Article

Multi-Scale Target Detection in Autonomous Driving Scenarios Based on YOLOv5-AFAM

Hang Ma, Wei Zhao, Bosi Liu and Wenbai Chen *

School of Automation, Beijing Information Science & Technology University, Beijing 100192, China; mh@bistu.edu.cn (H.M.); 2022020401@bistu.edu.cn (W.Z.); 2022010622@bistu.edu.cn (B.L.)
* Correspondence: chenwb@bistu.edu.cn; Tel.: +86-1336-628-5716

Abstract: Multi-scale object detection is critically important in complex driving environments within the field of autonomous driving. To enhance the detection accuracy of both small-scale and large-scale targets in complex autonomous driving environments, this paper proposes an improved YOLOv5-AFAM algorithm. Firstly, the Adaptive Fusion Attention Module (AFAM) and Down-sampling Module (DownC) are introduced to increase the detection precision of small targets. Secondly, the Efficient Multi-scale Attention Module (EMA) is incorporated, enabling the model to simultaneously recognize small-scale and large-scale targets. Finally, a Minimum Point Distance IoU-based Loss Function (MPDIou-LOSS) is introduced to improve the accuracy and efficiency of object detection. Experimental validation on the KITTI dataset shows that, compared to the baseline model, the improved algorithm increased precision by 2.4%, recall by 2.6%, mAP50 by 1.5%, and mAP50-90 by an impressive 4.8%.

Keywords: multi-scale object detection; autonomous driving; YOLOv5; Adaptive Fusion Attention Module; Efficient Multi-scale Attention Module; MPDIou-LOSS Loss Function; KITTI Datasets

check for updates

Citation: Ma, H.; Zhao, W.; Liu, B.; Chen, W. Multi-Scale Target Detection in Autonomous Driving Scenarios Based on YOLOv5-AFAM. *Appl. Sci.* **2024**, *14*, 4633. https://doi.org/10.3390/app14114633

Academic Editors: Silvia Liberata Ullo and Li Zhang

Received: 1 April 2024
Revised: 23 May 2024
Accepted: 23 May 2024
Published: 28 May 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Over the past few decades, autonomous driving technology has emerged as a focal point in the research of intelligent transportation systems and robotics. Within this domain, target detection plays a crucial role, serving as one of the key elements to ensure vehicular safety. In the realm of autonomous driving research, the recognition of multi-scale targets is not only highly challenging but also a critical task for ensuring the precision and reliability of the system.

The development of target detection technology can be generally divided into two key phases: traditional methods and deep learning-based methods. In classical target detection, the Viola–Jones (VJ) detector, introduced by P. Viola and M. Jones, marked a significant milestone as a major breakthrough in the field [1,2]. Furthermore, the Deformable Part Models (DPMs) proposed by P. Felzenszwalb, and later improved by R. Girshick, once dominated the detection challenges in VOC 2007, 2008, and 2009 [3–5]. Due to the swift advancement in deep learning, especially with the widespread implementation of Convolutional Neural Networks (CNNs), the target detection field has experienced a significant transformation. In this revolution, R. Girshick and others introduced the Region-based Convolutional Neural Network (R-CNN) in 2016, incorporating CNN features and bringing a fresh perspective to the domain of target detection [6]. Subsequent advancements, such as Fast R-CNN [7] and Faster R-CNN [8], improved detection precision and significantly increased processing speed while reducing computational costs by integrating the Region Proposal Network (RPN).

In the field of target detection technology, the introduction of one-stage detectors such as YOLO [9] and SSD [10] represents significant technological breakthroughs. The advent of RetinaNet further propelled the development in this domain, as it effectively addressed the issue of class imbalance through the adoption of Focal Loss technology, significantly enhancing the model's efficiency and accuracy in processing various types of samples [11]. Moreover, RetinaNet conducts object detection at multiple feature levels, further improving detection performance.

The evolution of the YOLO series has been significant in target detection technology. Beginning with YOLOv2 [12] and YOLOv3 [13], the introduction of anchor boxes and multi-scale detection greatly improved the model's accuracy for small objects and complex backgrounds. YOLOv4 [14] and YOLOv5 [15] introduced further innovations, utilizing efficient feature extractors, attention modules, and data augmentation strategies to improve the model's precision and robustness. Subsequently, variants like YOLO-R [16], YOLOX [17], YOLOv6 [18], and YOLOv7 [19] continued to refine the network architecture and algorithms. Particularly in the domain of autonomous driving, these models ensured safe and efficient driving by quickly and accurately identifying and locating vehicles, pedestrians, and traffic signs. The latest, YOLOv8 [20], has further increased detection speed and accuracy, demonstrating the maturity and optimization potential of deep learning in practical applications.

Recent progress in object detection has resulted in the creation of more efficient and precise algorithms, particularly within the realms of autonomous driving and traffic monitoring. Ning and Wang (2022) developed an improved YOLOv5 network tailored for automatic driving scene target detection, achieving notable accuracy in complex driving scenarios [21]. Li et al. (2022) investigated the application of standard vision transformer backbones for object detection, contributing to advancements in transformer-based models [22]. Avşar and Avşar (2022) utilized deep learning techniques for detecting and tracking moving vehicles at roundabouts, demonstrating the efficacy of trajectory union methods [23]. Jeon and Jeon (2022) addressed computational challenges by quantizing the YOLOv5x6 model using TensorRT, achieving faster processing speeds while maintaining comparable detection accuracy to non-quantized models [24]. Hamzenejadi and Mohseni (2023) optimized YOLOv5 for real-time vehicle detection in UAV imagery by incorporating architectural enhancements that markedly improved performance [25]. Zheng et al. (2023) introduced YOLOv5s FMG, a refined algorithm aimed at detecting small targets in low visibility conditions, which enhances accuracy in challenging environments [26]. Despite these advancements, there remains a gap in multi-scale object detection research, highlighting the need for algorithms capable of effectively detecting and tracking objects of various sizes across diverse environments.

To address the issue of multi-scale target detection, this paper proposes an improved YOLOv5-AFAM algorithm, with the main steps as follows: (the link to the code for this article can be found in reference [27]).

(1) The AFAM and the DownC downsampling module are introduced to extract semantic information, thereby enhancing the detection accuracy of small objects.

(2) The EMA module is incorporated to enhance the model's multi-scale detection capabilities.

(3) The MPDIou loss function is utilized to enhance the precision and effectiveness of object detection.

2. YOLO Algorithm and Its Improvements

2.1. YOLOv5 and Enhanced Model Architecture

The YOLOv5 architecture includes five distinct network scales: YOLOv5s, YOLOv5m, YOLOv5l, YOLOv5x, and YOLOv5n. These variants are all built on the same framework but differ in the depth and width of CSPNet, leading to variations in model size and parameter count. This architecture is divided into three primary components: the backbone, the neck, and the head. The backbone is tasked with extracting image features through a series of convolutional and pooling layers. The neck network employs a feature pyramid structure [28] for top-down feature extraction, enhancing the detection capabilities for targets of various scales. The head network consists of three detection layers that handle specific tasks such as classification, detection, or segmentation, and utilizes a grid-based anchoring strategy for object detection, particularly improving the detection of small objects. Through the collaborative functioning of these three parts, YOLOv5 achieves efficient and accurate detection of multi-scale targets while maintaining speed and flexibility, making it suitable for a wide range of application scenarios. The network structure of YOLOv5 is illustrated in Figure 1.

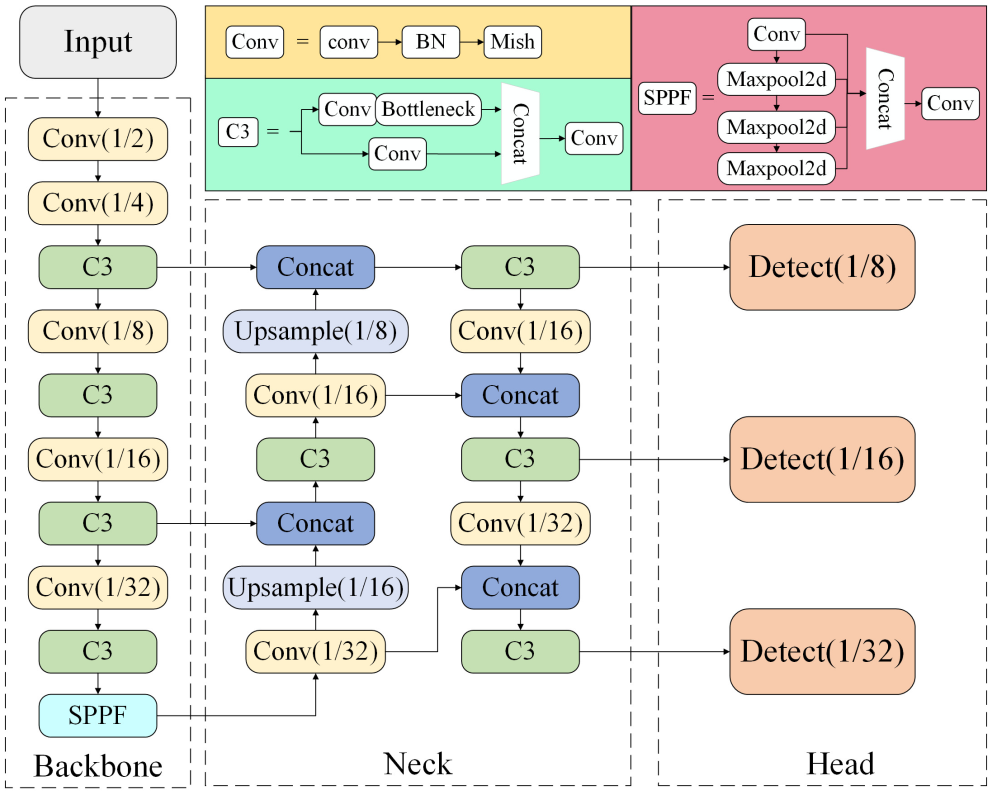


Figure 1. YOLOv5 Model Architecture. The YOLOv5 framework comprises three core components: the backbone, the neck, and the head. The backbone is composed of multi-scale convolutional layers, batch normalization, and Mish activation functions for feature extraction. The C3 modules enhance feature processing through convolutional and bottleneck layers. The neck employs a feature pyramid structure and upsampling, using Concat operations to integrate features across scales, thereby improving multi-scale target detection efficiency. The head network includes three detection layers designed for classification, detection, or segmentation.

This paper proposes a novel YOLOv5 model architecture. Firstly, the AFAM is introduced into the C3 modules of the backbone network, creating new C3AFAM modules. Subsequently, the convolution modules in the backbone network are replaced with DownC downsampling modules. In the neck network structure, an additional detection layer is added specifically for small object detection, and EMA modules are incorporated from the second to fifth C3 modules. Finally, features downsampled by factors of 4×, 8×, and 16× are combined and input into the detection head for prediction. The proposed YOLOv5 model architecture is illustrated in Figure 2.

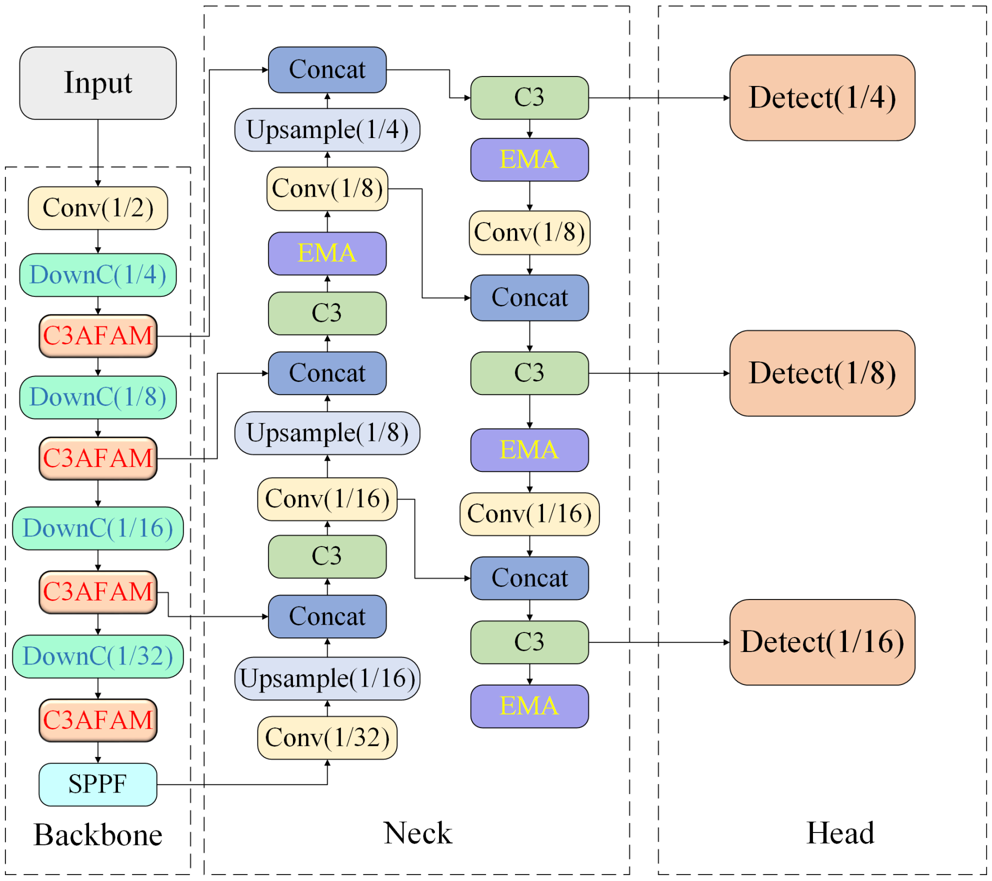


Figure 2. Enhanced YOLOv5 model architecture diagram. This diagram highlights the modifications made to the original YOLOv5 architecture shown in Figure 1. In the backbone, the standard downsampling convolution modules are replaced with DownC modules (DownC(1/4), DownC(1/8), DownC(1/16), and DownC(1/32)), and C3 modules are replaced with C3AFAM modules to enhance feature extraction with attention mechanisms. In the neck, EMA modules are added after each C3 module to further improve multi-scale feature integration. Additionally, an extra detection layer (Detect(1/4)) is included in the head to enhance the detection of small objects.

2.2. Improvements to the Backbone Section

To precisely implement multi-scale target detection, this study introduces the AFAM [29]. AFAM comprises two sub-modules: the Adaptive Channel Module (ACM) and the Fusion Spatial Module (FSM). When integrated into the backbone network of YOLOv5, the AFAM module is embedded within the C3 module. As illustrated in Figure 3, the process begins with the feature mapping $F \epsilon \mathbb{R}^{C \times H \times W}$. As inputs to the network, these feature maps are subsequently passed through the ACM and FSM, sequentially integrating attention information in both channel and spatial dimensions. The mathematical expression is shown in Equation (1); F_A represents the features F obtained after processing through the ACM, and F' represents the features F_A obtained after processing through the FSM. This demonstrates that this structural design not only enhances the model's ability to handle multi-scale features but also improves the accuracy of detecting small objects, thereby effectively boosting the overall performance of object detection.

$$F_A = f_A(F) \otimes F, \quad F' = f_F(F_A) \otimes F_A \tag{1}$$

In this context, $f_A(\cdot)$ represents the ACM and $f_F(\cdot)$ represents the FSM. $\otimes$ denotes an element-wise multiplication operation.

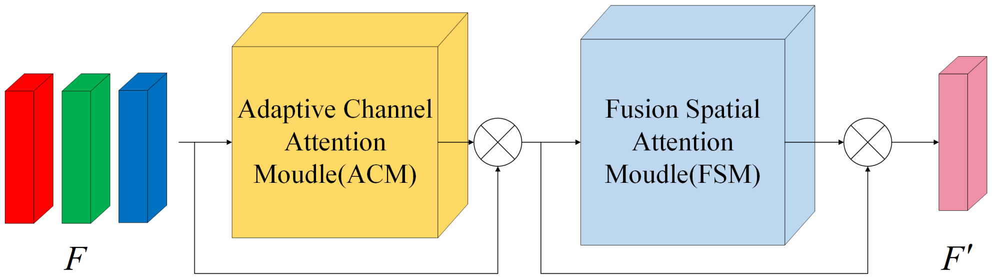

Adaptive Channel Attention Moudle(ACM)
Fusion Spatial Attention Moudle(FSM)
F
F′

Figure 3. Adaptive Fusion Attention Module. The module is composed of two primary components: the ACM and the FSM. The ACM adjusts the feature channels adaptively, enhancing the most informative channels. The FSM then applies spatial attention to the fused feature maps, focusing on the most relevant spatial information.

The ACM, illustrated in Figure 4, initiates with parallel convolutions using kernels of three distinct sizes (1×1, 3×3, and 5×5) to capture multi-scale feature information. These convolutional outputs are subsequently integrated and processed via a weighted summation method to generate fused features (denoted as F_f). The mathematical details of this process are provided in Equation (2). Following this, the fused features undergo average pooling (Avgpool) and max pooling (Maxpool) operations to produce descriptors that characterize the targets. Finally, these descriptors are further analyzed by a multi-layer perceptron (MLP) to yield the ultimate adaptive receptive field and channel attention features. This process is depicted in Equation (3).

$$F_f = \sum_{i=1}^{3} (Conv_i(F) \times \omega_i) \tag{2}$$

$$F_A = \sigma(MLP(Avgpool(F_f)) + MLP(MaxPool(F_f))) \tag{3}$$

In Equation (2), $Conv(\cdot)$ signifies the convolution operation using three different kernel sizes, and ω_i represents the ith weight factor for each convolution operation. In Equation (3), $\sigma(.)$ denotes the *sigmoid* function. $AvgPool$ and $MaxPool$ refer to the average pooling and max pooling operations, respectively.

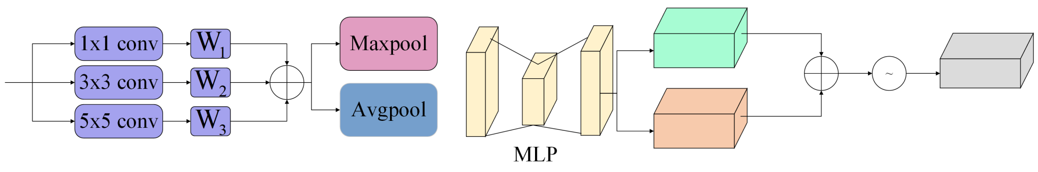


Figure 4. ACM network architecture diagram. It begins with three convolution layers (1×1, 3×3, and 5×5), each weighted by W_1,W_2, and W_3, respectively. The outputs are concatenated and passed through max pooling and average pooling layers. The pooled features are then fed into an MLP to generate channel-wise attention weights.

The FSM, as illustrated in Figure 5, inputs the feature maps generated by ACM (denoted as F_A) into three distinct branches in parallel, aiming to extract multi-dimensional feature information, as shown in Equation (4).

$$F_A^{max} = f_{max}(MaxPool(F_A)), \quad F_A^{avg} = AvgPool(F_A), \quad F_A^{conv} = f_{3\times3}(F_A) \tag{4}$$

In Equation (4), $f_{max}(\cdot)$ represents the convolution operation with a kernel size of $h \times \omega$.

Subsequently, the output features from these three branches are merged and further refined through a convolutional module (denoted as $f_C(\cdot)$). Afterwards, the merged feature maps are normalized via a Sigmoid activation function to obtain the FSM features F'. This process is represented in Equation (5).

$$F' = \sigma(f_C(Concat(F_A^{max}; F_A^{avg}; F_A^{conv}))) \tag{5}$$

In Equation (5), F_A^{max}, F_A^{avg} and F_A^{conv} correspond to the features after the $MaxPool(\cdot)$, $AvgPool(\cdot)$, and 3×3 convolutional layer, respectively.

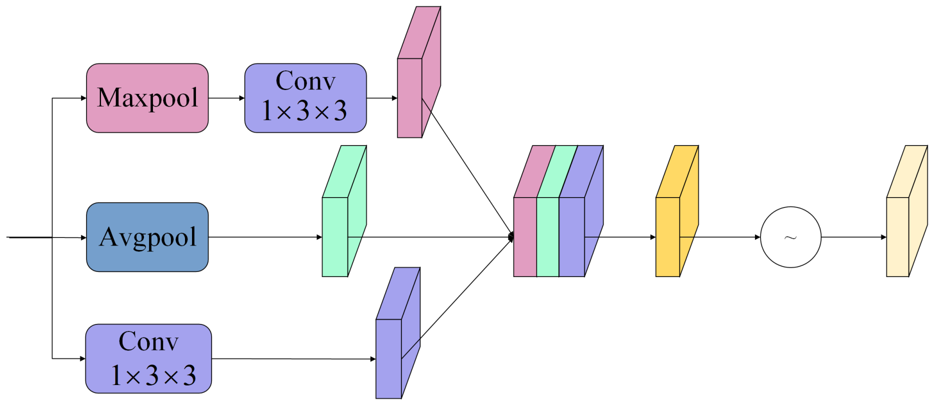


Figure 5. Schematic diagram of the FSM. The input feature map is processed through max pooling and average pooling layers, followed by a $1 \times 3 \times 3$ convolutional layer. The outputs from these operations are concatenated and passed through additional convolutional layers to generate spatial attention maps.

To tackle the challenges of detecting small objects and the difficulty in feature extraction, this paper introduces a DownC module. This module enlarges the receptive field and reduces the model's parameter count, which helps prevent overfitting during training. The MaxPool layer within this module decreases computational load while preserving texture features and retaining rich information.

As demonstrated in Figure 6, the DownC module comprises two branches for downsampling. One branch is the "All-Conv" branch, and the other is the "MaxPool-Conv" branch. To combine features from both branches while maintaining the same number of channels, each branch's channels are halved. The "All-Conv" branch consists of two convolutional modules, with the first module having a stride of 1 to halve the channels. The "MaxPool-Conv" branch begins with a max pooling operation, followed by a 1×1 convolutional module. In the "MaxPool-Conv" branch, the convolutional module also has a stride of 1 to reduce the number of channels.

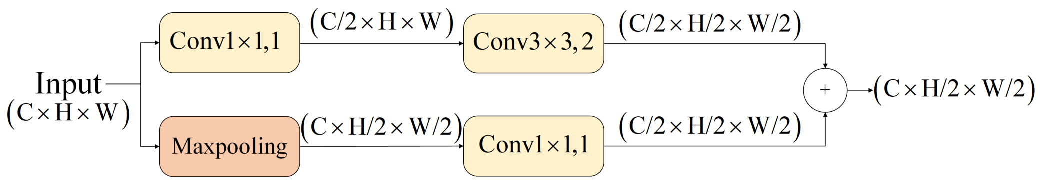


Figure 6. Schematic diagram of the DownC. The input feature tensor, with dimensions $C \times H \times W$, is processed through a 1×1 convolutional layer followed by a max pooling layer. The resulting feature maps are then processed through a 3×3 convolutional layer with a stride of 2 and another 1×1 convolutional layer. The outputs of these layers are concatenated and summed, resulting in a feature map with reduced dimensions $C/2 \times H/2 \times W/2$.

2.3. Improvements to the Neck Section

As depicted in Figure 7, the EMA [30] method is implemented on any input feature map $X \in \mathbb{R}^{C \times H \times W}$. EMA segments X into G groups of sub-features along the channel axis, facilitating the learning of unique semantic characteristics. This grouping method can be expressed as $X = [X_0, X_1, \cdots, X_{G-1}]$, where each sub-feature X_i belongs to $\mathbb{R}^{C/G \times H \times W}$.

Typically, G is much smaller than C. It is assumed that the learned attention weight descriptors will improve the feature representation capability of regions of interest within each sub-feature group.

EMA derives the attention weight descriptors from the clustered feature maps using three parallel routes, consisting of two 1 × 1 branches and one 3 × 3 branch. The 1 × 1 branches utilize global average pooling operations, while the 3 × 3 branch employs a 3 × 3 convolutional kernel to expand the expression of features.

Specifically, EMA incorporates two tensors that correspond to the outputs of the 1 × 1 and 3 × 3 branches, respectively. Next, it performs 2D global average pooling on the output of the 1 × 1 branch to capture comprehensive spatial information. Before activating the channel features jointly, the output of the 3 × 3 branch is directly reshaped into the corresponding dimensional format, denoted as $\mathbb{R}_1^{1\times C//G} \times \mathbb{R}_3^{C//G\times HW}$. The operation of 2D global pooling is illustrated in Equation (6).

$$z_c = \frac{1}{H \times W} \sum_j^H \sum_i^W x_c(i,j) \tag{6}$$

In Equation (6), z_c represents the scalar value obtained after global average pooling of the feature map for a specific channel c. The term $\frac{1}{H\times W}$ acts as the normalization factor for the average pooling, where H and W are the height and width of the feature map, respectively. $\sum_j^H \sum_i^W x_c(i,j)$ denotes the summation of all pixel values $x(i,j)$ in the feature map for channel c. This strategy not only enhances the pixel-level attention of CNNs to high-level feature maps but also, through the use of parallel convolution kernels, improves the handling of both short-range and long-range dependencies, making the model both efficient and adaptable to modern architectures.

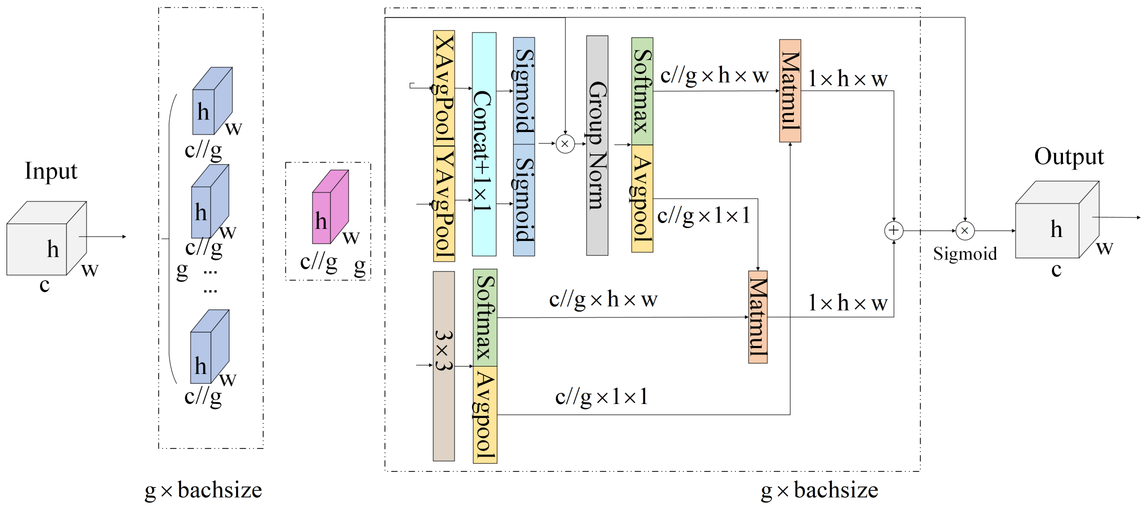


Figure 7. Schematic diagram of the EMA. The input feature map is divided into g groups, each processed separately. For each group, the features are passed through an average pooling layer and concatenated. This is followed by sigmoid and softmax activation functions applied to the pooled features. The outputs are then normalized using group normalization. The resultant features are multiplied and summed, producing the final output feature map, which retains the input dimensions $h \times w \times c$ after passing through a sigmoid activation function.

2.4. MPDIou Loss Function

The Minimum Point Distance Intersection over Union (MPDIou) loss [31], proposed by Siliang Maa, Yong Xua, et al., addresses the issues of CIou failure when the actual and predicted bounding box centers coincide and aspect ratios match, as well as the problem of extensive overlap among multiple predicted boxes. MPDIou encompasses all relevant factors considered in existing loss functions, such as overlap or non-overlap area, center point deviation, and deviations in width and height, while simplifying the computational process.

This approach directly minimizes the distance between the top-left and bottom-right corners of the predicted bounding box and the ground truth box. During the training phase, the formula forces the bounding box $B^{prd} = [x^{prd}, y^{prd}, w^{prd}, h^{prd}]$ to approach its actual bounding box $B_{gt} = [x_{gt}, y_{gt}, w_{gt}, h_{gt}]$:

$$L = \underset{\Theta}{MIN} \sum_{B_{gt} \epsilon B_{gt}} L(B_{gt}, B_{prd}|\Theta) \tag{7}$$

In Equation (7), B_{gt} represents the set of actual annotated boxes, while Θ denotes the parameters used for regression in deep learning. The loss function of MPDIou is defined as shown in Equation (8).

$$\begin{aligned}
&d_1^2 = (x_1^B - x_1^A)^2 + (y_1^B - y_1^A)^2, \\
&d_2^2 = (x_2^B - x_2^A)^2 + (y_2^B - y_2^A)^2, \\
&MPDIou = \frac{A \cap B}{A \cup B} - \frac{d_1^2}{w^2 + h^2} - \frac{d_2^2}{w^2 + h^2}, \\
&L_{MPDIou} = 1 - MPDIou
\end{aligned} \tag{8}$$

Thus, all factors of the existing bounding box loss function are determined by four points, with the transformation formula presented in Equation (9).

$$\begin{gathered}
|C| = (\max(x_2^{gt}, x_2^{prd}) - \min(x_1^{gt}, x_1^{prd})) \times (\max(y_2^{gt}, y_2^{prd}) - \min(y_1^{gt}, y_1^{prd})), \\
x_c^{gt} = \frac{x_1^{gt} + x_2^{gt}}{2}, y_c^{gt} = \frac{y_1^{gt} + y_2^{gt}}{2}, \\
y_c^{prd} = \frac{y_1^{prd} + y_2^{prd}}{2}, x_c^{prd} = \frac{x_1^{prd} + x_2^{prd}}{2}, \\
w_{gt} = x_2^{gt} - x_1^{gt}, h_{gt} = y_2^{gt} - y_1^{gt}, \\
w_{prd} = x_2^{prd} - x_1^{prd}, h_{prd} = y_2^{prd} - y_1^{prd}
\end{gathered} \tag{9}$$

In Equation (9), $|C|$ represents the area of the minimum bounding rectangle that encloses B_{gt} and B_{prd}. The coordinates (x_c^{gt}, y_c^{gt}) and (x_c^{prd}, y_c^{prd}) represent the central coordinates of the actual annotated bounding box and the predicted bounding box, respectively. The width and height of the actual bounding box are denoted as w_{gt} and h_{gt}, while those of the predicted bounding box are denoted as w_{prd} and h_{prd}.

3. Experimental Design and Results

3.1. Dataset

The KITTI dataset, jointly created by the Karlsruhe Institute of Technology in Germany and the Toyota Technological Institute at Chicago, gathers data from various sensors, including cameras and LiDAR, capturing a variety of real-world driving scenarios.

In the KITTI dataset used for this experiment, the 2D image target categories were originally divided into "Truck", "Van", "Tram", "Car", "Person _sitting", "Cyclist", and "Pedestrian". In this study, these categories were simplified and reclassified. The three vehicle types "Truck", "Van", and "Tram" were consolidated into the "Car" category, while "Person _sitting" and "Cyclist" were merged into the "Pedestrian" category. This adjustment was made to facilitate a more effective experimental analysis.

3.2. Preparation Work

The configuration required for this experiment is as shown in Table 1. Two RTX 4090 GPUs were utilized, and the training was conducted on an Ubuntu 20.04 system using the PyTorch 1.10.0+cu113 framework.

Table 1. Experimental platform configuration. The GPU configuration includes two RTX 4090 cards, each with 24 GB of memory. The CPU used is a 24 vCPU Intel(R) Xeon(R) Platinum 8352V running at 2.10 GHz. The operating system is Ubuntu 20.04, with CUDA version 11.3 and cuDNN version 8.2. PyTorch version 1.10.0+cu113 is used as the deep learning framework.

Name	Configuration Details
GPU	RTX 4090(24GB) $\times$ 2
CPU	24 vCPU Intel(R) Xeon(R) Platinum 8352V CPU @ 2.10 GHz
Operating System	Ubuntu 20.04
CUDA	11.3
cuDNN	8.2
PyTorch	1.10.0+cu113

The experimental parameter settings are shown in Table 2, with the number of training epochs set to 500, the number of categories set to 2, the batch size set at 8, and the image size set to 640 × 640.

Table 2. Experimental parameter settings. The model was trained for 500 epochs. There are 2 categories in the dataset. The batch size used during training is 8, and the input image size is 640 $\times$ 640 pixels.

Parameter	Configuration Details
Epoch	500
Number of Categories	2
Batch size	8
Image size	640 × 640

3.3. Evaluation Metrics

The relevant evaluation metrics for this study include precision, recall, and the mean average precision (mAP) at different thresholds, specifically mAP@0.5 and mAP@0.5:0.95.

The formula for calculating precision is presented in Equation (10).

$$Precision = \frac{TP}{TP + FP} \tag{10}$$

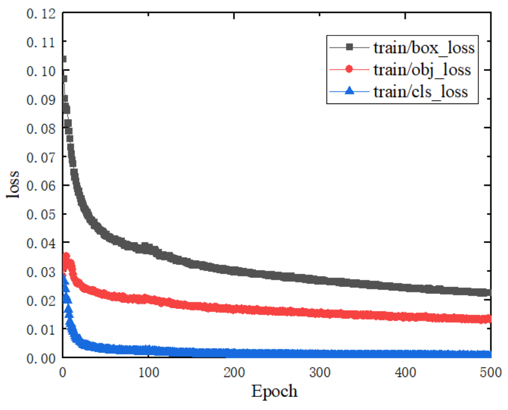
The calculation of recall is shown in Equation (11).

$$Recall = \frac{TP}{TP + FN} \tag{11}$$

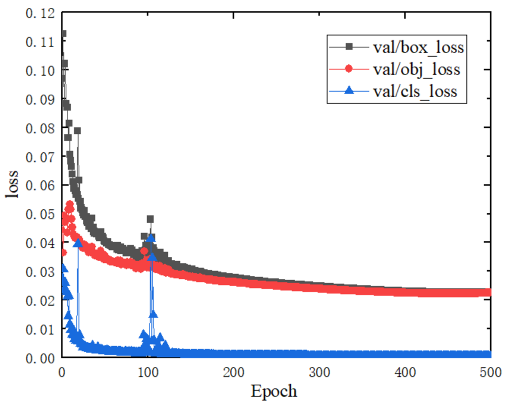
3.4. Experimental Results and Analysis

In this study, a baseline model was constructed, incorporating the DownC downsampling module, AFAM, and specialized layers for small object detection.

The dataset in this study was divided into a training set and a validation set, distributed at a ratio of 8:2. As shown in Figure 8, in Figure 8a displays the loss function graphs for the training set, while Figure 8b shows the loss graph for the validation set. box _loss represents the loss function value for the model in locating targets, obj_loss indicates the loss function value for predicting the presence of targets, and clc_loss denotes the classification loss, measuring the model's performance in classifying targets. In Figure 8a, all loss functions decrease and stabilize over the course of training rounds, indicating gradual improvement in model performance. Although there are some fluctuations in Figure 8b, the overall trend of the loss function is downward, suggesting good generalization of the model.



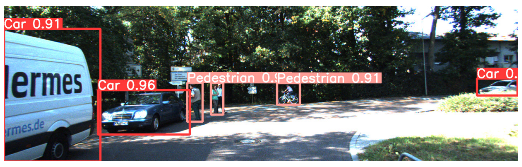
(**a**)



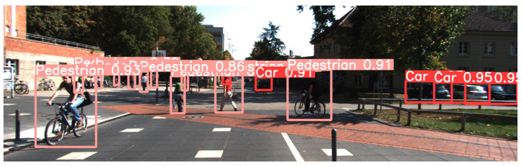
(**b**)

Figure 8. Baseline loss function graphs. (**a**) Training set loss. This graph shows the training loss for box, object, and class predictions over 500 epochs. (**b**) Validation set loss. This graph shows the validation loss for box, object, and class predictions over 500 epochs.

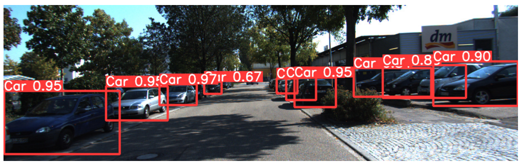
Figure 9 illustrates the detection results of the baseline model. In Figure 9a, all cars and pedestrians in the images are successfully detected with confidence scores exceeding 90%. In Figure 9b, while some distant and smaller targets, such as vehicles and pedestrians, are present, most of them are successfully identified. Only a small portion of these targets are not detected, yet the overall confidence scores remain high. In Figure 9c, the detection results for an image consisting entirely of cars are shown. It is observed that the two cars furthest from the camera were not successfully detected, while the remaining cars were detected with high confidence. In Figure 9d, pedestrians are the dominant class. All pedestrians were successfully detected with confidence scores above 79%, indicating that there is still room for improvement in detection performance.



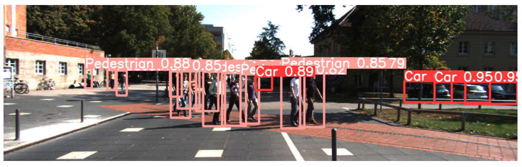
(**a**)



(**b**)



(**c**)



(**d**)

Figure 9. Detection results of the baseline model. (**a**) Close-up view. Detection results of large targets in close-up scenarios. (**b**) Distant view. Detection results of small targets in distant scenarios. (**c**) Car-dominant Image. This image predominantly contains cars, with only a few instances not being detected. (**d**) Pedestrian-dominant Image. All instances of pedestrians in this image are successfully detected.

The baseline model data presented in Table 3 show performance variations across different categories. The overall precision is 91.3%, with the Car category achieving a precision of 92.1% and the Pedestrian category at 90.5%. In terms of recall, the overall rate is 83.6%, with a higher 90.1% for the Car category and 77.2% for the Pedestrian category. For the mAP50 metric, the model scores 92.1% overall, with the Car category at 95.9% and the

Pedestrian category at 88.3%. In the mAP50-90 metric, the overall performance is 66.47%, with the Car category at 74.2% and the Pedestrian category at 55.2%.

Table 3. Baseline model evaluation metrics.This table presents the precision, recall, and mean average precision (mAP) metrics for different classes using the baseline model. The results are shown for both mAP@50 and mAP@50-90 thresholds. The "All" category represents the overall performance across all classes, while specific metrics are provided for the "Car" and "Pedestrian" classes.

Class	Precision	Recall	mAP50	mAP50-90
All	0.913	0.836	0.921	0.647
Car	0.921	0.901	0.959	0.742
Pedestrian	0.905	0.772	0.883	0.552

This study introduced the EMA module on top of the baseline model. The improved loss function graph is shown in Figure 10.

After the introduction of the EMA module, the loss function graph becomes noticeably smoother and more stable, with minimal fluctuations, especially when compared to the baseline model. Particularly in terms of validation loss, the incorporation of the EMA module results in a smoother trend with almost no significant fluctuations. The improved model demonstrates enhanced generalization capabilities and overall better performance.

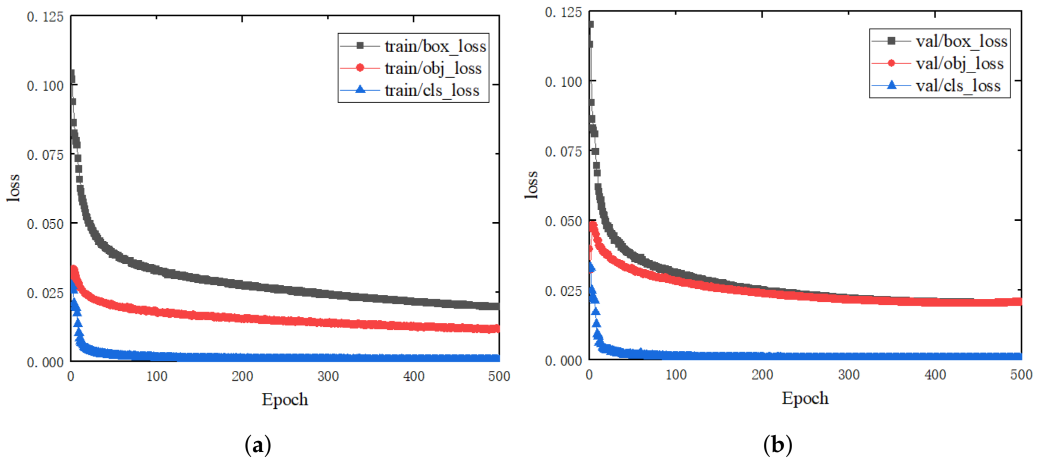


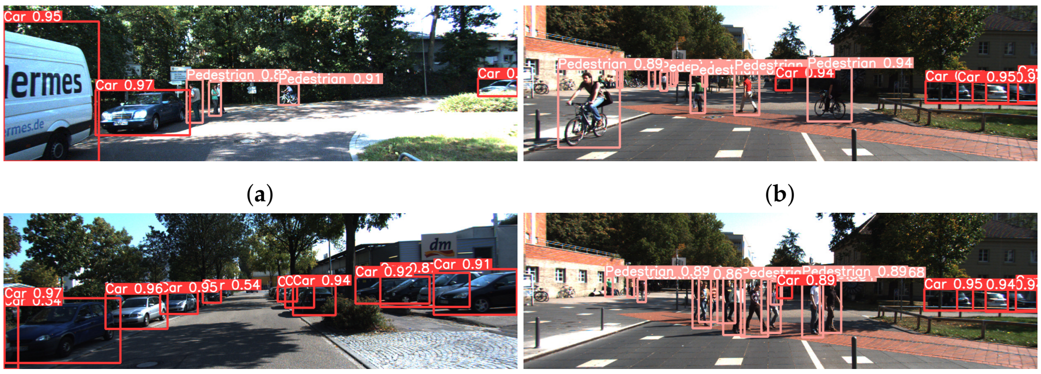
Figure 10. Incorporation of EMA loss function graphs. (**a**) Training set loss. This graph shows the training loss for box, object, and class predictions over 500 epochs with the incorporation of the EMA module. (**b**) Validation set loss. This graph shows the validation loss for box, object, and class predictions over 500 epochs with the incorporation of the EMA module.

Table 4 demonstrates that the introduction of the EMA module led to significant improvements across all evaluation metrics. Specifically, the overall precision reached 93.9%, with the precision for the Car category at 94.3% and for the Pedestrian category at 93.4%. In terms of recall, the overall rate increased to 85.9%, with the Car category achieving a high recall rate of 91.5%, and the Pedestrian category reaching 80.3%. For the mAP50 metric, the overall performance was 93.8%, with the Car category reaching 96.7% and the Pedestrian category at 90.8%. Regarding the mAP50-90 metric, the overall performance was 69.1%, with the Car category at 77.9% and the Pedestrian category at 60.4%.

The detection results after integrating the EMA module are shown in Figure 11. In Figure 11a, the confidence scores are significantly improved, with the highest reaching 97%, demonstrating a clear enhancement in performance. In Figure 11b, the improvement is more pronounced for the Car category compared to the Pedestrian category. Figure 11c shows better detection performance for close-range targets, with increased confidence scores. In Figure 11d, overall detection performance is improved, with all targets showing higher confidence scores. These results indicate that the integration of the EMA module leads to a significant enhancement in the overall detection performance of the model.

Table 4. Evaluation metrics for the introduced EMA module. This table presents the precision, recall, and mAP metrics for different classes after incorporating the EMA module. The results are shown for both mAP@50 and mAP@50-90 thresholds. The "All" category represents the overall performance across all classes, while specific metrics are provided for the "Car" and "Pedestrian" classes.

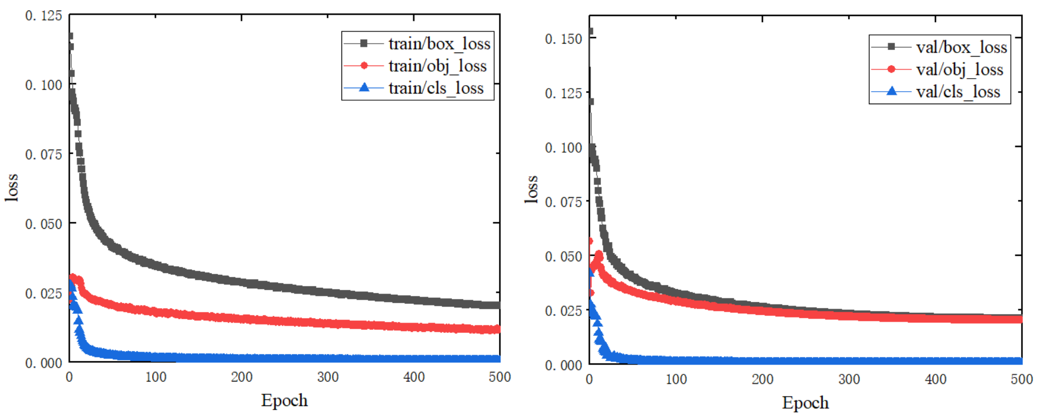
Class	Precision	Recall	mAP50	mAP50-90
All	0.939	0.859	0.938	0.691
Car	0.943	0.915	0.967	0.779
Pedestrian	0.934	0.803	0.908	0.604



(**a**) (**b**) (**c**) (**d**)

Figure 11. Detection results with EMA integration. (**a**) Close-up view. Detection results of large targets in close-up scenarios. (**b**) Distant view. Detection results of small targets in distant scenarios. (**c**) Car-dominant image. This image predominantly contains cars, with only a few instances not being detected. (**d**) Pedestrian-dominant image. All instances of pedestrians in this image are successfully detected.

To address more complex driving environments, we utilized the MPDIou loss function in the enhanced model. The improved loss function graph is depicted in Figure 12.

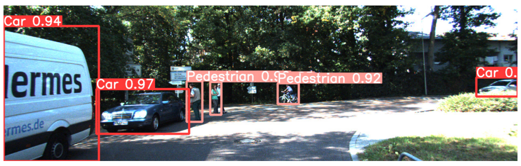


(**a**) (**b**)

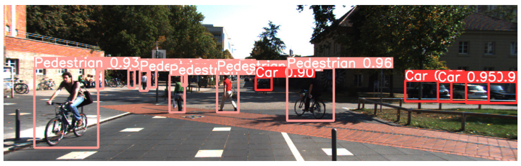
Figure 12. Graphs with modified MPDIoU loss function. (**a**) Training Set Loss. This graph shows the training loss for box, object, and class predictions over 500 epochs with the modified MPDIoU loss function. (**b**) Validation Set Loss. This graph shows the validation loss for box, object, and class predictions over 500 epochs with the modified MPDIoU loss function.

After adopting the MPDIou loss function, an initial increase in some loss function values was observed, along with significant drops and fluctuations in the early stages of training. However, as the number of training iterations increased, these loss functions eventually stabilized, showing a smooth curve trend.

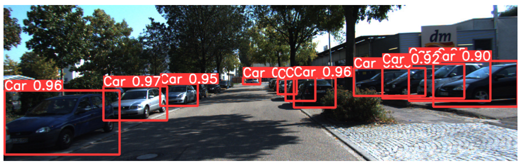
Figure 13 illustrates the results after integrating the EMA module and replacing the loss function with MPDIoU. In Figure 13a, the confidence scores for all detected categories exceed 90%, demonstrating better performance compared to just adding the EMA module. Similarly, in Figure 13b, the confidence scores for all detected categories are above 90%, showing significant improvement. Figure 13c highlights the detection of categories that were not identified by the previous two models, with confidence scores exceeding 90% and some reaching up to 97%. In Figure 13d, a slight improvement is observed, with most categories having confidence scores above 88%, except for one category at 45%. These results indicate that replacing the loss function with MPDIoU further enhances detection performance, proving the effectiveness of this improvement.



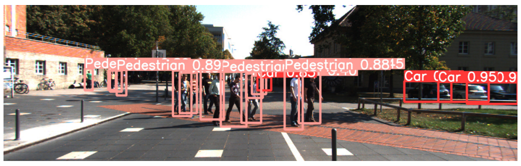
(**a**)



(**b**)



(**c**)



(**d**)

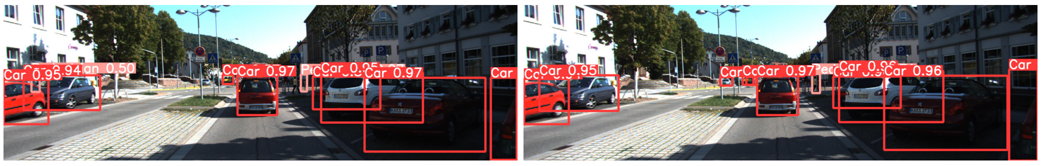
Figure 13. Detection results with EMA integration. (**a**) Close-up view. Detection results of large targets in close-up scenarios. (**b**) Distant view. Detection results of small targets in distant scenarios. (**c**) Car-dominant image. This image predominantly contains cars, with only a few instances not being detected. (**d**) Pedestrian-dominant image. All instances of pedestrians in this image are successfully detected.

Table 5 shows the changes in evaluation metrics after adopting the MPDIou loss function. The data reveal slight improvements in certain metrics. Specifically, the overall precision was 93.7%, with the precision for the Car category at 94.5% and for the Pedestrian category at 92.8%. In terms of recall, the overall rate was 86.2%, with the Car category at 91.3% and the Pedestrian category at 81.1%. For the mAP50 metric, the overall performance was 93.6%, with the Car category at 96.4% and the Pedestrian category at 90.8%. As for the mAP50-90 metric, the overall performance was 69.5%, with the Car category remaining at 77.9% and the Pedestrian category increasing to 61.2%. After adopting the MPDIou loss function, the model experienced stable and slight improvements across various evaluation metrics.

Table 5. Evaluation metrics utilizing MPDIoU. This table presents the precision, recall, and mAP metrics for different classes using the modified MPDIoU loss function. The results are shown for both mAP@50 and mAP@50-90 thresholds. The "All" category represents the overall performance across all classes, while specific metrics are provided for the "Car" and "Pedestrian" classes.

Class	Precision	Recall	mAP50	mAP50-90
All	0.937	0.862	0.936	0.695
Car	0.945	0.913	0.964	0.779
Pedestrian	0.928	0.811	0.908	0.612

Figure 14 presents a comparison of detection results. In Figure 14a, the left side shows an incorrect detection of a pedestrian where there is none, and distant vehicles are not successfully detected. Additionally, the pedestrian to the right of the red car in the middle is not detected, while a lamppost is mistakenly identified as a pedestrian, indicating mediocre detection performance. In contrast, Figure 14b corrects the errors observed in Figure 14a by not detecting any pedestrian on the left side. Moreover, it successfully detects the distant car that was missed in Figure 14a and correctly identifies the pedestrian to the right of the red car without misidentifying the lamppost as a pedestrian. The comparison between Figure 14a,b demonstrates that the improved model significantly outperforms the baseline model, with notably enhanced detection performance.



(**a**) (**b**)

Figure 14. Comparison of detection results. (**a**) Baseline model detection results, illustrating the detection performance for each category using the baseline model. (**b**) Improved model detection results, showcasing the detection performance of the proposed improved model.

3.5. Ablation Experiment

To verify the effectiveness of the improvements made, this study conducted a comparative analysis between the baseline model and each improved model. The comparison results are shown in Figure 15.

In Figure 15a, the precision of the baseline model is lower compared to the performance after introducing the EMA and MPDIou modules. Notably, with the initial introduction of the EMA module, there is a significant increase in precision, which then stabilizes at a higher level, consistently outperforming the baseline model. Upon integrating MPDIou, although the initial precision is slightly lower than that of the EMA module, it surpasses the previous two in the later stages of training, demonstrating superiority. In Figure 15b, the introduction of the EMA module significantly improves the recall rate, creating a noticeable gap with the baseline model. After adopting MPDIou, the model similarly surpasses the results with EMA in the later stages of training, showing a stable and gradual upward trend. For Figure 15c, the baseline model experiences fluctuations and a declining trend in the early stages of training, which then gradually and steadily increases. After introducing EMA, the mAP50 metric shows a stable upward trend and a relatively smooth curve, with a significant initial difference from the baseline model, although this difference diminishes later. With MPDIou, the performance exceeds that of the EMA result by the end of training. In the case of Figure 15d, the baseline model also undergoes fluctuations and declines before stabilizing and rising. After integrating EMA, the improvement in the mAP50-90 metric is more pronounced, showing the largest gap with the baseline model. When MPDIou is adopted, the model similarly outperforms the EMA result in the later stages.

As shown in Table 6, after introducing the EMA module to the baseline model, there was an increase of 2.6% in precision, a 2.3% improvement in recall rate, a 1.7% rise in mAP50, and a significant 4.4% increase in mAP50-90. The introduction of the EMA module led to substantial enhancements across all key performance metrics, thereby confirming the significant impact of the improvements made. Upon adopting the MPDIou loss function, the model's performance changes were minor adjustments. Specifically, there was a slight decrease in precision by 0.2%, while the recall rate increased by 0.3%. A minor decline of 0.2% was observed in the mAP50 metric. However, there was a 0.4% improvement in the mAP50-90 metric. These changes reflect the subtle balance and adjustment in the model's performance across different metrics after implementing the MPDIou loss function.

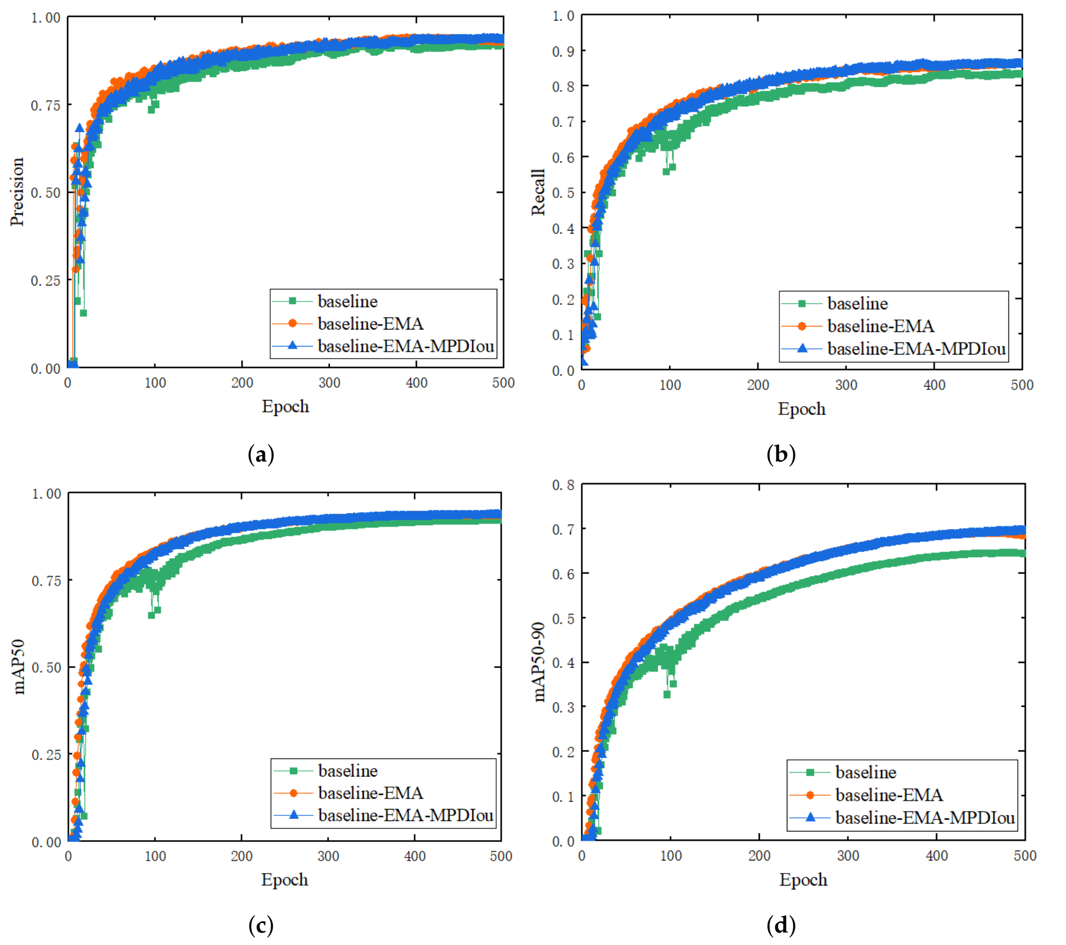


Figure 15. Improved comparison charts. (**a**) Precision graph. This graph shows the precision values over 500 epochs for the baseline model, the model with EMA, and the model with EMA and MPDIoU. (**b**) Recall graph. This graph depicts the recall values over 500 epochs for the three models. (**c**) mAP50 graph. This graph illustrates the mAP at 50% IoU over 500 epochs. (**d**) mAP50-90 graph. This graph shows the mAP across IoU thresholds from 50% to 90% over 500 epochs.

Table 6. Comparative experiment table. This table compares the evaluation metrics of three models: baseline, EMA, and MPDIoU. The metrics include precision, recall, and mAP at 50% and 50-90% IoU thresholds. Arrows indicate the change in performance relative to the baseline model. The EMA model shows improvements in precision, recall, and mAP metrics, while the MPDIoU model shows a slight decrease in precision but improvements in recall and mAP.

Model	**Precision**	**Recall**	**mAP50**	**mAP50-90**
Baseline	0.913	0.836	0.921	0.647
EMA	0.939↑	0.859↑	0.938↑	0.691↑
MPDIou	0.937↓	0.862↑	0.936↓	0.695↑

Table 7 presents a comparison between the experimental model and YOLOv8. The results indicate that while both models achieve the same accuracy on the validation set, the experimental model surpasses YOLOv8 in recall, with an improvement of 2.7%. Additionally, the experimental model outperforms YOLOv8 by 1.7 percentage points in mAP50. The only metric where the experimental model falls slightly behind YOLOv8 is in mAP50-90, with a decrease of one percentage point. Overall, the experimental model demonstrates superior performance compared to YOLOv8.

Table 7. Comparison with YOLOv8. This table presents the comparison of various metrics between the experimental model and YOLOv8.

Model	Precision	Recall	mAP50	mAP50-90
YOLOv8	0.937	0.835	0.919	0.705
Ours	0.937	0.862	0.936	0.695

4. Conclusions

In this experiment, we evaluated the improved YOLOv5 model to investigate its performance on the KITTI dataset. This study introduced the AFAM module, DownC downsampling module, EMA module, and MPDIou loss function, aiming to enhance the overall performance of the model. Experimental validation demonstrated that the improved model showed significant improvements across all key evaluation metrics. Specifically, compared to the baseline model, there was a 2.4% increase in precision, a 2.6% increase in recall rate, a 1.5% rise in mAP50, and a 4.8% increase in mAP50-90. These results strongly indicate that the improvement strategies proposed in this study effectively enhanced the model's performance, making it superior to the baseline model.

Author Contributions: Conceptualization, B.L.; methodology, H.M. and W.Z.; software, W.Z.; validation, W.C., H.M. and W.Z.; formal analysis, W.C.; investigation, B.L.; resources, H.M.; data curation, W.Z.; writing—original draft preparation, W.Z.; writing—review and editing, H.M. and W.C.; visualization, W.Z.; supervision, H.M. and W.C.; project administration, W.C. All authors have read and agreed to the published version of the manuscript.

Funding: The National Natural Science Foundation of China (Grant No. 62276028).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data for this thesis were obtained from the following link: https://github.com/luobo-1231/Multi-Scale-Target-Detection-in-Autonomous-Driving-Scenarios-Based-on-YOLOv5-AFAM/tree/master (accessed on 22 May 2024).

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Viola, P.; Jones, M. Rapid object detection using a boosted cascade of simple features. In Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2001), Kauai, HI, USA, 8–14 December 2001; Volume 1, p. I.
2. Viola, P.; Jones, M.J. Robust real-time face detection. *Int. J. Comput. Vis.* **2004**, *57*, 137–154. [CrossRef]
3. Felzenszwalb, P.; McAllester, D.; Ramanan, D. A discriminatively trained, multiscale, deformable part model. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2008), Anchorage, AK, USA, 23–28 June 2008; pp. 1–8.
4. Felzenszwalb, P.F.; Girshick, R.B.; McAllester, D. Cascade object detection with deformable part models. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2010), San Francisco, CA, USA, 13–18 June 2010; pp. 2241–2248.
5. Felzenszwalb, P.F.; Girshick, R.B.; McAllester, D.; Ramanan, D. Object detection with discriminatively trained part-based models. *IEEE Trans. Pattern Anal. Mach. Intell.* **2010**, *32*, 1627–1645. [CrossRef] [PubMed]
6. Girshick, R.; Donahue, J.; Darrell, T.; Malik, J. Region-based convolutional networks for accurate object detection and segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2016**, *38*, 142–158. [CrossRef] [PubMed]
7. Girshick, R. Fast r-cnn. In Proceedings of the IEEE International Conference on Computer Vision (ICCV 2015), Santiago, Chile, 7–13 December 2015; pp. 1440–1448.
8. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster r-cnn: Towards real-time object detection with region proposal networks. In Proceedings of the IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI 2017), Piscataway, NJ, USA, 1 June 2017; pp. 1137–1149.
9. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You only look once: Unified, real-time object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2016), Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788.
10. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.-Y.; Berg, A.C. SSD: Single shot multibox detector. In Proceedings of the European Conference on Computer Vision (ECCV 2016), Amsterdam, The Netherlands, 11–14 October 2016; Springer: Cham, Switzerland, 2016; pp. 21–37.

11. Lin, T.-Y.; Goyal, P.; Girshick, R.; He, K.; Dollár, P. Focal Loss for Dense Object Detection. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; pp. 2999-3007.
12. Redmon, J.; Farhadi, A. YOLO9000: Better, faster, stronger. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2017), Honolulu, HI, USA, 21–26 July 2017; pp. 7263–7271.
13. Redmon, J.; Farhadi, A. YOLOv3: An incremental improvement. *arXiv* **2018**, arXiv:1804.02767.
14. Bochkovskiy, A.; Wang, C.-Y.; Liao, H.-Y.M. YOLOv4: Optimal speed and accuracy of object detection. *arXiv* **2020**, arXiv:2004.10934.
15. Jocher, G. YOLOv5 by Ultralytics. Available online: https://github.com/ultralytics/yolov5 (accessed on 9 June 2020).
16. Wang, C.-Y.; Yeh, I.-H.; Liao, H.-Y.M. You only learn one representation: Unified network for multiple tasks. *arXiv* **2021**, arXiv:2105.04206.
17. Ge, Z.; Liu, S.; Wang, F.; Li, Z.; Sun, J. YOLOX: Exceeding YOLO series in 2021. *arXiv* **2021**, arXiv:2107.08430.
18. Li, C.; Li, L.; Jiang, H.; Weng, K.; Geng, Y.; Li, L.; Ke, Z.; Li, Q.; Cheng, M.; Nie, W.; et al. YOLOv6: A single-stage object detection framework for industrial applications. *arXiv* **2022**, arXiv:2209.02976.
19. Wang, C.-Y.; Bochkovskiy, A.; Liao, H.-Y.M. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. *arXiv* **2022**, arXiv:2207.02696.
20. Jocher, G.; Chaurasia, A.; Qiu, J. YOLO by Ultralytics. Available online: https://github.com/ultralytics/ultralytics (accessed on 10 January 2023).
21. Ning, J.; Wang, J. Automatic Driving Scene Target Detection Algorithm Based on Improved YOLOv5 Network. In Proceedings of the IEEE International Conference on Consumer Electronics-Asia (ICCNEA 2022), Xi'an, China, 23–25 September 2022; pp. 218–222.
22. Li, Y.; Mao, H.; Girshick, R.; He, K. Exploring plain vision transformer backbones for object detection. In Proceedings of the Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, 23–27 October 2022; Proceedings, Part IX; Springer: Cham, Switzerland, 2022; pp. 280–296.
23. Avşar, E.; Avşar, Y.Ö. Moving vehicle detection and tracking at roundabouts using deep learning with trajectory union. *Multimed. Tools Appl.* **2022**, *81*, 6653–6680. [CrossRef]
24. Jeon, H.-J.; Jeon, J. Quantized YOLOv5x6 for Traffic Object Detection. In Proceedings of the 2022 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia), Yeosu, Republic of Korea, 26–28 October 2022; pp. 1–3.
25. Hamzenejadi, M.H.; Mohseni, H. Fine-tuned YOLOv5 for real-time vehicle detection in UAV imagery: Architectural improvements and performance boost. *Expert Syst. Appl.* **2023**, *231*, 120845. [CrossRef]
26. Zheng, Y.; Zhan, Y.; Huang, X.; Ji, G. YOLOv5s FMG: An Improved Small Target Detection Algorithm Based on YOLOv5 in Low Visibility. *IEEE Access* **2023**, *11*, 75782–75793. [CrossRef]
27. Zhao, W. Multi-Scale Target Detection in Autonomous Driving Scenarios Based on YOLOv5-AFAM. Repository at GitHub. Available online: https://github.com/luobo-1231/Multi-Scale-Target-Detection-in-Autonomous-Driving-Scenarios-Based-on-YOLOv5-AFAM/tree/master (accessed on 22 May 2024).
28. Lin, T.; Dollár, P.; Girshick, R.B.; He, K.; Hariharan, B.; Belongie, S.J. Feature Pyramid Networks for Object Detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2017), Honolulu, HI, USA, 21–26 July 2017; pp. 936–944.
29. Wang, Y.; Zhao, L.; Ma, Y.; Shi, Y.; Tian, J. Multiscale YOLOv5-AFAM-Based Infrared Dim-Small-Target Detection. *Appl. Sci.* **2023**, *13*, 7779. [CrossRef]
30. Ouyang, D.; He, S.; Zhang, G.; Luo, M.; Guo, H.; Zhan, J.; Huang, Z. Efficient Multi-Scale Attention Module with Cross-Spatial Learning. In Proceedings of the 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2023), Rhodes Island, Greece, 4–10 June 2023; pp. 1–5.
31. Zhai, H.; Cheng, J.; Wang, M. Rethink the IoU-based loss functions for bounding box regression. In Proceedings of the 2020 IEEE 9th Joint International Information Technology and Artificial Intelligence Conference (ITAIC 2020), Chongqing, China, 11–13 December 2020; pp. 1522–1528.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.