A Lightweight Blind Obstacle Detection Network for Mobile Side

Jianghong Zhao, Ziyu Liu, Ziling Liu, Ailin Xu, Jifu Zhao

School of Surveying, Mapping and Urban Spatial Information, Beijing University of Civil Engineering and Architecture, Beijing,102616,China - zhaojiangh@bucea.edu.cn, 956828753@qq.com, 940568635@qq.com, ailinna2000113@163.com, 1403029118@qq.com

Keywords: Obstacle Detection, Deep Learning, Lightweight, Mobile Side.

Abstract

China has the longest and widest distribution of blind corridors in the world, but in many cities they are virtually non-existent, with all kinds of obstacles affecting the movement of the blind. Thus, guaranteeing safe traveling for the blind has become an increasingly important topic. Some existing assistive traveling devices for the blind have problems such as poor portability and low-cost performance. With the rise of the mobility era and the rapid development of deep learning technology, the target detection of blind obstacles on cell phones has become feasible. In this paper, we take the target detection model YOLOv8n as the base network, redesign the neck of the network, use GS convolution as the basis, adopt one-time aggregation and cross-channel branching to build a lightweight module, and use the CARAFE operator as the up-sampling method, and stack the lightweight module with CARAFE operator to build a new feature fusion layer, forming a lightweight feature-aware enhanced target detection network. The results show that the accuracy still reaches 99.5% of the original network under the premise that the model size is reduced by 1.2%, and the improved network achieves a balance between accuracy and lightweight.

1. Introduction

Statistics show that there are about 36 million blind people in the world, and as of 2023, the number of blind people in China exceeds 17.3 million, which is currently the largest number of blind people in the world. Normal people can perceive the position and distance of objects in their surroundings through vision, while blind people lack this ability, which greatly affects their freedom of movement and quality of life; for example, it is difficult to detect and avoid obstacles near blind alleys. In order to address the mobility needs of the blind and ensure their safety, more and more researchers are investing in the development of blind mobility systems or devices. Nowadays, the means of assisting the blind to travel can be roughly divided into three categories: the first category is to utilize sensors in different aspects such as vision and hearing, for example, ultrasonic detectors etc., to accomplish various tasks such as distance measurement and detection through integration and stacking. This method is very high precision, but the disadvantages are also very obvious: the sensor will be affected by environmental factors, and this type of equipment carries a variety of sensors, resulting in poor portability; the price is also very high; the second category is with the development of the field of computer vision, the use of different types of cameras or devices from the real environment to obtain the image, and the use of computer vision based on the algorithm to detect the obstacle. This method does not need to carry a lot of sensors and can be more direct and convenient detection. However, the problem of object obstruction in the image will make the algorithm biased in processing, and also needs to be used with a wearable device, increasing the additional cost of use for blind users. The third category is mobile phone-based blind obstacle target detection. This technology has high scalability, excellent hardware and software integration, and greatly reduces the cost of use for the blind community. As a basic task of computer vision, object detection is generally categorized into “two-stage” and "one-stage." The "two-stage" object detection network has high accuracy and precision and is suitable for complex scenes and small object detection. However, the speed is relatively slow due to the need for two-stage processing. For mobile deployments, choosing the "one-stage" object detection network with faster detection speed is better. At the same time, the network model should be lightweight while maintaining accuracy.

YOLOv8n has significant effects in real-time object detection., and this paper improves on YOLOv8n. Many researchers have replaced lightweight modules in the trunk part in large quantities to obtain a network structure with a few parameters. While the backbone is the key part of extracting the target feature map, the lightweight module, although it can reduce the number of parameters to a certain extent, will lead to a decrease in accuracy due to the abandonment of the completely dense connection. Therefore, seeking lightweight in the feature extraction part will lead to the loss of the most important target feature part used for detection, which directly affects the accuracy of the final detection result. Some studies have superimposed various attention mechanisms in the backbone to compensate for this loss, leading to a further increase in the number of parameters and not necessarily an increase in accuracy. Some other studies add new detection head branches in order to enhance the detection capability of small targets, usually superimposed in depth or parallelized in width. However, it increases the computational volume, making it difficult to balance accuracy and lightweight. To solve the above problems, we modify the neck of the YOLOv8n base network and propose a lightweight module C2f _GS based on GSConv without any addition and replacement of the backbone, which ensures that the extracted features are comprehensively accurate. At the same time, the content-aware reorganization operator CARAFE is used to compensate for the feature information loss problem caused by the lightweight module to balance accuracy and lightweight. At the same time, considering the lack of datasets for obstacles near blind alleys in public datasets, we establish an exclusive dataset for the field of view of blind alleys.

2. Related Work

2.1 Object Detection

object detection is an important task in computer vision; early object detection by manual feature extraction and traditional target detection algorithms are computationally intensive, poorly robust, and weakly generalized. With the AlexNet proposed in the literature (Krizhevsky et al., 2012), the development of a Convolutional Neural Network (CNN) (Fukushima, 1980) is successfully promoted. Literature (Girshick et al., 2014) proposed R-CNN, a CNN-based object detection algorithm, which significantly improved detection accuracy. Nowadays, object detection algorithms are mainly categorized into two types: traditional object detection algorithms and deep learning-based object detection algorithms. Deep learning-based object detection can be divided into Anchor-based and Anchor-free object detection.

2.1.1 Anchor-based Object Detection : Anchor-based object detection can be further categorized into one-stage object detection and two-stage object detection.

The pioneering work of two-stage object detection is R-CNN, which innovatively uses a convolutional neural network for feature extraction in the candidate region; compared with traditional algorithms, this method has made a qualitative leap in accuracy, but the detection speed of the network is slow. In order to solve the redundant computation caused by the large number of overlapping frames in R-CNN when extracting features, the literature (He et al., 2015) proposed SPPNet. When using the SPPNet network for object detection, the entire image must be computed only once to generate a fixed-size feature map, regardless of the size of the candidate frame, which greatly improves the network inference speed. However, like R-CNN, the SPPNet training process is still complex. To solve the above problems, in 2015, R. Girshick et al. proposed Fast RCNN (Girshick et al., 2015). Instead of multi-stage pipeline training, the method puts a detector and edge regressor into the CNN at the same time for training. The detection speed and accuracy are greatly improved. This network still uses a selective search algorithm to find candidate regions, a slow process. In order to solve the problem of long time for region of interest selection, literature (Ren et al., 2016) proposed the FasterR-CNN algorithm. Faster R-CNN introduces Region Proposal Network (RPN). The whole model can be trained for object detection and candidate region generation at the same time, which guarantees end-to-end training and real-time. In 2017, He K M et al. proposed Mask R-CNN (He et al., 2017) based on of Faster R-CNN, which introduces the Region of Interest aggregation (RolAlign) layer instead of the traditional ROI pooling and solves the problem that the corresponding position of the feature maps in the original image deviates from the real position. In the same year, T.-Y. Lin et al. proposed the Feature Pyramid Networks (FPN) (Lin et al., 2017) technique. FPN utilizes a multi-scale feature pyramid and top-down and bottom-up feature propagation mechanism, which enables the network to better utilize multi-scale information for object detection.

One-stage object detection networks are faster in detection speed. Liu W et al. proposed the SSD algorithm in 2016 (Liu et al., 2016). SSD uses feature maps of different scales for object detection of different sizes. These feature maps are extracted by a convolutional neural network (CNN). They are detected at different levels so that targets of various sizes can be effectively detected, with better detection of small targets. YOLOv1 (Redmon et al., 2016) treats the problem of object detection as an end-to-end regression problem, which greatly improves the speed of detection but with lower precision. To address the low recall and accuracy of YOLOv1, the literature (Redmon and Farhadi, 2017) proposed YOLOv2 based on YOLOv1. In YOLOv2, a batch normalization layer is introduced after each convolutional layer, while Darknet-19 is used as the network infrastructure. The target detection accuracy was improved without decreasing the detection speed.YOLOv3 (Redmon and Farhadi, 2018) improved the network's backbone, utilized multi-scale feature maps for detection, and improved the logistic regression classifier to replace softmax for predicting category classification.YOLOv4 (Bochkovskiy et al., 2020) backbone partially changed to incorporate CSP as a network architecture in the DarkNet53 network structure while using Mish activation function.YOLOv5 used PANet (Path Aggregation Network) as a feature fusion module at the input side to better utilize the different scales of feature information. In addition, the Adaptive Anchor Box (AAB) mechanism is introduced to better adapt to targets of different shapes and sizes. Meituan launched YOLOv6 in 2022 (Li et al., 2022), YOLOv6 adopts the feature pyramid network (Rep-PAN) to realize feature fusion, which fuses low-level semantic features containing detailed information and high-level semantic features rich in contextual information to construct a multi-scale pyramid feature mapping. This fusion strategy significantly improves the model's accuracy in the object detection task. YOLOv7 (Wang et al., 2022) improves on YOLOv5 by introducing an extended efficient layer aggregation network and model scaling based on the Concatenate model and incorporating techniques such as auxiliary training header and label assignment in network training. YOLOv7 achieved excellent performance, with accuracy and response speed outperforming most previous single-stage object detection algorithms. In 2023, Ultralytics released YOLOv8, which replaced the C3 structure of YOLOv5 with a C2f structure with richer gradients, and adapted different numbers of channels for different scaling models, and the network head was structured with a decoupled header structure, which substantially improved the model performance. The authors of YOLOv7 released YOLOv9 in 2024 (Wang et al., 2024), which addresses the problem of input data losing information during the transmission process, and proposes PGI (Programmable Gradient Information) and GELAN (Generalized ELAN), which retains important information in the deeper features, and improves the parameter utilization of the network structure and the computational Efficiency.

2.1.2 Anchor-free Object Detection : The anchor-based object detection algorithm generates a large number of anchors during the training process, which will increase the computational cost and the time complexity of both training and inference. At the same time, it is prone to the problem of positive and negative sample imbalance. To improve this situation, the literature (Law and Deng, 2018) proposed the CornerNet algorithm. The algorithm performs target detection by predicting the upper-left and lower-right corner points of the target, and this key-point-dependent approach avoids the problem of Anchor position offset.CenterNet (Zhou and Wang, 2019), on the other hand, focuses on predicting the center point position of the target, removes the time-consuming NMS post-processing operation, and greatly improves the speed of network detection. The FCOS network (Tian et al., 2020) is an FCN-based pixel-by-pixel object detection algorithm that adopts the design ideas of Anchor-free and Proposal-free and introduces the concept of Center-Ness, which saves a large amount of memory space during the training process and thus reduces the total training memory occupancy by about two times.

2.2 Lightweighting Methods

2.2.1 Compression of Network Models: The main purpose of this approach is to generate compact neural networks, which usually do not involve modification of the overall structure of the model, but rather removal of redundant parts or compression of the whole. Model compression is achieved in three main ways: pruning, quantization and distillation.

Network pruning removes weights or neurons according to certain rules to avoid unnecessary computation (Molchanov et al., 2016). Previously, unstructured pruning in terms of weights and other elements is often used. However this pruning introduces a large number of null values in the network parameters, which cannot reduce the computation of neural networks without lower hardware optimization, so scholars have proposed structured pruning in terms of the entire channel and other blocks, structured pruning does not need to combine with a specific hardware to achieve the inference acceleration, but due to the larger granularity of its pruning, it has a greater impact on the model's accuracy is also greater.

Model quantization techniques that convert weight parameters in neural networks from high-precision representations (e.g., floating-point numbers) to low-precision representations (e.g., integer or binary) can reduce the model's storage requirements and computational overhead, thus enabling neural network models to run more efficiently on resource-constrained devices (Han et al., 2015). Some scholars have quantized the weights and activation values to the same bit width, but the redundancy and importance of each layer of a neural network vary, so subsequent studies have used mixed precision to assess the importance of each layer with varying degrees of quantization for each layer.

Knowledge distillation uses the prediction results of a larger, complex model (teacher model) to guide the training of a lightweight model (student model) to achieve higher performance (Hinton et al., 2015). However, its biggest problem is that it is limited by the pre-trained neural network, and the accuracy of the teacher model directly affects the accuracy of the student model.

2.2.2 Neural Network Automated Search Architecture: The NasNet (Elsken et al., 2019) method proposed by Elsken et al. automatically explores and discovers the optimal neural network architecture, the main principle of which is to find the optimal hyperparameter combinations in a predefined search space based on a specific search strategy, and then combines the basic units to design a high-performance neural network architecture that is capable of solving a given task. Next, the overall performance of the neural network architecture is measured by a performance evaluation strategy, similar to the role of a loss function. The evaluation results are fed back to the search strategy as a way to iteratively optimize and eventually obtain the best solution.

2.2.3 Manual Design of Lightweight Network Structures: Convolutional neural networks are made up of multiple convolutions of different sizes and roles stacked on top of each other, so designing a lightweight network structure usually starts with designing lightweight convolutions. In the early stage, group convolution or dot convolution with depth convolution is used to reduce the computational effort of standard convolution, but group convolution also has limitations, which can lead to the information between feature maps is not fluent, so the subsequent model design is usually constructed by using the depth separable convolution which is a combination of dot convolution and depth convolution as the base convolution, such as the classic MobileNet (Howard et al., 2019), ShuffleNet (Ma, 2019).In addition, lightweight module design is also a key point. The dense connection design of DenseNet helps the full transfer and reuse of information, but because each layer is directly connected to all the previous layers, which leads to high memory consumption, so many scholars use this dense connection as the basis for reconstructing the lightweight module in a more efficient way, such as CSPNet (Wang et al., 2020).

3. Method

3.1 YOLOv8 Network Architecture

YOLOv8, the best comprehensive performance model, has the advantages of fast speed and high accuracy. In order to be better deployed on mobile, YOLOv8n, which has the smallest model size, is used as the base network. YOLOv8n consists of backbone, neck, and head parts. Backbone borrows the idea of VGG network and ResNet, uses more 3×3 convolution and residual connectivity, and consists of multiple CBS (Conv ordinary convolution + BN batch processing) modules. Normalization + SiLU activation function) modules. The neck combines FPN+PAN, where FPN passes down the strong semantic features at the higher level and PAN passes up the strong localization features at the lower level. There are also cross-layer connections between the two branches to achieve better feature fusion. The head uses a decoupling head, which regresses the classification and detection frames as two branches to improve the model generalization ability.

3.2 Improved YOLOv8 Model

3.2.1 Lightweight Module C2f _GS: Many studies have substantially modified the feature extraction layer in order to lighten the network, replacing the ordinary convolution with a lightweight convolution such as the DW convolution, which reduces the number of parameters to a certain extent, but loses more feature information and reduces the accuracy of the whole network. The feature extraction layer, as the core of the whole network, greatly impacts the subsequent detection if it fails to extract accurate and rich features. Based on the above considerations, this experiment starts from the feature fusion layer and constructs the lightweight module by stacking based on the lightweight convolution GSConv (Li et al., 2022), which is a new type of convolution that fuses the standard convolution, depth-separable convolution, and shuffle channel shuffling to be used, and its structure is shown in the following Figure. 1. The convolution first performs a standard convolution to change the size and channel of the input map, retaining all the feature information, and then does a depth-separable convolution on top to reduce the number of parameters. The results of the two branches obtained from different convolutions are spliced together and then output using channel shuffling operation to cross-fertilize the features of each channel, which completes the fusion of the standard convolution and the lightweight convolutions, and makes the output result as close as possible to the standard convolution, retains more feature information and reduces the number of parameters.

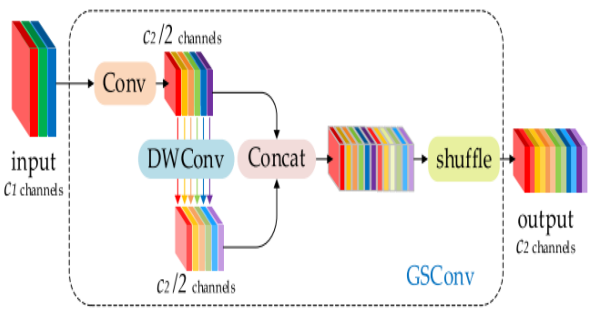
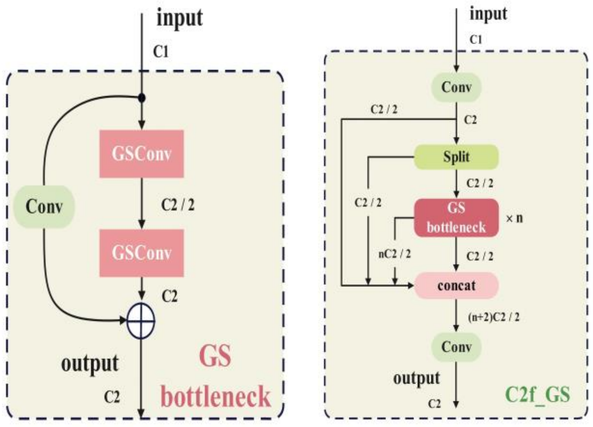


Figure 1. GSConv structure.

As shown in Fig. 2(a), the GS bottleneck structure incorporates GS convolutional layers and is designed with shortcut branches. The input feature map is successively convolved with GS twice to obtain a new output and then spliced with the original input feature map in the channel direction to obtain the final output feature map.

On this basis, the lightweight module C2f _GS, which is mainly based on GS bottleneck, is constructed, as shown in Figure. 2 (b)below, and the input feature map enters into C2f _GS and first undergoes one standard convolution to get the output. Inspired by VovNet (Lee et al., 2019) and CSPNet, when the intermediate layer aggregation ability is lower, the final layer aggregation ability is stronger. The middle layer performs dense connectivity to bring not high gains, so only the splicing of branches at the end (Fig. 2(b) contact operation) is performed, and the input and output channels of the middle layer are made equal, so that a 1 × 1 convolution is not needed to change the number of channels, avoiding the parameter redundancy and making the computation more efficient. Meanwhile, to prevent the duplication of gradient information, we truncate the gradient flow and add a separate cross-channel connection branch (Fig. 2(b) Split operation) without adding redundant fragmentation branches to avoid efficiency loss.



(a) GS bottleneck structure (b) C2f _GS
Figure 2. Lightweight module C2f _GS.

3.2.2 Upsampling Operator: The nearest neighbor interpolation of the up-sampling method used in the original network only uses the spatial location of the (2*2) domain around the pixel point to interpolate the reduced feature maps. The too-small perceptual region makes the reduced-size feature maps not contain enough semantic information, whiletheaddition of the lightweight module makes the feature information even less comprehensive. The CARAFE operator (Wang et al., 2019) treats the up-sampling process as the dot product of the up-sampling kernel and the original feature map, which consists of two parts: up-sampling kernel prediction and feature reorganization. The global information of the original feature map is content encoded to generate the up-sampling kernel. The dot product feature reorganization uses a 5*5 domain, which can effectively aggregate the context information and enhance the feature fusion ability of different size targets. Its structure is shown in Figure. 3.

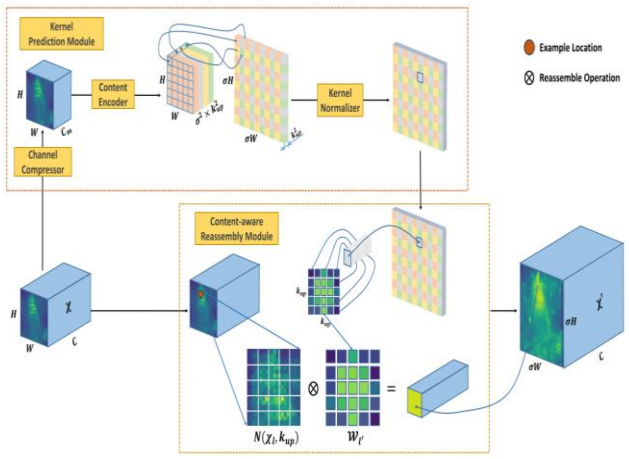


Figure 3. Structure of the CARAFE operator.

3.2.3 Neck Optimization: Based on the structure of the original network, the C2f_GS lightweight module is stacked with the CARAFE operator to construct a new feature fusion layer (i.e., Lightweight Enhanced Feature-neck, LEF-neck), which replaces the neck of the original network to form a lightweight feature-aware enhanced object detection network.

The overall structure of the feature fusion layer is shown in Figure 4 below. This part up-samples the fifth-layer features (high-level features) obtained from the network feature extraction layer so that they have the same spatial dimensions as the fourth-layer feature maps, and then the up-sampled fifth-layer feature maps are channel-spliced with the fourth-layer feature maps to obtain a richer feature representation. The spliced feature map is input into the C2f _GS module for processing, and the processed feature map is up-sampled once more so that it has the same spatial dimensions as the third-layer feature map and is spliced with the third-layer feature map, and the spliced feature map is input into the C2f _GS for a second time, and then the obtained result is inputted into the detection layer to obtain a large-size object detection result; the second C2f _ GS processing results obtained after GS downsampling convolution with the first C2f _GS processing results are spliced, after which the third C2f _GS processing is carried out, and the obtained results are input into the detection layer to get the medium-sized object detection results; the results obtained from the third C2f _GS processing are then spliced with the fifth layer of feature maps after GS downsampling convolution, and then the fourth C2f _ GS processing, the obtained results are input to the detection layer to get the small-size object detection results, and finally the detection results of large, medium, small and three scales are output. By performing processing and detection on different layers of feature maps, the detection performance for different sized objects is improved.

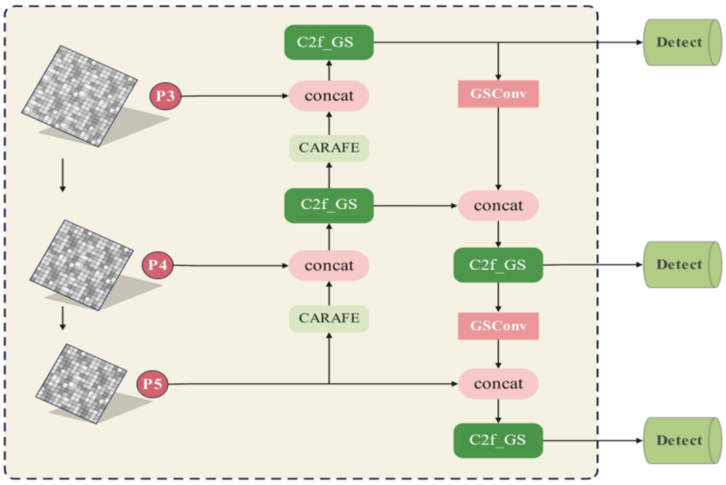


Figure 4. Structure of the LEF-neck feature fusion layer.

(val)	mAP(%)	FLOPs(G)	Number of parameters (M)	Prediction time (ms)	Model size (mb)
v8n	77.6	8.1	3.0	1.9	5.95
v5n	75.2	4.2	1.77	1.2	3.66
v7-tiny	81.0	13.1	6.03	2.2	11.7
GOLD	77.2	11.9	6.0	3.7	11.8
Ours	**77.2**	**7.5**	**2.94**	**3.7**	**5.88**

Table 1. Results of comparison experiments with homemade datasets.

(val)	mAP(%)	FLOPs(G)	Number of parameters (M)	Prediction time (ms)	Model size (mb)
v8n	77.6	8.1	3.0	1.9	5.95
v8n+GS	77.1	7.3	2.80	2.0	5.61
v8n+GS+CARAFE	**77.2**	**7.5**	**2.94**	**3.7**	**5.88**

Table 2. Comparison of ablation experiment results.

4. Experiments

4.1 Dataset

4.1.1 MS COCO Dataset： MS COCO's full name is Microsoft Common Objects in Context, a large image dataset released by Microsoft in 2014. COCO dataset annotation format is json format. The dataset is mainly used for tasks such as object detection, human key point detection, semantic segmentation and subtitle generation. The COCO dataset has a total of 91 categories (80 categories are used for the detection task), covering four major categories: people, animals, transportation, and home appliances.

4.1.2 PASCAL VOC Dataset： The PASCAL VOC dataset has been released since 2005, and many versions have been released until 2012; the most widely used ones are VOC2007 and VOC2012. VOC2007 and VOC2012 contain four categories: people, animals, transportation, and furniture. The VOC dataset is stored in XML format, and each image has the corresponding category and bounding box information. The VOC dataset is stored in XML format, and each image has a corresponding category and bounding box information. This dataset can be used for image classification, image recognition, and object detection tasks.

4.1.3 Self-constructed Dataset： Through preliminary research, we established the type of dataset that would meet the needs of the blind. The dataset includes ten categories of bicycles, automobiles, people, utility poles, stone piers, tree pits, fire hydrants, motorcycles, steps, and conical barricades, and the number of images for each category is not less than 500, totaling 16,296 images. These images were partially selected from existing datasets such as MS COCO and PASCAL VOC for included category pictures and labeling information. The categories not included are acquired by taking pictures with web crawlers. The dataset is uniformly labeled in VOC format by labeling software, and the labeling process is shown in Figure 5. The dataset is divided into training sets, validation sets, and test sets according to the ratio of 7:2:1.

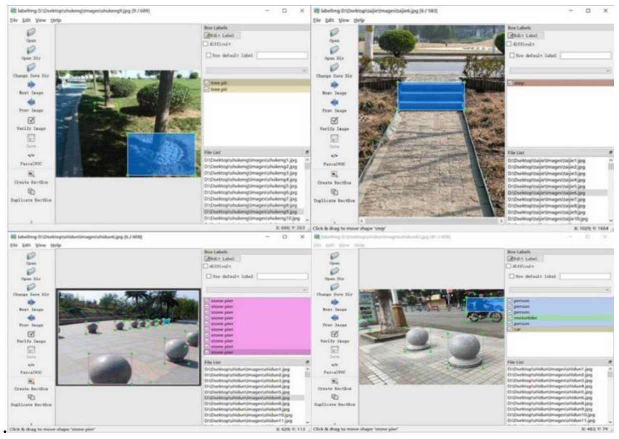


Figure 5.Process of dataset production.

4.2 Set up

The experiments were conducted using a homemade blind obstacle dataset with an RTX 2080 Ti, a graphics card with 11 GB of video memory, an Ubuntu system, a python version 3.8.10, a CUDA version 11.1, and a pytorch version 1.9.0 configuration, with the batchsize set to 32, an epoch of 200, and pre-training weights yolov8n.pt.

4.3 Contrast Experiment

Our experiments compared our network with the same type of state-of-the-art single-stage object detection networks in our experiments. As shown in Table 1, YOLOv5n is still the lightest network, but its accuracy decreases significantly compared to other networks, and these accuracy losses have a significant impact on real-world applications of object detection. Compared with YOLOv7-tiny, our model computation is reduced by 42.7%, parameter amount is reduced by 51.2%, model size is reduced by 49.7%, and the accuracy reaches 95.3% of that of YOLOv7-tiny, which is a great improvement in lightweight. Compared to the original network of YOLOv8n, our model achieves 99.5% accuracy with a 7.4% reduction in computation, a 2% reduction in the number of parameters, and a 1.2% reduction in model size. We also embedded Huawei's proposed GOLD-YOLO module (Wang et al., 2023) into the original network for a separate comparison experiment, and its computation amount increased by 46.9%, the number of parameters doubled, and the model size increased by 98.3% compared to our network with the same accuracy, which does not meet the requirement of lightweight. This fully demonstrates the advanced performance of our network in detecting blind obstacles.

4.4 Ablation Experiments

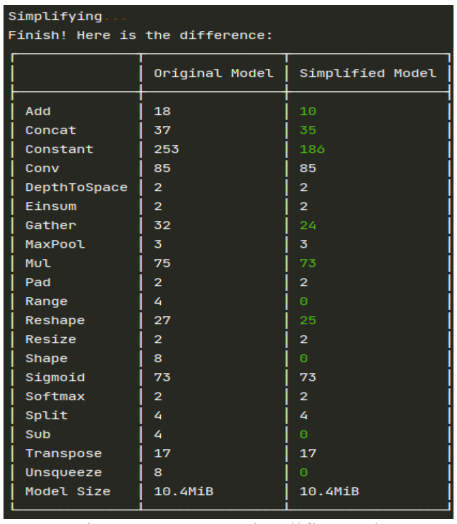
As shown in Table 2, v8n is the original base network; v8n+GS uses only the GS lightweight module; v8n+GS+CARAFE indicates that both the GS lightweight module and the CARAFE operator are used. When only the GS module is added, the network computation decreases by 9.8%, the number of parameters decreases by 6.7%, and the model size decreases by 5.7%, yet the accuracy remains 99.4% of the original network. Adding the CARAFE upsampling operator on top of this, the model accuracy is improved by 0.1%, and the addition of the CARAFE operator does compensate for the loss of feature information brought about by the GS lightweight module to a certain extent. When the C2f-GS module is used in combination with the CARAFE operator (i.e., LEF-neck), the accuracy achieves 99.5% of the original network on the premise that the model computation, parameter counts, and the model size are comprehensively reduced, and lightweight is realized.

5. Mobile deployment of the blind obstacle detection network

5.1 Deployment of An Obstacle Detection Network Based on The NCNN Framework

NCNN is a high-performance neural network forward computation framework developed by Tencent and designed for mobile devices. It can minimize model size and computational load to adapt to resource-constrained embedded development devices and mobile application scenarios. Written in C++, NCNN supports multiple computing platforms, including CPU, GPU, and DSP, etc., and does not require any third-party library dependencies, enabling efficient model inference and training. Developers can register custom layers to extend the model design while supporting quantization and half-precision floating-point storage, which is conducive to the portability of multiple model files.

5.1.1 Model Conversion: ONNX is an open file format designed for machine learning by Microsoft and Facebook, which allows different deep learning frameworks (e.g., Pytorch, TensorFlow, Paddle) to use the same format to store model data. Pt files cannot be directly loaded into NCNN frameworks when deploying network models, so pt files are converted to a more inclusive common format, ONNX, and then simplified by the ONNX-simplifier tool to remove useless operators generated during the conversion process. are converted to the more inclusive universal format ONNX, and then the model is simplified by the ONNX-simplifier tool to remove the useless operators generated in the process of converting the two formats, as shown in Figure 6 below, where the green part is the number of simplified structures. After obtaining the ONNX format file, the model is then converted to the format file supported by the NCNN framework using the online conversion tool.



```
Simplifying...
Finish! Here is the difference:
┌──────────────┬────────────────┬──────────────────┐
│              │ Original Model │ Simplified Model │
├──────────────┼────────────────┼──────────────────┤
│ Add          │ 18             │ 10               │
│ Concat       │ 37             │ 35               │
│ Constant     │ 253            │ 186              │
│ Conv         │ 85             │ 85               │
│ DepthToSpace │ 2              │ 2                │
│ Einsum       │ 2              │ 2                │
│ Gather       │ 32             │ 24               │
│ MaxPool      │ 3              │ 3                │
│ Mul          │ 75             │ 73               │
│ Pad          │ 2              │ 2                │
│ Range        │ 4              │ 0                │
│ Reshape      │ 27             │ 25               │
│ Resize       │ 2              │ 2                │
│ Shape        │ 8              │ 0                │
│ Sigmoid      │ 73             │ 73               │
│ Softmax      │ 2              │ 2                │
│ Split        │ 4              │ 4                │
│ Sub          │ 4              │ 0                │
│ Transpose    │ 17             │ 17               │
│ Unsqueeze    │ 8              │ 0                │
│ Model Size   │ 10.4MiB        │ 10.4MiB          │
└──────────────┴────────────────┴──────────────────┘
```

Figure 6. ONNX-simplifier tool.

5.1.2 Model Deployment: After the model files are all converted, they are loaded into the Android Studio project for deployment. The device for this experiment is a Redmi K40 mobile phone with 8GB of RAM, Qualcomm Snapdragon 870 CPU, and MIUI 12 operating system. The mobile phone is paired and connected to the Android Studio platform and then synchronised with Gradle. Gradle is a tool used to manage third-party libraries. It packages, which can automatically download multiple types of dependent libraries, and also supports customized packaging scripts, providing a convenient basis for multi-channel packaging. Gradle synchronization will check whether the downloaded packages and Android Studio version of the project are the same and, at the same time, parse the configuration to determine the build output target, dependencies, and tasks to be performed during the build process and then synchronize the project to package the whole program on the phone.

5.2 Results Display

Figure 7 shows the results of using the mobile device deployed with the blind obstacle object detection network for field detection. As can be seen from the detection results graph, the object detection obstacles can be detected almost all the time, and the mobile terminal's FPS can reach 20 and above, which fully meets the demand for real-time detection. The prediction of the detected object detection box category also does not appear to misdetect the situation, the recognition effect is good.

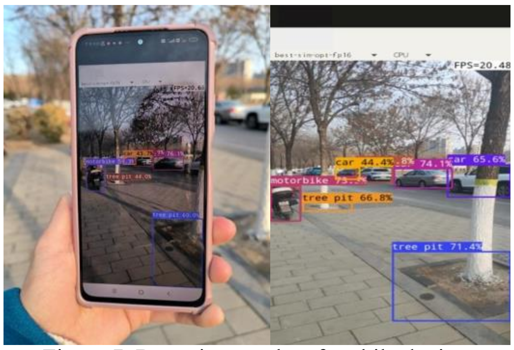


Figure.7. Detection results of mobile device.

6. Conclusion

In this paper, the feature fusion layer of the network is redesigned by using the C2f-GS module with the CARAFE up-sampling operator, which reduces the computational effort while enhancing the feature representation. Compared with the original YOLOv8n network, we achieve lossless lightweight with the same accuracy and a relative balance between accuracy and efficiency. Meanwhile, our network is deployed on mobile phones with FPS of 20 and above, which meets the demand of mobile deployment and achieves real-time detection.

Reference

Bochkovskiy, A., Wang, C.-Y., Liao, H., 2020. YOLOv4: Optimal Speed and Accuracy of Object Detection. *arXiv preprint arXiv:*2004.10934.

Elsken, T., Metzen, J.H., Hutter, F., 2019. Neural Architecture Search: A Survey. *Journal of Machine Learning Research* 20, 1–21.

Fukushima, K., 1980. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biol. Cybernetics* 36(4), 193–202.

Girshick, R., Donahue, J., Darrell, T., Malik, J., 2014. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 580–587.

Girshick, R., 2015. Fast R-CNN. *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1440–1448.

He, K., Zhang, X., Ren, S., Sun, J., 2015. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37, 1904–1916. doi.org/10.1109/TPAMI.2015.2389824.

He, K., Gkioxari, G., Dollar, P., Girshick, R., 2017. Mask R-CNN. *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2961–2969.

Hinton, G., Vinyals, O., Dean, J., 2015. Distilling the Knowledge in a Neural Network. *arXiv preprint arXiv*:1503.02531. doi.org/10.48550/arXiv.1503.02531.

Han, S., Mao, H., Dally, W.J., 2016. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding. *arXiv preprint arXiv*:1510.00149.doi.org/10.48550/arXiv.1510.00149.

Howard, A., Sandler, M., Chu, G., Chen, L.-C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., Le, Q.V., Adam, H., 2019. Searching for MobileNetV3. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1314–1324.

Krizhevsky, A., Sutskever, I., Hinton, G.E., 2017. ImageNet classification with deep convolutional neural networks. *Commun.ACM* 60, 84–90. doi.org/10.1145/3065386.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., Berg, A.C., 2016. SSD: Single Shot MultiBox Detector. *Computer Vision – ECCV 2016*. Springer, Cham, pp. 21–37. doi.org/10.1007/978-3-319-46448-0_2.

Li, C., Li, Lulu, Jiang, H., Weng, K., Geng, Y., Li, Liang, Ke, Z., Li, Q., Cheng, M., Nie, W., Li, Y., Zhang, B., Liang, Y., Zhou, L., Xu, X., Chu, X., Wei, Xiaoming, Wei, Xiaolin, 2022. YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications.*arXiv preprint arXiv:2209.02976.* doi.org/10.48550/arXiv.2209.02976.

Lin, T.-Y., Dollar, P., Girshick, R., He, K., Hariharan, B., Belongie, S., 2017. Feature Pyramid Networks for Object Detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2117–2125.

Long, J., Shelhamer, E., Darrell, T., 2015. Fully Convolutional Networks for Semantic Segmentation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3431–3440.

Law, H., Deng, J., 2018. CornerNet: Detecting Objects as Paired Keypoints. *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 734–750.

Lee, Y., Hwang, J., Lee, S., Bae, Y., Park, J., 2019. An Energy and GPU-Computation Efficient Backbone Network for Real-Time Object Detection. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 0–0.

Li, H., Li, J., Wei, H., Liu, Z., Zhan, Z., Ren, Q., 2024. Slim-neck by GSConv: a lightweight-design for real-time detector architectures. *J Real-Time Image Proc* 21, 62. doi.org/10.1007/s11554-024-01436-6.

Molchanov, P., Tyree, S., Karras, T., Aila, T., Kautz, J., 2017. Pruning Convolutional Neural Networks for Resource Efficient Inference.*arXiv preprint arXiv:1611.06440.* doi.org/10.48550/arXiv.1611.06440.

Ma, N., Zhang, X., Zheng, H.-T., Sun, J., 2018. ShuffleNet V2: Practical Guidelines for Efficient CNN Architecture Design. *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 116–131.

Ren, S., He, K., Girshick, R., Sun, J., 2017. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Trans. Pattern Anal. Mach. Intell.* 39, 1137–1149.

Redmon, J., Divvala, S., Girshick, R., Farhadi, A., 2016. You Only Look Once: Unified, Real-Time Object Detection. 2016

IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 779–788. doi.org/10.1109/CVPR.2016.91.

Redmon, J., Farhadi, A., 2017. YOLO9000: Better, Faster, Stronger. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* pp. 7263–7271.

Redmon, J., Farhadi, A., 2018. YOLOv3: An Incremental Improvement. *arXiv preprint arXiv*:1804.02767.

Tian, Z., Shen, C., Chen, H., He, T., 2022. FCOS: A Simple and Strong Anchor-Free Object Detector. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44, 1922–1933.

Wang, C.-Y., Bochkovskiy, A., Liao, H.-Y.M., 2022. YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 7464-7475. doi.org/10.48550/arXiv.2207.02696.

Wang, C.-Y., Yeh, I.-H., Liao, H.-Y.M., 2024. YOLOv9: Learning What You Want to Learn Using Programmable Gradient Information. *arXiv preprint arXiv:*2402.13616. doi.org/10.48550/arXiv.2402.13616.

Wang, C.-Y., Liao, H.-Y.M., Wu, Y.-H., Chen, P.-Y., Hsieh, J.-W., Yeh, I.-H., 2020. CSPNet: A New Backbone That Can Enhance Learning Capability of CNN. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 390–391.

Wang, J., Chen, K., Xu, R., Liu, Z., Loy, C.C., Lin, D., 2019. CARAFE: Content-Aware ReAssembly of FEatures. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3007–3016.

Wang, C., He, W., Nie, Y., Guo, J., Liu, C., Wang, Y., Han, K., 2023. Gold-YOLO: Efficient Object Detector via Gather-and-Distribute Mechanism. *Advances in Neural Information Processing Systems* 36, 51094–51112.

Zhou, X., Wang, D., Krähenbühl, P., 2019. Objects as Points. *arXiv preprint arXiv*:1904.07850. doi.org/10.48550/arXiv.1904.07850.