YOLO-ADual: Lightweight Traffic Sign Detection Model on Mobile Driving System

Simin Fang , Chengming Chen * , Zhijian Li , Meng Zhou , Renjie Wei

Posted Date: 15 July 2024

doi: 10.20944/preprints202407.1126.v1

Keywords: traffic sign detection; small object detection; YOLO-ADual; attention mechanism; dual convolution

 doi:10.20944/preprints202407.1126.v1

Article

YOLO-ADual: Lightweight Traffic Sign Detection Model on Mobile Driving System

Simin Fang, Chengming Chen *, ZhiJian Li, Meng Zhou and Renjie Wei

College of Engineering Science and Technology, Shanghai Ocean University, Shanghai 201306, China; 2130303@st.shou.edu.cn (S.F.); cmchen@shou.edu.cn (C.C.)

* Correspondence: cmchen@shou.edu.cn;

Abstract: Traffic sign detection plays a pivotal role in autonomous driving systems. The intricacy of the detection model necessitates high-performance hardware. Real-world traffic environments exhibit considerable variability and diversity, posing challenges for effective feature extraction by the model. Therefore, it is imperative to develop a detection model that is not only highly accurate but also lightweight. In this paper, we proposed YOLO-ADual, a novel lightweight model. Our method leverages the C3Dual and Adown lightweight modules as replacements for CPS and CBL modules in YOLOv5. The Adown module effectively mitigates feature loss during downsampling while reducing computational costs. Meanwhile, C3Dual optimizes the processing power for kernel feature extraction, enhancing computation efficiency while preserving network depth and feature extraction capability. Furthermore, the inclusion of the CBAM module enables the network to focus on salient information within the image, thus augmenting its feature representation capability. Our proposed algorithm achieves a mAP@0.5 of 70.1% while significantly reducing the number of parameters and computational requirements to 51.83% and 64.73% of the original model, respectively. Compared to various lightweight models, our approach demonstrates competitive performance in terms of both computational efficiency and accuracy.

Keywords: traffic sign detection; small object detection; YOLO-ADual; attention mechanism; dual convolution

1. Introduction

With the rapid advancement of artificial intelligence and computer hardware, autonomous driving has become an essential feature in contemporary intelligent vehicles (IVs). This technology allows vehicles to transport passengers to their destinations safely without human intervention, significantly enhancing both comfort and safety. A crucial component of autonomous driving systems is Traffic Sign Detection (TSD), which ensures safe operation through precise recognition of traffic signs. Any inaccuracies in TSD can jeopardize driver safety [1]. Various sophisticated high-precision base networks have been proposed, including VGGNet [2], RetinaNet [3], and SSD [4]. However, these models are often complex, with a significant number of parameters and computations. For effective traffic sign recognition, TSD systems must be efficiently implementable in embedded vehicle devices. Large models frequently struggle to provide the real-time performance needed for industrial applications. Thus, the target detection algorithm for autonomous driving must combine high accuracy with a lightweight structure and rapid detection speed.

Most contemporary traffic sign detection (TSD) algorithms can be categorized into two main groups: traditional approaches and deep learning techniques. Traditional algorithms rely on shape, edge, and color characteristics, such as HSV, SIFT, and HOG, to identify traffic signs. However, they often underperform in complex backgrounds. Deep learning algorithms, in contrast, have gained popularity due to their robust feature extraction capabilities, enabling effective identification of small traffic signs even in intricate environments. Despite their advantages, deep learning models require substantial matrix operations and parallel processing, necessitating high-performance processors.

 doi:10.20944/preprints202407.1126.v1

GPUs are commonly used in deep learning due to their superior parallel processing capabilities. Neural network models contain millions of parameters, leading to high memory requirements for both storage and processing of extensive intermediate data. To mitigate these demands, the storage space required for neural networks can be reduced by pruning unimportant weights or lowering the number of parameters.

The deep learning approach to object detection categorizes target detection algorithms into two main categories: one-stage and two-stage detection algorithms. Typical examples of two-stage detection algorithms include R-CNN [5], SPPNet [6], and Fast R-CNN [7]. Despite their excellent accuracy, these two-stage models lag in training time and detection speed. In contrast, one-stage detection algorithms offer high speed, making them particularly well-suited for mobile applications. Several lightweight one-stage detection algorithms have been developed, such as Xception [8], SqueezeNet [9], MobileNet [10], ShuffleNet [11], and You Only Look Once (YOLO) [12], have been proposed by researchers. Xception combines depth-wise convolution derived from Inception-V3 [13] with pointwise convolution to reduce computation and parameters while maintaining high accuracy. SqueezeNet uses the VGG stacking concept and a compression method by substituting 3x3 convolutions with 1x1 convolutions, achieving only 2.14% of the parameters of AlexNet [14] while maintaining the same level of performance. While SqueezeNet reduces the network's parameters by increasing depth and minimizing the number of parameters, this approach can impact the network's parallel processing capabilities. This often results in slower convergence during training and heightened sensitivity to hyperparameter selection, complicating the tuning process. Additionally, SqueezeNet employs conventional convolution computation techniques.

In contrast, the subsequently introduced MobileNet model employs a more efficient deeply separable convolutional computation method, effectively reducing computational effort and model size, thereby accelerating the implementation of convolutional networks on mobile devices. MobileNetV3 [15] incorporates the VGG concept along with additional components such as the Squeeze and Excitation (SE) module and the novel nonlinear h-swish activation function. The SE module dynamically adjusts the importance of the feature channels by learning the dependencies between the channels, which enhances the expressive power of the model. h-swish activation function, on the other hand, is an improvement of the traditional Swish function, which provides higher computational efficiency and is particularly suitable for environments with limited computational resources. MobileNet utilizes the depth separable convolution, which offers enhanced computational efficiency and compatibility with quantization techniques. However, this may lead to poor dissemination of information. To address this, ShuffleNet employs channel shuffling, allowing information exchange across different groups of feature channels, thus enhancing feature representation and reducing computation. Additionally, ShuffleNet introduces the ResNet concept, which accelerates the model training process and improves the training efficiency of deep networks, setting it apart from MobileNet and SqueezeNet.

In the end, we concentrate on the YOLOv family. YOLO's core principle is to treat object detection as a single regression problem, directly converting image pixels into bounding box coordinates and class probabilities. YOLO partitions the input image into grids, each responsible for predicting objects whose centers are within that grid. By simplifying detection to a single network forward propagation, YOLO is particularly well-suited for real-time applications.YOLOv5 [16] , the most classic and widely adopted single-stage algorithm in the YOLO family, excels in precision, accuracy, and confidence scores over other YOLO algorithms. Despite this, YOLOv5 lacks an effective attention mechanism, leading to suboptimal performance for small target detection. Furthermore, these algorithms do not fulfill the lightweight and high-accuracy requirements essential for mobile devices.

Within the limitations of YOLOv5, the primary aim of this study is to develop a traffic sign detection algorithm that combines lightweight design with high precision, ensuring a balance between accuracy and efficiency. Moreover, this research aims to improve the detection of smaller targets. We present a novel traffic sign detection method, named YOLO-ADual, whose structure is shown in Figure 1.

 doi:10.20944/preprints202407.1126.v1

The key contributions of our research are summarized below:

1. The YOLO5s-ADual model is introduced in this study. By utilizing DualConv and C3, we propose a more efficient C3Dual architecture to replace the CBL module in the backbone of the YOLOv5 object detection model. Additionally, this architecture incorporates ADown from YOLOv9 in place of the Conv module in both the head and backbone of YOLOv5s. As a result, the model becomes lightweight, enabling faster reasoning and reducing the need for hardware resources, thereby making it more suitable for mobile device deployment.
2. In the actual scenario of TSD, since there are more small objects for traffic sign detection, the CBAM attention mechanism is adopted to improve the small object detection performance.
3. The experimental outcomes on the TT100k dataset demonstrate that the proposed method halves the parameters of the original YOLOv5s model while enhancing the mAP by 5 points. Compared to several contemporary lightweight object detection methods, the model presented in this paper demonstrates superior accuracy and a more lightweight structure.

The structure of the remaining sections is as follows: Section 2 provides an overview of recent research on two-stage object detection and YOLOv5s. Section 3 details the design of the C3Dual module, the structure of the ADown module, the CBAM attention mechanism, and the overall model structure. Section 4 describes the dataset, experimental setup, and the results obtained from the experiments. Finally, Section 5 offers a summary and the paper's conclusions.

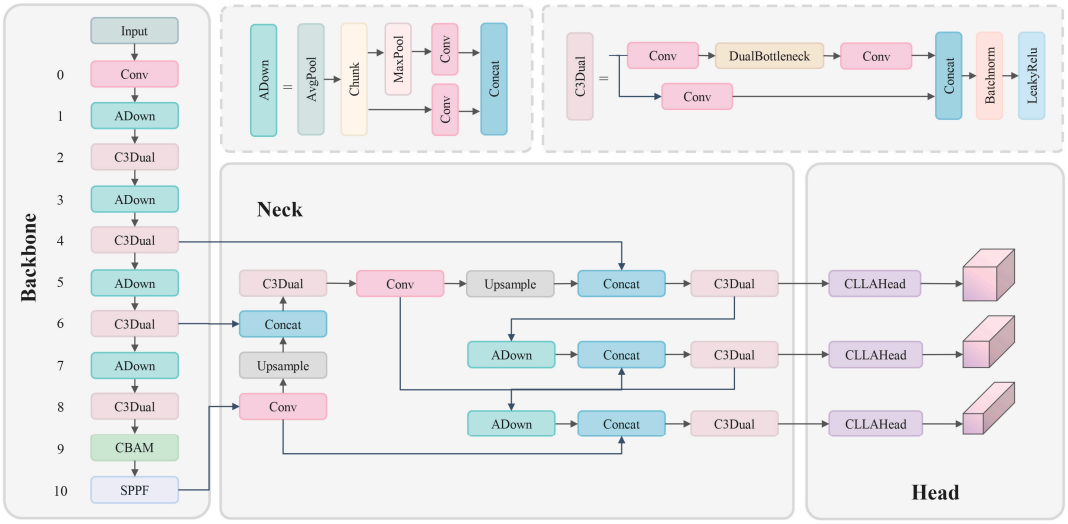

Figure 1. The overall architecture of YOLO-ADual.

2.1. Research on Two-Stage Approaches in Object Detection

The two-stage object detection algorithm is acclaimed for its remarkable precision in detecting targets. Within this algorithm, the task of detecting targets is segregated into two fundamental components: proposing regions and classifying targets. These components are adept at handling complex scenarios and identifying diminutive objects. The initial deployment of the R-CNN (Region-based Convolutional Neural Network) model marked the introduction of employing convolutional neural networks (CNNs) for the recognition of objects. Notably, the deep features retrieved by this model supplanted the conventional HOG and SIFT features. Significantly enhance the performance of detection. Nevertheless, due to up to 2,000 candidate regions in the first image search, each candidate frame necessitates characterization by CNN and subsequent classification by SVM, leading to a substantial computational burden. Spatial Pyramid Pooling Network (SPPNet) [17] employs a single convolutional neural network (CNN) operation on the entire image. It also utilizes a pyramid spatial pooling operation, significantly enhancing both speed and accuracy. Fast R-CNN incorporates an ROI Pooling layer, using Sofmax for classification, and utilizes a multi-task loss function. However, it still employs selective search to acquire 2000 candidate origins, which is very time-consuming. Consequently, the Faster R-CNN, later created, abandoned the selective search and

 doi:10.20944/preprints202407.1126.v1

introduced the region proposal network (RPN) instead. RPN generated region proposals and introduced the idea of Anchor, enabling the training of the entire object detection process from end to end. Region-based Fully Convolutional Networks (R-FCN) [18] employ convolutional operations to produce position-sensitive score maps. These maps encode the positional information of the target, enabling all Region proposals to share all parameters. This feature significantly enhances the efficiency of the detection. Feature Pyramid Networks for Object Detection (FPN) [19] introduces a feature pyramid network that emulates the down-top path of Convolutional Neural Networks (CNNs). The system employs lateral connections and a down-top path to extract features from images of varying scales, generating multi-scale feature representations. Feature maps at all levels possess robust semantic information, including specific high-resolution feature maps.

2.2. Lightweight traffic sign detection network

In real-world application settings, mobile devices often lack sufficient storage capacity to accommodate the extensive parameters of deep convolutional networks. Consequently, researchers have suggested viable methods. Zhang et al. [20](2023) proposed a lightweight YOLO model called Ghost-YOLO. This model uses the Ghost *module* and C3Ghost to achieve a lightweight detection model. Liu et al. [21](2023) integrated ConvNeXt-V2 to propose a module named C3_CN2. In the Head section, the lightweight receptive field attention module, LPFAConv, is employed to enhance the detection capability. Li et al. [22](2024) predict categories in the first stage of detection and fuses two-stage category prediction. An effective post-processing method, named SA-NMS, is also proposed for the first stage detection. The final results were obtained with high accuracy and inference speed. These studies show that the lightweight YOLO model performs better in detection and accuracy for mobile devices.

2.3. YOLOv5s

Before the introduction of YOLOv1, the R-CNN series of algorithms held a prominent position in the domain of target detection. The R-CNN series exhibited a notable level of detection accuracy. Still, its detection speed fell short of real-time performance due to its two-stage detection network architecture, resulting in substantial criticism. YOLO's core idea is to convert the target identification task into a regression issue. This transformation involves using the entire map as input to the neural network and only using a neural network to obtain the position of the bounding box and its corresponding class. YOLOv5 is an advancement iteration derived from YOLOv3 [23] and YOLOv4 [24].The overall architecture of YOLOv5 is shown in Figure 2. The dataset comprises five distinct models, specifically YOLOv5n, YOLOv5s, YOLOv5m, YOLOv5l, and YOLO5x. Those YOLOv5 models exhibit a progressive enhancement in detection accuracy, accompanied by a gradual decline in detection speed. Individuals can select an appropriate model to attain an optimal balance between precision and efficiency. In contrast to its predecessors, YOLOv4 and YOLOv3, YOLOv5 employs the Focus structure within its Backbone component and incorporates three distinct output Heads to facilitate multi-scale prediction. The same project team develops YOLOv8 as YOLOv5. This model is advanced and currently at the forefront of technology. The general framework of this model bears resemblance to YOLOv5. YOLOv8 gave up the prior Anchor-Base approach and instead adopted the Anchor-Free concept. Moreover, YOLOv8 introduced the C2f module to replace the CBL module in YOLOv5. The architecture of C2f primarily draws inspiration from the ELAN concept in YOLOv7 [25] and C3 modules. This enables YOLOv8 to get ampler gradient flow information while maintaining a lightweight approach.

Selcuk et al. [26] conducted a mobile UI detection experiment. It was observed that YOLOv5 outperforms YOLOv8 when applied to smaller UI solid parts. For comparison, Bian et al. [27] employed three models in their UAV image detection experiment, namely YOLOv5, YOLOv7, and Roboflow 2.0. The findings revealed that the YOLOv5s model achieved the highest average precision. Yusof et al. [28] conducted a road detection experiment. It was observed that YOLOv5 exhibited the highest mean average precision (mAP) when compared to YOLOv7 and YOLOv6. While YOLOv5 and YOLOv7 demonstrated similar inference image capabilities, YOLOv5 outperformed YOLOv7

 doi:10.20944/preprints202407.1126.v1

regarding confidence. Ultimately, the experiment concluded that YOLOv7 is twice as fast as YOLOv5, yet YOLOv5 outperformed YOLOv7 in accuracy and precision. Based on the three experiments above, it can be concluded that despite not being the most recent addition to the YOLO family, YOLOv5's algorithm continues to exhibit superior precision compared to YOLOv6, YOLOv7, and YOLOv8. The YOLOv5 model remains highly valuable for further research and enhancement. Hence, we choose YOLOv5 as the foundation for enhancement to design a detection model that is simultaneously lightweight and highly accurate.

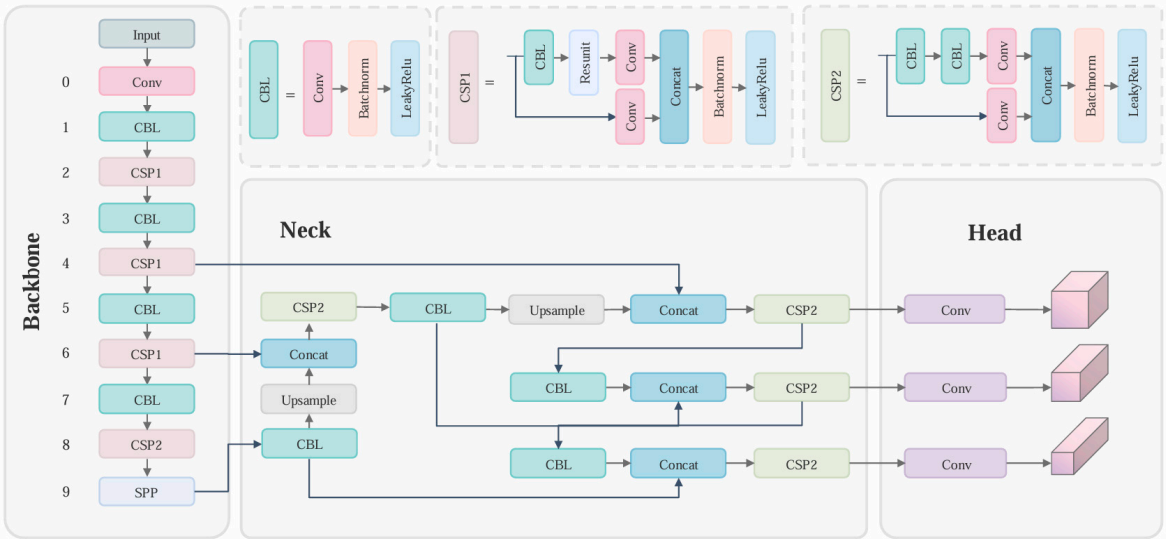

Figure 2. The overall architecture of YOLOv5s.

3.1. Overview of YOLO-ADual

This paper introduces a novel and lightweight object detection method named YOLO-ADual, intended to overcome the challenges of integrating intelligent transportation systems on mobile devices. The method integrates three key components into the YOLOv5s framework:

3.2. Adown

Traditional Convolutional Neural Networks (CNNs) often necessitate an extensive number of parameters and FLOPS to attain acceptable levels of accuracy. This is due to the significant redundancy present in the intermediate feature maps generated by prevalent CNN architectures. In this study, we utilize the Adown module introduced in YOLOv9 [29] to improve the Conv module of YOLOv5s. The Adown module is enhanced from the downsample network transition block in YOLOv7. A comparison of the structures of these two downsample networks is shown in Figure 3. The Transition-Block module's right section employs Maxpool and a convolution kernel with a stride of 1 for processing. Conversely, the branch's left section utilizes convolution kernels of 1×1 and 3×3 for the convolution operation [30]. The Adown module performs the AvgPool operation before splitting, the right part of the branch retains the original Transition-Block operation, and the other half only uses 3×3 convolution kernels for the convolution operation and deletes the 1×1 convolution. The Adown module has been improved to reduce the feature loss and enhance the feature learning efficiency.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 15 July 2024 doi:10.20944/preprints202407.1126.v1

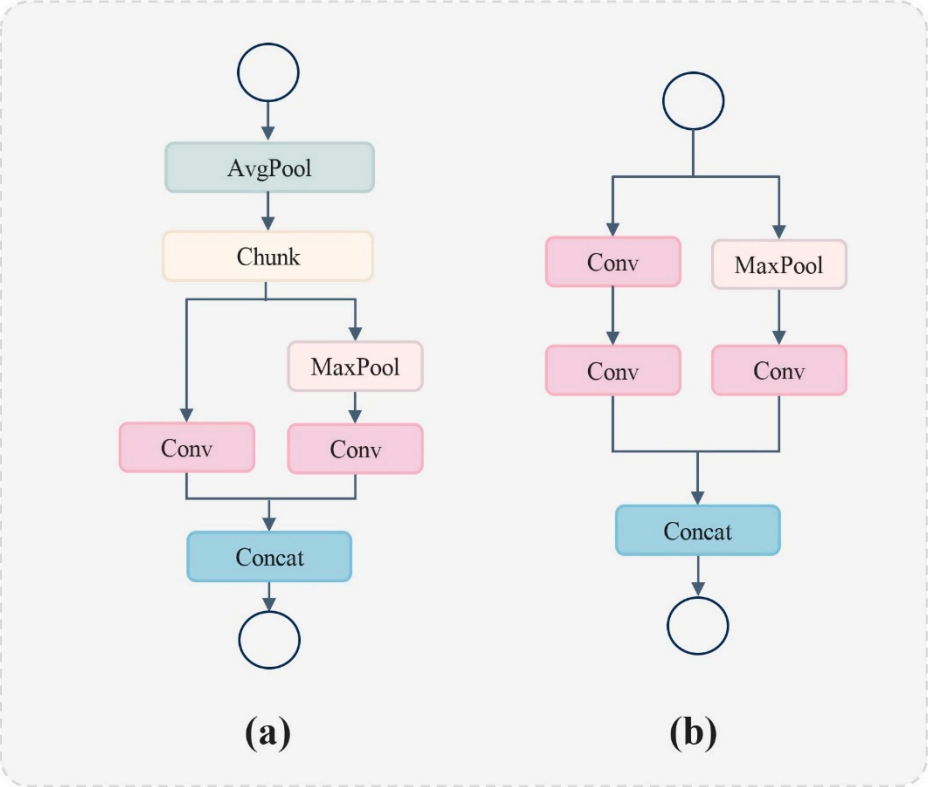

Figure 3. (**a**) The structure of the Adown module, which is used in YOLOv9 as a downsample network, (**b**) The structure of Transition-Block, which is used in YOLOv7 as a downsample network.

3.3. C3Dual

DualConv is a lightweight deep neural network designed to optimize both spatial information capture and computational efficiency. As illustrated in Figure 4, DualConv combines 3×3 and 1×1 convolutional kernels [31]. The 3×3 convolutional kernel enhances spatial information extraction, while the 1×1 convolutional kernel facilitates interactions between feature channels and information integration, minimizing additional parameters and computational complexity. By integrating these two convolutional kernels, DualConv processes the same input feature mapping channels concurrently, with each group of convolutional kernels handling a subset of the input channels separately. The outputs are then merged, promoting efficient information flow and integration across different feature map channels. This structural design preserves the network depth and characterization capability, optimizing information processing for kernel feature extraction, and reducing computational complexity and model size. Thus, DualConv is particularly suitable for resource-constrained environments.

Furthermore, DualConv employs group convolution to effectively reduce the number of parameter kernel computations. In group convolution, the input and output feature maps are divided into multiple groups, and each group of convolution filters processes only a portion of the corresponding input feature map. This reduces model complexity by allowing different convolutional kernels within a group (e.g., 3×3 and 1×1) to process the same set of input channels in parallel. This design leverages the spatial feature extraction capabilities of large-size convolutional kernels and the computational efficiency of small-size convolutional kernels, thereby lowering the parameter and computational costs of the model while maintaining accuracy.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 15 July 2024 doi:10.20944/preprints202407.1126.v1

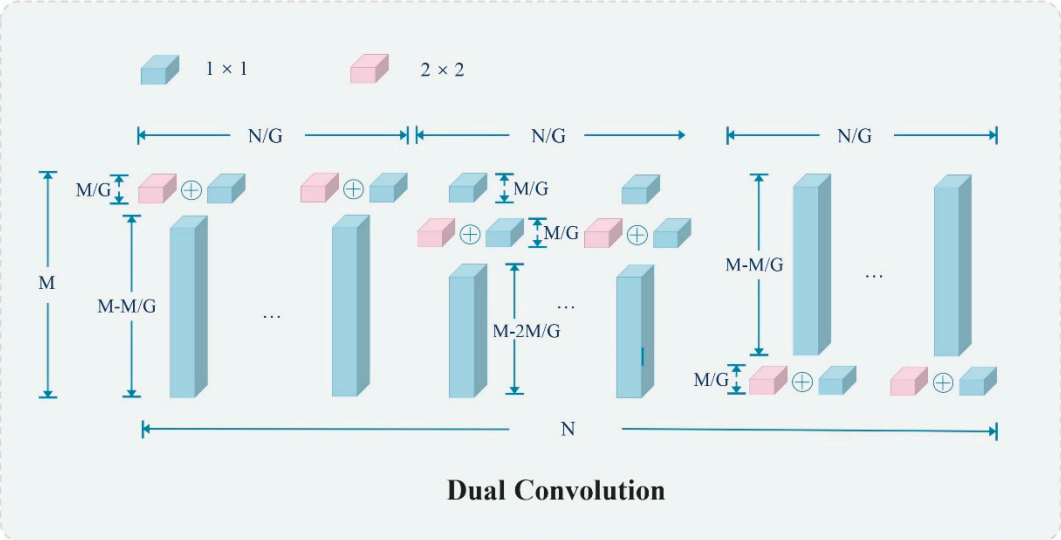

Figure 4. Diagram of the structure of the group convolution technique utilized by DualConv

In this study, we leverage the advantages of DualConv to introduce a compact feature extraction framework named C3Dual. The C3Dual architecture consists of three 1×1 convolutional layers and n sequentially arranged DualBottlenecks. The initial 1×1 convolutional layer halves the channel count relative to the output channels. Subsequent layers extract features through sequentially arranged DualBottlenecks and separate residual branches. This method captures deep semantic information from the input image through both branches, and the two feature sets are combined via the Contact module. The Contact operation fuses multiple feature maps based on channel count, enhancing semantic information utilization across various scales and improving performance through channel expansion. The combined features are processed through a BatchNorm module and activated using LeakyReLU. Each DualBottleneck comprises two 1×1 convolutional layers flanking a Dual convolutional layer, with the 1×1 layers reducing and expanding dimensions, and the Dual convolutional layer serving as the bottleneck for smaller dimensions. By replacing all CSP modules in YOLOv5 with C3Dual modules, computational demand is significantly reduced and the model size is compressed without compromising detection accuracy..

3.3. CBAM

Convolutional Block Attention Module (CBAM) focuses on essential features and screens out unnecessary features to enhance the network representation [32]. Figure 5 (**a**) shows the schematic structure of CBAM. It contains a Channel Attention Module(CAM) and a Spatial Attention Module(SAM). Firstly, CBAM focuses on the important features through CAM and then focuses on the important locations of these features through SAM. This method can effectively help the network focus on the image's key information and improve the features' representation strength. Given the input feature $F \in R^{C*H*W}$, first, perform the one-dimensional convolution $M_c \in R^{C*1*1}$, of the channel attention module, multiply the convolution result by the original image to get the CAM output result F' as shown in equation (1):

$$F' = M_c(F) \times F, \quad (1)$$

Take the CAM output F' as input, perform the two-dimensional convolution $M_s \in R^{1*H*W}$, of spatial attention module, and multiply the output result with the original image. As shown in equation (2)

$$F'' = M_s(F') \times F', \quad (2)$$

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 15 July 2024 doi:10.20944/preprints202407.1126.v1

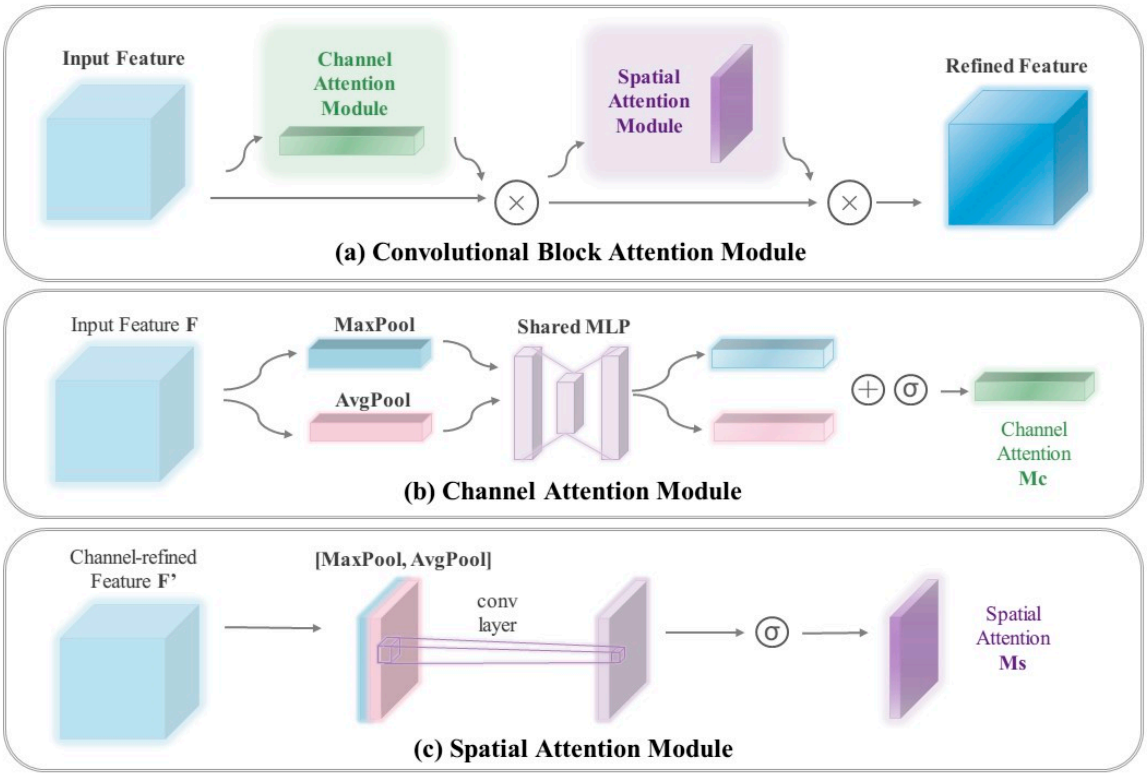

Figure 5. (**a**) The structure of the Convolutional Block Attention Module(CBAM), (**b**) The structure of the Channel Attention Module(CAM), (**c**) The structure of the Spatial Attention Module(SAM).

3.3.1. Channel Attention Module

The Channel Attention Module (CAM) is designed to enhance the feature representation of each channel. It maintains the channel dimension while compressing the spatial dimension. As illustrated in Figure 5(**b**), the input feature F is processed through parallel MaxPool and AvgPool layers, transforming the feature map from a size of C*H*W to C*1*1. Subsequently, it undergoes processing in the Shared MLP module, where it first reduces the channel count to 1/r of its initial value and then restores it to the original count. Finally, it generates two activated outputs using the ReLU activation function [33]. Element-wise summation is performed on the two output results, followed by the application of a sigmoid activation function to obtain the Channel Attention weight. This weight is then applied to each channel of the original feature map, resulting in a weighted feature map with dimensions C*H*W. By employing parallel MaxPool and AvgPool layers, a more diverse and comprehensive set of high-level features can be extracted. The weighting of the feature map serves to highlight channels beneficial for the current task while diminishing the impact of irrelevant channels. The formula for channel attention is:

$$M_C(F) = \sigma(MLP(AP(F)) + MLP(MP(F))) \quad (3)$$

where σ denotes the sigmoid function, $MLP \in R^{\frac{C}{r}\times C}$, and $AP \in R^{C\times C/r}$. AP represents AvgPool, MP represents MaxPool, MLP represents the connect operation. Note that the MLP weights, MLP and AP, are shared for both inputs and the ReLU activation function is followed by AP.

3.3.2. Spatial Attention Module

The Spatial Attention Module (SAM) concentrates on the spatial information of the target without altering the spatial dimensions; it only compresses the channel dimensions. Figure 5(**c**) illustrates the schematic structure of SAM. The output F' from the Channel Attention Module (CAM) is processed through maximum and average pooling to generate two 1*H*W feature maps. These

 doi:10.20944/preprints202407.1126.v1

maps are then concatenated and transformed into a single-channel feature map through a 7*7 convolution. Spatial attention weights are subsequently derived using a sigmoid function. These weights are applied to the original feature map to emphasize features at specific spatial locations, converting the map back to a C*H*W size. This process accentuates significant regions of the image while diminishing the impact of less important areas. The formula for spatial attention is:

$$M_S(F) = \sigma(f^{7\times7}([AP(F) + MP(F)])) \quad (4)$$

where σ denotes the sigmoid function and $f^{7\times7}$ represents a convolution operation with the filter size of 7×7. $AP \in R^{1\times H\times W}$ and $MP \in R^{1\times H\times W}$. AP represents AvgPool, MP represents MaxPool. Each denotes average-pooled features and max-pooled features across the channel.

4. Experiment Analysis

4.1. Datasets

The Tsinghua-Tencent 100K dataset is a massive, high-resolution dataset of traffic signs created by Tsinghua University and Tencent. The collection comprises 100,000 photos with a resolution of 2048 x 2048, gathered from the Tencent Street View map under various lighting and weather situations. The TT100K dataset represents a practical scenario for traffic sign detection on mobile devices, containing 221 categories of Chinese traffic signs and includes over 30,000 annotated occurrences of traffic signs across 10,000 pictures. Sample images are shown in Figure 6. In accordance with the experimental findings, the native dataset contains a substantial amount of traffic signs, with a small proportion that cannot be effectively learned. We analyzed the number of traffic signs in each dataset category to address this issue. To mitigate the sampling variations induced by the diverse categories of traffic signs in the dataset, we selected 45 categories with a count exceeding 100. The chosen images depict traffic signs of various sizes, ranging from 8 × 8 to 400 × 400 pixels. These signs constitute a small portion of the entire image, representing between 0.001% and 4% of the total area.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 15 July 2024 doi:10.20944/preprints202407.1126.v1

Figure 6. Sample images of the TT100k dataset

Figure 7 depicts a sample from the dataset. Signs prefixed with "w" denote warning signs, while those starting with "p" represent prohibition signs, and those beginning with "i" signify indication signs. Additionally, signs ending with an asterisk (*) signify numerical values, such as pl50, pl60, and pl100. The dataset comprises 9457 images, divided into three categories: 6598 for training, 970 for testing, and 1889 for validation.

Figure 7. 45 categories of traffic signs selected from TT100k

4.2. Experimental Environment

Our experimental setup is detailed in Table 1. We used high-end hardware components, including an Intel i5-13490F CPU and an RTX 4070Ti GPU, complemented by 12 GB of VRAM and 16 GB of RAM. For software, we opted for Windows 11 as the operating system and Python 3.9.12 as our primary programming language. The deep learning framework employed is PyTorch 1.10.0, with CUDA 11.2 enhancing computational speed. The detailed configuration is essential for ensuring the accuracy and reproducibility of our results. We conducted 300 training epochs, and Table 2

 doi:10.20944/preprints202407.1126.v1

summarizes the setup parameters for YOLO-ADual. PyTorch facilitates model training with its comprehensive suite of tools and libraries for constructing and training neural networks. Algorithm 1 outlines the training algorithm used in this study.

Table 1. Experiment configuration environment.

Configuration	Name	Specific Information
Hardware Environment	CPU	Intel(R) core(TM)i5-13490F
	GPU	NVIDIA GeForce RTX4070Ti
	VRAM	12 GB
	Memory	16 GB
Software Environment	Operating System	Windows 11
	Python Version	3.9.12
	PyTorch Version	2.0.0
	CUDA Version	11.8

4.3. Experimental Details

Step1 (data preprocessing): To comprehensively evaluate the performance of the target detection algorithm in traffic scenarios, we utilized the Tsinghua-Tencent 100K dataset. This dataset comprises 100,000 photos collected by Tencent Street View Map under various lighting and weather conditions, covering 221 categories of Chinese traffic signs and over 30,000 traffic signs labeled in 10,000 images. For the experiment's accuracy, we selected 9,457 images, using 6,598 for training, 970 for testing, and 1,889 for validation. To address issues of missing labels or incorrect annotations, we manually corrected and labeled them using the LabelImg tool.

Before formal model training, all images underwent a rigorous preprocessing step to optimize learning. Initially, images were normalized by scaling pixel values to the range [0, 1] to reduce pixel value differences. Subsequently, images were uniformly resized to the model's required size, generally using bilinear interpolation to maintain quality, and cropping or padding as necessary. Finally, image enhancement techniques were applied to overexposed or low-contrast images: exposure compensation restored details in overexposed images, while contrast stretching or histogram equalization enhanced low-contrast images. These enhancements improved image quality and, consequently, the model's accuracy in recognizing traffic signs.

Step2(model selection): In the field of target detection technology, numerous advanced algorithms have been developed and applied to monitor the transportation environment. Each algorithm has specific design features and corresponding advantages and disadvantages. When selecting algorithms for our experiments, it was crucial to consider not only their inherent performance but also their adaptability to specific datasets. We explored seven cutting-edge target detection algorithms: YOLOv5n, YOLOv5s, SSD, YOLOv7-tiny, EfficientDet, YOLOX, Faster R-CNN, and Zhang et al. Additionally, we introduced and explored an innovative detection method, YOLO-ADual. Although these algorithms have demonstrated excellent performance in various application scenarios, our primary goal was to evaluate their performance differences in traffic sign detection. We benchmarked these algorithms on the same dataset to identify the most suitable model for traffic scene detection.

Step3(model training): In this study, we employed a transfer learning approach to rigorously train the YOLO-ADual model. Initially, the YOLO-ADual model was initialized using pre-trained weights on the COCO dataset. Then, it was carefully fine-tuned on specific datasets to meet our application requirements. During training, we set the batch size to 32, the initial learning rate to 0.01, and performed 300 training cycles. The parameters related to the YOLO-ADual configuration are detailed in Table 2. We used PyTorch as the deep learning framework due to its comprehensive tools and libraries for building and training neural networks. Adam was chosen as the initial optimizer for its combination of momentum method and adaptive learning rate technique, which automatically adjusts each parameter's learning rate. Unlike SGD, Adam simplifies the optimization process by not

 doi:10.20944/preprints202407.1126.v1

requiring manual learning rate adjustments, thereby finding the minimum value of the loss function faster and accelerating model convergence. Algorithm 1 in the article details the training process used.

Table 2. YOLO-ADual configuration parameters.

Parameters	Specific Information
Epoch	300
Image size	640
Batch size	32
Number of images	9457
Parameters	3,817,609
Layers	246

Algorithm 1: YOLO-ADual training algorithm with transfer learning

Data: COCO Pretrained Weights, Specialized Datasets: TT100k
Result: Fine-tuned YOLO-ADual Model
Initialize YOLO-ADual neural network with COCO pretrained weights;
Initialize training parameters: batch size = 32, initial learning rate = 0.01, total epochs = 300;
Initialize loss function: Loss;
Initialize optimizer: Adam;
Initialize evaluation metrics: Recall, Precision;
for *epoch* ← 1 **to** *total_epochs* **do**
    for *batch* ← 1 **to** *total_batches* **do**
        // Load a batch of training data from TT100k
        image_batch, ground_truth_batch ← LoadBatchFromDatasets(TT100k, batch_size);
        // Forward pass through the network
        predicted_boxes ← YOLO-ADual(image_batch);
        // Calculate Loss
        loss ← CalculateIoULoss(predicted_boxes, ground_truth_batch);
        // Backpropagation and weight update
        BackpropagateAndOptimize(loss);
    end
    // Adjust learning rate (e.g., learning rate decay)
    if *epoch % learning_rate_decay_interval == 0* ***then***
        AdjustLearningRate(optimizer, new_learning_rate);
    end
    // Evaluate the model on validation data
    recall, precision ← EvaluateModel(YOLO-ADual, ValidationData);
    if *recall > threshold_recall* ***and*** *precision > threshold_precision* ***then***
        // Save the model if recall and precision meet the criteria
        SaveModel(YOLO-ADual, 'trained_model_epoch_' + epoch);
    end
end

Step 4 (model evaluation): For a comprehensive and systematic evaluation of the model's performance in target detection for traffic scenes, we utilize standardized evaluation metrics such as precision, average precision (AP), mean average precision (mAP), recall, F1 score, and processing speed (fps). These metrics provide a quantitative overview of the model's effectiveness in complex traffic environments. If the model underperforms in specific areas, optimizing hyperparameters or

 doi:10.20944/preprints202407.1126.v1

expanding the training dataset may enhance its capabilities. Ensuring a balance between overfitting and underfitting is essential for excellent generalization to unseen data. Precision, for instance, indicates the proportion of true positive (TP) samples out of all samples predicted as positive.

$$precision = \frac{TP}{TP + FP} \tag{4}$$

where TP denotes the number of images that have properly recognized items with an Intersection over Union (IoU) of more than 0.5. In other words, it refers to the count of images with correctly identified positive samples by the model. Conversely, FP represents the count of images where objects are detected with an IoU below 0.5, classified as false positives.

Recall the ratio of accurate predictions to the total number of positive samples, as follows:.

$$recall = \frac{TP}{TP + FN}, \tag{5}$$

where FN (false negative) represents the number of pictures mistakenly classified as not containing things of interest.

The average precision (AP) is the measure of the area enclosed by the precision-recall curve and the X-axis., calculated as follows:

$$AP = \int_0^1 p(r)dr, \tag{6}$$

where the precision function for a given recall r is denoted by p(r)

The F1 score is calculated as the harmonic mean of precision and recall, with values ranging from 0 to 1, as follows:

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}. \tag{7}$$

The mean average precision (mAP) is calculated as the average AP value across all object classes, as follows:

$$mAP = \frac{\sum AP}{N_{classes}}, \tag{8}$$

where $N_{classes}$ denotes the number of classes.

4.4. Result Analysis

Figure 8 displays the comprehensive results of the road detection model proposed in this study. Considering a range of performance metrics, our model demonstrates outstanding performance across all parameters. To illustrate the effectiveness of our proposed technique in Traffic Sign Detection (TSD) tasks, we conducted a comparative analysis between our YOLO-ADual model and several other models, including Faster R-CNN, Zhang et al., YOLOv7-tiny, SSD, EfficientDet, YOLOX, YOLOv5n, and YOLOv5s. Among these, Faster R-CNN serves as a two-stage detector, whereas YOLO is the one-stage detector. As depicted in Table 3, Faster R-CNN exhibits suboptimal performance as a two-stage detector, achieving a mAP@0.5 of merely 53.1%. Conversely, our YOLO-ADual model demonstrates commendable results on the TT100k dataset, with an accuracy of 71.8%, a recall of 63.23%, and a mAP@0.5 of 70.1%. for two-stage detectors, our model's mAP@0.5 surpasses EfficientDet by 18.8%,Faster R-CNN by 17% and SSD by 16.84%. Moreover, Relative to the YOLOv5-s model, our model shows an enhancement of 2.46% in accuracy, 0.27% in recall, and 2.5% in mAP@0.5. In comparison with other YOLO variants, YOLO-ADual exhibits competitive performance across all detection metrics.

 doi:10.20944/preprints202407.1126.v1

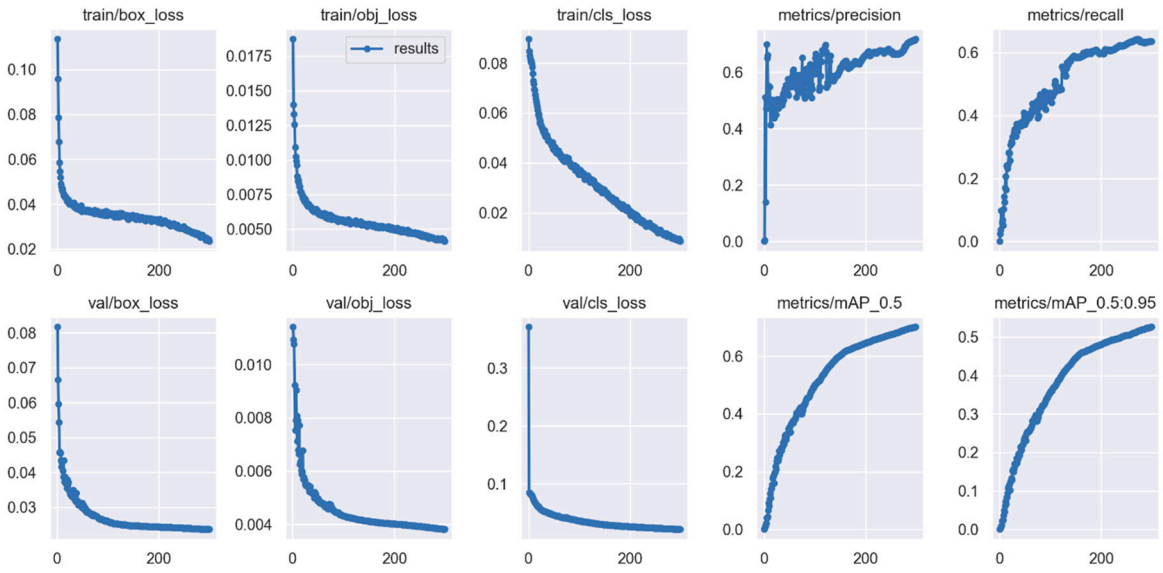

Figure 8. Sample images of the TT100k dataset

In terms of network size, the parameter count and computational requirements of YOLO-ADual are reduced to 51.83% and 64.73% of the original, respectively. YOLOv7-tiny, a notable lightweight model, has its parameters and computational load reduced by 37.68% and 17.03%, respectively, when compared to YOLO-ADual. Additionally, we compared our model with Zhang et al., a lightweight network optimized for traffic sign detection. YOLO-ADual outperforms Ghost-YOLO with a 37.68% reduction in parameters and a 14.6% improvement in mAP@0.5. YOLOX introduces an anchor-free approach divergent from the anchor-based method used in YOLOv3, YOLOv4, and YOLOv5. It has a parameter count and computational intensity that are comparable to YOLOv5s in terms of mAP@0.5. However, our YOLO-ADual model still achieves a 24% reduction in parameter count and a 4.7% higher mAP@0.5 compared to YOLOX.

In addition to evaluating intuitive loss performance, this paper presents a classification confusion matrix, as illustrated in Figure 9. The confusion matrix summarizes the results of classification, serving as an accuracy assessment tool. Darker shades within the matrix indicate a higher recognition rate for the respective targets.

Table 3. Comparative experimental results between YOLO-ADual and other object lightweight detection algorithms. Bold represents the best result

Method	Precision	Recall	mAP@0.5	Params	GFLOPs
Faster R-CNN[34]	47.91	53.79	53.1	-	-
Zhang et al. [20]	56.10	52.10	55.5	6,655,232	8.6
SSD	51.45	53.76	53.26	641,473	3.1
EfficientDet	68.80	46.40	51.30	5,524,683	9.4
YOLOv7-tiny[25]	61.70	57.20	59.9	6,125,934	13.5
YOLOX[35]	64.47	58.27	65.4	5,044,797	15.30
YOLOv5n	67.90	45.10	48.8	1,820,743	4.4
YOLOv5s	69.34	63.23	67.6	7,365,671	17.3
YOLO-ADual	71.80	63.50	70.1	3,817,609	11.2

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 15 July 2024 doi:10.20944/preprints202407.1126.v1

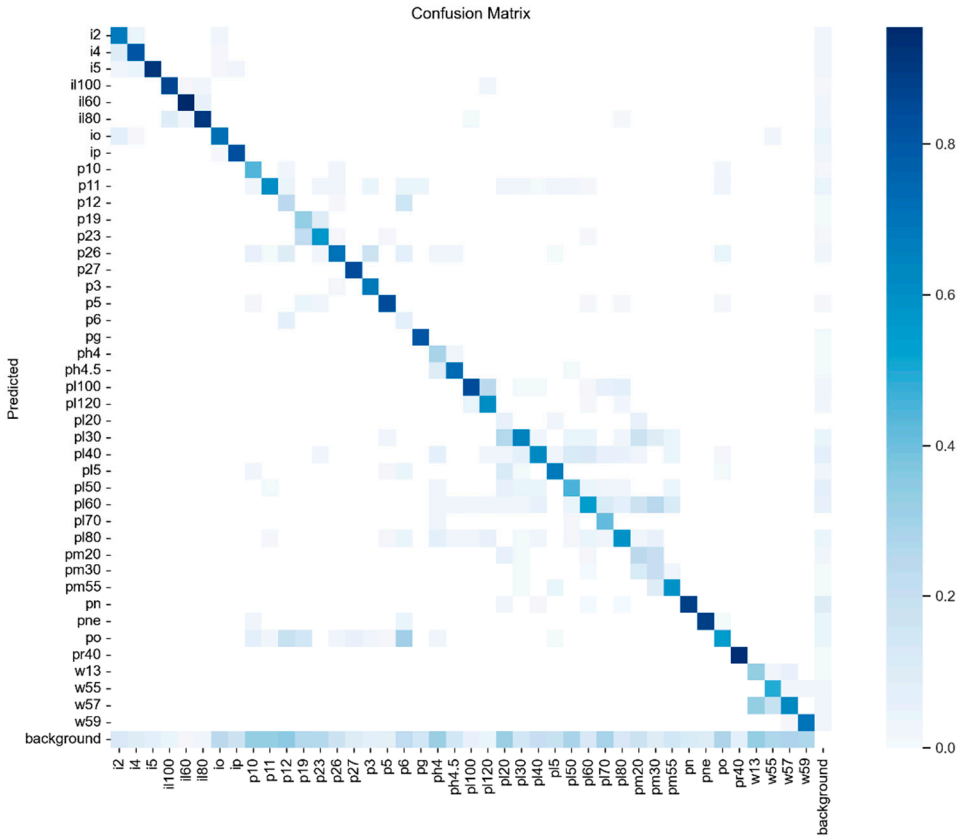

Figure 9. YOLO-ADual Confusion matrix on test set

4.5. Ablation Study

To further validate the efficacy of our proposed YOLOv5s-ADual model, we conducted ablation experiments using the TT100K dataset, with the original YOLOv5s as the baseline model. The results, presented in Table 4, compare performance metrics including mAP@0.5, parameters, and GFLOPs. Each innovation demonstrates improvement over the baseline. Our implementation of the CBAM attention mechanism exhibits significant enhancements across all evaluation metrics, notably increasing mAP@0.5 by 9.87% while simultaneously reducing both parameters and GFLOPs. This underscores the suitability of CBAM for detecting small traffic signs on mobile devices. Furthermore, employing solely the Adown module and C3Dual led to a reduction of approximately two million arithmetic parameters and GFLOPs, resulting in respective mAP@0.5 improvements of 4.6% and 9.5%. The YOLO-ADual model, in contrast to the original, reduces approximately three and a half million parameters and 6.1 GFLOPs, significantly diminishing computational demands within the detection network. Notably, YOLO-ADual exhibits the lowest parameter count among the ablation experiments, demonstrating a 2.5% improvement in mAP@0.5. These findings affirm that YOLO-ADual not only reduces the number of operations and compresses model size but also achieves superior accuracy.

Table 4. Comparison of ablation experiment results

Method	CBAM	ADown	C3Dual	mAP@0.5	Params	GFLOPs
YOLOv5s				67.6	7,365,671	17.3
YOLOv5s	√			77.47	7,156,265	16.2
YOLOv5s		√		72.2	5,468,903	12.9
YOLOv5s			√	77.1	5,828,903	13.0
YOLOv5s	√	√		67.87	5,501,769	12.9
YOLOv5s		√	√	69.6	4,163,719	9.5

 doi:10.20944/preprints202407.1126.v1

YOLOv5s	√	√	√	70.1	3,817,609	11.2

4.6. Visualization Result

To comprehensively evaluate the detection capabilities of our model, we have visualized its outputs. Figure 10(a) presents the detection results for both YOLOv5s and YOLO-ADual. Both models were initialized with the same learning rate and utilized the Adam optimizer. The original image is displayed in the first column, followed by the visual detection outcomes of YOLOv5s and YOLO-ADual in the second and third columns, respectively. Given the small size of the traffic sign in the original image, an enlarged version is provided in the corner to facilitate closer examination of the detection details. Upon inspecting the first comparison row in Figure 10(a), it is evident that YOLO-ADual successfully detects the 'pn' traffic sign, whereas YOLOv5s does not. Similarly, in the second comparison row, YOLO-ADual detects the 'io' traffic sign, while YOLOv5s not only fails to detect 'io' traffic signs but also shows a significantly lower confidence level compared to our model. Figure 10(b) illustrates the detection results of YOLO-ADual on small targets, demonstrating the model's proficiency in identifying distant and diminutive traffic signs with high confidence, which is crucial for Traffic Sign Detection (TSD) tasks and the safety of autonomous vehicles. Additionally, as depicted in Figure 10(c), the model showcases its capability to detect traffic signs under challenging conditions, including occlusion and shadows, and to accurately identify small targets. This highlights the model's robustness in recognizing traffic signs amidst complex surroundings.

(**a**)

(**b**)

 doi:10.20944/preprints202407.1126.v1

(**c**)

Figure 10. Visualization outcomes on the TT100K dataset include (**a**) a comparative analysis of the results between YOLOv5s and YOLO-ADual, (**b**) detection outcomes for small targets, (**c**) detection performance in challenging conditions such as shadows, occlusions, and cloudy weather.

5. Conclusions

In this study, we present YOLO-ADual, an innovative and lightweight architecture based on the YOLOv5 framework, designed to overcome the challenges of traffic sign detection while addressing the limitations of the original YOLOv5. We enhance the model's ability to detect smaller objects by integrating the Channel Attention Mechanism (CBAM). Additionally, we replace the original model's Cross-Stage Partial (CSP) module with Adown, a downsampling component adapted from YOLOv9. We also introduce the Dual convolution technique within the C3 structure, creating a novel C3Dual module. This innovation not only improves the model's precision in identifying small objects but also significantly reduces the computational and parameter requirements, achieving a lightweight configuration. Evaluation on the TT100K dataset demonstrates that YOLO-ADual strikes a commendable balance between accuracy and lightweight performance, showing resilience in detecting small targets in complex environments. Our model improves accuracy by 2.46%, recall by 0.27%, mAP by 2.5%, reduces parameters by 51.83%, and decreases computational load by 64.73%. In future research, we aim to develop even lighter models and enhance the model's speed for mobile applications. To more closely match in-vehicle hardware configurations, we plan to experiment with lower configuration devices or vehicle-specific hardware, such as the NVIDIA Drive PX, used by Tesla for the Autopilot system. We also intend to explore the impact of diverse autonomous driving conditions, including extreme weather scenarios (e.g., snow, fog, tornado) and varying levels of exposure or low light environments. These conditions may influence detection accuracy, necessitating future investigations into illumination invariant techniques and visual attention mechanisms. Overall, this study contributes to reducing hardware dependency for target detection algorithms designed for mobile devices, with significant implications for enhancing the safety of autonomous driving.

Author Contributions: Conceptualization, M.Z. and S.F.; Methodology, S.F.; Software, M.Z.; Validation S.F.; Resources, C.C. and Z.L.; Data curation, M.Z.; Writing—original draft preparation, S.F. and R.W.; Writing—review and editing, S.F. and C.C.; Visualization, S.F. and R.W; Supervision, C.C.; All authors have read and agreed to the published version of the manuscript.

Funding: This work was funded by the National Natural Science Foundation of China (Grant No. 62027810)

Data Availability Statement: Data are contained within the article.

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

References

1. Geiger, A.; Lenz, P.; Urtasun, R. Are We Ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition; IEEE: Providence, RI, June 2012; pp. 3354–3361.
2. Simonyan, K.; Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. *ArXiv Prepr. ArXiv14091556* **2014**.

3. Lin, T.-Y.; Goyal, P.; Girshick, R.; He, K.; Dollár, P. Focal Loss for Dense Object Detection. In Proceedings of the Proceedings of the IEEE International Conference on computer vision; 2017; pp. 2980–2988.
4. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.-Y.; Berg, A.C. SSD: Single Shot MultiBox Detector. In *Computer Vision – ECCV 2016*; Leibe, B., Matas, J., Sebe, N., Welling, M., Eds.; Lecture Notes in Computer Science; Springer International Publishing: Cham, 2016; Vol. 9905, pp. 21–37.
5. Mittal, U.; Chawla, P.; Tiwari, R. EnsembleNet: A Hybrid Approach for Vehicle Detection and Estimation of Traffic Density Based on Faster R-CNN and YOLO Models. *Neural Comput. Appl.* **2023**, *35*, 4755–4774, doi:10.1007/s00521-022-07940-9.
6. Ghahremannezhad, H.; Shi, H.; Liu, C. Object Detection in Traffic Videos: A Survey. *IEEE Trans. Intell. Transp. Syst.* **2023**, *24*, 6780–6799, doi:10.1109/TITS.2023.3258683.
7. Arora, N.; Kumar, Y.; Karkra, R.; Kumar, M. Automatic Vehicle Detection System in Different Environment Conditions Using Fast R-CNN. *Multimed. Tools Appl.* **2022**, *81*, 18715–18735, doi:10.1007/s11042-022-12347-8.
8. Chollet, F. Xception: Deep Learning with Depthwise Separable Convolutions. In Proceedings of the Proceedings of the IEEE conference on computer vision and pattern recognition; 2017; pp. 1251–1258.
9. Iandola, F.N.; Han, S.; Moskewicz, M.W.; Ashraf, K.; Dally, W.J.; Keutzer, K. SqueezeNet: AlexNet-Level Accuracy with 50x Fewer Parameters And< 0.5 MB Model Size. *ArXiv Prepr. ArXiv160207360* **2016**.
10. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. Mobilenets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *ArXiv Prepr. ArXiv170404861* **2017**.
11. Zhang, X.; Zhou, X.; Lin, M.; Sun, J. Shufflenet: An Extremely Efficient Convolutional Neural Network for Mobile Devices. In Proceedings of the Proceedings of the IEEE conference on computer vision and pattern recognition; 2018; pp. 6848–6856.
12. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. In Proceedings of the Proceedings of the IEEE conference on computer vision and pattern recognition; 2016; pp. 779–788.
13. Xia, X.; Xu, C.; Nan, B. Inception-v3 for Flower Classification. In Proceedings of the 2017 2nd international conference on image, vision and computing (ICIVC); IEEE, 2017; pp. 783–787.
14. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet Classification with Deep Convolutional Neural Networks. In Proceedings of the Advances in Neural Information Processing Systems; Pereira, F., Burges, C.J., Bottou, L., Weinberger, K.Q., Eds.; Curran Associates, Inc., 2012; Vol. 25.
15. Koonce, B. MobileNetV3. In *Convolutional Neural Networks with Swift for Tensorflow*; Apress: Berkeley, CA, 2021; pp. 125–144.
16. Yifan Liu; BingHang Lu; Jingyu Peng; Zihao Zhang Research on the Use of YOLOv5 Object Detection Algorithm in Mask Wearing Recognition. *World Sci. Res. J.* **2020**, *6*, doi:10.6911/WSRJ.202011_6(11).0038.
17. Purkait, P.; Zhao, C.; Zach, C. SPP-Net: Deep Absolute Pose Regression with Synthetic Views. *ArXiv Prepr. ArXiv171203452* **2017**.
18. Dai, J.; Li, Y.; He, K.; Sun, J. R-FCN: Object Detection via Region-Based Fully Convolutional Networks. In Proceedings of the Advances in Neural Information Processing Systems; Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., Garnett, R., Eds.; Curran Associates, Inc., 2016; Vol. 29.
19. Lin, T.-Y.; Dollár, P.; Girshick, R.; He, K.; Hariharan, B.; Belongie, S. Feature Pyramid Networks for Object Detection. In Proceedings of the Proceedings of the IEEE conference on computer vision and pattern recognition; 2017; pp. 2117–2125.
20. Zhang, S.; Che, S.; Liu, Z.; Zhang, X. A Real-Time and Lightweight Traffic Sign Detection Method Based on Ghost-YOLO. *Multimed. Tools Appl.* **2023**, *82*, 26063–26087, doi:10.1007/s11042-023-14342-z.
21. Liu, P.; Xie, Z.; Li, T. UCN-YOLOv5: Traffic Sign Object Detection Algorithm Based on Deep Learning. *IEEE Access* **2023**, *11*, 110039–110050, doi:10.1109/ACCESS.2023.3322371.
22. Li, Z.; Chen, H.; Biggio, B.; He, Y.; Cai, H.; Roli, F.; Xie, L. Toward Effective Traffic Sign Detection via Two-Stage Fusion Neural Networks. *IEEE Trans. Intell. Transp. Syst.* **2024**, 1–12, doi:10.1109/TITS.2024.3373793.
23. Redmon, J.; Farhadi, A. Yolov3: An Incremental Improvement. *ArXiv Prepr. ArXiv180402767* **2018**.
24. Bochkovskiy, A.; Wang, C.-Y.; Liao, H.-Y.M. Yolov4: Optimal Speed and Accuracy of Object Detection. *ArXiv Prepr. ArXiv200410934* **2020**.
25. Li, S.; Wang, S.; Wang, P. A Small Object Detection Algorithm for Traffic Signs Based on Improved YOLOv7. *Sensors* **2023**, *23*, 7145, doi:10.3390/s23167145.
26. Selcuk, B.; Serif, T. A Comparison of YOLOv5 and YOLOv8 in the Context of Mobile UI Detection. In *Mobile Web and Intelligent Information Systems*; Younas, M., Awan, I., Grønli, T.-M., Eds.; Lecture Notes in Computer Science; Springer Nature Switzerland: Cham, 2023; Vol. 13977, pp. 161–174.
27. Bian, H.; Liu, Y.; Shi, L.; Lin, Z.; Huang, M.; Zhang, J.; Weng, G.; Zhang, C.; Gao, M. Detection Method of Helmet Wearing Based on UAV Images and Yolov7. In Proceedings of the 2023 IEEE 6th Information Technology,Networking,Electronic and Automation Control Conference (ITNEC); IEEE: Chongqing, China, February 24 2023; pp. 1633–1640.

28. Mohd Yusof, N. 'Izzaty; Sophian, A.; Mohd Zaki, H.F.; Bawono, A.A.; Embong, A.H.; Ashraf, A. Assessing the Performance of YOLOv5, YOLOv6, and YOLOv7 in Road Defect Detection and Classification: A Comparative Study. *Bull. Electr. Eng. Inform.* **2024**, *13*, 350–360, doi:10.11591/eei.v13i1.6317.
29. Wang, C.-Y.; Yeh, I.-H.; Liao, H.-Y.M. YOLOv9: Learning What You Want to Learn Using Programmable Gradient Information. *ArXiv Prepr. ArXiv240213616* **2024**.
30. Zheng, Y.; Cui, Y.; Gao, X. An Infrared Dim-Small Target Detection Method Based on Improved YOLOv7. In Proceedings of the Proceedings of the 2023 Asia Conference on Computer Vision, Image Processing and Pattern Recognition; ACM: Phuket Thailand, April 28 2023; pp. 1–5.
31. Zhong, J.; Chen, J.; Mian, A. DualConv: Dual Convolutional Kernels for Lightweight Deep Neural Networks. *IEEE Trans. Neural Netw. Learn. Syst.* **2022**.
32. Woo, S.; Park, J.; Lee, J.-Y.; Kweon, I.S. Cbam: Convolutional Block Attention Module. In Proceedings of the Proceedings of the European conference on computer vision (ECCV); 2018; pp. 3–19.
33. Li, Y.; Gong, Z.; Zhou, Y.; He, Y.; Huang, R. Production Evaluation of Citrus Fruits Based on the YOLOv5 Compressed by Knowledge Distillation. In Proceedings of the 2023 26th International Conference on Computer Supported Cooperative Work in Design (CSCWD); IEEE: Rio de Janeiro, Brazil, May 24, 2023; pp. 1938–1943.
34. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 1137–1149, doi:10.1109/TPAMI.2016.2577031.
35. Ge, Z.; Liu, S.; Wang, F.; Li, Z.; Sun, J. Yolox: Exceeding Yolo Series in 2021. *ArXiv Prepr. ArXiv210708430* **2021**.