 \end{cases} \tag{9}$$

where $LN()$, $FFN()$, and $GL()$ denote layer normalization, feed-forward neural network, and group axial attention layer, respectively $T_{L-1}$ denotes the output of the layer $L-1$ encoder; $Z'_L$ and $Z_L$ denote the results from the layer $L$ encoding process; and $Y \in R^{N\times C}$ denotes the output of the layer 6 encoder, i.e., the final encoding result for the feature map $T \in R^{H\times W\times C}$.

### 2.2.2 Group axial attention layer

As shown in Figure 4, for the input feature vector $X \in R^{H\times W\times C}$, $H$ and $W$ denote the height and width, respectively. $Liner1$ maps the channel dimension $C$ to $dim$, yielding $X \in R^{H\times W\times \dim}$, which serves as the input to group axial attention layer. The hyperparameter $K$ is set to the number of heads, and $C$ is divided into two parts, $X^h \in R^{H\times W\times \dim/2}$ and $X^v \in R^{H\times W\times \dim/2}$, according to the $dim$, representing the horizontal and vertical groups, respectively. In the horizontal group, the length of the horizontal layer $h_i$ is set to $s$, and $h_i$ is $\frac{W}{s}$. Similarly, the number of vertical layers $j$ is $\frac{H}{s}$. In each horizontal layer $h_i$, $h_i \in R^{s\times W\times \dim/2}$ is serialized as a vector $\eta_i \in R^{sW\times \dim/2}$, and $K/2$ heads are set for the computation of the multi-head attention. The vertical group divides the vertical layer $v_j$ by $s$ and performs the same operation. The horizontal and vertical groups are spliced, and $dim$ is mapped back to $C$ by the $Liner2$. The group axial attention computation process is shown in Equation 10:

$$\begin{cases} X = Liner1(X) \\ X^h, X^v = Spilt(X) \\ [h_1, h_2, ..., h_i] = Spilt(X^h), [v_1, v_2, ..., v_j] = Spilt(X^v) \\ H - Attention = Concat[MSA(h_1)MSA(h_2), ..., MSA(h_i)] \\ V - Attention = Concat[MSA(v_1)MSA(v_2),..,MSA(v_j)] \\ G - Attention = Concat\left[V - Attention, H - Attention\right] \\ \mathrm{X} = Liner2\left(G - Attention\right) \end{cases} \tag{10}$$

The group axial attention layer combines $H - Attention$ and $V - Attention$ in the channel dimension to create group axial attention. This layer maintains the feature map dimensions while evenly distributing the heads into different attention groups. This distribution helps capture various features and patterns in image sequences, ultimately improving the model's expressive and generalization capabilities. In the horizontal and vertical groups, the range of attention can be changed by adjusting $s$. Experiments have shown that the best accuracy is achieved when $s = [1, 1, 2, 2, 6, 6]$.

### 2.2.3 Analysis and variants

Our approach differs from traditional attention mechanisms in that it captures extensive features and manages the attention scope by separating horizontal and vertical layers. This method effectively reduces the computational complexity. When computing the group axial attention, the computational overhead of operations such as division is negligible, whereas the computation of the two branches is parallel. As shown in Figure 4, in each branch, features are learned from the subspace divided between multi-layer blocks, and mappings are added to the output to further enhance the encoder.

This section further explores the relationship between the attention mechanism and computational complexity. Assuming that the feature map size to be processed is $H \times W \times C$, the computational complexity is given by Equation 11 when using the multi-head attention mechanism (MSA). The attention range is set to s. When group axial attention (GA), the computational complexity is given by Equation 12. On this basis, Window G-Attention (WGA) is designed, and only the operations of splitting and merging windows are added, as shown in Figure 4, where the size of the window is set as $p \times p$, and calculation of the group axial

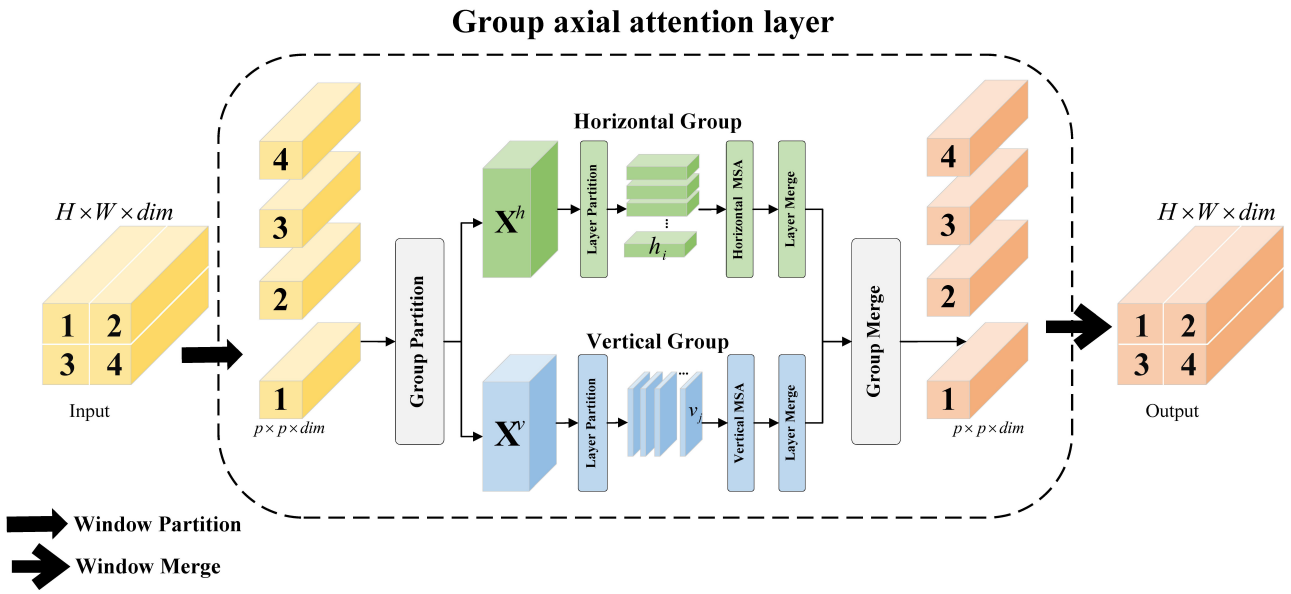


FIGURE 4
Illustration of GA and WGA.

attention is performed in a fixed window, as shown in Equation 13:

$$\Omega(MSA) = HWC * (4C + 2HW) \tag{11}$$

$$\Omega(GA) = HWC * (4C + sH + sW) \tag{12}$$

$$\Omega(WGA) = HWC * (4C + 2sp) \tag{13}$$

In practice, the computational volume increases sharply when $H$ and $W$ increase. $s$ is usually much smaller than $H$ and $W$, which requires less attention computation. In the calculation of WGA, $H = W$ is adjusted, and the operation of dividing the window is performed first. The window size is $p \times p$, which further reduces the amount of computation and facilitates faster inference. The value of $s$ should be appropriately adjusted as the encoders continue to stack. To reduce the limitation of window size on WGA, the value of $p$ should not be too small. The core principle of this approach is to divide the input feature map into multiple windows and compute local self-attention separately within the horizontal and vertical directions. The results are then concatenated and fused through linear projection to extract both local and global features. This significantly reduces computational complexity, making it efficient and suitable for high-resolution images.

## 2.3 Dynamic hyperparameter tuning training method based on Pareto efficiency

In the model training stage, different classes of loss functions are typically utilized, including intersection over union (IoU), classification, and localization losses, as illustrated in Figure 5. This study employed the SIoU loss function to enhance the regression efficiency by guiding the direction of the prediction box toward the ground truth box (Rezatofighi et al., 2019). For classification loss, the focal loss was employed, while Smooth-L1 loss was used for localization loss. The weights of the loss functions are essential for balancing the various components to improve the training outcomes, particularly in object detection tasks. Traditionally, manual methods are used to set fixed hyperparameters as weights for weight assignment. Nevertheless, this strategy might disregard the possibility that the complexity of training different loss functions could vary over the course of training, potentially restricting the learning capacity. To address this issue, a dynamic hyperparameter tuning training method based on Pareto efficiency is introduced in this section. This method treats loss functions as multiple interacting objectives and continuously optimizes them by dynamically weighting and coordinating these objectives, thereby accelerating the convergence process and enhancing accuracy at convergence.

### 2.3.1 Introduction to loss functions

#### 2.3.1.1 SIoU loss

##### 2.3.1.1.1 Angle cost

Angle loss plays a crucial role in expediting the convergence process before the prediction box and ground truth box are matched. This is achieved by initially regulating the angle factor to align them on the same horizontal or vertical line. As shown in Figure 6, the loss function minimizes $\partial$ when $\partial < \frac{\pi}{4}$. Otherwise, it minimizes $\beta = \frac{\pi}{2} - \partial$. The angular loss is defined as

$$\Lambda = 1 - 2 * \sin^2\left(\arcsin(x) - \frac{\pi}{4}\right) \tag{14}$$

where $x = \frac{c_h}{\sigma} = \sin(\partial)$, $\sigma = \sqrt{\left(b_{c_x}^{gt} - b_{c_x}\right)^2 + \left(b_{c_y}^{gt} - b_{c_y}\right)^2}$, and $c_h = max(b_{c_y}^{gt} - b_{c_y}) - min(b_{c_y}^{gt} - b_{c_y})$.

10.3389/fnbot.2024.1484276

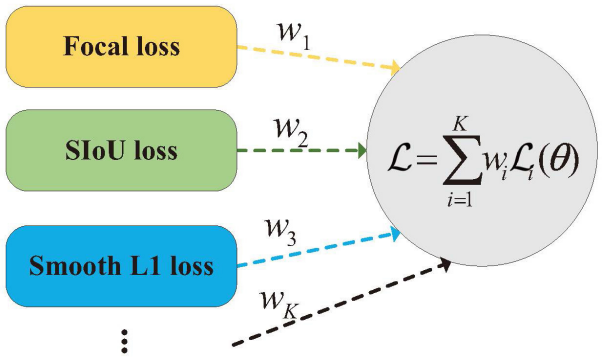


FIGURE 5
Illustration of functions and algorithm.

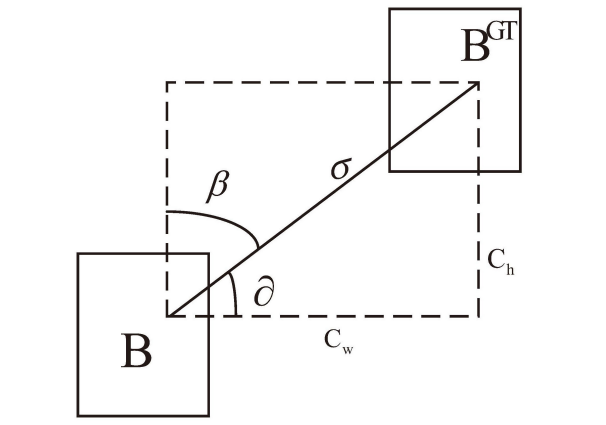


FIGURE 6
Illustration of angle cost.

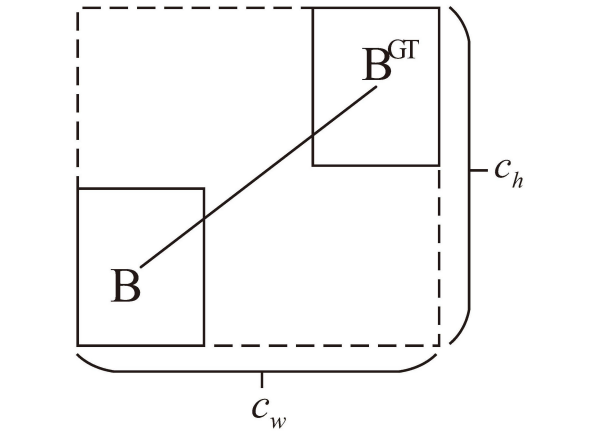


FIGURE 7
Illustration of distance cost.

#### 2.3.1.1.2 Distance cost

As demonstrated in Figure 7, when the prediction box and ground truth box are aligned either horizontally or vertically but remain significantly apart, it is crucial to impose a constraint on their separation distance. Building on the angle cost, the distance cost is defined by Equations 15, 16:

$$\Delta = \sum_{t=x,y} \left(1 - e^{-\gamma \rho_t}\right) \tag{15}$$

$$\rho_x = \left(\frac{b_{c_x}^{gt} - b_{c_x}}{c_w}\right)^2, \rho_y = \left(\frac{b_{c_y}^{gt} - b_{c_y}}{c_h}\right)^2, \gamma = 2 - \Lambda \tag{16}$$

where the width and height of the outer rectangle are represented by $c_w$ and $c_h$ for the prediction box and ground truth box, respectively. $\gamma$ is utilized to regulate the impact of the angular loss on the distance cost.

#### 2.3.1.1.3 Shape cost

Shape loss describes the similarity between the shapes of the prediction box and ground truth box, as defined in Equations 17, 18:

$$\Omega = \sum_{t=w,h} \left(1 - e^{-\omega_t}\right)^{\theta} \tag{17}$$

$$\omega_w = \frac{|w - w^{gt}|}{\max\left(w, w^{gt}\right)}, \omega_h = \frac{|h - h^{gt}|}{\max\left(h, h^{gt}\right)} \tag{18}$$

where $(w, h)$ and $(w^{gt}, h^{gt})$ denote the width and height of the ground truth box and prediction box, respectively. The value of $\theta$ indicates the degree of shape control, where a smaller $\theta$ indicates a higher degree of control; typically, $\theta \in [2, 6]$.

#### 2.3.1.1.4 IoU cost

The IoU quantifies the extent of overlap between the predicted bounding box and actual ground truth box. IoU loss is defined as follows:

$$\text{IoU} = \frac{\left|B \cap B^{GT}\right|}{\left|B \cup B^{GT}\right|} \tag{19}$$

where $B \cap B^{GT}$ denotes the overlapping area of the prediction box and ground truth box, and $B \cup B^{GT}$ denotes the concurrent area of the prediction box and ground truth box. SIoU loss is defined by Equation 20:

$$L_{\text{siou}} = 1 - IoU + \frac{\Delta + \Omega}{2} \tag{20}$$

### 2.3.1.2 Focal loss

Focal loss aims to enhance the focus on challenging samples by reducing the weight of easy-to-classify samples and amplifying the weight of difficult-to-classify samples, as defined in Equations 21, 22:

$$p_t \begin{cases} p, y = 1 \\ 1 - p, others \end{cases} \tag{21}$$

$$Focalloss(p_t) = -\alpha_t (1 - p_t)^{\gamma} \log(p_t) \tag{22}$$

where $p$ represents the probability of the model output, $y$ denotes the true label, and $\gamma$ is the weight factor.

#### 2.3.1.3 Smooth-L1 loss

The Smooth-L1 loss function combines the advantages of the L1 and L2 losses, with a smooth and robust training process, defined as

$$L_{\text{Smooth}-L1}\left(b_i^{gt}, \hat{b}_i\right) = \begin{cases} \frac{1}{2}\sum_{i=0}^{N}\left(b_i^{gt} - \hat{b}_i\right)^2, \text{ if } \left|b_i^{gt} - \hat{b}_i\right| < 1 \\ \sum_{i=0}^{N}\left|b_i^{gt} - \hat{b}_i\right| - 0.5, \text{ other} \end{cases} \tag{23}$$

where $b_i^{gt}$ denotes the position coordinates of the $i$-th ground truth box, and $\hat{b}_i$ denotes the position coordinates of the $i$-th prediction box.

#### 2.3.1.4 Overall loss function

In summary, the function is defined as

$$L = w_1 L_{cls} + w_2 L_{\text{siou}} + w_3 L_{Smooth-L1} \tag{24}$$

where $L_{\text{cls}}$ denotes the category focal loss, $L_{\text{siou}}$ denotes SIoU loss, $L_{Smooth-L1}$ denotes Smooth-L1 loss, and $w_1$, $w_2$, $w_2$ and are the respective weight parameters.

### 2.3.2 Algorithm of dynamic hyperparameter tuning training method based on Pareto efficiency

#### 2.3.2.1 Pareto efficiency and loss functions

Pareto efficiency is a concept in multi-objective optimization. To minimize a set of objective functions $f_1, ..., f_K$ in a given system, Pareto efficiency is a state in which it is impossible to improve one objective without hurting others.

Definition 1. To minimize objectives, denote the outcomes of two solutions by $s_i = [f_1^i, ..., f_K^i]$ and $s_j = [f_1^j, ..., f_K^j]$, where $s_i$ dominates if and only if $f_1^i \le f_1^j, f_2^i \le f_2^j, ..., f_K^i \le f_K^j$ .

Definition 2. A solution $s_i = [f_1^i, ..., f_K^i]$ is Pareto efficient if no other solution $s_j = [f_1^j, ..., f_K^j]$ dominates $s_i$.

Our goal is to discover Pareto efficient solutions. It is important to recognize that these solutions are not unique, leading to the establishment of the Pareto frontier. In summary, the training of object detection loss functions can be viewed as an optimization task involving the minimization of multiple loss functions. As illustrated in Figure 5, a set of weights, denoted as **w**, is determined to enable Pareto efficiency to be achieved by the loss functions. By iteratively solving for these weights **w** during the training process, we can continually and effectively optimize the objectives.

#### 2.3.2.2 Conditions of the algorithm

It is assumed that there are $K$ differentiable loss functions $\mathcal{L}_i(\theta)$, where $\theta$ denotes the model parameters in the object detection model $F(\theta)$, $\forall i \in \{1, ..., K\}$. The $K$ loss functions correspond to the $K$ objectives to be optimized, and multiple objectives are merged into a single one by setting a scalarization weight for the objectives, as shown in Equation 25:

$$\mathcal{L}(\theta) = \sum_{i=1}^{K} \omega_i \mathcal{L}_i(\theta) \tag{25}$$

where $\sum_{i=1}^{K} \omega_i = 1$, $\omega_i \ge 0$ , $\forall i \in \{1, ..., K\}$. Boundary constraints of $\omega_i$ on $c_i$ are set as $\omega_i \ge c_i$, $\sum_{i=1}^{K} c_i \le 1, c_i \in [0, 1]$, $\forall i \in \{1, ..., K\}$.

To obtain Pareto efficient solutions, the aggregated objective loss function must be minimized, and the model parameters should satisfy the KKT (Chen, 2022) condition such that Equations 26, 27 are satisfied:

$$\sum_{i=1}^{K} \omega_i = 1, \exists \omega_i \ge c_i, i \in \{1, ..., K\} \tag{26}$$

$$\sum_{i=1}^{K} \omega_i \nabla_\theta \mathcal{L}_i(\theta) = 0 \tag{27}$$

where $\nabla_\theta \mathcal{L}_i(\theta)$ represents the gradient of $\mathcal{L}_i$. Considering the specific problem to be solved, we transform the KKT condition is as follows:

$$\min . \left\| \sum_{i=1}^{K} \omega_i \nabla_\theta \mathcal{L}_i(\theta) \right\|_2^2 \text{ s.t.} \sum_{i=1}^{K} \omega_i = 1, \omega_i \ge c_i, \forall i \in \{1, \ldots, K\} \tag{28}$$

A solution satisfying Equation 28 is a Pareto efficient solution. It has been demonstrated that these solutions result in gradient directions that minimize all loss functions (Sener and Koltun, 2018).

#### 2.3.2.3 Framework of the algorithm

The framework begins with a uniform scalarization weight and proceeds by alternately updating the weights and model parameters. An optimizer is then utilized to ensure that the model converges effectively. As shown in Table 1 and Algorithm 1, the key part of the algorithm is to solve for conditionally generated weights for Pareto efficiency.

According to Equation 28, the problem is transformed into a quadratic programming algorithm by denoting $\hat{\omega}_i$ as $\omega_i - c_i$, and the Pareto efficiency condition becomes

$$\min . \left\| \sum_{i=1}^{K} (\hat{\omega}_i + c_i) \nabla_\theta \mathcal{L}_i(\theta) \right\|_2^2 \text{ s.t.} \sum_{i=1}^{K} \hat{\omega}_i = 1 - \sum_{i=1}^{K} c_i \tag{29}$$

The Pareto efficiency condition is equivalent to Equation 29. However, addressing the issue at hand is not straightforward, given its quadratic programming structure. Initially, we opt to ease these limitations by focusing solely on the equation restrictions. Subsequently, we implement a projection technique that produces effective outcomes from the viable set that encompassed all restrictions. When all other constraints are omitted except for the equational constraints, as shown in Equation 30, the solution is given by Theorem 1.

$$\min . \left\| \sum_{i=1}^{K} (\hat{\omega}_i + c_i) \nabla_\theta \mathcal{L}_i(\theta) \right\|_2^2 \text{ s.t.} \sum_{i=1}^{K} \hat{\omega}_i = 1 - \sum_{i=1}^{K} c_i \tag{30}$$

THEOREM 1. The solution to Equation 30 is $\hat{\omega}^* = \left(\left(\mathbf{M}^\top \mathbf{M}\right)^{-1} M\tilde{z}\right)[1:K]$ where $\mathbf{G} \in \mathbf{R}^{K \times \mathrm{m}}$ is the stacking matrix of $\nabla_\theta \mathcal{L}_i(\theta)$, $\mathbf{e} \in \mathbf{R}^K$ is the vector whose elements are all 1, $\mathbf{C} \in \mathbf{R}^K$

TABLE 1 Notations and description.

| Notations | Description |
|---|---|
| $F(\theta)$ | The object detection model |
| $\theta$ | The model parameters |
| $\mathcal{L}_i(\theta)$ | The loss function of for $i$-th objective |
| $\omega_i$ | The weight of $i$-th objective for scalarization |
| $c_i$ | The boundary constraint for $i$-th objective |
| $\nabla_\theta \mathcal{L}_i(\theta)$ | The gradient of loss $\mathcal{L}_i(\theta)$with respect of $\theta$ |
| $G$ | The stacking matrix of $\nabla_\theta \mathcal{L}_i(\theta)$ |
| $e$ | The vector whose elements are all 1 |

**Input:** The loss functions of multiple objectives correspondingly: $\mathcal{L}_i(\theta), \forall i \in \{1, \cdots, K\}$; the scalarization of weights initialized uniformly: $\omega_i = \frac{1}{K}, \forall i \in \{1, \cdots, K\}$: The bounds for the objectives:$c_i$, $\forall i \in \{1, \cdots, K\}$;
**Output:** The model parameters: $\theta$;
Get the single aggregated objective functions: $\mathcal{L}(\theta) = \sum_{i=1}^{K} \omega_i \mathcal{L}_i(\theta)$;
**for** each batch **do**
  Optimize $\mathcal{L}(\theta)$ with the optimizer and update the $F(\theta)$ parameters: $\theta$;
  The problem to be solved as Equation 30;
  Use Theorem 1 to obtain solutions to Equation 30;
  The weights are derived through Equation 31: $\omega_i, \cdots, \omega_k$;
  Aggregated the objectives: $\mathcal{L}(\theta) = \sum_{i=1}^{K} \omega_i \mathcal{L}_i(\theta)$
**end for**

Algorithm 1. Dynamic hyperparameter tuning training method based on Pareto efficiency.

is the concatenated vector of $c_i$, and $\tilde{\boldsymbol{Z}} \in \mathbf{R}^K$ is the concatenated vector of $-\mathbf{G}\mathbf{G}^\top \mathbf{c}$ and $1 - \sum_{i=1}^{K} c_i$ and $\mathbf{M} = \begin{pmatrix} \mathbf{G}\mathbf{G}^\top & \mathbf{e} \\ \mathbf{e} & 0 \end{pmatrix}$.

During the solution process, the inverse operation of the matrix is negligible because the number of loss functions for object detection is small. However, the solution $\hat{\omega}^*$ of Equation 30 may be invalid because the non-negative constraints are ignored. Therefore, we obtain an effective solution using the projection method, as shown in Equation 31:

$$\min . \left\| \tilde{\omega} - \hat{\omega}^* \right\|_2^2 \text{s.t.} \sum_{i=1}^{K} \hat{\omega}_i = 1, \tilde{\omega}_i \geq 0, \forall i \in \{1, ..., K\} \quad (31)$$

Equation 31 represents a non-negative least squares problem, which can be easily solved using the active set method (Arnström and Axehill, 2021).

## 2.4 Summary

In the previous section, we presented three aspects of the improvement approach for unmanned object detection. First, we introduced an architecture to solve the insufficiency of the model for multi-scale object localization and detection. Subsequently, we developed a transformer encoder with group axial attention to reduce computation and enhance the inference speed. Finally, we presented a novel training technique that utilizes dynamic hyperparameter tuning inspired by the principle of Pareto efficiency. By dynamically adjusting the weights to align the training states of different loss functions, this approach effectively addresses issues related to manually assigning fixed weights. As a result, it enhances both the speed and accuracy of model convergence.

# 3 Experimental results and analysis

## 3.1 Setups

### 3.1.1 Dataset

Dataset 1 was selected from the COCO 2017 (Lin et al., 2014) dataset with category objectives related to autonomous driving, consisting of 10 categories, 35,784 images for training, and 2431 images for validation. Dataset 2 combines the original categories from the PASCAL VOC 2012 (Everingham et al., 2010) dataset, which include Person, Car, Train, Motorcycle, Bicycle, and Other, with 11,540 images for training and 2913 images for validation. Dataset 3 is sourced from the KITTI professional autonomous driving dataset (Geiger et al., 2013), primarily including categories such as Car, Pedestrian, Cyclist, Van, Truck, and Tram, with 7,481 images for training and 7,518 images for validation.

### 3.1.2 Evaluation metrics

The experimental evaluation metrics were AP, FPS, and GFLOPs. FPS and GFLOPs denote the inference speed and computation of the model, respectively. Specifically, $AP_S$ and $AP_M$ denote the AP for small- and medium-sized objects, respectively. AP is the area under the precision-recall (PR) curve, and precision (P) and recall (R) are calculated using Equations 32, 33:

$$P = \frac{TP}{TP + FP} \quad (32)$$

$$R = \frac{TP}{TP + FN} \quad (33)$$

where TP, FP, and FN denote the accurately recognized positive samples, erroneously recognized positive samples, and erroneously recognized negative samples, respectively. Mean average precision (mAP) can be computed by taking the average of the AP values across different categories, as illustrated in Equations 34, 35:

$$AP = \int_0^1 P(R)dR \quad (34)$$

$$mAP = \frac{1}{n} \sum_{i=1}^{n} AP_i \quad (35)$$

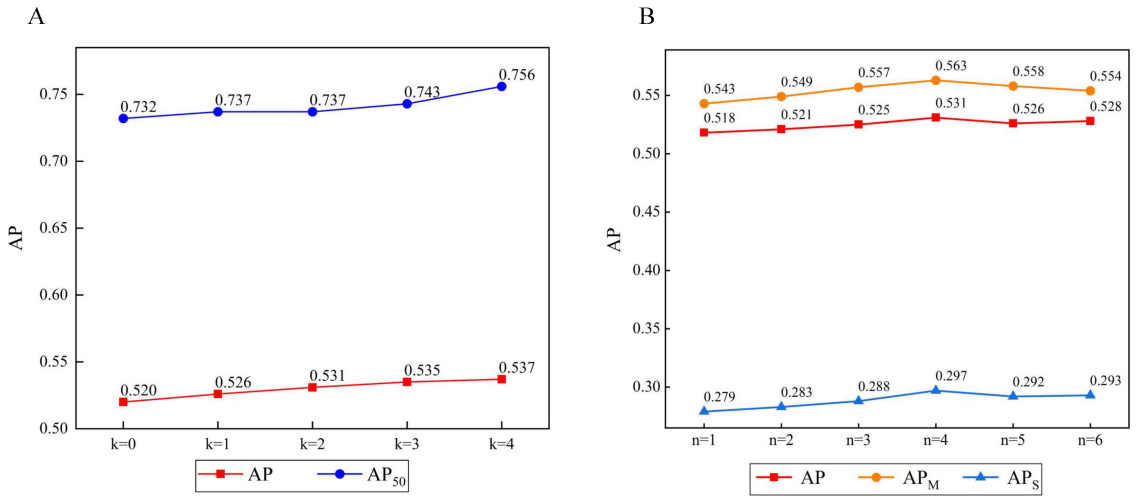


FIGURE 8
**(A)** The result of the number of CA modules. **(B)** The result of the number of scales.

In the context of object detection, the average precision (AP) is calculated across different Intersection over Union (IoU) thresholds ranging from 0.5 to 0.95 with increments of 0.05. When the IoU threshold is specifically set at 0.5, it is denoted as $AP_{50}$. The mAP is then computed as the average of the AP values for each category in the dataset. Therefore, all individual AP values mentioned correspond to the overall mAP.

#### 3.1.3 Implementation details

The model was trained on an NVIDIA V100 GPU. In each stage the multi-scale feature and location information extraction method (MLEM), $N$ was set to 3, 4, 6, and 3 correspondingly. The Adam optimizer was used for all experiments: initial_learning_ rate = 0.0005, weight_decay = 0.0001, and batchsize = 8.

## 3.2 Analysis of model parameters

#### 3.2.1 Parameter analysis of multi-scale feature extraction module

To explore the effect of the scale $n$ on this approach, $n$ was set as a scale control parameter. Six experiments were designed with $n = 1, 2, 3, 4, 5, 6$. $n = 1$ represents a scale of one with no multi-scale fusion. Similarly, $n = 2$ represents a scale of 2, and the data are divided into two parts for multi-scale fusion. We adopted AP, $AP_M$, and $AP_s$ as indicators, as shown in Figure 8.

The experimental results show that the multi-scale fusion operation has a significant effect on $AP_S$. An increase of $n$ indicates that the number of different scales of the feature map fusion increases, and all show an upward trend. When $n = 4$, the multi-scale effect is obvious and optimal. When $n = 5$ or 6, the value of AP decreases slightly. Probably owing to limitations on the image size, the multi-scale feature extraction ability remains almost unchanged. However, an excessive number of branches can lead to a model degradation.

#### 3.2.2 Parameter analysis of coordinate attention module

To investigate the effects of the CA module on the experiments, $k$ was set as the experimental parameter. Five experiments were designed with $k = 0, 1, 2, 3, 4$. There are four stages: $k = 0$ denotes no use of CA, $k = 1$ denotes that the module is deployed in stage 1, $k = 2$ denotes that the module is deployed in the first two stages, etc. We adopted AP as an indicator, as shown in Figure 8.

The experimental results show that the CA module effectively improves the AP compared with the case of $k = 0$. Building on the multi-scale feature maps obtained in the previous stages helps further improve the effectiveness of CA in later stages. When $k = 1$ or 2, shallow features, such as space and details, are retained, which helps improve the AP. When $k = 4$, the enhancement effect of AP is weakened, but AP reaches its peak.

#### 3.2.3 Parameter analysis of group axial attention layer

To investigate the effect of the attention range $s$ on the feature map in the group axial attention layer, we set the number of encoders $n = 6$, the attention layer in the encoder adopts a single attention range for each encoder to the feature map, $s_i$ denotes the $i$-th range combination, $s_0$ denotes the original method, $s_1 = [1, 1, 1, 1, 1, 1]$, $s_2 = [2, 2, 2, 2, 2, 2]$, $s_3 = [1, 1, 1, 2, 2, 2]$, $s_4 = [1, 1, 2, 2, 4, 4]$, $s_5 = [1, 1, 2, 2, 6, 6]$, $s_6 = [2, 2, 4, 4, 6, 6]$, and we performed seven experiments, as shown in Figure 9.

When $s_0$ becomes $s_1$, the AP gradually increases. When the attention range is $s_2$ the accuracy is further improved, probably due to the expansion of the attention calculation range. As the encoders continue to stack, the image feature level continuously increases. However, $s_1$ and $s_2$ do not take this case into account. From $s_3$ to $s_6$, the attention range increases. In the early stages, the smaller attention range facilitates learning of the local details of the image. In the later stages, a larger range is more conducive to learning the global information of the image. Therefore, $s_3$ from to $s_5$, $AP_M$ is

10.3389/fnbot.2024.1484276

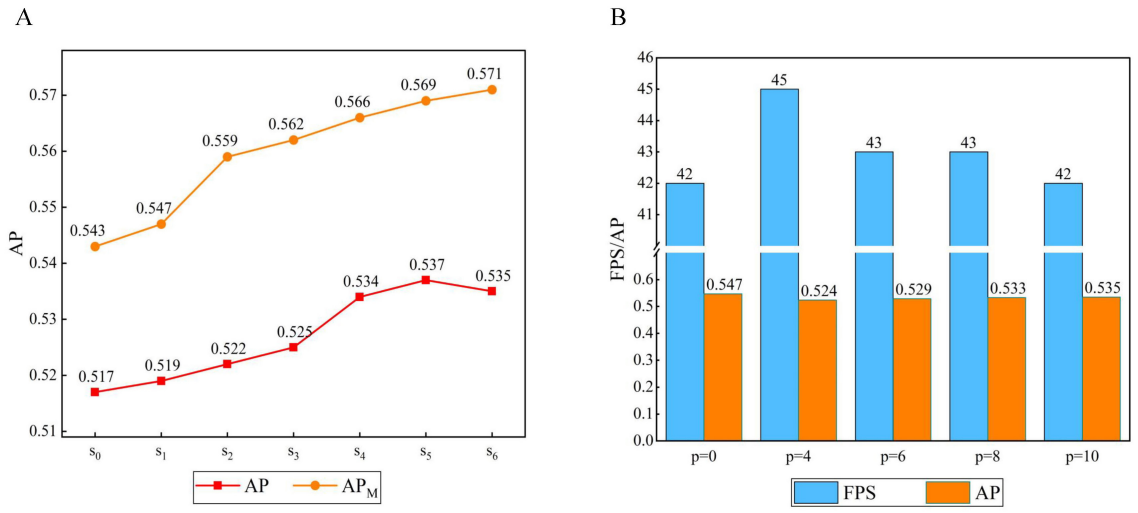


FIGURE 9
**(A)** Result of the range of group axial attention layer. **(B)** Result of the window size of window axial group attention.

TABLE 2 Ablation experiments number and results.

| ID | MLEM | TEGA | DHMP | AP | $AP_{50}$ |
|---|---|---|---|---|---|
| Exp.1 | | | | 0.519 | 0.721 |
| Exp.2 | ✓ | | | 0.535 | 0.723 |
| Exp.3 | | ✓ | | 0.528 | 0.725 |
| Exp.4 | | | ✓ | 0.525 | 0.734 |
| Exp.5 | | ✓ | ✓ | 0.541 | 0.736 |
| Exp.6 | ✓ | | ✓ | 0.543 | 0.733 |
| Exp.7 | ✓ | ✓ | | 0.539 | 0.737 |
| Exp.8 | ✓ | ✓ | ✓ | 0.552 | 0.741 |

TABLE 3 Ablation experiments number and results.

| ID | WGA | Dataset 1 | Dataset 2 | FPS ($p = 4$) | FPS ($p = 8$) |
|---|---|---|---|---|---|
| Exp.1 | | ✓ | | 40 | 40 |
| Exp.2 | | | ✓ | 41 | 41 |
| Exp.3 | ✓ | ✓ | | 46 | 44 |
| Exp.4 | ✓ | | ✓ | 45 | 43 |

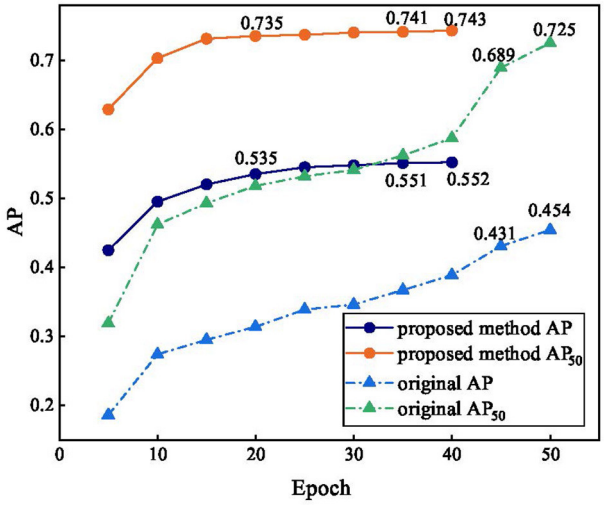


FIGURE 10
Comparison of accuracy trends before and after improvement.

significantly improved compared to the cases of $s_1$ and $s_2$, and the AP reaches its best for $s_5$. Comparing $s_6$ and $s_1$, it can be observed that a larger attention range is more favorable for targets at the medium scale.

### 3.2.4 Parameter analysis of window group axial attention

To explore the effect of window size on window group axial attention, we set the window size $p$ as the experimental parameter, adjusted the size of the feature map to $W = H$ in the encoder group axial attention layer, set the number of encoders to $n = 6$, and the attention range of the feature map in the group axial attention layer in each encoder was set to $s_5 = [1, 1, 2, 2, 6, 6]$, where $p = 0$ denotes no window partition, $p = 1$ denotes a partition size of $1 \times 1$, etc. Five experiments were conducted, as shown in Figure 9.

It follows that dividing the attention range within a fixed window can further reduce the computational complexity. When $p = 4$, the window size is smaller, the computation complexity is minimized, and the FPS increases; however, this will have a greater impact on dividing the attention region, which will lead to a lower AP. When p is larger, the impact on the operation of dividing the attention region is reduced, and although the individual window complexity increases, it leads to further increases in FPS and AP.

TABLE 4 The results of comparison.

| Model | AP | $AP_{50}$ | $AP_S$ | Epoch | GFLOPs | Params |
|---|---|---|---|---|---|---|
| YOLOv6 (Li et al., 2023) | 0.542 | 0.727 | 0.327 | 100 | 150 | 59M |
| YOLOv7 (Wang et al., 2023) | 0.546 | 0.733 | 0.331 | 100 | 104 | 36M |
| YOLOv8 (Talaat and ZainEldin, 2023) | 0.551 | 0.739 | **0.350** | 100 | 165 | 43M |
| DETR (Carion et al., 2020) | 0.519 | 0.720 | 0.291 | 110 | 187 | 41M |
| H-DETR (Zong et al., 2023) | 0.539 | 0.732 | 0.332 | 70 | 268 | 42M |
| Anchor-DETR (Li et al., 2022) | 0.521 | 0.721 | 0.294 | 80 | 172 | 39M |
| Deformable-DETR (Zhu et al., 2020) | 0.539 | 0.726 | 0.316 | 50 | 173 | 40M |
| DAB-DETR (Liu et al., 2022) | 0.543 | 0.732 | 0.325 | 50 | 256 | 44M |
| DN-Deformable-DETR (Li et al., 2022) | 0.545 | 0.731 | 0.327 | 50 | 265 | 48M |
| DINO-DETR (Zhang et al., 2022) | 0.550 | 0.737 | 0.332 | **25** | 279 | 47M |
| **Proposed method** | **0.552** | **0.741** | **0.336** | **40** | **161** | **37M** |

Bold values indicate the best or second-best values.

TABLE 5 The results of comparison.

| Model | AP | $AP_{50}$ | $AP_S$ | Epoch | GFLOPs | Params |
|---|---|---|---|---|---|---|
| YOLOv6 (Li et al., 2023) | 0.571 | 0.672 | 0.316 | 100 | 150 | 59M |
| YOLOv7 (Wang et al., 2023) | 0.473 | 0.676 | 0.319 | 100 | 104 | 36M |
| YOLOv8 (Talaat and ZainEldin, 2023) | 0.477 | 0.678 | **0.323** | 100 | 165 | 43M |
| DETR (Carion et al., 2020) | 0.433 | 0.643 | 0.251 | 110 | 187 | 41M |
| H-DETR (Zong et al., 2023) | 0.443 | 0.653 | 0.268 | 70 | 268 | 42M |
| Anchor-DETR (Li et al., 2022) | 0.438 | 0.652 | 0.256 | 80 | 172 | 39M |
| Deformable-DETR (Zhu et al., 2020) | 0.461 | 0.663 | 0.273 | 50 | 173 | 40M |
| DAB-DETR (Liu et al., 2022) | 0.465 | 0.669 | 0.278 | 50 | 256 | 44M |
| DN-Deformable-DETR (Li et al., 2022) | 0.469 | 0.673 | 0.309 | 50 | 265 | 48M |
| DINO-DETR (Zhang et al., 2022) | 0.475 | 0.678 | 0.318 | **25** | 279 | 47M |
| **Proposed method** | **0.478** | **0.687** | **0.321** | **35** | **161** | **37M** |

Bold values indicate the best or second-best values.

TABLE 6 The results of comparison.

| Model | AP | $AP_{50}$ | $AP_S$ | Epoch | GFLOPs | Params |
|---|---|---|---|---|---|---|
| YOLOv6 (Li et al., 2023) | 0.546 | 0.745 | 0.347 | 100 | 150 | 59M |
| YOLOv7 (Wang et al., 2023) | 0.548 | 0.766 | 0.352 | 100 | 104 | 36M |
| YOLOv8 (Talaat and ZainEldin, 2023) | 0.554 | 0.821 | **0.403** | 100 | 165 | 43M |
| DETR (Carion et al., 2020) | 0.526 | 0.722 | 0.286 | 110 | 187 | 41M |
| H-DETR (Zong et al., 2023) | 0.539 | 0.732 | 0.332 | 70 | 268 | 42M |
| Anchor-DETR (Li et al., 2022) | 0.537 | 0.729 | 0.327 | 80 | 172 | 39M |
| Deformable-DETR (Zhu et al., 2020) | 0.541 | 0.736 | 0.349 | 50 | 173 | 40M |
| DAB-DETR (Liu et al., 2022) | 0.545 | 0.739 | 0.353 | 50 | 256 | 44M |
| DN-Deformable-DETR (Li et al., 2022) | 0.549 | 0.811 | 0.364 | 50 | 265 | 48M |
| DINO-DETR (Zhang et al., 2022) | 0.553 | 0.819 | 0.381 | **25** | 279 | 47M |
| **Proposed method** | **0.556** | **0.823** | **0.397** | **35** | **161** | **37M** |

Bold values indicate the best or second-best values.

## 3.3 Ablations

Ablation experiments were conducted to evaluate the effectiveness of the proposed method. We trained our model on the COCO dataset and ensured that the experimental conditions were consistent. Exp. 1 represents the baseline, based on which we adopted the MLEM the transformer encoder based on the group axial attention mechanism (TEGA), and dynamic hyperparameter tuning training method based on Pareto efficiency (DHMP). A total of eight ablation experiments were performed, as shown in Table 2. Exp. 2 shows that the addition of MLEM resulted in a 1.6% increase in AP over DETR, demonstrating that multi-scale features have a more significant impact on the results. Exp. 3 shows that after adding TEGA, AP is improved by 0.9% compared to DETR, and compared to MLEM, the improvement is minimal, which indicates that the accuracy of the detection may depend on the pre-processing of the features; however, other experiments demonstrated that encoders have a material impact in terms of reducing model complexity and increasing inference speed. Exp. 3 shows that adding DHMP can make the model converge with a higher accuracy. A comparison of the results of Exps. 5–7 with those of Exp. 8, respectively, shows that each improvement is necessary to improve detection. Exp. 8 shows that applying all three improvements simultaneously significantly improves the accuracy, with a 2.4% improvement in AP.

As shown in Table 3, we conducted several groups of ablation experiments to demonstrate the effectiveness of Window G-Attention (WGA). Dataset 1 and Dataset 2 represent the COCO and PASCAL VOC datasets, respectively. Exp. 1 refers to the improved method without using WGA. By comparing the results of Exps. 1 and 3, as well as Exps. 2 and 3, we conclude that WGA contributes to improving FPS. It is worth noting that the degree of FPS improvement achieved by WGA varies across different datasets, which we believe is related to the distribution of objects within the images. In addition, we observed that the smaller the window size p of WGA, the lower the computational complexity and the higher the FPS, which is consistent with the conclusion we reached in the article.

Figure 10 reflects the accuracy trends before and after the improvement in the algorithm. The original method still has no convergence trend over 50 epochs, the AP and $AP_{50}$ increase slowly, and the accuracy only reaches approximately 50% that of our method after 25 epochs, and the accuracy only reaches approximately 5% that of our method after 25 epochs. The accuracy of our method increased faster in the early stages of training, with AP and $AP_{50}$ reaching 0.535 and 0.735, respectively, at epoch 20, which was higher than the accuracy of DETR at epoch 50. After 25 epochs, our method converges, indicating that DLMP is effective, and AP and $AP_{50}$ finally reach 0.552 and 0.743, respectively.

## 3.4 Comparison with other methods

In this section, we compare our method with current mainstream algorithms on the COCO, PASCAL VOC and KITTI datasets, as shown in Tables 4–6. As shown in Table 4, the AP of our method is 0.552, and its $AP_{50}$ is 0.741; both are the best, although

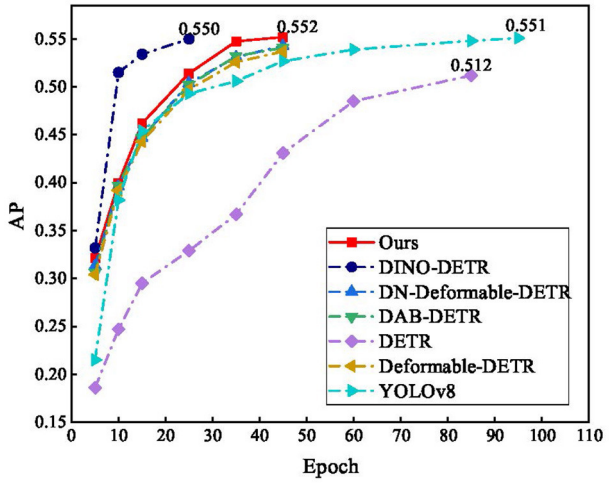


FIGURE 11
Results of comparison.

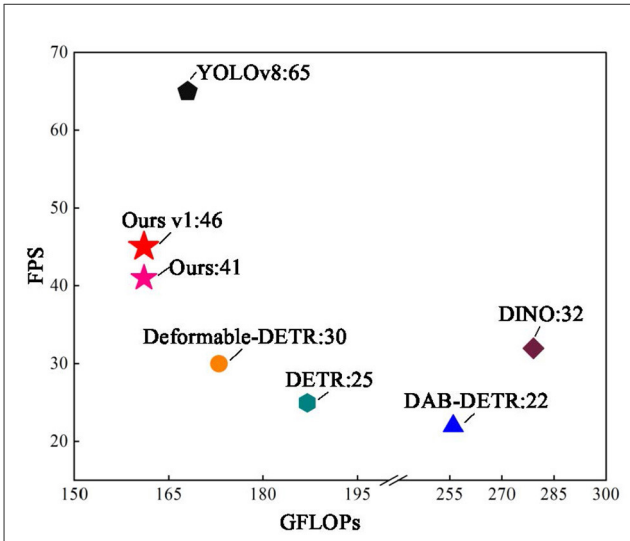


FIGURE 12
Comparison of FPS with different algorithms.

$AP_S$ is 1.4% lower than that of YOLOv8, and the actual epochs required for training are less than for YOLOv8. Compared with DN-Deformable-DETR and DINO-DETR, the proposed method maintains $AP_S$ at the same level as the former and 0.4% higher than the latter while significantly reducing the GFLOPS and params. Compared to DETR, our method reduces the number of parameters by approximately 10% and improves the AP, $AP_{50}$, and $AP_S$ by 3.3%, 2.1%, and 4.5%, respectively, which is advantageous for DETR-like models.

Table 5 shows the experimental results of the different methods on PASCAL VOC. The AP of our method is 0.477, which is only lower than that of YOLOv8. The proposed method reaches convergence in 35 epochs, which is only higher than that of DINO-DETR, showing good convergence speed and more satisfactory detection accuracy.

FIGURE 13
Visualization of the proposed method compared with current methods in suburban road scenes.

FIGURE 14
Visualization of the proposed method compared with current methods in suburban street scenes.

Table 6 shows the results of different methods on the KITTI dataset. Our proposed method achieves the highest AP value of 0.556, representing a 3% improvement compared to the baseline method. Our method converges within 35 epochs, second only to DINO-DETR. In terms of $AP_S$ evaluation, our method is only slightly behind YOLOv8 and is nearly on par with it, while having fewer model parameters and lower computational complexity.

The convergence of the different algorithms during training is shown in Figure 11. The training of the proposed method essentially converged in fewer than 45 epochs, which is a remarkable improvement over DETR and YOLOv8. DINO-DETR converges in the 25th epoch, but in actual training, owing to its high complexity, the actual training of an epoch is approximately three times as long as that of the proposed method.

Figure 12 shows the relationship between the model computation GFLOPs and FPS and test images from the COCO and PASCAL VOC datasets, with an image size of 900 × 900. The proposed method had the smallest GFLOPs and best FPS in the DETR series. Version v1 uses Window G-Attention, and there is still a large gap in the FPS compared to YOLOv8; however, the FPS is improved by 84% compared to DETR.

In conclusion, the proposed method demonstrates robust performance across three distinct datasets. First, the AP values underscore the superior detection accuracy of the method,

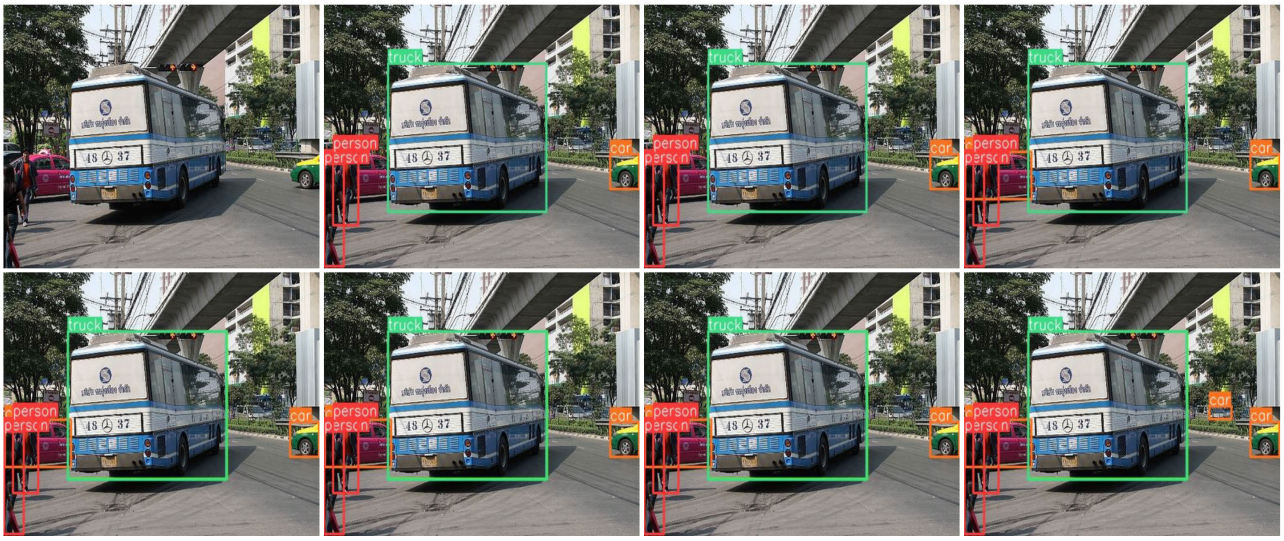


FIGURE 15
Visualization of the proposed method compared with current methods in suburban road scenes.

effectively identifying objects of varying scales in autonomous driving scenarios, including obstacles, vehicles, pedestrians, and traffic lights. Second, considering the inherent constraints of computational resources and energy consumption in autonomous driving hardware, our model is designed to minimize parameter size while achieving notable computational efficiency compared to mainstream object detection algorithms. With respect to the FPS metric, our method achieves the best performance among DETR-based algorithms, satisfying the stringent real-time detection requirements of autonomous driving systems. Furthermore, the proposed model is implemented using the PyTorch framework, ensuring seamless deployment on vehicle-embedded hardware. Notably, our method exhibits slightly lower performance on the $AP_S$ metric compared to YOLOv8, likely due to the embedded data augmentation techniques employed by YOLOv8. This insight highlights a promising direction for further optimization and refinement of our algorithm.

## 3.5 Visualization

We tested the visualization of the different methods in various driving scenarios. The threshold for each detector was set to 0.6, and the performances are shown in Figures 13–15. Each image shows visualization of several approaches (Corresponding order from top to bottom, left to right): (a) the image to be detected, (b) DETR, (c) Deformable-DETR, (d) DAB-DETR, (e) DN-Deformable-DETR, (g) DINO-DETR (f) YOLOv8, and (h) the proposed method. According to the experimental results, the proposed method addresses the issues of omission and false detection caused by mutual occlusion, mutilation, and the small size of the target, and it has excellent adaptability to complex scenes.

Figure 13 shows a suburban road scene with clearer vehicle targets, but the problem of missing the car occurs in both (b) and (c), where the person in the car is a more difficult target to identify, and the remaining methods did not produce a missed detection. Both the proposed method and YOLOv8 detected traffic signs and the farthest vehicle.

Figure 14 shows a street scene, where people stand densely and the targets are small; however, the proposed method showed the least number of missed detections and best object detection. Figure 15 shows a city road scene, where there are cases of mutual occlusion and mutilation of detected objects, and only the proposed method successfully detected the vehicles behind the grass.

# 4 Conclusion

This paper presented an improved autonomous driving object detection method based on DETR. This approach includes a feature extraction technique that incorporates position-sensitive attention to improve multi-scale object detection. In addition, a transformer encoder with a group axial attention mechanism was developed to enhance the inference speed and reduce model computation. Furthermore, a dynamic hyperparameter tuning training method based on Pareto efficiency was implemented to adjust the training state of the loss function by dynamically modifying the weights. This approach aims to overcome the limitations associated with manually setting fixed weights, accelerate convergence, and improve model accuracy. Experimental results demonstrated that this approach outperforms traditional methods.

It should be emphasized that our method stands out for its exceptional inference speed compared to other DETR-like algorithms. However, it lags behind the YOLO series of algorithms in this regard. Achieving a high inference speed often results in a tradeoff with detection accuracy, posing a challenge for DETR-type algorithms to strike a balance between the two. In addition, our approach proved that the object detection model can be enhanced by synchronizing the training phases of the loss function. Developing a more intricate training strategy for the loss function is a promising prospect for future research.

Finally, addressing faults in autonomous driving through fault detection, data reconstruction, decision optimization, and fault-tolerant mechanisms based on deep learning models is of great significance for improving the robustness and safety of autonomous driving systems.

## Data availability statement

The raw data supporting the conclusions of this article will be made available by the authors, without undue reservation.

## Ethics statement

The studies involving humans were approved by Academic Committee of Jiamusi University. The studies were conducted in accordance with the local legislation and institutional requirements. Written informed consent for participation was not required from the participants or the participants' legal guardians/next of kin in accordance with the national legislation and institutional requirements.

## Author contributions

HZ: Writing – original draft, Writing – review & editing. SZ: Writing – original draft, Writing – review & editing. XP: Writing – original draft, Writing – review & editing. ZL: Writing – original draft, Writing – review & editing. GL: Writing – original draft, Writing – review & editing.

## Funding

The author(s) declare that financial support was received for the research, authorship, and/or publication of this article. The work was supported by the Natural Science Foundation of Heilongjiang (LH2022E114), 2024 Heilongjiang Province Higher Education Institutions' Basic Research Operating Expenses Project (2024-KYYWF-0553), Municipal Science and Technology Programme Innovation Incentive Category Projects of Jiamusi (GY2023JL0002), Innovation Team Project of Heilongjiang Provincial Department of Education (2024-KYYWF-0625), Jiamusi University Education and Teaching Reform Research Project (2021JY2-02), East Pole Academic Team Project of Jiamusi University (DJXSTD202417), Jiamusi University Horizontal Project(JMSUHXXM2024082101), and Jiamusi University Horizontal Project (JMSUHXXM2024122501).

## Acknowledgments

First of all, SZ would like to express his thanks to HZ, GL, and XP for their guidance and help. Meanwhile, we would like to thank Editage for English language editing.

## Conflict of interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

## Publisher's note



## References

Arnström, D., and Axehill, D. (2021). A unifying complexity certification framework for active-set methods for convex quadratic programming. *IEEE Trans. Automat. Contr.* 67, 2758–2770. doi: 10.1109/TAC.2021.3090749

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., and Zagoruyko, S. (2020). "End-to-end object detection with transformers," in *European Conference on Computer Vision* (Cham: Springer International Publishing), 213–229. doi: 10.1007/978-3-030-58452-8_13

Chen, S. (2022). The kkt optimality conditions for optimization problem with interval-valued objective function on hadamard manifolds. *Optimization* 71, 613–632. doi: 10.1080/02331934.2020.1810248

Dalal, N., and Triggs, B. (2005). "Histograms of oriented gradients for human detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)* (IEEE), 886–893. doi: 10.1109/CVPR.2005.177

Dong, X., Bao, J., Chen, D., Zhang, Z., Wang, L., Yuan, L., et al. (2022). "Cswin transformer: a general vision transformer backbone with cross-shaped windows," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 12124–12134. doi: 10.1109/CVPR52688.2022.01181

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., et al. (2020). An image is worth 16x16 words: transformers for image recognition at scale. *arxiv preprint arxiv:2010.11929*.

Everingham, M., Van Gool, L., Williams, C. K., Winn, J., and Zisserman, A. (2010). The pascal visual object classes (VOC) challenge. *Int. J. Comput. Vis.* 88, 303–338. doi: 10.1007/s11263-009-0275-4

Gao, S.-H., Cheng, M.-M., Zhao, K., Zhang, X.-Y., Yang, M.-H., and Torr, P. H. (2019). Res2net: a new multi-scale backbone architecture. *IEEE Trans. Pattern Anal. Mach. Intell.* 43, 652–662. doi: 10.1109/TPAMI.2019.2938758

Ge, Z., Liu, S., Wang, F., Li, Z., and Sun, J. (2021). Yolox: exceeding yolo series in 2021. *arxiv preprint arxiv:2107.08430*.

Geiger, A., Lenz, P., Stiller, C., and Urtasun, R. (2013). Vision meets robotics: the kitti dataset. *Int. J. Rob. Res.* 32, 1231–1237. doi: 10.1177/0278364913491297

Hou, Q., Zhou, D., and Feng, J. (2021). "Coordinate attention for efficient mobile network design," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 13713–13722. doi: 10.1109/CVPR46437.2021.01350

Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. (2017). "Densely connected convolutional networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4700–4708. doi: 10.1109/CVPR.2017.243

Huang, Z., Ben, Y., Luo, G., Yuan, Y., Zhang, X., Li, Z., et al. (2021). Shuffle transformer: Rethinking spatial shuffle for vision transformer. *arxiv preprint arxiv:2106.03650*.

Kendall, A., Gal, Y., and Cipolla, R. (2018). "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 7482–7491. doi: 10.1109/CVPR.2018.00781

Li, C., Li, L., Geng, Y., Song, W., Zhang, Z., Liu, H., et al. (2023). Yolov6 v3. 0: a full-scale reloading. *arxiv preprint arxiv:2301.05586*.

Li, F., Zhang, H., Liu, S., Guo, J., Yang, J., Wang, X., et al. (2022). "Dn-detr: accelerate detr training by introducing query denoising," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 13619–13627. doi: 10.1109/CVPR52688.2022.01325

Li, Z., Wang, W., Li, H., Xie, E., Sima, C., Lu, T., et al. (2024). Bevformer: learning bird's-eye-view representation from lidar-camera via spatiotemporal transformers. *arxiv preprint arXiv:2203.17270*.

Lin, B., Ye, F., Zhang, Y., Su, Y., and Chen, B. (2021). Reasonable effectiveness of random weighting: a litmus test for multi-task learning. *arxiv preprint arxiv:2111.10603*.

Lin, J., Mao, X., Chen, Y., Zhu, X., Chen, S., Yu, N., et al. (2022). $D^2$etr: decoder-only detr with computationally efficient cross-scale attention. *arxiv preprint arxiv:2203.00860*.

Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., and Belongie, S. (2017). "Feature pyramid networks for object detection," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 936–944. doi: 10.1109/CVPR.2017.106

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., et al. (2014). "Microsoft coco: Common objects in context," in *Computer Vision-ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6–12, 2014, Proceedings, Part V 13* (Springer International Publishing), 740–755. doi: 10.1007/978-3-319-10602-1_48

Lin, X., Chen, H., Pei, C., Yu, D., and Yang, M. (2019). "A pareto-efficient algorithm for multiple objective optimization in e-commerce recommendation," in *Proceedings of the 13th ACM Conference on Recommender Systems*, 20–28. doi: 10.1145/3298689.3346998

Liu, S., Li, F., Zhang, H., Liu, L., Liu, C., Hu, H., et al. (2022). Dab-detr: Dynamic anchor boxes are better queries for detr. *arxiv preprint arxiv:2201.12329*.

Liu, X., Tong, X., and Liu, Q. (2021). "Profiling pareto front with multi-objective stein variational gradient descent," in *Advances in Neural Information Processing Systems*, 14721–14733.

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., et al. (2021). "Swin transformer: hierarchical vision transformer using shifted windows," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 10012–10022. doi: 10.1109/ICCV48922.2021.00986

Mahapatra, D., and Rajan, V. (2020). "Multi-task learning with user preferences: gradient descent with controlled ascent in pareto optimization," in *International Conference on Machine Learning* (PMLR), 6597–6607.

Mehta, S., and Rastegari, M. (2021). Mobilevit: light-weight, general-purpose, and mobile-friendly vision transformer. *arxiv preprint arxiv:2110.02178*.

Mushtaq, H., Deng, X., Azhar, F., Ali, M., and Raza Sherazi, H. H. (2024). PLC-fusion: perspective-based hierarchical and deep lidar camera fusion for 3D object detection in autonomous vehicles. *Information* 15:739. doi: 10.3390/info15110739

Neubeck, A., and Van Gool, L. (2006). "Efficient non-maximum suppression," in *18th International Conference on Pattern Recognition (ICPR'06)* (IEEE), 850–855. doi: 10.1109/ICPR.2006.479

Ou, Z., Wang, Z., Xiao, F., Xiong, B., Zhang, H., Song, M., et al. (2023). AD-RCNN: adaptive dynamic neural network for small object detection. *IEEE Internet Things J.* 10, 4226–4238. doi: 10.1109/JIOT.2022.3215469

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). "You only look once: unified, real-time object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 779–788. doi: 10.1109/CVPR.2016.91

Ren, S., He, K., Girshick, R., and Sun, J. (2015). "Faster r-CNN: towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems*, 28.

Rezatofighi, H., Tsoi, N., Gwak, J., Sadeghian, A., Reid, I., and Savarese, S. (2019). Generalized intersection over union: a metric and a loss for bounding box regression," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 658–666. doi: 10.1109/CVPR.2019.00075

Sener, O., and Koltun, V. (2018). "Multi-task learning as multi-objective optimization," in *Advances in Neural Information Processing Systems*, 31.

Simonyan, K., and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arxiv preprint arxiv:1409.1556*.

Talaat, F. M., and ZainEldin, H. (2023). An improved fire detection approach based on yolo-v8 for smart cities. *Neural Comput. Applic.* 35, 20939–20954. doi: 10.1007/s00521-023-08809-1

Tang, C., Zhang, L.-L., Jiang, H., Liu, Y., Song, S., and Wang, G. H. X. (2023). "Elasticvit: conflict-aware supernet training for deploying fast vision transformer on diverse mobile devices," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 5829–5840. doi: 10.1109/ICCV51070.2023.00536

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., et al. (2017). "Attention is all you need," in *Advances in Neural Information Processing Systems*, 30.

Wang, C.-Y., Bochkovskiy, A., and 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*, 7464–7475. doi: 10.1109/CVPR52729.2023.00721

Wang, Y., Zhang, X., Yang, T., and Sun, J. (2022). "Anchor detr: query design for transformer-based detector," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2567–2575. doi: 10.1609/aaai.v36i3.20158

Yu, F., Wang, D., Shelhamer, E., and Darrell, T. (2018). "Deep layer aggregation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2403–2412. doi: 10.1109/CVPR.2018.00255

Yung, N. D. T., Wong, W. K., Juwono, F. H., and Purnamasari, A. (2022). "Safety helmet detection using deep learning: Implementation and comparative study using yolov5, yolov6, and yolov7," in *2022 International Conference on Green Energy, Computing and Sustainable Technology (GECOST)* (IEEE), 164–170. doi: 10.1109/GECOST55694.2022.10010490

Zhang, H., Li, F., Liu, S., Hu, H., Li, L., Wang, X., et al. (2022). Dino: detr with improved denoising anchor boxes for end-to-end object detection. *arxiv preprint arxiv:2203.03605*.

Zhu, X., Su, W., Lu, L., Li, B., Wang, X., and Dai, J. (2020). Deformable detr: Deformable transformers for end-to-end object detection. *arxiv preprint arxiv:2010.04159*.

Zong, Z., Song, G., and Liu, Y. (2023). "Detrs with collaborative hybrid assignments training," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 6748–6758. doi: 10.1109/ICCV51070.2023.00621