Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2024.051728

ARTICLE

Improved YOLOv8n Model for Detecting Helmets and License Plates on Electric Bicycles

Qunyue Mu[1,2], Qiancheng Yu[1,2,*], Chengchen Zhou[1,2], Lei Liu[1,2] and Xulong Yu[1,2]

[1]The College of Computer Science and Engineering, North Minzu University, Yinchuan, 750021, China

[2]The Key Laboratory of Images and Graphics Intelligent Processing of State Ethnic Affairs Commission, North Minzu University, Yinchuan, 750021, China

*Corresponding Author: Qiancheng Yu. Email: 1999019@nmu.edu.cn

Received: 13 March 2024 Accepted: 06 May 2024 Published: 18 July 2024

ABSTRACT

Wearing helmets while riding electric bicycles can significantly reduce head injuries resulting from traffic accidents. To effectively monitor compliance, the utilization of target detection algorithms through traffic cameras plays a vital role in identifying helmet usage by electric bicycle riders and recognizing license plates on electric bicycles. However, manual enforcement by traffic police is time-consuming and labor-intensive. Traditional methods face challenges in accurately identifying small targets such as helmets and license plates using deep learning techniques. This paper proposes an enhanced model for detecting helmets and license plates on electric bicycles, addressing these challenges. The proposed model improves upon YOLOv8n by deepening the network structure, incorporating weighted connections, and introducing lightweight convolutional modules. These modifications aim to enhance the precision of small target recognition while reducing the model's parameters, making it suitable for deployment on low-performance devices in real traffic scenarios. Experimental results demonstrate that the model achieves an mAP@0.5 of 91.8%, showing an 11.5% improvement over the baseline model, with a 16.2% reduction in parameters. Additionally, the model achieves a frames per second (FPS) rate of 58, meeting the accuracy and speed requirements for detection in actual traffic scenarios.

KEYWORDS

YOLOv8; object detection; electric bicycle helmet detection; electric bicycle license plate detection

1 Introduction

Electric bicycles, serving as a convenient, cost-effective, and eco-friendly mode of short-distance transportation, are increasingly becoming the preferred choice for individuals' local commutes. As of 2022, the total number of electric bicycles in China has surged to 350 million. However, this rise in electric bicycle usage has been paralleled by a concerning increase in traffic accidents involving electric bicycles. Notably, head injuries account for the highest proportion of driver injuries and fatalities in such accidents. Wearing a safety helmet correctly while riding an electric bicycle can significantly mitigate the risks of severe head injuries and fatalities resulting from traffic collisions.

This work is licensed under a Creative Commons Attribution 4.0 International License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

With the nationwide implementation of registration policies for electric bicycles in China and the enforcement of the "One Helmet, One Belt" initiative by the Ministry of Public Security, there has been a notable reduction in accidents caused by traffic violations. However, despite the current national regulations mandating compulsory helmet usage for electric bicycle riders, instances of non-compliance persist on the roads. Moreover, law enforcement authorities face significant challenges in deploying manpower at scale for continuous monitoring and enforcement of helmet-wearing regulations. The research objective is to explore detection technologies and apply them to identify helmets and license plates on electric bicycles, aiming to standardize safety practices among electric bicycle riders. This research holds substantial practical importance.

In traditional helmet and number plate detection methods, most rely on physical texture or colour features for localisation and recognition. Jia et al. [1] proposed a model that combines helmet colour, local features and histogram of oriented gradient (HOG) features to identify safety helmets used in construction sites. Hsu et al. [2] proposed a combination of directional gradient histogram (HOG) and support vector machine (SVM) to detect motorcycle number plates. Zhong et al. [3] proposed a method based on character edge detection operators to detect car license plates, and an algorithm based on colour features to extract license plate characters, but the effectiveness is poor when the colour of the vehicle body is close to the license plate. These traditional machine learning methods rely heavily on specific scenes. For example, the effectiveness of identifying safety helmets based on colour features in construction sites is better when the helmets are of a single colour and uniform style. However, their effectiveness needs to be verified in scenarios with different types and styles of electric vehicle helmets, as well as complex traffic environments. In traffic scenes where the environment is complex, traditional machine learning methods face significant challenges in target detection. With the development of deep learning technology, the use of convolutional neural networks enables object detection in more complex background environments. This is particularly beneficial for detecting helmets and number plates in complex traffic scenarios, such as the non-motorised vehicle lane studied in this paper.

This study introduces an enhanced detection model for identifying electric bicycle helmets and license plates using YOLOv8n in non-motorized vehicle lane scenarios. The model is designed to improve detection accuracy while minimizing parameter usage. The efficacy of the upgraded algorithm is confirmed through ablation experiments. Comparative analyses with other object detection algorithms underscore its superior performance. The key research methodologies include:

(1) Refinement of the YOLOv8n network structure: By enhancing the network structure and incorporating a small target detection head, the model can better capture features across various scales and abstraction levels, leading to enhanced overall performance.

(2) Improvement of the Concat module by integrating weighted connections: This enables the learning of weights for input tensors from different directions, thereby reinforcing the influence of input tensors with higher weights.

(3) Development of lightweight convolutional modules to replace original convolutional layers: These lightweight modules significantly reduce the model's parameter count and computational complexity, with minimal impact on accuracy. This adaptation enhances the model's suitability for deployment on edge devices in traffic settings.

The paper is structured as follows: Section 2 provides an overview of the current research status on electric bicycle license plates and helmets, introducing the YOLOv8 model. Section 3 elaborates on the enhancements made to the YOLOv8 model, offering detailed derivations of the necessary equations. Section 4 discusses the experimental setup and result analysis. Finally, in Section 5, conclusions are presented along with potential avenues for future work.

2 Related Work

2.1 Related Research

Research on the detection of electric bicycle license plates and helmets currently predominantly relies on deep learning, which is further divided into one-stage object detection and two-stage object detection tasks. One-stage object detection involves directly generating object category confidence and position coordinates through a single detection pass. Representative models include the YOLO series [4], SSD [5], RT-DETR [6], and others. One-stage object detection models exhibit fast recognition speed and lightweight model size, making them suitable for lightweight devices or real-time detection scenarios where hardware requirements are not stringent. However, these models may have slightly lower detection accuracy. Two-stage object detection models identify the location of the target in the first stage, obtaining proposed bounding boxes to improve accuracy and recall. In the second stage, these proposed boxes undergo classification to refine the position further. Representative models for two-stage detection include Faster CNN [7], among others. Two-stage object detection models typically have larger model sizes, higher detection accuracy, but slower detection speed, making them suitable for high precision detection tasks.

Many scholars have conducted research on the detection of targets such as helmets or license plates on electric bicycles. Wang et al. [8] improved the two-stage object detection algorithm Faster R-CNN. They integrated Inception Res-Net_v2 into the overall architecture of Faster R-CNN to extract features specific to car license plates. The algorithm then employed threshold binarization for character segmentation, followed by the use of the mLetNet5 network for license plate character recognition. Qi et al. [9] performed row projection correction on license plates, utilized the MobileNet-SSD algorithm to detect single-row and stacked characters, and achieved good recognition results by feeding stacked characters into a recognition network based on the CTC loss. Islam et al. [10] explored the use of two separate models for car license plate detection and number recognition, comparing the detection performance of different models. They proposed using Faster R-CNN for license plate detection and sending the recognized license plate images to an SSD model for license plate number recognition, ultimately achieving a 98% accuracy in license plate detection and a 91.67% accuracy in license plate character recognition. While these algorithms demonstrate high detection accuracy, their network models are relatively large, leading to slow prediction speeds, which may not be conducive to real-time detection in traffic environments. Alharbi et al. [11] proposed a method for license plate recognition using the YOLO model, followed by license plate number recognition using the RCNN model. They enhanced the YOLO model by modifying its architecture and introduced K-means++ clustering for precise identification. The final model achieved promising results in terms of detection accuracy. Zhang et al. [12] proposed an improved YOLOv5s-BC algorithm for recognizing electric bicycle helmets. They introduced soft pooling into the SPP layer of YOLOv5s, replaced the PAN network with BiFPN, and added CA attention. Finally, they changed the loss function to EIOU. The overall model mAP reached 98.4%, representing a 6.3% improvement compared to the initial YOLOv5 model. Aboah et al. [13] introduced a small data sampling technique for dataset processing and used YOLOv8 for model training to identify motorcycles and driver helmets. They achieved the seventh place in the 2023 AI City Challenge. Zhuang et al. [14] proposed an improved YOLOv5m algorithm, modifying the DIOU loss function, adding ECA attention between the Backbone and Neck, and redesigning anchor boxes using the K-means clustering algorithm. They also applied Mosaic augmentation to the dataset. The overall recognition accuracy (mAP) for helmets and license plates increased from 90.55% to 92.7%, effectively enhancing model precision. This model not only detects helmets but also recognizes license plates, although it does not identify license plate numbers, which is a slight limitation.

In the current research, there is a scarcity of studies that discuss the integration of electric bicycle license plate numbers with helmet detection within a single model. This paper independently constructed a dataset of nonmotorized vehicles captured from an overhead perspective on footbridges. The dataset includes annotations for 22 classes of targets, such as helmets, license plates, and license plate numbers. The aim is to utilize a single model to recognize electric bicycle riders, helmets, and license plate numbers. This approach is intended to assist law enforcement officers in using deep learning-based object detection technology to address issues related to electric bicycle helmets.

2.2 YOLOv8 Model

The YOLOv8 model is one of the mainstream single-stage object detection models, known for its outstanding detection speed among numerous models, making it highly suitable for real-time detection tasks.

Compared to the YOLOv5 version, YOLOv8 replaces the original C3 module with the C2f module, which can fuse more features; removes the convolutional layers before up sample in the Head part; adopts the concept of anchor free, enhancing the model's ability to recognize the same object at different scales, etc. The YOLOv8 model consists of five different network depths: YOLOv8n, YOLOv8s, YOLOv8m, YOLOv8l, and YOLOv8x. Among them, YOLOv8n is the model with the smallest network depth in this series, suitable for deployment on embedded devices with lower performance. As traffic cameras themselves belong to low-performance embedded devices, this paper improves the YOLOv8n model structure to achieve the detection of electric bicycle license plates, helmets, drivers, and other targets.

The YOLOv8 model is composed of a Backbone and a Head. The Backbone part uses convolutional layers and C2f layers to extract input features, integrating features in SPPF. The head part uses Path Aggregation Network (PAN) [15] and Feature Pyramid Network (FPN) [16]. PAN is a bottom-up feature extraction structure, while FPN is a top-down feature extraction structure. After concatenating the ensample layers with the outputs of the C2f layers in the backbone with consistent sizes, feature fusion is enhanced. Finally, the features extracted by the C2f layer in the head are sent to the detection head, which calculates the position of the target box and classification confidence, predicting targets of different sizes on different feature maps. Fig. 1 shows the network structure of the YOLOv8 model.

3 Improved Model

The main focus of this paper is to address the issues of missed detections and low accuracy in detecting small objects, such as license plates, in the baseline YOLOv8n model. This paper first deepens the network architecture to enable the extraction and fusion of a wider range of features. Additionally, a new small object detection head has been added to enhance the accuracy of detecting small objects. Secondly, in the Concat module for network feature fusion, weighted concatenation is used to emphasise the importance of features that conteibute significantly to model performance improving, thus enhancing model accuracy. Finally, a lightweight convolution module, Group Cross Conv (GCC), is designed to replace the original Conv module in the Backbone section. The final model shuows significant improvements in accuracy while reducing the number of model parameters and the size of the model files. The structure of the improved model is shown in Fig. 2. The first improvement involves the addition of a new network structure and small target detection head on a gray background. In the second enhancement, the Concat module in the Head section is replaced with the Weighted Concat module, highlighted in green. The third enhancement occurs in the BackBone, where the Conv module is replaced with the lightweight GCC module.

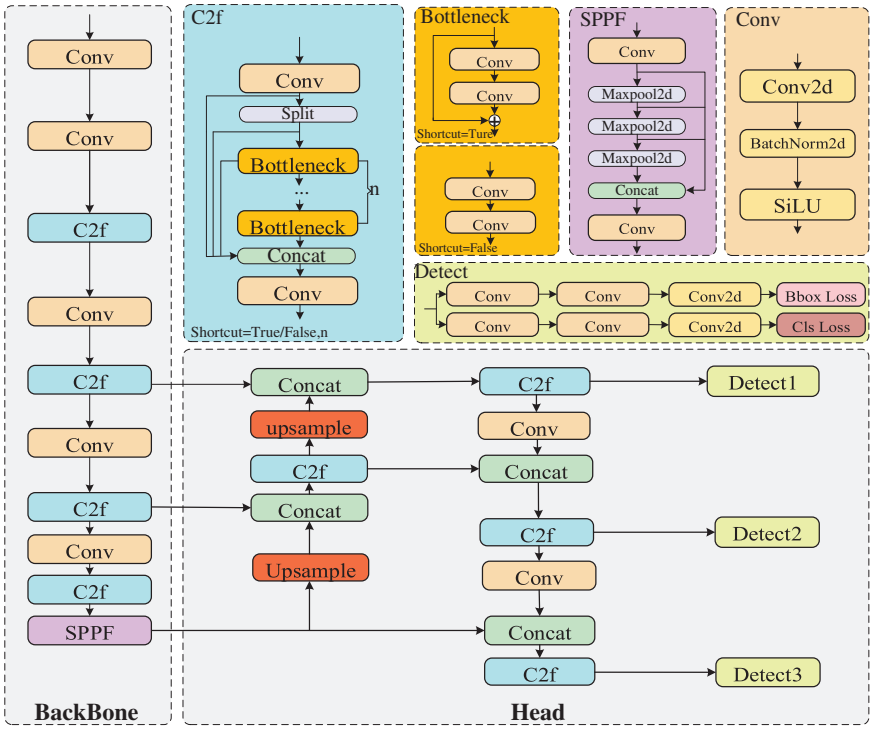

Figure 1: YOLOv8 model structure diagram

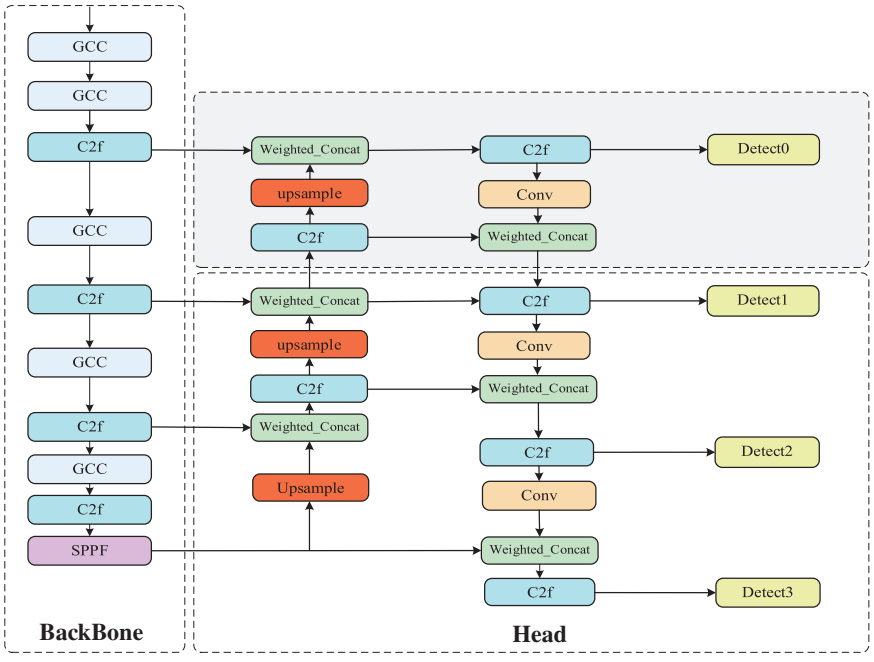

Figure 2: Improved YOLOv8n model structure diagram

3.1 Network Structure and Detection Head Improvements

In the baseline YOLOv8 model, inherited from previous YOLO version, three detection heads are characteristic, covering typical object detection scenarios. However, in real traffic scenes, small objects such as license plates occupy few pixels, contain limited information, and belong to various categories, posing challenges for accurate recognition by the model.

To address these issues, this paper extends the depth of the network structure in the head region of the YOLOv8 model, allowing for better extraction and fusion of image features. An additional C2f layer is introduced to extract deeper information and fuse shallow information, enriching semantic representation. An added up sample layer adjusts the feature map size for easier fusion with other deep feature maps. Furthermore, a connecting layer concatenates shallower and deeper information, allowing the network to combine deep abstract features with texture and shape features obtained from shallower layers. After feature extraction, another C2f layer is employed for feature fusion. Lastly, a detection head is appended to the C2f layer to strengthen feature detection for small objects and continue feature extraction through convolutional layers, transmitting features to different depths. The original YOLOv8 detection head consists of three feature map sizes: 20×20, 40×40, and 80×80. The newly added small object detection head can detect targets on the output feature map with a size of 160×160, effectively improving the accuracy of recognizing small objects such as license plates and helmets.

Assuming an input image size of 1280×1280, the original YOLOv8n detection head has a size of 80×80, with a receptive field of $1280/80 = 16$, meaning targets smaller than 16×16 pixels are likely to be ignored. The newly added small object detection head has a size of 160×160, with a receptive field of $1280/160 = 8$, capable of detecting targets as small as 8×8 pixels, thus enhancing sensitivity to small objects. The specific improvements in the network structure and detection head are depicted in the gray background section of Fig. 2.

3.2 Introduction of Weighted Concatenation

In the Head section of YOLOv8, the Concat module is used to merge features from different depths to enhance the network's semantic representation. Traditional Concat operations simply concatenate two tensors without considering weights for tensors from shallow and deep layers. Inspired by the Bi-directional Feature Pyramid Network (BiFPN) structure [17], this paper introduces weighted concatenation, where tensors from different inputs are multiplied by calculated weights before being connected, thus enhancing the contribution of features significant to the model.

BiFPN designs addition operations for tensors at different depths, merging shallow information such as edges and textures with deep abstract features, enriching network connections and semantic representation. Moreover, after normalizing the weights, this approach prevents scaling issues, ensuring stable feature fusion and mitigating problems like gradient explosion or vanishing during model learning. Weighted Concat calculates weights for different tensors, enhancing the significance of valuable input branches and overall model recognition accuracy. Considering the original Concat operation in the model simply concatenates tensors from different inputs, this paper replaces the addition operation in BiFPN with concatenation to maintain consistency in the number of channels after Concat operations and avoid sharp declines in model performance due to significant reductions in channel numbers. From a lightweight perspective, the Weighted Concat operation eliminates cross-network depth connection features in BiFPN, only improving the weights of different tensors during feature fusion, thereby reducing complexity costs and parameter quantities, and better representing

input features. The equation for Weighted Concat is as follows:

$$P = Concat\left(\frac{w_1}{w_1 + w_2 + \varepsilon} \cdot X_1, \frac{w_2}{w_1 + w_2 + \varepsilon} \cdot X_2\right) \tag{1}$$

In Eq. (1), P represents the output tensor, *Concat* denotes the concatenation operation, W_1 and W_2 are learnable parameters used to learn the weights of different input tensors, X_1 and X_2 are two input tensors, and ε is a constant with a value of 0.0001, which helps to avoid division by zero. During backpropagation, W_1 and W_2 continuously learn and adjust their values to connect the two input tensors according to different weight coefficients, thereby enhancing the accuracy of the model. The Weighted Concat is represented by the green module in Fig. 2 and the pseudo-code of module algorithm is shown in Fig. 3.

Algorithm Weighted Concatenation

Input : Tensors X_1,X_2
Output : Tensors Y
Parameters:
X_1 - input tensors.
X_2 - another input tensors.
w_1 - weighted params of X_1, with an initial value of 1.
w_2 - weighted params of X_2, with an initial value of 1.
ε - ε=0.0001 to mitigate the occurrence of division by zero as much as possible.

1. $X_1 \leftarrow \frac{w_1}{w_1 + w_2 + \varepsilon} \bullet X_1$ //Compute the weights of X_1, X_2 and learn the weight. // parameters w_1, w_2 during backpropagation.
2. $X_2 \leftarrow \frac{w_2}{w_1 + w_2 + \varepsilon} \bullet X_2$
3. $Y \leftarrow ConCat(X_1, X_2)$ //Concatenate two tensors.
4. Return Y

Figure 3: The algorithm for weighted concatenation

3.3 Introduction of Group Cross Conv (GCC) Lightweight Convolution Module

Building on these improvements, while the YOLOv8 model deepens the network structure and introduces Weighted Concat for normalised connection weights, the increase in model performance also increases the number of parameters and computational complexity, resulting in increased GFLOPs. This makes the model bulky and less conducive to running on low-performance embedded devices. Therefore, this paper proposes a plug-and-play lightweight convolution module, GCC, to replace the original Conv module.

The structure of the GCC model is depicted in Fig. 4, wherein the distinction between the GCC module and the Conv module lies in the colored portion. The GCC module consists of Group Convolution (GConv) [18], Cross Convolution (Cross Conv), Concat, Shuffling [19], Batch Normalization [20], and the SiLU activation function [21]. GConv reduces the parameter count of convolutional operations, while Cross Conv utilizes the results of GConv as input and replaces 3×3 convolutional kernels with 1×3 and 3×1 convolutional kernels, reducing the convolutional kernel parameters from 9 to 6. The results of both convolution operations are concatenated and then subjected to shuffling. Finally, the concatenated result is normalized and passed through the SiLU activation function to produce the final output.

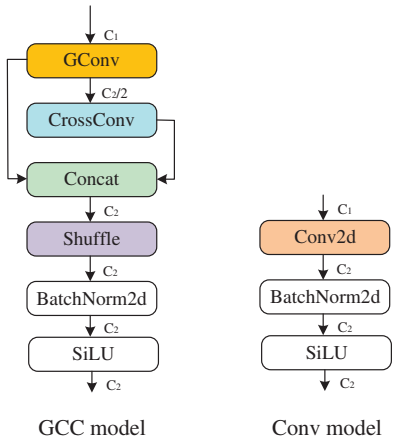

Figure 4: GCC module structure diagram and Conv module structure diagram

The module designed in this paper allows users to specify input and output channel numbers, enabling plug-and-play functionality without altering channel numbers when replacing other types of convolution modules. Assuming the original convolution layer has an input channel number of C_1 and an output channel number of C_2, the input tensor with C_1 channels undergoes group convolution operation, resulting in an output channel number of $C_2/2$ with $C_1/2$ groups. The output tensor of the grouped convolution is then subjected to a CrossConv operation, using 1×3 and 3×1 convolutions for feature extraction, respectively. The results of the grouped convolution and CrossConv are concatenated to obtain a tensor with C_2 channels. To enhance inter-channel relationships, a shuffle operation is performed as the final step, yielding the ultimate result. The pseudo-code of the GCC module algorithm is shown in Fig. 5.

Algorithm Group Cross Conv

Input : Tensors X
Output : Tensors Y

```
1.  out1 ← GroupConv(X)          //The input tensor is first divided into groups,
                                 //followed by convolution on each group.
2.  out2 ← CrossConv(out1)       //The input tensor undergoes a 3×1 convolution
                                 //followed by another 3×1 convolution.
3.  out3 ← ConCat(out1,out2)     //Concatenates two tensors.
4.  out3 ← Shuffle(out3)         //Shuffling tensors along the channel dimension.
5.  out3 ← BatchNorm(out3)       //Normalizing tensors.
6.  Y    ← SiLU(out3)            //Applying activation function to tensors.
7.  Return Y
```

Figure 5: The algorithm for Group Cross Conv

To further demonstrate the lightweight effect of the GCC module when replacing the Conv module, the calculation equations for parameters (Params) is introduced.

$$Conv_{\text{params}} = K_{\text{w}} \times K_h \times C_{in} \times C_{out} \tag{2}$$

$$GConv_{\text{params}} = K_{\text{w}} \times K_h \times \frac{C_{in}}{Group} \times \frac{C_{out}}{Group} \times Group \tag{3}$$

$$CrossConv_{\text{params}} = 1 \times K_h \times C_{in} \times C_{out} + K_{\text{w}} \times 1 \times C_{in} \times C_{out} \tag{4}$$

Eqs. (2)–(4) represent the parameter calculation equations for Conv, Gconv, and CrossConv, respectively. For simplicity, the computations here omit consideration for bias parameters. In fact, within the GCC module, the bias parameter is configured as False. The meanings and values of each symbol in the equations are defined as follows: K_{w} and K_h represent the width and height of the convolution kernel, as mentioned earlier, the convolution kernel size for regular convolution and group convolution is 3×3, and for group convolution, it is 3×1 and 1×3; C_{in} and C_{out} are the input and output channel numbers, both taken as C_1 and C_2; Group is the number of groups in group convolution, taking the value $C_1/2$.

The result of the parameter calculation for *Conv* is obtained as follows:

$$\begin{aligned} Conv_{\text{params}} &= K_{\text{w}} \times K_h \times C_{in} \times C_{out} \\ &= 3 \times 3 \times C_1 \times C_2 \\ &= 9C_1C_2 \end{aligned} \tag{5}$$

The result of the parameter calculation for *GConv* is obtained as follows:

$$\begin{aligned} GConv_{\text{params}} &= K_{\text{w}} \times K_h \times \frac{C_{in}}{Group} \times \frac{C_{out}}{Group} \times Group \\ &= 3 \times 3 \times \frac{C_1}{C_1/2} \times \frac{C_2/2}{C_1/2} \times \frac{C_1}{2} \\ &= 9C_2 \end{aligned} \tag{6}$$

The result of the parameter calculation for *CrossConv* is obtained as follows:

$$\begin{aligned} CrossConv_{\text{params}} &= 1 \times K_h \times C_{in} \times C_{out} + K_{\text{w}} \times 1 \times C_{in} \times C_{out} \\ &= 1 \times 3 \times \frac{C_2}{2} \times \frac{C_2}{2} + 3 \times 1 \times \frac{C_2}{2} \times \frac{C_2}{2} \\ &= \frac{3}{2}C_2^2 \end{aligned} \tag{7}$$

Therefore, GCC_{params} is equal to the sum of $Gconv_{\text{params}}$ and $CrossConv_{\text{params}}$.

$$\begin{aligned} GCC_{\text{params}} &= Gconv_{params} + GrossConv_{params} \\ &= 9C_2 + \frac{3}{2}C_2^2 \end{aligned} \tag{8}$$

To facilitate a deeper comparison between $Conv_{\text{params}}$ and GCC_{params}, taking into account that the output channel count is practically non-zero, we can utilize Eqs. (5) and (8) to eliminate C_2, resulting in $9C_1$ and $9 + (3/2)C_2$, respectively.

In the YOLOv8 Backbone, the input channel count C_1 typically doubles after passing through the Conv module. In the YOLOv8n model architecture, there are a total of five convolutional modules within the Backbone. The first Conv module transforms a 3-channel input through convolution to yield a 16-channel output. For this specific module, the values for input and output channels are employed in the following equations: $9C_1 = 27$ and $9 + (3/2)C_2 = 33$. The lightweight of the GCC module in our study did not yield the expected results; instead, it slightly increased the parameter count. However, in the subsequent Conv modules of the Backbone section, when the GCC module was employed as a replacement, a parameter lightweight effect was achieved due to the output channel count being twice that of the input channel count. For instance, in the second convolutional module where the input tensor is 16 channels and the output tensor is 32 channels, applying the results derived from the equations: $9C_1 = 144$ and $9 + (3/2)C_2 = 57$ demonstrates the superior performance of the GCC module in reducing parameter count. Moreover, as one moves closer to the bottom of the Backbone, the reduction in parameter count becomes more pronounced. This trend persists in subsequent Conv modules within the Backbone, achieving parameter lightweight. Further calculations in this regard are omitted for brevity. Upon substituting the values into the calculations, it is observed that GCC_{params} is smaller. Therefore, replacing the original Conv convolution module with GCC can achieve lightweight.

In fact, in Eqs. (5) and (8), it can be observed that in terms of the comparison of parameter quantity between the Conv module and the GCC module, when the number of channels is high, the coefficients become crucial in determining the size of the parameters. When extended to the typical Conv modules in YOLOv8, under common scenarios where the output channel number is twice the input channel number, C_2 can be replaced by $2 \times C_1$. This substitution clearly highlights the contribution of the GCC module in parameter optimization.

Although reducing parameters may lead to a potential decrease in accuracy, the use of different convolutions for feature extraction results in features that are richer compared to the original convolution. This richness in features can help mitigate the accuracy impact caused by lightweight. The results of the GConv module and the CrossConv module are concatenated, followed by a Shuffle operation. The concatenation and shuffling operations incur minimal additional computational and spatial costs. Channel shuffling allows the model to learn features from each channel, compensating for the weakened inter-channel connections introduced by previous grouped convolutions. Therefore, the designed GCC module in this paper achieves higher lightweight when replacing the Conv convolution module. The addition of the GCC module is illustrated in the Backbone section of Fig. 2.

4 Experimental Results and Analyses

4.1 Dataset

Due to the lack of publicly available datasets for identifying targets such as electric bicycles, license plates, and helmets in traffic environments, this paper constructs a dataset of images from non-motorized vehicle lanes at traffic intersections, manually annotating information such as electric bicycles, electric tricycles, helmets, and license plates.

The images in the dataset are manually collected by the authors, taken at the Huaiyuan Night Market Overpass in the Xixia District of Yinchuan City, Ningxia. The photographs were captured using a Redmi K20 Pro smartphone. The collection method involved fixing the smartphone on a bracket above the pedestrian overpass and recording videos at a rate of 30 frames per second with a 4K resolution. Subsequently, frames were extracted from the videos. Additionally, a small number of shared electric bicycle photos parked by the roadside were added to enrich the dataset.

The resolution of the collected images is 3840 × 2160. In YOLOv8's object detection task, the default practice is to resize input images to 640 × 640 resolution. Simple resizing of images may lead to loss of details, especially for small targets like license plates. However, resizing input images to match the original image size would require significant GPU memory and might lead to memory overflow during model training. Therefore, this paper segments the images in the self-built dataset without altering the original RGB values of pixels. In each image, as the background information predominates, target segmentation is performed to remove unnecessary background information, facilitating subsequent training. During segmentation, the background information around the target is appropriately retained. Because electric bicycles of different brands vary in size, and some targets may appear closer or farther in images, the pixel sizes of the targets also vary, resulting in non-uniform image resolutions in the dataset. For annotation, a total of 22 labels are used, including numeric characters 0–9, uppercase letters A, H, M, Q, Chinese characters, and general categories such as helmets, license plates, people, electric bicycles, bicycles, and electric tricycles. The dataset used in this paper comprises a total of 5546 images, which are randomly split into training, validation, and test sets in an 8:1:1 ratio. Specifically, the training set consists of 4474 images, the validation set consists of 506 images, and the test set consists of 566 images.

In Table 1, all types and quantities of labels are summarized. Although some labels have low counts, Focal Loss in YOLOv8 can effectively handle this class imbalance issue by reducing the weights of easily classified samples during training, enabling the model to focus on hard-to-classify samples. This helps mitigate the poor performance of the model on categories with fewer instances.

Table 1: Types and quantities of dataset labels

Label	Count	Label	Count	Label	Count
Human	6768	0	624	7	295
e-bike	4020	1	476	8	318
Bikecard	1180	2	511	9	296
Bike	947	3	390	A	664
Casque (helmet)	629	4	239	M	56
e-3bike	336	5	376	H	76
Ning (chinese character)	653	6	370	Q	52
Lin (chinese character)	354				

4.2 Experimental Setup and Training Process

Software Environment: Windows Server 2019 Datacenter 64-bit operating system, PyCharm 2023.1.2 Community Edition, CUDA 11.8, Python 3.8, PyTorch 2.0.1.

Hardware Environment: 2 Intel® Xeon® Gold 6154 @3.00 GHz processors, 256 GB DDR4 2666 MHz memory, NVIDIA TITAN V with 12 GB memory, 4 TB mechanical hard drive (for storing datasets), and 1 TB solid-state drive (system disk).

The default input image size for YOLOv8n is 640 × 640 pixels. When the input image exceeds this size, it is compressed before object detection. However, most images in the self-built dataset in this paper are around 1000 × 1000 pixels. Using the default size may result in loss of license plate details during image compression, leading to poor recognition performance. Therefore, the model's imgsz parameter was adjusted to 1280 to identify complete image features.

The hyperparameters of the proposed model and various comparison models are consistent, as shown in Table 2.

Table 2: Hyperparameters setting of model training

Parameter names	value
Batchsize	4
Imgsz	1280
Epochs	200
Optimizer	SGD
Momentum	0.937
Close mosaic	10
Model (pretrained weights)	NULL

4.3 Evaluation Metrics

In this paper, multiple metrics are used to evaluate the performance of the models in the experiments. The selected metrics include:

1. Mean Average Precision (mAP): MAP is calculated by computing the average precision and recall for each individual class, and then taking the mean. It represents the overall detection accuracy of the model across all classes.

2. Parameters: Parameters refer to the number of parameters in the model. A smaller value is preferable.

3. Frames per Second (FPS): FPS indicates the number of images that can be processed per second on the current hardware environment. A higher value is desirable.

4. Floating Point Operations (FLOPs): FLOPs represent the number of floating-point operations required per second when running the model. A lower value is preferable.

5. Precision: Quantifies the proportion of accurately predicted instances among all positive predictions. A higher Precision value indicates fewer false positives, thereby reflecting a more accurate model performance.

6. Recall: Delineates the proportion of correctly predicted positive instances out of all actual positive instances. A greater Recall value signifies fewer instances being missed, thus indicating a more comprehensive model performance.

4.4 Benchmark Model Comparison Experiment

Various mainstream object detection models are compared using the self-built dataset in this paper. The performance results of each model are presented in Table 3.

Table 3: Performance comparison of different models

Model	mAP@ 0.5	mAP@ 0.5–0.95	Precision	Recall	Parameters /M	FLOPs /G	FPS	Training time/H
Faster RCNN	18.8	11.2	15.2	20.1	137.1	370	13	65.6
SSD	30.4	11.4	29.6	28.4	26.4	141.2	51	33
RT-DETR-l	61.2	40.1	66.2	61.7	32	–	18	33.9
PP-PicoDet-l	14.2	–	–	–	5.8	–	8.2	12.2
TTFNET	3.7	1	1.2	13.8	45.7	–	26	34.4
TOOD	15.7	7.9	15.4	24.3	32.1	–	13.7	28.2
YOLOv5n	75.4	42.2	65.1	71	1.7	4.2	74	9.5
YOLOv6n	57.3	36.7	45.4	68	4.2	11.8	123	7.6
YOLOv7tiny	72.9	40.8	67.5	69.8	6.1	13.2	69	15.1
YOLOv8n	80.3	50.2	75.8	71.7	3	8.1	128	8.2
YOLOv8s	89.6	55.9	81.8	85.2	11.1	28.5	67	8.1
Our model	91.8	56.7	85.5	86.3	2.5	11.2	58	10.5

In the experimental results, Faster RCNN exhibits lower precision. Upon examining the precision for each class, it was observed that this model performs well on medium-sized and large objects, but its performance on small targets such as license plates is poor. This contributes to the overall lower mean Average Precision (mAP). SSD, PP-PicoDet-l [22], TOOD [23], TTFNet [24], and YOLOv6n [25] show significant differences in detection metrics compared to the YOLOv8n model. Although the RT-DETR-l model slightly improves precision compared to other models, its large parameter count and low FPS make it unsuitable for real-time detection in traffic scenarios. YOLOv5n offers some advantages in terms of parameter count and computational complexity, but still falls short in precision compared to YOLOv8. YOLOv7n [26] outperforms most models but still lags behind YOLOv8n in each metric. The YOLOv8n model demonstrates high precision, relatively low parameter count, and computational complexity, with an FPS suitable for real-time detection in traffic environments. This is one of the reasons why this model was chosen as the benchmark in this study.

Comparing these model performance metrics, it can be observed that the proposed model in this paper exhibits significant improvements in precision compared to other benchmark models. Not only does it achieve higher accuracy than YOLOv8s, but it also has clear advantages in terms of parameter count and computational complexity, making it easily deployable on low-cost, performance-limited embedded devices.

4.5 Ablation Experiment

To validate the effectiveness of the proposed improved model, ablation experiments were conducted to assess the impact of each module improvement on the overall model performance. Sequentially, improvements including increased network depth and small target detection head, Weighted

Concat, and GCC lightweight convolution replacing Backbone convolution layers were compared. These improvements were added to the original YOLOv8n model, and training was conducted for 200 epochs under the same experimental conditions. The training results are summarized in Table 4.

Table 4: YOLOv8n ablation experiment results

Model	Network depth & detection head	Weighted Concat	GCC	mAP @0.5/%	mAP @0.95/%	Precision	Recall	FLOPs /G	Parameters /M	FPS	Model size /KB
YOLOv8n-1				80.3	50.2	75.8	71.7	8.1	3.0	128	6135
YOLOv8n-2	√			91.4	56.3	83.6	86.9	12.2	2.9	88	6261
YOLOv8n-3	√	√		91.7	57.4	85.9	87.4	12.2	2.9	88	6266
Our model	√	√	√	91.8	56.7	85.5	86.3	11.2	2.5	58	5507

YOLOv8n-1 serves as the baseline model employed in our experiments. Without any model modifications, its mAP@0.5 metric is observed to be only 80.3%, indicating suboptimal detection accuracy. YOLOv8n-2, an enhanced version of the baseline model, features a deeper network structure and introduces a specialized small-object detection head. This design significantly improves the model's precision to 91.4%. However, this enhancement comes at the cost of increased computational complexity. YOLOv8n-3 builds upon YOLOv8n-2 by replacing the original Concat connection with Weighted Concat, achieving a marginal 0.3% performance improvement with almost no change in model parameters or computational complexity. In this paper, we propose a novel model that further refines the architecture based on YOLOv8n-3. We replace Conv modules in the Backbone with lightweight convolutional modules (GCC). This adjustment results in a substantial reduction in both model computation complexity and parameter count. Despite marginal improvements in the mAP@0.5 metric, there is a noticeable reduction in the model's parameter count. Furthermore, the generated model files exhibit a significant decrease in storage space requirements. This presents a cost-effective storage and computational solution for the practical deployment of the model.

4.6 Detection Results and Analysis

In Fig. 6, a selection of images is presented to test the detection performance of the proposed model. To ensure privacy, one digit of the license plate number is obscured in the displayed images. The first row shows the original images, the second row shows the detection results of the Faster RCNN model, the third row shows the detection results of the YOLOv6n model, the fourth row shows the detection results of the SSD model, the fourth row shows the detection results of the TOOD model, the fifth row shows the detection results of the YOLOv8n baseline model, and the last row shows the detection results of our model proposed in this paper. The selected comparison images include three images captured in normal weather conditions and two images captured in rainy weather. The selection of targets includes electric bicycles, electric tricycles, bicycles, helmets, license plates, license plate numbers, and pedestrians. The confidence threshold used for detection is set to 0.5, meaning that the model annotates targets with a confidence score greater than 50%.

Figure 6: Comparison of model detection effects

By comparing the detection results in the images, it can be observed that all models perform well in recognizing large objects but exhibit poorer performance in identifying small objects. Faster RCNN

and TOOD failed to recognize any characters on the license plates, and Faster RCNN also missed detecting helmets. YOLOv5n and YOLOv6n were able to identify some characters on the license plates based on this, but not comprehensively. Among these benchmark models, YOLOv8n performs the best, being able to recognize the most license plate characters. This indicates that under the framework of deep learning, feature extraction for small objects remains challenging. In models based on anchor boxes, default anchor boxes may not meet the requirements for detecting small objects, necessitating careful design of anchor boxes based on the characteristics of object detection tasks. YOLOv8n's anchor-free design conveniently circumvents this complexity. Additionally, in small object detection, existing detection heads may not satisfy the requirements for detecting tiny objects, necessitating better optimization of the model network to enable the extraction of features from small objects. This is precisely the work carried out in Sections 3.1 and 3.2 of this paper. In contrast, the improved YOLOv8n model proposed in this paper accurately detects all targets, regardless of whether it is sunny or rainy. This demonstrates that after the improvement of the YOLOv8n baseline model, the model is more sensitive to small targets such as license plate numbers, enabling high-precision detection of electric bicycles, helmets, and license plates in traffic scenarios.

5 Conclusions and Prospect

This paper conducted experiments using various baseline models to detect targets such as electric bicycle drivers, helmets, and license plates, with the aim of identifying the optimal baseline model for further improvement. To address the limitations of the baseline model in detecting small targets like license plate numbers, the YOLOv8n model's network structure was enhanced by deepening it and introducing a small object detection head. The feature concatenation component of the model was refined by incorporating weighted parameters. Additionally, to improve accuracy, a lightweight convolution module (GCC) was devised to reduce the model's computational complexity and parameter count, making it more suitable for deployment on embedded devices like traffic cameras. The model developed in this study enables different types of target detection on non-motorised vehicle lanes in traffic scenarios, prompting e-bike riders to wear safety helmets to reduce injuries from traffic accidents.

The experimental results illustrate a substantial enhancement achieved by the improved YOLOv8n model presented in this paper compared to the original baseline model. The mAP@0.5 and mAP@0.5–0.95 metrics saw notable increases of 11.5% and 6.5%, respectively, while reducing the parameter count from 3 to 2.5 M through lightweight optimization. Despite an increase in computational complexity (FLOPs) due to network depth and the addition of a small object detection head, this was offset by a decrease following the replacement of the original convolution module with the lightweight GCC module. This adaptation enables the model to run operate on devices with limited computational capabilities for effectively detecting helmets and license plates of electric bicycle drivers. Consequently, the enhanced model proposed in this study can serve as a high-performance and efficient tool to support traffic police in enforcing helmet regulations for electric bicycle drivers in practical scenarios.

Nevertheless, the model is not without limitations. For example, the identified license plate of an electric bicycle requires post-processing to ascertain the sequential arrangement of the license plate numbers through localization. Additionally, there is reduced accuracy in detecting pedestrians wearing raincoats on rainy days. Future endeavors will concentrate on developing software to transform the license plate numbers detected by the model into a more realistic character sequence, aligning with their respective positions. Moreover, efforts will be directed towards refining the model and enriching

the dataset with images captured in diverse weather conditions, including rainy, foggy, and dusty atmospheres, to bolster the model's resilience and precision.

Acknowledgement: The authors of this article would like to express our sincere gratitude to the editors and reviewers for their invaluable advice and expert guidance, which significantly enhanced the quality of this research.

Funding Statement: This work was supported by the Ningxia Key Research and Development Program (Talent Introduction Special Project) Project (2022YCZX0013); North Minzu University 2022 School-Level Scientific Research Platform "Digital Agriculture Enabling Ningxia Rural Revitalization Innovation Team" (2022PT_S10); Yinchuan City University-Enterprise Joint Innovation Project (2022XQZD009); Ningxia Key Research and Development Program (Key Project) Project (2023BDE02001).

Author Contributions: The authors confirm contribution to the paper as follows: study conception and design: Qunyue Mu, Qiancheng Yu; data collection: Qunyue Mu, Xulong Yu; analysis and interpretation of results: Qunyue Mu, Chengchen Zhou, Lei Liu; draft manuscript preparation: Qunyue Mu, Qiancheng Yu. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: The data that support the findings of this study are available from the corresponding author.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] J. Jia, Q. Bao, and H. Tang, "Safety helmet wearing detection based on deformable component model," *Comput. Appl. Res.*, vol. 33, no. 3, pp. 953–956, 2016.

[2] G. Hsu, S. Zeng, C. Chiu, and S. Chung, "A comparison study on motorcycle license plate detection," in *2015 IEEE Int. Conf. Multimed. Expo Workshops (ICMEW)*, Turin, Italy, 2015, pp. 1–6.

[3] W. Zhong, Z. Du, X. Xu, X. Huang, and T. Zhu, "License plate localization method based on character edge point extraction," *Comput. Eng. Design*, vol. 38, no. 3, pp. 798–800+813, 2017.

[4] R. Joseph, K. D. Santosh, B. G. Ross, and F. Ali, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Las Vegas, Nevada, USA, 2016, pp. 779–788.

[5] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed and C. Fu, "SSD: Single shot multibox detector," in *Comput. Vis.–ECCV 14th Eur. Conf.*, Amsterdam, The Netherlands, 2016, pp. 21–37.

[6] W. Lv *et al.*, "Detrs beat yolos on real-time object detection," arXiv preprint arXiv:2304.08069, 2023.

[7] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1137–1149, 2017. doi: 10.1109/TPAMI.2016.2577031.

[8] Y. Wang, J. Zhang, and Q. Yin, "License plate recognition algorithm based on faster R-CNN," *J. Beijing Normal Univ.: Nat. Sci.*, vol. 56, no. 5, pp. 647–653, 2020.

[9] Z. Qi, K. Tu, S. Wu, and S. Zhang, "Deep learning based license plate recognition algorithm with stacked characters," *Appl. Res. Comput.*, vol. 38, no. 5, pp. 1550–1554+1558, 2021.

[10] T. Islam and R. Rasel, "Real-time bangla license plate recognition system using faster R-CNN and SSD: A deep learning application," in *2019 IEEE Int. Conf. Robot., Autom., Artif.-Intell. Internet-of-Things (RAAICON)*, Dhaka, Bangladesh, 2019, pp. 108–111.

[11] F. Alharbi, R. Alshahrani, M. Zakariah, A. Aldweesh, and A. Alghamdi, "YOLO and blockchain technology applied to intelligent transportation license plate character recognition for security," *Comput., Mat. Contin.*, vol. 77, no. 3, pp. 3697–3722, 2023. doi: 10.32604/cmc.2023.040086.
[12] R. Zhang, F. Dong, and X. Cheng, "Application of improved YOLOv5s algorithm in non-motor vehicle helmet wear detection," *J. Henan Univ. Sci. Technol.: Nat. Sci.*, vol. 44, no. 1, pp. 44–53+7, 2023.
[13] A. Aboah, B. Wang, U. Bagci, and A. Yaw, "Real-time multi-class helmet violation detection using few-shot data sampling technique and yolov8," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Vancouver, BC, Canada, 2023, pp. 5349–5357.
[14] J. Zhuang and J. Ye, "Rider helmet and license plate detection method based on improved YOLOv 5 m electric bicycle," *J. Nanjing Univ. Inform. Sci. Technol.: Nat. Sci. Edition*, vol. 16, no. 1, pp. 1–10, 2023.
[15] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path aggregation network for instance segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Salt Lake City, UT, USA, 2018, pp. 8759–8768.
[16] T. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan and S. Belongie, "Feature pyramid networks for object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Honolulu, HI, USA, 2017, pp. 2117–2125.
[17] M. Tan, R. Pang, and Q. Le, "Efficientdet: Scalable and efficient object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Seattle, WA, USA, 2020, pp. 10781–10790.
[18] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Honolulu, HI, USA, 2017, pp. 1492–1500.
[19] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An extremely efficient convolutional neural network for mobile devices," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Salt Lake City, UT, USA, 2018, pp. 6848–6856.
[20] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *32nd Int. Conf. Mach. Learn.*, Lille, France, 2015, pp. 448–456.
[21] S. Elfwing, E. Uchibe, and K. Doya, "Sigmoid-weighted linear units for neural network function approximation in reinforcement learning," *Neural Netw.*, vol. 107, pp. 3–11, 2018. doi: 10.1016/j.neunet.2017.12.012.
[22] G. Yu *et al.*, "PP-PicoDet: A better real-time object detector on mobile devices," arXiv preprint arXiv:2111.00902, 2021.
[23] C. Feng, Y. Zhong, Y. Gao, M. R. Scoot, and W. Huang, "Tood: Task-aligned one-stage object detection," in *2021 IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Montreal, QC, Canada, 2021, pp. 3490–3499.
[24] Z. Liu, T. Zheng, G. Xu, Z. Yang, H. Liu and D. Cai, "Training-time-friendly network for real-time object detection," in *Proc. AAAI Conf. Artif. Intell.*, New York, USA, 2020, pp. 11685–11692.
[25] C. Li *et al.*, "YOLOv6: A single-stage object detection framework for industrial applications," arXiv preprint arXiv:2209.02976, 2022.
[26] C. Wang, A. Bochkovskiy, and H. Liao, "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Vancouver, Canada, 2023, pp. 7464–7475.