electronics

Article

Lightweight DB-YOLO Facemask Intelligent Detection and Android Application Based on Bidirectional Weighted Feature Fusion

Bin Qin [1], Ying Zeng [2], Xin Wang [1,*], Junmin Peng [1], Tao Li [3], Teng Wang [4,*] and Yuxin Qin [5]

[1] School of Electrical and Information Engineering, Hunan University of Technology, Zhuzhou 412007, China; qinbin1@hut.edu.cn (B.Q.); 14255@hut.edu.cn (J.P.)
[2] School of Computer Science, Hunan University of Technology, Zhuzhou 412007, China; m21085400026@stu.hut.edu.cn
[3] College of Railway Transportation, Hunan University of Technology, Zhuzhou 412007, China; litao@hut.edu.cn
[4] School of Computer Science, South China Normal University, Guangzhou 510631, China
[5] School of Computer Science, University of Glasgow, Glasgow G12 8QQ, UK; y.qin.1@research.gla.ac.uk
* Correspondence: xinwang@hut.edu.cn (X.W.); 20011039@m.scnu.edu.cn (T.W.)

Abstract: Conventional facemask detection algorithms face challenges of insufficient accuracy, large model size, and slow computation speed, limiting their deployment in real-world scenarios, especially on edge devices. Aiming at addressing these issues, we proposed a DB-YOLO facemask intelligent detection algorithm, which is a lightweight solution that leverages bidirectional weighted feature fusion. Our method is built on the YOLOv5 algorithm model, replacing the original YOLOv5 backbone network with the lightweight ShuffleNetv2 to reduce parameters and computational requirements. Additionally, we integrated BiFPN as the feature fusion layer, enhancing the model's detection capability for objects of various scales. Furthermore, we employed a CARAFE lightweight upsampling factor to improve the model's perception of details and small-sized objects and the EIOU loss function to expedite model convergence. We validated the effectiveness of our proposed method through experiments conducted on the Pascal VOC2007+2012 and Face_Mask datasets. Our experimental results demonstrate that the DB-YOLO model boasts a compact size of approximately 1.92 M. It achieves average precision values of 70.1% and 93.5% on the Pascal VOC2007+2012 and Face_Mask datasets, respectively, showcasing a 2.3% improvement in average precision compared to the original YOLOv5s. Furthermore, the model's size is reduced by 85.8%. We also successfully deployed the model on Android devices using the NCNN framework, achieving a detection speed of up to 33 frames per second. Compared to lightweight algorithm models like YOLOv5n, YOLOv4-Tiny, and YOLOv3-Tiny, DB-YOLO not only reduces the model's size but also effectively improves detection accuracy, exhibiting excellent practicality and promotional value on edge devices.

Keywords: object detection; YOLOv5; lightweight; BiFPN; edge deployment

check for updates

Citation: Qin, B.; Zeng, Y.; Wang, X.; Peng, J.; Li, T.; Wang, T.; Qin, Y. Lightweight DB-YOLO Facemask Intelligent Detection and Android Application Based on Bidirectional Weighted Feature Fusion. *Electronics* **2023**, *12*, 4936. https://doi.org/10.3390/electronics12244936

Academic Editors: Antoine Vacavant, KC Santosh, Aaisha Makkar, Ashutosh Kumar Singh, Anas Abou El Kalam, Mohamed-Rafik Bouguelia, Myra Conway and Ravindra Hegadi

Received: 16 October 2023
Revised: 21 November 2023
Accepted: 6 December 2023
Published: 8 December 2023

1. Introduction

The global impact of the COVID-19 pandemic has been profound, prompting the widespread use of facemasks as effective tools for the prevention and control of this epidemic [1]. As the pandemic fluctuates, facemask detection remains crucial in the efforts to combat its spread. However, traditional manual facemask detection methods suffer from inefficiency, high costs, and susceptibility to subjectivity. Hence, researchers are focusing on automated facemask detection methods using computer vision and deep learning technologies.

Currently, object detection models based on the YOLO (You Only Look Once) series algorithms have been widely applied [2,3]. However, these models face challenges such as low accuracy, large model size, and slow computation speed in practical applications, limiting their deployment in real-world scenarios, especially on edge devices. To address these issues, researchers like Fu [4] and Yu [5] have proposed integrating the CBAM [6] (Convolutional Block Attention Module) attention mechanism and introducing BiFPN [7] (Bidirectional Feature Pyramid Network) to improve model accuracy. Zhang et al. [8] have achieved lightweight effects by using MobileNetv3 [9] as the backbone feature extraction network. Nevertheless, to achieve real-time detection, further model size reduction is required. However, with the development of the YOLO series algorithms, the commonly used YOLOv5 not only has achieved significant improvements in detection accuracy but also has reduced the model size considerably. Moreover, it has been applied to various scenarios [10–13]. Thus, this paper proposes a lightweight facemask object detection algorithm based on DB-YOLO, built upon the YOLOv5 architecture, ensuring high detection accuracy, speed, and significantly reduced model size. The following sections will provide a detailed description of the implementation process and present a series of experiments to validate its effectiveness.

The research achievements of this paper are mainly as follows:

(1) Adoption of ShuffleNetv2 [14] as the YOLOv5 backbone network: ShuffleNetv2 is a lightweight network structure that significantly reduces the number of parameters and computational requirements while maintaining high accuracy.
(2) Introduction of BiFPN as the feature fusion layer: To better integrate multi-scale feature information, we employ BiFPN as the feature fusion layer. Unlike traditional FPN + PAN [15,16] (Feature Pyramid Network + Path Aggregation Network), BiFPN utilizes both top-down and bottom-up paths as a single feature network layer and iteratively repeats the process to achieve higher-level feature fusion. This design effectively enhances the model's detection capability for objects of different scales, improving the precision and robustness of object detection.
(3) Use of CARAFE (Content-Aware Reassembly of Features) [17] as the lightweight upsampling factor in BiFPN: In the feature fusion process, we introduce CARAFE as a lightweight upsampling factor. CARAFE effectively increases the resolution of the feature map, enhancing the model's perception of details and small-sized objects.
(4) Adoption of EIOU [18] (Efficient Intersection Over Union) as the loss function: To further improve the model's performance, we utilize EIOU as the loss function. Compared to the traditional CIOU [19] (Complete Intersection Over Union), the EIOU loss function considers not only the differences in distance and aspect ratio between the target and anchor box centers but also the differences in width and height. By directly minimizing these differences, the EIOU loss function accelerates the model's convergence process and improves its accuracy and stability.

Through experimental evaluation, our model has outperformed other lightweight facemask detection models based on YOLO [20–23] (e.g., YOLOv3/v4-Tiny/v5). The model size is 1.92 M, and the mAP (mean average precision) value reaches 93.5%. Additionally, the real-time detection inference speed on Android devices has reached 33 frames per second, making it suitable for edge computing scenarios, such as real-time facemask detection. This indicates that the proposed method is practical and valuable for intelligent facemask detection and other object detection tasks, especially in the field of real-time object detection [24–26].

2. Introduction of YOLOv5 Algorithm

The YOLOv5 algorithm represents an advancement over previous versions in the YOLO series. It adopts a deeper and wider network architecture, incorporating multiple convolutional layers, pooling layers, and residual connections [27]. This design aims to better capture object features.

Currently, YOLOv5 offers five versions: YOLOv5n, YOLOv5s, YOLOv5m, YOLOv5l, and YOLOv5x. These versions vary in terms of network depth and width, allowing users to select the appropriate one according to their specific needs. Among them, YOLOv5n is the smallest network structure with the fastest speed.

The YOLOv5 network structure in Figure 1.

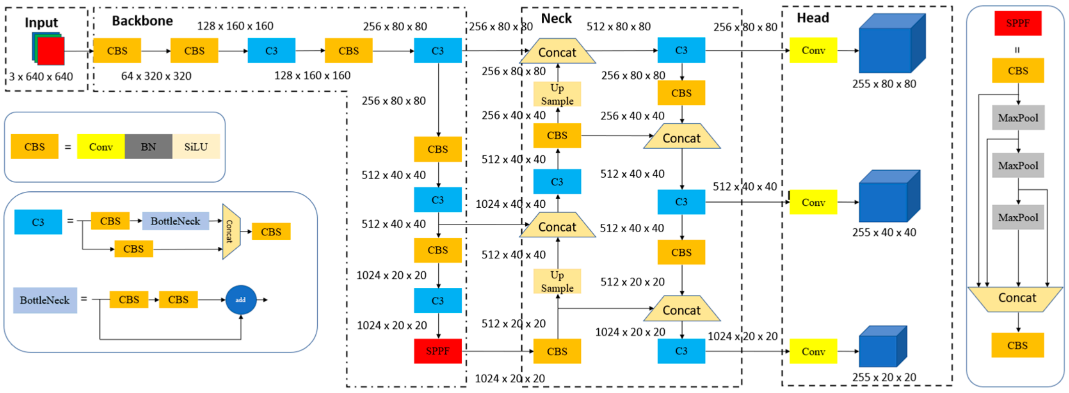


Figure 1. YOLOv5 network structure.

The YOLOv5 network architecture [28] comprises four key components: Input, Backbone, Neck, and Head.

(1) The input component includes image data augmentation, image size processing, and adaptive anchor box calculation:

- Image Data Augmentation [29]: During the training process, data augmentation techniques are applied, including random operations such as rotation, scaling, and flipping, to augment the training data and enhance the model's robustness.
- Image Size Processing: Adjustment of input image sizes to ensure they meet the model's input dimension requirements.
- Adaptive Anchor Box Calculation: YOLOv5 dynamically calculates anchor boxes suitable for object detection based on the training dataset, adapting to different sizes and proportions of targets.

(2) Backbone Network:

A moderately deep and wide backbone network, incorporating multiple convolutional layers, pooling layers, and residual connections employed in YOLOv5. This design aims to better capture target features, and the backbone network is responsible for extracting high-level feature representations from input images.

(3) Feature Pyramid [15]:

The feature pyramid is a pyramid structure composed of feature maps at different levels, facilitating object detection with varying sizes and proportions. By integrating features from different levels of the backbone network, the model's perception of targets is enhanced.

(4) Head component handles box prediction generation and box filtering, as follows:

- Box Prediction Generation: The output component generates predictions for target boxes. YOLOv5 utilizes anchor boxes and convolutional layers to generate coordinate information, class probabilities, and confidence scores for the predicted target boxes.
- Box Filtering: The model filters out target boxes with confidence scores below a set threshold through a series of filtering and thresholding processes, re-training target predictions with high confidence.

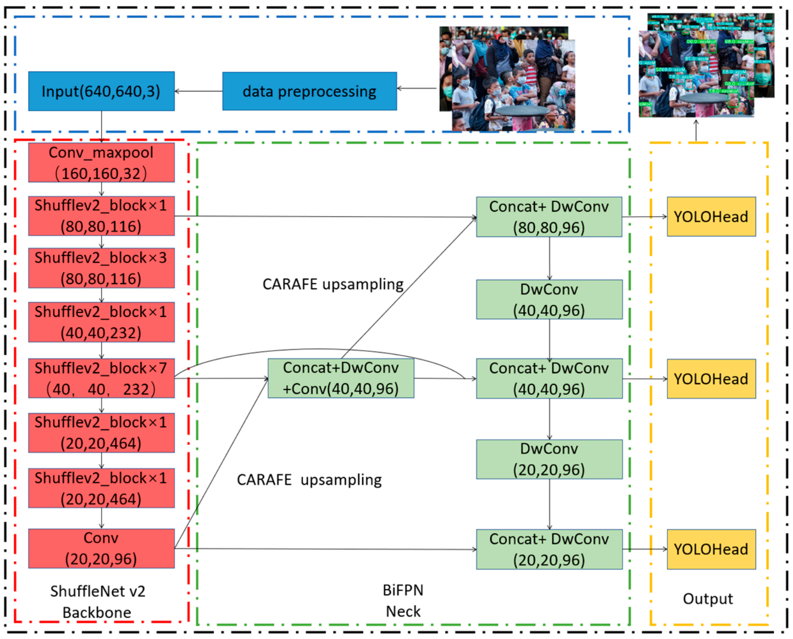
This comprehensive architecture enables YOLOv5 to effectively process input data, extract relevant features through the backbone and feature pyramid, and generate accurate predictions through the output component, ultimately enhancing its object detection capabilities.

Compared to YOLOv4 [30] and YOLOv3 [31], YOLOv5 has undergone significant improvements in its network architecture. These enhancements have resulted in substantial performance gains in terms of speed and accuracy. With its advantages in speed, accuracy, and ease of use, YOLOv5 has become one of the leading algorithms in the field of object detection.

3. Design of DB-YOLO Network Architecture

To efficiently perform object detection tasks in resource-constrained environments, the DB-YOLO network architecture incorporates a lightweight ShuffleNetv2 as the backbone network, BiFPN as the feature fusion layer, and EIOU as the loss function. This network design combines a lightweight model structure, multi-scale feature fusion, and an improved loss function to provide an effective solution for object detection with smaller computational and storage requirements.

The design of the DB-YOLO network architecture is illustrated in Figure 2.



Input(640,640,3)
data preprocessing
Conv_maxpool (160,160,32)
Shufflev2_block×1 (80,80,116)
Shufflev2_block×3 (80,80,116)
Shufflev2_block×1 (40,40,232)
Shufflev2_block×7 (40, 40, 232)
Shufflev2_block×1 (20,20,464)
Shufflev2_block×1 (20,20,464)
Conv (20,20,96)
ShuffleNet v2 Backbone
CARAFE upsampling
Concat+DwConv +Conv(40,40,96)
CARAFE upsampling
Concat+ DwConv (80,80,96)
DwConv (40,40,96)
Concat+ DwConv (40,40,96)
DwConv (20,20,96)
Concat+ DwConv (20,20,96)
BiFPN Neck
YOLOHead
YOLOHead
YOLOHead
Output

Figure 2. The structure diagram of DB-YOLO network model.

3.1. ShuffleNetv2 Backbone Network

To address the resource limitations of edge devices, we use the lightweight ShuffleNetv2 as the backbone network in the YOLOv5 architecture. This choice aims to minimize the model size and the number of parameters while maintaining high detection accuracy.

The ShuffleNetv2 network comprises two types of blocks: the pointwise convolution block and the depthwise separable convolution block. The pointwise convolution block handles dimension reduction and expansion using a 1×1 convolution, while the depthwise separable convolution block is responsible for feature extraction and integration. Furthermore, ShuffleNetv2 introduces a special operation known as Channel Shuffle, which rearranges the channels in the feature map to enhance feature interaction.

The Channel Shuffle operation proceeds as follows: the input feature map is divided into several subsets based on channels, and then the channels within each subset are recombined to form a new feature map. This Channel Shuffle operation endows ShuffleNetv2 with highly parallel characteristics, effectively reducing the computational workload.

The basic unit structure of ShuffleNetv2 [14] is illustrated in Figure 3.

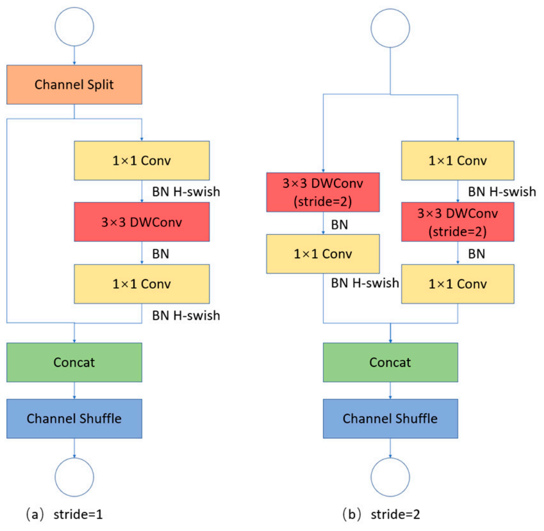


Figure 3. ShuffleNetv2 cell structure.

In summary, ShuffleNetv2 incorporates channel shuffling, a process that divides the input feature map into subsets and then recombines them through channel reordering operations. This enhances the interaction between different channels in the feature map, resulting in excellent feature extraction capabilities and computational efficiency. As a result, the network achieves high detection accuracy even with limited resources.

3.2. Depthwise Separable Convolution

Depthwise separable convolution [32] is a specific type of convolutional operation. The depthwise separable convolution block consists of a 3 × 3 depthwise convolution followed by a 1 × 1 pointwise convolution. Its primary purpose is to reduce the number of parameters and the computational complexity of the model while preserving effective feature extraction capabilities.

Depthwise convolution involves performing separate convolutions on each input channel, with each channel convolved using a dedicated kernel. This enables channel-wise feature extraction without increasing the model's parameter count. The number of convolutional kernels in the depthwise convolution equals the number of input channels.

Pointwise convolution, also known as a 1 × 1 convolution, follows the depthwise convolution. It employs a 1 × 1 kernel to perform convolution on the feature map obtained from the depthwise convolution. Pointwise convolution linearly combines the feature maps, reducing the number of channels to a smaller size. It can be seen as a weighted linear combination of features from each channel to create a new feature representation.

A comparison between regular convolution and depthwise separable convolution is depicted in Figure 4.

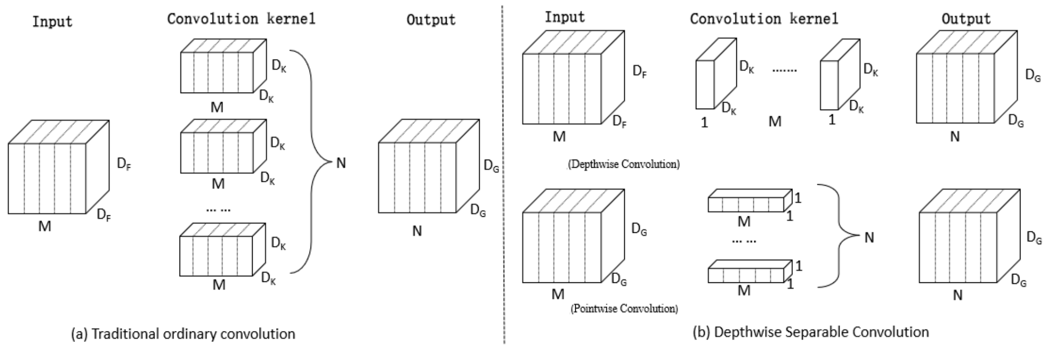


Figure 4. Comparison of convolutions.

We assume that the input feature dimension is $D_F \times D_F \times M$, with D_K as the size of the convolution kernel, M as the number of input channels, N as the number of output channels, and G as the number of groups.

In a regular convolution, the number of parameters is $D_K \times D_K \times M \times N$. However, in the depthwise convolution and pointwise convolution, the number of parameters is $D_K \times D_K \times 1 \times M$ and $1 \times 1 \times M \times N$, respectively. Therefore, the number of parameters in the depthwise separable convolution is $D_K \times D_K \times M + M \times N$. The parameter ratio between the depthwise separable convolution and standard convolution is given by:

$$\frac{D_K \times D_K \times M + M \times N}{D_K \times D_K \times M \times N} = \frac{1}{N} + \frac{1}{D_k^2} \tag{1}$$

3.3. H-Swish Activation Function

To enhance the neural network's accuracy while reducing computational costs, we replace the ReLU activation function with the H-swish activation function [33]. Derived from the swish function [34], the H-swish function offers improved performance. Compared with ReLU, the H-swish function maintains accuracy while reducing the computational load and memory usage, thus boosting the neural network's efficiency and speed.

The formula for the H-swish activation function is as follows:

$$H - \mathrm{swish}(x) = x\frac{\mathrm{ReLU6}(x+3)}{6} \tag{2}$$

The function image is depicted in Figure 5 below.

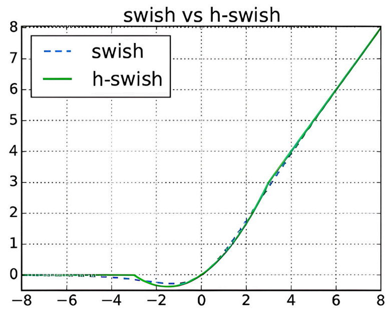


Figure 5. H-swish function image.

4. BiFPN Feature Fusion

The feature fusion layer in DB-YOLO employs BiFPN [7], which provides several advantages over the traditional FPN + PAN [15,16] approach:

(1) Improved Information Exchange: BiFPN incorporates bidirectional connections and weighted fusion mechanisms, allowing for better information exchange and fusion across different feature levels. By adaptively learning the importance weights of features, BiFPN enables mutual influence and complementarity among different levels, thereby enhancing feature representation and information propagation.

(2) Reduced Information Loss: The traditional FPN + PAN approach may lead to information loss or redundancy during the feature fusion process. In contrast, BiFPN, with its bidirectional connections, preserves more fine-grained feature information, avoiding information loss and contributing to an improved performance and accuracy of the detection model.

(3) Higher Accuracy and Stability: BiFPN employs an adaptive weighted fusion mechanism during feature fusion, dynamically adjusting based on the quality and importance of features. This allows high-quality and important features to have a greater impact on the results of the final detection, leading to a higher accuracy and stability of the model.

The inclusion of BiFPN helps enhance the accuracy and stability of the detection model, particularly in scenarios with significant variations in target scales or complex scenes.

The structures of FPN, PAN, and BiFPN for feature fusion [7] are illustrated in Figure 6.

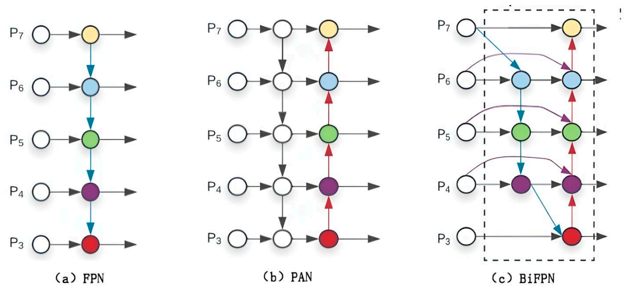


Figure 6. Feature Fusion.

4.1. Lightweight Upsampling with CARAFE

Upsampling is a commonly used operation in various network architectures. Here, we employ a lightweight and versatile upsampling operator called CARAFE [17]. Compared to traditional upsampling operators like nearest-neighbor and bilinear interpolation, CARAFE has demonstrated significant improvements in the detection process while introducing minimal parameters and computational cost.

The CARAFE algorithm consists of two main modules: the upsampling kernel prediction module and the feature reassembly module.

The upsampling kernel prediction module generates an upsampling kernel for each pixel position. It takes the input feature map as the input and uses a convolutional neural network structure to output an upsampling kernel of the same size as the output feature map. The generation of the upsampling kernel is dynamically adjusted based on the content of the input feature map, with each pixel position having a corresponding upsampling kernel. These upsampling kernels can be considered as weights that define the relationship between pixels in the input feature map and pixels in the output feature map.

The feature reassembly module utilizes the upsampling kernels to reassemble the input feature map. It adopts the concept of spatial transformation networks and performs transformation and reassembly operations on the input feature map based on the upsampling kernels. The feature reassembly module combines each pixel in the input feature map with its surrounding pixels based on the weights in the upsampling kernels to generate the pixel values in the output feature map. The weights in the upsampling kernels determine the contribution of each input pixel to the output pixel.

The implementation process of CARAFE upsampling is illustrated [17] in Figure 7.

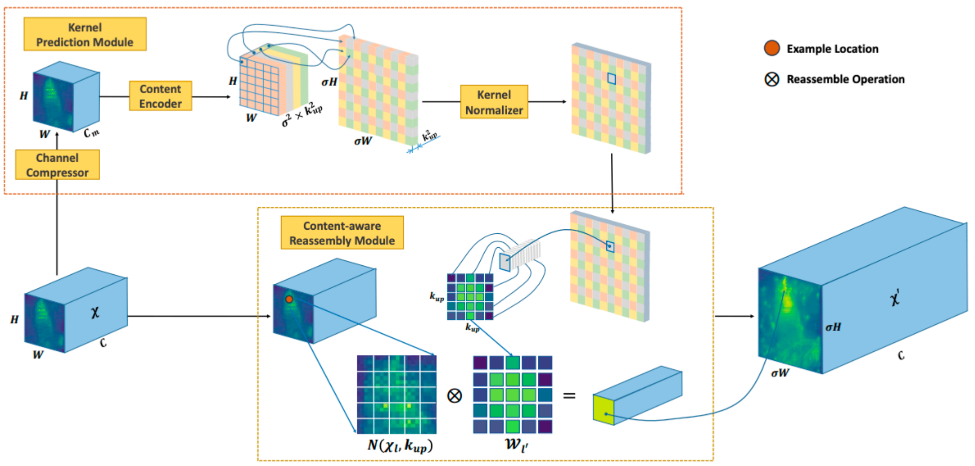


Figure 7. CARAFE upsampling process.

One of the advantages of CARAFE is its ability to adaptively adjust the upsampling operation based on the content of the input feature map, avoiding the introduction of artifacts and blurring effects that may occur in traditional upsampling algorithms. Additionally, CARAFE introduces minimal parameters and computational costs, ensuring high-quality upsampling without significantly increasing the model size and computational burden.

4.2. EIOU Loss Function

This study adopts the EIOU [18] loss function, which builds upon the CIOU [19] loss function and introduces improvements by incorporating separate impact factors for the width and height calculations of the target and anchor boxes. The EIOU loss function consists of three components: overlap loss (L_{IOU}), center distance loss (L_{dis}), and aspect ratio loss (L_{asp}).

In the overlap loss and center distance loss components, EIOU follows the methodology of CIOU, considering the overlap between the target and anchor boxes and the distance between their centers, providing a comprehensive evaluation of their similarity and matching.

In the aspect ratio loss component, EIOU adopts a direct minimization approach for the differences in width and height between the target and anchor boxes. By considering the width and height differences as independent loss terms, EIOU achieves a faster convergence and more efficient optimization of the model's performance.

The formula for the CIOU loss function is as follows:

$$L_{CIOU} = 1 - IOU + \frac{\rho^2(b, b^{gt})}{c^2} + \alpha v \tag{3}$$

$$\alpha = \frac{v}{(1 - IOU) + v} \tag{4}$$

$$v = \frac{4}{\pi^2}\left(\arctan\frac{w^{gt}}{h^{gt}} - \arctan\frac{w}{h}\right)^2 \tag{5}$$

The formula for the EIOU loss function is as follows:

$$L_{EIOU} = L_{IOU} + L_{dis} + L_{asp} = 1 - IOU + \frac{\rho^2(b, b^{gt})}{c^2} + \frac{\rho^2(w, w^{gt})}{c_w^2} + \frac{\rho^2(h, h^{gt})}{c_h^2} \tag{6}$$

In the EIOU loss function, "b" and "b^{gt}" represent the center points of the predicted bounding box and the ground truth bounding box, respectively. "$\rho^2()$" represents the Euclidean distance, "w, w^{gt}" and "h, h^{gt}" represent the width and height of the predicted bounding box and the ground truth bounding box, respectively, and "c" is the length of the diagonal of the smallest outer frame covering the predicted bounding box and the ground truth bounding box.

5. Experimental Results and Analysis

5.1. Experimental Datasets

To validate the effectiveness of the proposed model, we conducted experiments using two datasets: the Pascal VOC2007+2012 dataset [35] for ablation experiments and the Face_Mask dataset for comparative experiments. The Face_Mask dataset [36] consists of 7959 images collected from publicly available datasets, such as MAFA [37] and Wilder-Face [38], as well as 1137 images obtained from the Internet. These images contain both masked and unmasked faces. Before training the algorithm model with this dataset, we performed data filtering to remove images that did not correspond to the labels and features. After processing, a total of 9096 images were retained. To enhance the robustness of the designed object detection model to images obtained from different environments, we subjected the initial dataset to various data augmentation techniques, such as flipping, shift, rotation, mix-up [39], and mosaic. Subsequently, we used the dataset for training and evaluation, randomly splitting it into training and testing sets with an 8:2 ratio. The labels for Class 1 correspond to "Mask", while Class 0 corresponds to "NoMask". The dataset categories are shown in Table 1.

Table 1. Data categories.

	Face_Mask	**Pascal VOC2007+2012**
Train	7277	16,551
Test	1819	4952
Category	2	20

5.2. Experimental Environment

The experiments were conducted on a computer running the Windows 10 operating system. We used Python 3.8 as the language environment. The model was built using the PyTorch deep learning framework, and the required libraries were installed according to the environment requirements. The computer was equipped with an Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.4 GHz processor with two processors and 64 GB of memory. GPU acceleration training was performed using an NVIDIA Tesla M40 graphics card with 24 GB of memory.

5.3. Evaluation Metrics

To assess the model's performance and accuracy in image detection, we utilized various evaluation metrics, including mAP, precision, recall, number of parameters, weight file size, and inference time.

Precision measures the ratio of true positive predictions to the total number of predicted samples, while recall measures the ratio of true positive predictions to the total number of actual positive samples.

The formulas for calculating precision and recall are shown in Equations (7) and (8). The mAP calculates the average precision across all classes and is determined by Equations (9) and (10).

$$\mathrm{Precision} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \tag{7}$$

$$\mathrm{Recall} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{8}$$

$$\mathrm{AP} = \int_0^1 \mathrm{P}(\mathrm{R})\mathrm{dR} \tag{9}$$

$$\mathrm{mAP} = \frac{1}{\mathrm{m}} \sum_{\mathrm{i}=1}^{\mathrm{m}} \mathrm{AP}_{\mathrm{i}} \tag{10}$$

Here, TP represents the number of true positive detections, FP represents the number of false positive predictions, FN represents the number of false negatives, AP represents the average precision for each class, and mAP represents the mean average precision across all classes; "mAP50" represents the mAP when the IOU (Intersection Over Union) threshold is 50% and "mAP50:90" represents mAP when the IOU threshold varies from 50% to 90%. In object detection tasks, IOU thresholds are commonly used to measure the overlap between the predicted and the real bounding boxes. A threshold of 50% indicates that a predicted bounding box is considered correct if its overlap with the real bounding box exceeds 50%.

By employing these evaluation metrics, we comprehensively assess the model's performance in image detection tasks. These metrics allow us to better understand and compare the accuracy, efficiency, and resource consumption of different models, providing valuable insights for model selection and optimization.

5.4. Ablation Experiment

To validate the effectiveness and robustness of the proposed algorithm, we conducted ablation experiments on the Pascal VOC2007+2012 dataset. These experiments involve removing or replacing key components in the algorithm to observe how these changes impact algorithm performance and analyze the significance of each component for object detection. The results of the ablation experiments are presented in Table 2.

Table 2. Results of ablation experiment.

Backbone	Neck	Loss Function	Upsampling	Parameters	Size	mAP50	Inference Time
ShuffleNetv2	FPN + PAN	CIOU	Nearest	0.70 M	1.72 MB	66.1%	4.2 ms
ShuffleNetv2	FPN + PAN	EIOU	Nearest	0.70 M	1.72 MB	66.4%	4.2 ms
ShuffleNetv2	BiFPN	CIOU	Nearest	0.74 M	1.80 MB	67.8%	4.4 ms
ShuffleNetv2	BiFPN	EIOU	Nearest	0.74 M	1.80 MB	68.3%	4.4 ms
ShuffleNetv2	BiFPN	EIOU	CARAFE	0.85 M	1.92 MB	70.1%	6.4 ms

Based on the experimental results, the combination of ShuffleNetv2-BiFPN-EIOU-CARAFE (DB-YOLO) outperforms previous combinations in terms of performance. By improving feature fusion, the loss function, and the upsampling method, this combination achieves significant improvements in accuracy, robustness, and object perception capability. As a result, it can more precisely localize and recognize objects. Although there is a slight increase in weight size and inference time, this is achieved with minimal resource sacrifice. Therefore, DB-YOLO maintains high efficiency and practicality while enhancing overall performance.

Figure 8 displays the P-R (precision–recall) curve of DB-YOLO on the VOC2007+2012 dataset. On the P-R curve, the larger the AUC (area under the curve) for a class, the higher its mAP50.

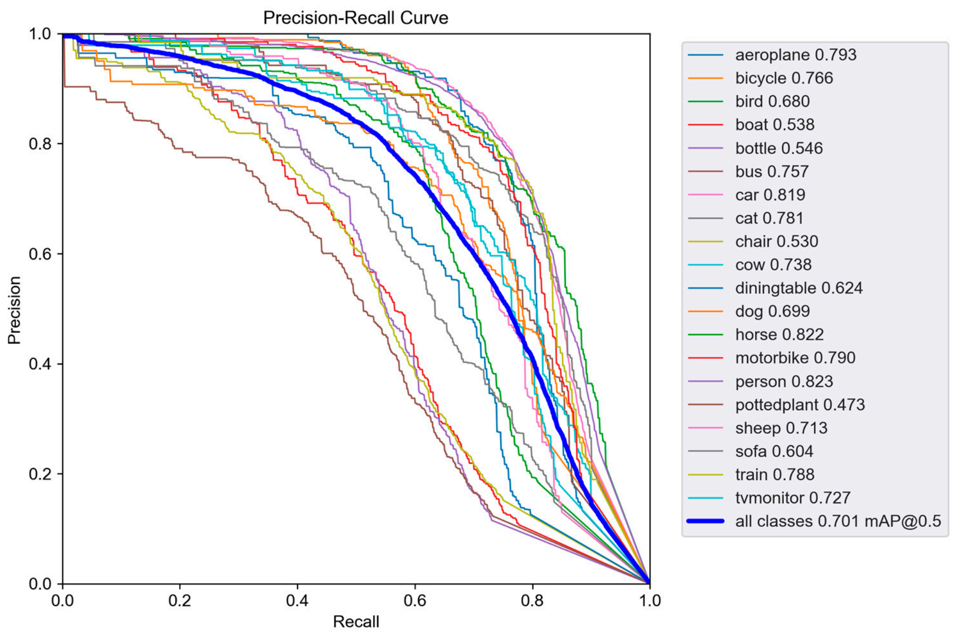


Figure 8. Precision–recall curve.

As shown in Figure 8, the performance of the DB-YOLO model across 20 categories in the VOC2007+2012 dataset can be observed. Among them, the "person" category has the highest mAP50 value of 82.3%, while the "pottedplant" category has the lowest mAP value of 47.3%. The average mAP50 value across all 20 categories is 70.1%, indicating that DB-YOLO demonstrates excellent detection performance.

5.5. Contrast Experiment

To further validate the effectiveness and reliability of our proposed method, we compared it with several commonly used lightweight object detection models. These models were trained and tested on the same Face_Mask dataset, and their performance in terms of detection accuracy, speed, and model size was evaluated. The detection results are summarized in Table 3.

Table 3. Comparative experimental results.

Name	Parameters	Size	Precision	Recall	mAP50	mAP50:95	Inference Time
YOLO-Fastest	0.28 M	1.15 MB	78.7%	89.3%	88.8%	57.3%	5.8 ms
YOLOv3-Tiny	8.27 M	33.1 MB	72.9%	92.4%	92.2%	62.5%	3.5 ms
YOLOv4-Tiny	5.60 M	22.4 MB	76.2%	92.8%	92.9%	64.5%	4.5 ms
YOLOv5s	6.69 M	13.6 MB	92.2%	85.9%	91.2%	62.9%	6.4 ms
YOLOv5n	1.68 M	3.5 MB	92.0%	81.6%	88.5%	59.4%	4.2 ms
DB-YOLO (ours)	0.85 M	1.92 MB	93.8%	89.0%	93.5%	66.4%	6.4 ms

Comparing the experimental results of different algorithms, we observed that the DB-YOLO method achieves higher precision and recall while reducing the number of parameters and model size. The mAP value reaches 93.5%, indicating that the model performs well in object detection tasks. Moreover, the model's inference time is 6.4 ms, making it suitable for real-time scenarios in facemask detection applications. These results demonstrate that DB-YOLO exhibits higher efficiency and accuracy, making it suitable for widespread use in various application scenarios.

Figure 9 illustrates the training results of DB-YOLO in Face_Mask dataset.

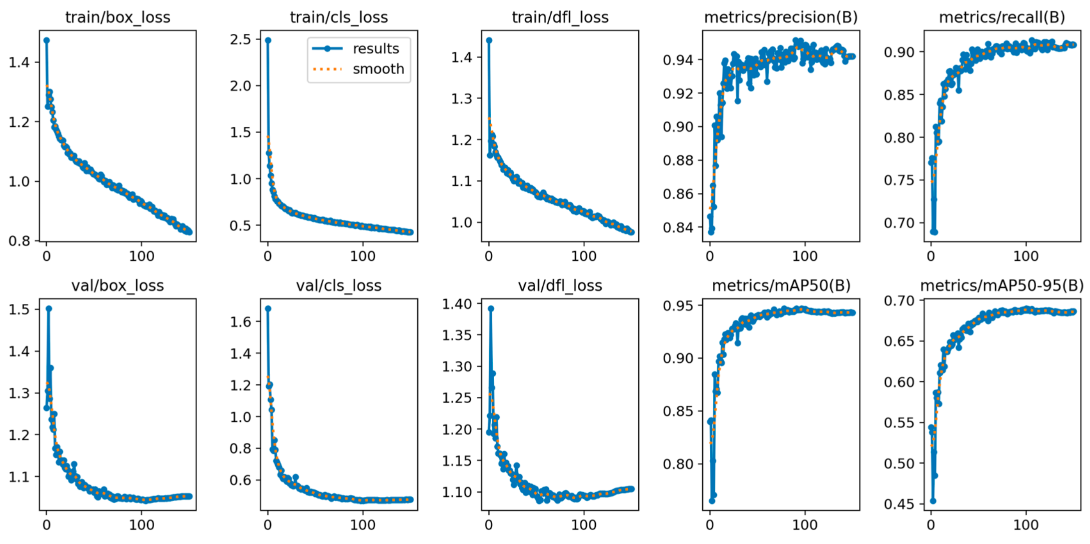


Figure 9. Training results.

As shown in Figure 9, the DB-YOLO model was trained for 120 epochs and the loss functions for both the training set and testing set rapidly decrease, stabilize, and eventually levels off with the increase of training epochs. Precision, recall, mAP50, and mAP50:90 also show a rapid increase, stabilize, and eventually levels off. Its mAP50 value stabilizes around 93.5%. This indicates that the DB-YOLO model performs well during the training process and achieves the expected performance.

6. Android Device Application Deployment

6.1. Deployment Process

To deploy the trained object detection model on an Android [40] device for testing in edge computing environments, we followed a series of steps as described below.

(1) Export the Trained Model to ONNX (Open Neural Network Exchange) [41]: The optimized object detection model is exported as an ONNX file to ensure its portability and cross-platform usability. This process involves saving the model's weights and structural information in the ONNX format, making it compatible with edge devices.

(2) Selection of NCNN [42] precision–recall framework: For deployment on Android devices, we chose the NCNN framework. NCNN is a high-performance neural network computing framework specifically designed for mobile and embedded devices. We utilized NCNN to load the exported ONNX file onto the Android device and leverage its capabilities to compile and optimize the model. This approach fully utilizes the device's processor architecture and computational power, enhancing the model's runtime efficiency and performance.

The deployment process is illustrated in Figure 10, where the exported ONNX model is loaded and optimized using the NCNN framework on the Android device.

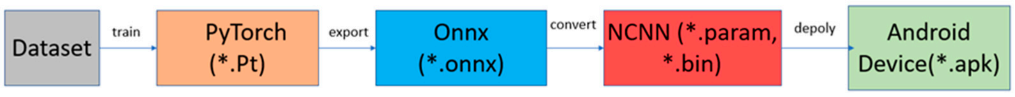


Figure 10. Deployment flowchart.

6.2. Testing Environment

During the testing phase, we evaluated the model's performance on a Xiaomi Mi 10 Pro device. The Mi 10 Pro is equipped with a powerful Qualcomm Snapdragon 865 processor and runs on Android 13, providing excellent computational capabilities and performance.

6.3. Application Results

We conducted comparative testing experiments using the following three models with the Android application: YOLOv5s, YOLOv5n, and DB-YOLO. The image input sizes were set to 320 and 640, and the evaluation metric was solely measured in FPS (frames per second) to assess the deployment performance on the Android device.

To assess the model's detection accuracy under various scenarios, we tested it with images containing various situations, such as no facemask, partially covered facemask, incorrect facemask wearing, and correct facemask wearing for real-time facemask detection.

The Android test results are as shown in Table 4.

Table 4. Android test results.

Input	Name	FPS
640 × 640	YOLOv5s	13
640 × 640	YOLOv5n	16
640 × 640	DB-YOLO	18
320 × 320	YOLOv5s	28
320 × 320	YOLOv5n	31
320 × 320	DB-YOLO	33

As shown in the test results from Table 4, the overall deployment performance of DB-YOLO on Android devices is superior to those of other models. However, if the detection input size is reduced for higher real-time performance, the detection accuracy may be compromised.

The Android application detection results for DB-YOLO are presented in Figure 11.

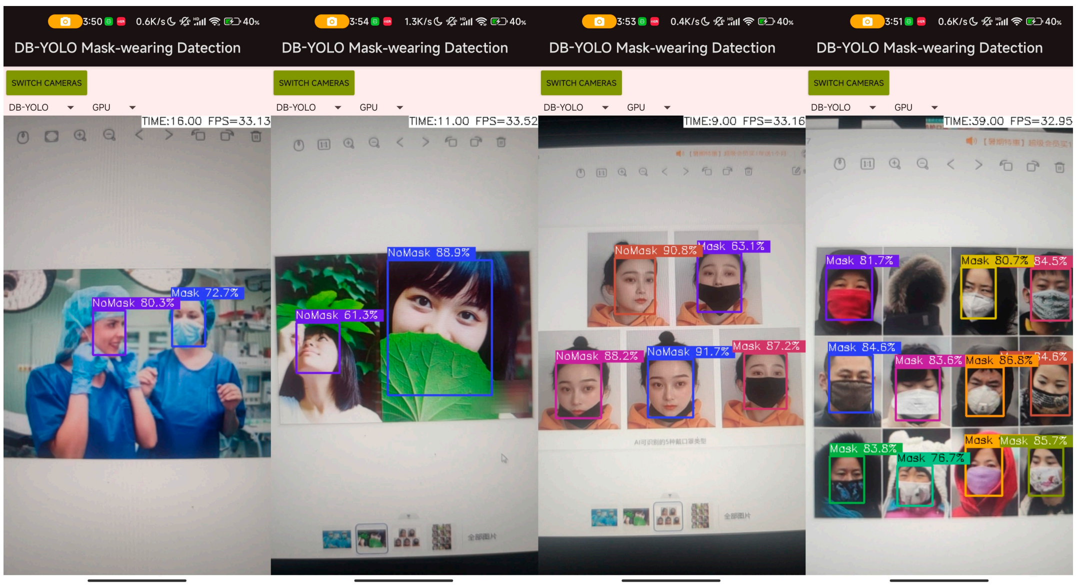


Figure 11. Real-time detection of Android application.

From the detection result images, it is evident that the DB-YOLO model maintains a high level of accuracy even in cases of face occlusion or improper facemask wearing. This demonstrates the robustness and adaptability of the model, enabling it to handle complex real-world scenarios. Additionally, the model achieves an average frame rate of approximately 33 FPS during runtime, showcasing its efficient performance on edge computing devices and providing strong support for practical applications.

7. Conclusions

In response to the challenges faced by traditional object detection models, we propose a lightweight DB-YOLO intelligent facemask detection algorithm based on bidirectional weighted feature fusion (DB-YOLO). This method effectively addresses issues related to accuracy and model size in detection algorithms. By incorporating the lightweight ShuffleNetv2 as the backbone network, utilizing BiFPN for feature fusion, employing CARAFE for lightweight upsampling, and using EIOU as the algorithm's loss function to expedite model convergence, DB-YOLO significantly improves the efficiency and detection accuracy of the model.

Experimental evaluations on the Pascal VOC2007+2012 and Face_Mask datasets validate the outstanding performance of the DB-YOLO model. The model occupies only approximately 1.92 M of storage space and achieves average precision values of 70.1% and 93.5% on the Pascal VOC2007+2012 and Face_Mask datasets, respectively. It has been successfully deployed on Android devices, achieving a real-time detection speed of up to 33 frames per second. Compared to other lightweight models, such as YOLOv5s, YOLOv5n, YOLOv4-tiny, and YOLOv3-tiny, DB-YOLO demonstrates significant reductions in model size while effectively improving its detection accuracy. Furthermore, it showcases excellent practicality and wide application potential on edge devices, providing valuable insights and inspirations for facemask detection and other object detection tasks.

Author Contributions: B.Q. contributed to the design of the entire system solution, providing ideas and methods, and to the improvement of the neural network algorithm and software design; Y.Z. contributed to data collection and software design debugging, Android model testing, and writing—initial draft preparation; X.W. contributed to data analysis, supervision, writing—review and editing, and procurement of research funding; J.P. contributed to validation and procurement of research funding; T.L. contributed to visualization and writing—review and editing; T.W. contributed to validation and writing—review and editing; Y.Q. contributed to validation, data processing,

and writing—review and editing. All authors have read and agreed to the published version of the manuscript.

Funding: Financial support was provided in part by the National Natural Science Foundation of China (grant numbers 62373142, 62033014, and 61903136) and the Natural Science Foundation of Hunan Province (grant numbers 2021JJ50006 and 2022JJ50074), Hunan Engineering Research Center of Electric Drive and Regenerative Energy Storage and Utilization.

Data Availability Statement: The Pascal VOC2007+2012 and Face_mask datasets used in this paper are open, and can be downloaded from the Internet. Pascal VOC2007+2012 (http://host.robots.ox.ac.uk/pascal/VOC/ (accessed on 5 December 2023)); Face_mask (https://github.com/waittim/mask-detector/tree/master/modeling/data/images (accessed on 5 December 2023)).

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

References

1. Chan, J.F.W.; Yuan, S.; Zhang, A.J.; Poon, V.K.M.; Chan, C.C.S.; Lee, A.C.Y.; Fan, Z.; Li, C.; Liang, R.; Cao, J.; et al. Surgical mask partition reduces the risk of noncontact transmission in a golden Syrian hamster model for coronavirus disease 2019 (COVID-19). *Clin. Infect. Dis.* **2020**, *71*, 2139–2149. [CrossRef]
2. Jiang, P.; Ergu, D.; Liu, F.; Cai, Y.; Ma, B. A Review of Yolo algorithm developments. *Procedia Comput. Sci.* **2022**, *199*, 1066–1073. [CrossRef]
3. Diwan, T.; Anirudh, G.; Tembhurne, J.V. Object detection using YOLO: Challenges, architectural successors, datasets and applications. *Multimed. Tools Appl.* **2023**, *82*, 9243–9275. [CrossRef] [PubMed]
4. Fu, H.; Song, G.; Wang, Y. Improved YOLOv4 marine target detection combined with CBAM. *Symmetry* **2021**, *13*, 623. [CrossRef]
5. Guo, Y.; Chen, S.; Zhan, R.; Wang, W.; Zhang, J. Sar ship detection based on yolov5 using cbam and bifpn. In Proceedings of the IGARSS 2022-2022 IEEE International Geoscience and Remote Sensing Symposium, Kuala Lumpur, Malaysia, 17–22 July 2022; pp. 2147–2150.
6. Woo, S.; Park, J.; Lee, J.Y.; Kweon, I.S. Cbam: Convolutional block attention module. In Proceedings of the European Conference on Computer Vision (ECCV), Glasgow, UK, 23–28 August 2018; pp. 3–19.
7. Tan, M.; Pang, R.; Le, Q.V. Efficientdet: Scalable and efficient object detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 10781–10790.
8. Zhang, X.; Li, N.; Zhang, R. An improved lightweight network MobileNetv3 Based YOLOv3 for pedestrian detection. In Proceedings of the 2021 IEEE International Conference on Consumer Electronics and Computer Engineering (ICCECE), Guangzhou, China, 15–17 January 2021; pp. 114–118.
9. Howard, A.; Sandler, M.; Chu, G.; Chen, L.C.; Chen, B.; Tan, M.; Wang, W.; Zhu, Y.; Pang, R.; Vasudevan, V.; et al. Searching for mobilenetv3. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Republic of Korea, 27 October–2 November 2019; pp. 1314–1324.
10. Du, X.; Song, L.; Lv, Y.; Qiu, S. A lightweight military target detection algorithm based on improved YOLOv5. *Electronics* **2022**, *11*, 3263. [CrossRef]
11. Wang, T.; Su, J.; Xu, C.; Zhang, Y. An intelligent method for detecting surface defects in aluminium profiles based on the improved YOLOv5 algorithm. *Electronics* **2022**, *11*, 2304. [CrossRef]
12. Hu, C.; Min, S.; Liu, X.; Zhou, X.; Zhang, H. Research on an Improved Detection Algorithm Based on YOLOv5s for Power Line Self-Exploding Insu-lators. *Electronics* **2023**, *12*, 3675. [CrossRef]
13. Liu, J.; Cai, Q.; Zou, F.; Zhu, Y.; Liao, L.; Guo, F. BiGA-YOLO: A Lightweight Object Detection Network Based on YOLOv5 for Autonomous Driving. *Electronics* **2023**, *12*, 2745. [CrossRef]
14. Ma, N.; Zhang, X.; Zheng, H.T.; Sun, J. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 116–131.
15. Lin, T.Y.; Dollár, P.; Girshick, R.; He, K.; Hariharan, B.; Belongie, S. Feature pyramid networks for object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 2117–2125.
16. Liu, S.; Qi, L.; Qin, H.; Shi, J.; Jia, J. Path aggregation network for instance segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–236 July 2018; pp. 8759–8768.
17. Wang, J.; Chen, K.; Xu, R.; Liu, Z.; Loy, C.C.; Lin, D. Carafe: Content-aware reassembly of features. In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Republic of Korea, 27 October–2 November 2019; pp. 3007–3016.
18. Zhang, Y.-F.; Ren, W.; Zhang, Z.; Jia, Z.; Wang, L.; Tan, T. Focal and efficient IOU loss for accurate bounding box regression. *Neurocomputing* **2022**, *506*, 146–157. [CrossRef]
19. Zheng, Z.; Wang, P.; Liu, W.; Li, J.; Ye, R.; Ren, D. Distance-IoU loss: Faster and better learning for bounding box regression. In Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020; Volume 34, pp. 12993–13000.

20. Wang, T.; Wang, X.; Shi, W.; Han, Y.Y.; Dong, J.W. Lightweight Real-Time Mask-wearing Detection Algorithm Based on YOLOv3. In Proceedings of the 2021 5th International Conference on Automation, Control and Robots (ICACR), Nanning, China, 25–27 September 2021; pp. 1–6.
21. Zhu, J.; Wang, J.L.; Wang, B. Lightweight mask detection algorithm based on improved YOLOv4-tiny. *Chin. J. Liq. Cryst. Disp.* **2021**, *36*, 1525–1534. [CrossRef]
22. Xu, S.; Guo, Z.; Liu, Y.; Fan, J.; Liu, X. An improved lightweight yolov5 model based on attention mechanism for face mask detection. In *International Conference on Artificial Neural Networks*; Springer Nature: Cham, Switzerland, 2022; pp. 531–543.
23. Yang, G.; Feng, W.; Jin, J.; Lei, Q.; Li, X.; Gui, G.; Wang, W. Face mask recognition system with YOLOV5 based on image recognition. In Proceedings of the 2020 IEEE 6th International Conference on Computer and Communications (ICCC), Chengdu, China, 11–14 December 2020; pp. 1398–1404.
24. Xiang, X.; Meng, F.; Lv, N.; Yin, H. Engineering Vehicles Detection for Warehouse Surveillance System Based on Modified YOLOv4-Tiny. *Neural Process. Lett.* **2023**, *55*, 2743–2759. [CrossRef]
25. Ayachi, R.; Said, Y.; Ben Abdelaali, A. Pedestrian detection based on light-weighted separable convolution for advanced driver assistance systems. *Neural Process. Lett.* **2020**, *52*, 2655–2668. [CrossRef]
26. Ayachi, R.; Afif, M.; Said, Y.; Atri, M. Traffic signs detection for real-world application of an advanced driving assisting system using deep learning. *Neural Process. Lett.* **2020**, *51*, 837–851. [CrossRef]
27. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
28. GitHub. YOLOV5-Master. 2021. Available online: https://github.com/ultralytics/yolov5/ (accessed on 8 November 2023).
29. Shorten, C.; Khoshgoftaar, T.M. A survey on image data augmentation for deep learning. *J. Big Data* **2019**, *6*, 1–48. [CrossRef]
30. Bochkovskiy, A.; Wang, C.Y.; Liao, H.Y.M. Yolov4: Optimal speed and accuracy of object detection. *arXiv* **2020**, arXiv:2004.10934.
31. Redmon, J.; Farhadi, A. Yolov3: An incremental improvement. *arXiv* **2018**, arXiv:1804.02767.
32. Chollet, F. Xception: Deep learning with depthwise separable convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 1251–1258.
33. Avenash, R.; Viswanath, P. Semantic Segmentation of Satellite Images using a Modified CNN with Hard-Swish Activation Function. In Proceedings of the VISIGRAPP (4: VISAPP), Prague, Czech Republic, 25–27 February 2019; pp. 413–420.
34. Ramachandran, P.; Zoph, B.; Le, Q.V. Searching for activation functions. *arXiv* **2017**, arXiv:1710.05941.
35. Shetty, S. Application of convolutional neural network for image classification on Pascal VOC challenge 2012 dataset. *arXiv* **2016**, arXiv:1607.03785.
36. Wang, Z.; Wang, P.; Louis, P.C.; Wheless, L.E.; Huo, Y. Wearmask: Fast in-browser face mask-wearing detection with serverless edge computing for COVID-19. *arXiv* **2021**, arXiv:2101.00784. [CrossRef]
37. Ge, S.; Li, J.; Ye, Q.; Luo, Z. Detecting masked faces in the wild with lle-cnns. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 2682–2690.
38. Yang, S.; Luo, P.; Loy, C.C.; Tang, X. Wider face: A face detection benchmark. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 5525–5533.
39. Zhang, H.; Cisse, M.; Dauphin, Y.N.; Lopez-Paz, D. mixup: Beyond empirical risk minimization. *arXiv* **2017**, arXiv:1710.09412.
40. Zeng, T.; Li, S.; Song, Q.; Zhong, F.; Wei, X. Lightweight tomato real-time detection method based on improved YOLO and mobile deployment. *Comput. Electron. Agric.* **2023**, *205*, 107625. [CrossRef]
41. GitHub. ONNX. 2021. Available online: https://github.com/onnx/onnx. (accessed on 8 November 2023).
42. GitHub. NCNN. 2021. Available online: https://github.com/Tencent/ncnn. (accessed on 8 November 2023).