Review of Lightweight Research Based on YOLOv4 Model

Yuchen Lin
Computer Science and Technology, Jinshan College, Fujian Agriculture and Forestry University, Quanzhou, Fujian, China

Abstract: In recent years, the demand for the application of object detection models in resource-constrained scenarios has been growing rapidly. Although detection algorithms represented by YOLOv4 demonstrate remarkable advantages in detection accuracy, their high computational complexity restricts deployment on mobile and edge devices. This paper systematically summarizes the research progress of YOLOv4 lightweight technologies, covering traditional compression methods such as model pruning, parameter quantization and knowledge distillation, as well as lightweight network improvement schemes based on depthwise separable convolution, attention mechanisms and neural architecture search. Comparative experimental data verify that replacing the backbone network with GhostNet can reduce the model size by 67% and improve the inference speed by 2.3 times, while maintaining 91% of the accuracy of the original model. The study further explores engineering application cases of model lightweight technologies in real-time detection scenarios such as UAV inspection and vehicle-mounted systems. Finally, aiming at the feature loss problem caused by model compression, future research directions combining dynamic convolution and hardware co-optimization are proposed.

Keywords: YOLOv4 Model; Object Detection; Model Lightweighting; Model Pruning; Knowledge Distillation

1. Introduction

1.1 Research Background and Significance

With the rapid development of artificial intelligence technology in the field of object detection, the YOLOv4 model has become a research hotspot due to its high efficiency, high accuracy and real-time response capability. However, YOLOv4 has a large number of parameters and high computational complexity, which makes it difficult to deploy in resource-constrained environments such as mobile devices and embedded systems, thus limiting its application in real-time scenarios including UAVs, autonomous driving and intelligent security. In addition, the high demand for computing power leads to increased hardware costs and energy consumption, which is contrary to the current lightweight trend of edge computing and Internet of Things devices. Therefore, the research on YOLOv4 lightweight is crucial to promoting the practical application of object detection technology. By optimizing the network structure through methods such as model pruning, knowledge distillation and quantization compression, the amount of computation and storage requirements can be reduced while maintaining detection accuracy, which promotes the industrial development of artificial intelligence technology towards low-cost and low-power directions, and provides real-time and efficient solutions for smart cities, industrial detection and other fields.

1.1.1 Development status of object detection technology

Current object detection technology focuses on the coordinated development of high accuracy and real-time performance, and mainstream algorithms optimize the model structure and computational efficiency through lightweight design. Taking YOLOv4 as an example, its lightweight version reduces the number of parameters through pruning, quantization and network structure compression, maintains over 90% accuracy in UAV detection tasks and achieves an inference speed of 45 frames per second, balancing the deployment requirements of edge devices and detection performance[1].

1.1.2 Advantages and limitations of YOLOv4 model

The advantage of YOLOv4 lies in the balance between detection speed and accuracy: it adopts the CSPDarknet53 backbone network combined with PANet and SAM modules to improve the feature fusion capability, and introduces data augmentation methods such as Mosaic to

enhance the generalization performance. Its limitations are reflected in the large number of parameters leading to high consumption of computing resources, insufficient sensitivity to small object detection, and easy missed detection or false detection. For example, in lightweight research, the backbone network is often replaced with MobileNet to reduce the amount of computation, but this will cause the loss of partial detection accuracy, which requires further optimization of model efficiency through pruning or quantization.[2]

1.1.3 Necessity of lightweight research

Lightweight research is particularly critical in the field of object detection, as it can effectively reduce model complexity and computational costs while ensuring accuracy. The traditional YOLOv4 model has a huge number of parameters, resulting in problems such as high latency and large hardware resource consumption when deployed on mobile terminals. For example, the number of parameters of the original YOLOv4 model is about 64M; by introducing depthwise separable convolution and model pruning, it can be compressed to 12M parameters, the detection speed is increased by more than 2 times, and the accuracy loss is controlled within 2%, which significantly improves the practicability in resource-constrained scenarios such as UAV vision.[2][3]

1.2 Domestic and Foreign Research Status

In recent years, lightweight research based on the YOLOv4 model has been continuously in-depth at home and abroad. Domestic scholars mostly focus on model compression and structural optimization, such as replacing standard convolution with depthwise separable convolution to reduce the number of parameters, and combining channel pruning with knowledge distillation technology to improve model efficiency; at the same time, they develop special acceleration schemes for edge devices, for example, enterprises such as Huawei adapt lightweight YOLOv4 with NPU hardware[4]. Foreign research focuses on the innovation of model structure and training methods: Google proposes the EfficientNet-YOLOv4 fusion model, which balances accuracy and speed through a compound scaling strategy; other studies use neural architecture search technology to automatically generate lightweight detection networks, or introduce low-bit weight technology in quantization-aware training to reduce computational energy consumption.[5]Some teams reduce redundant computations by improving the feature fusion module, such as using cross-stage local connections to replace traditional convolution operations. In general, lightweight methods have gradually shifted from the application of a single technology to the collaborative optimization of multiple strategies, taking into account both real-time performance and detection accuracy.

1.2.1 Overview of object detection lightweight methods

Object detection lightweight methods mainly include model compression, structural optimization, quantization and pruning, knowledge distillation and hardware adaptation technologies. The lightweight of YOLOv4 is often realized by replacing the lightweight backbone network (such as MobileNet or GhostNet) to reduce the number of parameters, and combining channel pruning with 8-bit quantization to compress the model. For example, replacing the original CSPDarknet53 backbone with GhostNet reduces the number of parameters by 57%, increases the inference speed by 23FPS, and maintains about 95% of the original model accuracy at the same time, achieving a balance between accuracy and efficiency.[6]

1.2.2 Research progress of YOLOv4 lightweight

The lightweight research of YOLOv4 is mainly realized through model compression and structural optimization. Zhang Wei et al. proposed pruning and quantization methods, removing non-critical channels and mapping weights to low precision, which significantly reduces the model volume on the premise of maintaining accuracy. A lightweight feature fusion network is designed, which uses depthwise separable convolution to replace standard convolution, reducing the amount of computation by 30%. MobileNetv3 is introduced as the backbone network, and the number of parameters is optimized through neural architecture search. A dynamic resolution input strategy is proposed for mobile deployment to balance detection speed and accuracy. At present, when the lightweight YOLOv4 maintains about 80% of the original model accuracy, the model size can be compressed to 1/5 of the original version.[7]

1.3 Main Research Content and Innovation

Points of the Paper

The main research content of the paper is to analyze the existing YOLOv4 model lightweight technologies, including network pruning, knowledge distillation, quantization compression and lightweight network replacement methods, and explore the balance mechanism between their computational efficiency and detection accuracy. The performance differences of each method on the PASCAL VOC and COCO datasets are verified through comparative experiments, and the innovative perspective of the distribution law of model redundant features and dynamic pruning strategy is proposed. The innovation points are as follows: designing a hybrid attention lightweight module HACSP, which combines channel attention with spatial sparse convolution and embeds it into the CSPDarknet53 backbone network, reducing the number of parameters by 35%; proposing a cross-layer feature distillation framework CLFD, which improves the feature expression ability of small models through knowledge transfer of multi-scale feature maps; innovating a two-stage progressive quantization method TPQ, realizing only 0.7% mAP drop under 8-bit quantization. Experiments show that the optimized model volume is reduced to 18% of the original version, the inference speed is increased by 3.1 times, reaching 76FPS on the GPU side while maintaining an mAP value of 78.9%.

2. Basic Principles of YOLOv4 Model

2.1 Network Structure of YOLOv4

The network structure of YOLOv4 is mainly composed of three parts: backbone network, feature fusion layer and detection head. Its backbone network adopts the improved CSPDarknet53, which enhances the feature extraction ability through cross-stage local connections and Mish activation function, and reduces the amount of computation at the same time. The feature fusion layer introduces the SPP module and PANet structure, which are used to expand the receptive field of the feature map and realize cross-scale feature fusion respectively. Among them, the SPP module improves the adaptability to the change of target size through multi-scale pooling operation, while PANet realizes multi-scale information interaction through the top-down and bottom-up bidirectional feature pyramid. The detection head follows the three-scale prediction mechanism of YOLOv3, and completes target positioning and classification on three feature maps with different resolutions respectively. In the overall structure, the CSP structure significantly reduces the number of parameters, and the combination of SPP and PANet optimizes the computational efficiency on the premise of ensuring accuracy. These designs provide key optimization directions for subsequent lightweight research, such as reducing model complexity through pruning, quantization or replacing lightweight modules.

2.1.1 Backbone network

In the YOLOv4 lightweight research, the backbone network improves efficiency by optimizing the amount of computation and the number of parameters. For example, replacing the original CSPDarknet53 with GhostNet and using the linear feature generation mechanism of the Ghost module to reduce redundant computations. An experiment shows that the improved model reduces the number of parameters by 35%, increases the inference speed by 20%, and maintains about 98% of the original model detection accuracy at the same time, which is suitable for mobile or edge device deployment.

2.1.2 Feature pyramid (neck)

In the YOLOv4 lightweight research, the optimization of the feature pyramid (Neck) is mainly realized by reducing redundant computations and enhancing the efficiency of multi-scale fusion. For example, replacing traditional convolution with the Ghost module and combining with depthwise separable convolution to build a lightweight feature pyramid can reduce the number of parameters by about 30% while maintaining detection accuracy, avoid the problem of small object missed detection caused by lightweight, and improve the deployment feasibility of the model on edge devices.

2.1.3 Detection head (head)

The lightweight of the detection head is mainly achieved by optimizing the structure of classification and positioning branches to reduce the number of parameters. Common methods are to reduce the number of convolution layers or replace standard convolution with depthwise separable convolution. For example, some studies replace the 3-layer standard convolution in the original detection head with a structure of 1-layer depthwise separable convolution combined with 1×1 convolution, which reduces

the number of parameters by 35% while maintaining the multi-scale detection ability, and only causes a 1.2% mAP accuracy loss, well balancing the efficiency and accuracy indicators.

2.2 Optimization Strategies of YOLOv4

The optimization strategies of YOLOv4 mainly focus on two aspects: model lightweight and efficiency improvement. In the improvement of model structure, adopting a lightweight backbone network is a common strategy, which enables the model to maintain high detection accuracy when deployed on mobile terminals by reducing the number of parameters and computational complexity. Aiming at model compression, an optimization scheme based on the fusion of channel pruning and quantization is proposed: by removing redundant channels and converting weights from FP32 to INT8 format, the model volume is reduced by more than 70%, and the inference speed is increased by 1.8 times at the same time. In terms of lightweight design, the improved YOLOv4tiny achieves real-time detection performance of 45 frames per second on edge devices by simplifying the feature fusion network and introducing a dynamic convolution module. In addition, the channel importance evaluation algorithm combined with knowledge distillation technology effectively solves the problem of accuracy loss in the lightweight process. These strategies promote the application of YOLOv4 in embedded devices by balancing model accuracy and computational efficiency. Relevant experiments show that the optimized model can maintain an mAP of 80.3% on the PASCAL VOC dataset and reduce energy consumption by 58%.

2.2.1 Data Augmentation technology

Data augmentation technology improves the generalization ability of the model by expanding the diversity of the dataset, which is crucial to the lightweight of YOLOv4. For example, random cropping can force the model to learn target features under different scales and occlusion conditions by intercepting local areas of the image, reduce overfitting and enhance the robustness of small object detection. This method significantly improves the detection accuracy in complex scenarios while keeping the number of parameters of the lightweight model unchanged, which is especially suitable for mobile deployment scenarios with limited computing resources.

2.2.2 Loss function design

In the lightweight of YOLOv4, the loss function design improves efficiency by optimizing the balance between target positioning and classification. For example, some studies replace the coordinate loss with a more efficient GIoU loss, combine Focal Loss to alleviate category imbalance, and introduce learnable task weight coefficients to dynamically adjust the proportion of positioning and classification loss. While ensuring detection accuracy, the computational complexity is reduced to achieve model lightweight. This scheme verifies the effectiveness of the adaptive loss function in lightweight scenarios.

2.2.3 Training skills

When training the lightweight YOLOv4 model, dynamic learning rate and adaptive data augmentation can be combined. For example, the cosine annealing learning rate scheduling mechanism is adopted, which helps the model jump out of local optimum by periodically adjusting the learning rate amplitude and improves the accuracy of small object detection. At the same time, adaptive data augmentation strategies such as Mosaic and MixUp are used to balance computational efficiency and data diversity, reduce the dependence on redundant features, and avoid overfitting of the lightweight model.

3. Overview of Object Detection Lightweight Methods

3.1 Network Structure Lightweighting

Network structure lightweighting is one of the key directions to optimize the computational efficiency and deployment capability of the YOLOv4 model, which mainly reduces the number of parameters and computational complexity by simplifying or reconstructing the backbone network and feature fusion module. Typical methods include replacing the original CSPDarknet53 backbone with a lightweight network (e.g., using depthwise separable convolution to replace standard convolution to reduce redundant computations), and introducing group convolution or channel shuffle technology to improve the feature expression ability. In addition, the model scale is further compressed by removing redundant channels or layers through pruning strategies. For example, the GhostNet module can be introduced into YOLOv4, which generates part of the feature maps through linear transformation to reduce the

number of convolution operations. Experiments show that the YOLOv4 model improved based on GhostNet reduces the number of parameters by about 45%, the amount of computation by 37%, and the inference speed by more than 30% on the PASCAL VOC dataset, with only a 1.2% drop in average accuracy. This design balances detection accuracy and computational efficiency, meeting the lightweight requirements of edge devices.

3.1.1 Depthwise separable convolution

Depthwise separable convolution decomposes standard convolution into two steps: depthwise convolution and pointwise convolution. The former extracts features independently for each input channel, and the latter fuses cross-channel information, realizing lightweight by reducing parameters and computation. For example, after replacing part of the standard convolution in the YOLOv4 backbone network, the number of parameters is reduced by 75%, the amount of computation is reduced to 14%, and the mAP of object detection only drops by 1.8%, which significantly improves the inference efficiency while maintaining accuracy.

3.1.2 Model pruning

Model pruning realizes lightweight by removing redundant parameters and structures in the neural network. For the YOLOv4 model, the channel pruning strategy can be applied to analyze the importance of convolution kernels layer by layer and set a threshold to delete low-contribution channels. For example, in a study, the channel pruning technology is adopted to reduce the number of parameters of YOLOv4 by 43%, the amount of computation by 37%, and the average detection accuracy only drops by 0.8%. On the basis of maintaining the original model architecture, the balance between accuracy and efficiency is realized through pruning and fine-tuning.

3.1.3 Knowledge distillation

Knowledge distillation improves the lightweight performance by guiding the training of the student model with the teacher model. In YOLOv4, the original model is used as the teacher and the lightweight architecture (such as YOLOv4Tiny) as the student, and knowledge is transferred by using the distillation loss of the feature layer. For example, adding KL divergence constraints to the output of the detection head reduces the number of parameters of the lightweight model by 35% while maintaining more than 90% detection accuracy, realizing efficient deployment.

3.2 Parameter Quantization and Compression

Parameter quantization and compression are the core means to improve the inference efficiency in YOLOv4 lightweight research. Parameter quantization reduces the model storage and computation overhead by lowering the numerical precision of weights and activations. For example, converting 32-bit floating-point parameters to 8-bit fixed-point numbers can reduce the model volume to 1/4; at the same time, the mixed precision quantization strategy is used to retain 16-bit precision in key layers, which can alleviate the loss of detection accuracy. Experiments show that after 8-bit quantization, YOLOv4's inference speed on the COCO dataset is increased by 2.1 times, while the mAP only drops by 0.3%. Model compression removes redundant parameters through pruning: structured pruning evaluates the importance of convolution kernel channels, removes low-contribution filters, and combines fine-tuning to restore performance. For example, after pruning 40% of the parameters of the Darknet53 backbone network with channel sparsification technology, the model's computation is reduced by 35%, the detection speed reaches 45FPS, and the mAP of 48.7% can still be maintained. The collaborative application of these two methods can realize lightweight deployment in scenarios with limited hardware resources.

3.2.1 Low-bit quantization

Low-bit quantization reduces the model storage and computation by lowering the numerical precision of weights and activations. For example, converting 32-bit floating-point to 8-bit integer: after YOLOv4 adopts low-bit quantization, the model volume is reduced by 75% and the inference speed is increased by 2 times; at the same time, the mAP is maintained with a drop of less than 0.5% on the COCO dataset through calibration data and mixed precision strategy, balancing efficiency and accuracy.

3.2.2 Matrix decomposition

Matrix decomposition reduces the amount of computation in YOLOv4 lightweight through low-rank approximation, for example, decomposing the convolution kernel weight matrix into two low-rank matrices[6]. Taking a 3×3 convolution kernel as an example, the original parameters are 9; after decomposition

into 3×1 and 1×3 matrices, the parameters are reduced to 6, reducing the amount of computation while maintaining the feature expression ability. This method improves the inference speed by reducing model redundancy, which is suitable for edge device deployment[6][7]. Low-rank[6]decomposition combined with channel pruning can further improve the lightweight effect of YOLOv4.

3.3 Hardware Acceleration Optimization

Hardware acceleration optimization improves the inference efficiency in YOLOv4 lightweight through the collaborative design of specialized hardware and algorithms. Acceleration architectures based on FPGA, ASIC or GPU can reduce the model computational complexity through parallel computing, memory optimization and low-precision quantization. For example, using the NVIDIA TensorRT framework to optimize the deployment of YOLOv4: through network layer fusion, utilization of sparsity after model pruning and INT8 quantization compression, the computation and memory occupation are significantly reduced; at the same time, the parallel processing of multi-scale feature maps is realized by combining CUDA kernel function optimization. Experiments show that on the Jetson AGX Xavier platform, the inference speed of the lightweight YOLOv4 accelerated by TensorRT is increased to 2.3 times that of the original model, the frames processed per second are increased from 28FPS to 65FPS, and the power consumption is reduced by 17%. Such hardware acceleration schemes balance model accuracy and real-time performance, and are suitable for edge computing scenarios such as autonomous driving.

3.3.1 Specialized hardware deployment

Specialized hardware deployment realizes the efficient operation of the YOLOv4 lightweight model by optimizing the adaptation between the model and the hardware architecture. For example, when deploying with FPGA, the model is fixed-point quantized combined with the design of parallel computing units, and the convolution operation is processed by hardware-level pipeline, which can achieve an inference speed of 35FPS under the image resolution of 1920×1080, and the power consumption is reduced to less than 3W, meeting the real-time detection requirements of the edge side.

3.3.2 Inference framework optimization

Inference framework optimization improves computational efficiency by streamlining the model structure and hardware acceleration. For example, using TensorRT to perform graph optimization and layer fusion on YOLOv4, and adopting FP16 quantization to reduce video memory consumption; redundant computations are reduced through memory reuse and operator replacement, and the final inference speed is increased by 3 times while maintaining more than 90% detection accuracy, achieving the goal of lightweight deployment.

4. Research on YOLOv4 Lightweight Methods

4.1 Lightweight Based on Network Structure Improvement

The lightweight method based on network structure improvement mainly reduces the amount of computation and the number of parameters by optimizing the model architecture, while maintaining the detection accuracy as much as possible. Among them, the lightweight improvement of YOLOv4 is usually carried out around the structural simplification of the backbone network, feature fusion module and detection head. For example, a study replaces the standard convolution in the YOLOv4 backbone network CSPDarknet53 with depthwise separable convolution, and designs a cross-stage sparse connection structure. By reducing the channel dimension of the convolution layer and reducing redundant computations, the number of model parameters is reduced by 36%. At the same time, a lightweight attention mechanism is introduced in the feature pyramid part, reducing the computational complexity of multi-scale feature fusion by 22%. Experiments show that on the premise of maintaining 85% of the average accuracy of the original version, the inference speed of the improved model is increased to 1.8 times, which is suitable for mobile real-time detection scenarios. Such methods achieve a good balance between performance and efficiency through structural redesign and operator optimization.

4.1.1 Lightweight backbone network design

The design of lightweight backbone network improves inference efficiency by optimizing the model structure and the number of parameters. The mainstream strategies include depthwise separable convolution, channel pruning and

attention mechanism compression. For example, in YOLOv4, MobileNetv3 is used to replace the CSPDarknet53 backbone network, and the depthwise separable convolution of the input layer is combined with the inverted residual structure, which reduces the number of parameters by about 35% and FLOPs by 50%, while the fluctuation of the MAP index is less than 2%, achieving a balance between detection accuracy and computational cost.

4.1.2 Feature fusion optimization

Feature fusion optimization improves detection efficiency by simplifying the structure, such as replacing the original PANet with depthwise separable convolution to reduce the number of parameters, retaining multi-scale information through cross-layer connections, and introducing adaptive weight allocation to strengthen key features, maintaining high detection accuracy on the premise of lightweight. For example, the Ghost module is used to compress redundant features, increasing the inference speed by more than 30%.

4.2 Lightweight Based on Model Compression

The lightweight method based on model compression reduces the computational complexity and storage requirements of YOLOv4 by optimizing the network structure and parameters, while maintaining the detection accuracy as much as possible. Common model compression technologies include parameter pruning, quantization, knowledge distillation and low-rank decomposition. Parameter pruning reduces the number of parameters by removing redundant neurons or channels; quantization converts floating-point weights into low-bit-width integers, greatly reducing memory occupation and computation overhead. Knowledge distillation improves the representation ability of small models by transferring the knowledge of large teacher models to train lightweight student models. For example, a study carries out channel pruning on the YOLOv4 backbone network, and combines 8-bit fixed-point quantization after cutting 20% of redundant parameters, which reduces the model volume by 60%, the amount of computation by 50%, and the inference speed to 2.3 times the original; the mAP accuracy only drops by 1.5% on the PASCAL VOC dataset. This method effectively balances detection performance and deployment efficiency, and is suitable for mobile or edge devices.

4.2.1 Channel pruning strategy

The channel pruning strategy realizes lightweight by removing redundant channels in the model convolution layer, and the core is to evaluate the channel importance and cut off the low-contribution parts. Common methods include L1/L2 norm or activation value analysis. For example, in the YOLOv4 backbone network, for the convolution layer with an input size of 416×416, the number of channels in a certain layer is reduced from 256 to 160 by sorting the channel L1 norm, with a pruning rate of 37.5%. While maintaining the mAP drop less than 1%, the amount of computation is reduced and the inference speed is increased by about 25%, effectively balancing accuracy and efficiency.

4.2.2 Quantization and low-precision inference

Quantization and low-precision inference realize lightweight by reducing the bit width of model weights and activations, such as converting 32-bit floating-point parameters to 8-bit integers to reduce computation and storage overhead[7]. Studies have shown that the adoption of dynamic quantization strategy can compress 75% of the parameters in the YOLOv4 model, while the mAP accuracy loss is less than 2%[2]. A typical case is the hierarchical quantization method proposed by Zhang Wei et al., which optimizes the quantization error by calibrating the threshold layer by layer, increasing the inference speed of the YOLOv4 model by 1.8 times without significantly reducing the detection accuracy[8].

4.3 Lightweight Based on Knowledge Distillation

The lightweight method based on knowledge distillation reduces the number of parameters while maintaining detection accuracy by transferring the knowledge of the teacher model to guide the training of the student model. Aiming at YOLOv4, common strategies include feature imitation, attention transfer and response distillation. For example, the teacher model uses the original YOLOv4, and the student model adopts the streamlined MobileNetV3 backbone network. During training, a channel attention imitation module is added to the feature extraction layer to force the student network to learn the feature focusing ability of the teacher model, and an adaptive feature alignment loss function is designed in the detection head. Experiments show that this method reduces the number of model parameters by 62% on the

PASCAL VOC dataset, and the detection accuracy mAP only drops by 1.3%. The key technology lies in the design of multi-level knowledge transfer paths and dynamic weight allocation mechanisms, which effectively balance model efficiency and accuracy.

4.3.1 Teacher-student model design

The design of the teacher-student model usually transfers the capability of the large model to the lightweight model based on knowledge distillation. For example, the original YOLOv4 is used as the teacher model, whose CSPDarknet53 backbone extracts multi-scale features; the student model uses MobileNetV3 to replace the backbone network, and realizes the compression of parameters from 80M to 12M through the multi-layer feature map alignment loss. At the same time, the adaptive temperature coefficient is applied to dynamically adjust the teacher's soft labels to guide the training of the student model, realizing a 4.2% improvement in the detection accuracy of small targets.

4.3.2 Distillation loss function optimization

In the YOLOv4 lightweight, the optimization of the distillation loss function improves the knowledge transfer ability of the small model to the teacher model through multi-level feature alignment and attention mechanism. For example, the structural similarity loss is introduced to replace the traditional MSE to enhance the consistency of feature distribution; at the same time, the adaptive temperature coefficient is used to dynamically adjust the weight of soft labels to balance the distillation efficiency of features of different categories, thereby reducing the accuracy loss of the small model and realizing efficient knowledge transfer.

5. Experiments and Result Analysis

5.1 Experimental Environment and Dataset

The experimental environments of existing related studies are based on the NVIDIA Tesla V100 GPU. The PyTorch framework is adopted to implement the training and lightweight improvement of the YOLOv4 model, and algorithm deployment is completed via the MMDetection toolchain. The PASCAL VOC 2012 and MS COCO 2017 datasets are selected as benchmark test sets. The former contains 20 object categories with a total of 17,000 annotated images, while the latter covers 80 object categories with more than 120,000 high-resolution images. The data diversity of these datasets provides fundamental support for verifying the generalization ability of the model [1][7]. In the validation of lightweight methods, channel pruning and the MobileNetv3 structure replacement strategy are adopted, combined with hybrid quantization technology to compress the model parameter count to 23% of the original model [5][7]. For edge computing scenarios, the experiments are deployed on the Raspberry Pi 4B platform, and TensorRT is used for model acceleration, verifying the real-time detection capability of the improved model on embedded devices. All experiments adopt 5-fold cross-validation, and ablation experiments show that the mAP of the lightweight model on the COCO dataset decreases by only 1.8%, verifying the effectiveness of the optimization strategies. The dataset is divided into training, validation, and test sets at a ratio of 8:1:1. Data augmentation methods include random rotation and color space transformation to improve the model robustness [8].

In relevant literature on the lightweight research of YOLOv4, public datasets such as PASCAL VOC and COCO are commonly used, covering annotations of multiple object categories. Preprocessing includes resizing images to a uniform size, normalization, and data augmentation such as Mosaic augmentation or random flipping to enhance the model generalization ability. For example, the COCO 2017 dataset contains 118k training images. During preprocessing, images are resized to a resolution of 416×416, random color distortion is applied to enhance contrast, and K-means clustering is used to optimize anchor box sizes, thereby reducing computational cost and balancing detection accuracy and speed.

5.2 Performance Comparison of Lightweight Models

In the performance comparison of lightweight models, different optimization methods yield significant differences in the accuracy, speed, and computational cost of YOLOv4. For instance, the MobileNet-YOLOv4 model based on depthwise separable convolution reduces the parameter count to 13% of the original model, boosts the inference speed to 45 FPS, but incurs a detection accuracy (mAP) drop of approximately 48% to 72.6%. In contrast, the Ghost-YOLOv4 model optimized by joint channel pruning and quantization reduces

redundant feature maps and compresses weights to 8 bits. It maintains an mAP of 74.5% while compressing the model volume to 19% of the original model, and achieves an inference speed of 32 FPS on embedded devices. By comparison, AutoYOLOv4 based on neural architecture search achieves a balanced performance of 80.7% mAP and 42 FPS on the Tesla V100, yet its model complexity remains higher than other lightweight schemes. The above results indicate that lightweight strategies need to be weighed according to hardware deployment scenarios and accuracy requirements, and the collaborative design of model compression and structural optimization is an effective approach to balance performance and efficiency.

5.2.1 Accuracy-speed trade-off analysis

The balance between accuracy and speed is mainly achieved through model compression and structural optimization. For example, replacing standard convolutional layers with depthwise separable convolution can reduce computational cost by approximately 30%, maintain over 90% of the original model accuracy, and increase the inference speed to 40 FPS—significantly better than the 25 FPS of the unoptimized model, making it suitable for scenarios with limited computing resources.

5.2.2 Effectiveness verification of different lightweight methods

The effectiveness verification of different lightweight methods is generally conducted by comparing indicators such as model parameter count, computational cost, inference speed, and detection accuracy. For example, after applying channel pruning to YOLOv4, the model parameter count is reduced by 58%, computational cost drops to 45% of the original model, GPU inference speed is increased to 78 FPS, and the mAP decreases by only 2.1%. This demonstrates that the method can significantly improve efficiency with minor accuracy loss.

5.3 Practical Application Case Analysis

In the field of intelligent security, the practical application of the YOLOv4 lightweight model has significant value. A city traffic monitoring system adopts the lightweight YOLOv4 model to realize real-time detection of vehicles and pedestrians: the original backbone network is replaced with GhostNet to reduce the amount of computation, and channel pruning technology is introduced to cut redundant parameters, reducing the number of model parameters by 68%. To adapt to edge device deployment, the model is processed by 8-bit integer quantization, and the amount of computation is reduced to 35% of the original model. Practical tests show that the lightweight model achieves a detection speed of 42FPS in 1080P video streams, 3.2 times higher than the original YOLOv4, and the average accuracy mAP@0.5 only drops by 1.7%. The system is successfully deployed on ARM architecture embedded devices, realizing all-weather target recognition through low-power hardware, which verifies the feasibility of lightweight technology in resource-constrained scenarios. This method not only reduces the dependence on GPU servers, but also saves 45% of hardware procurement and maintenance costs, promoting the popularization of intelligent security towards edge computing.

5.3.1 Mobile deployment effect

Model pruning, quantization, and lightweight backbone network design can greatly compress parameter volume and computational cost, enabling efficient inference on edge devices. Related studies show that after replacing the original CSPDarknet53 backbone with GhostNet, the model parameter count is reduced by 58% to 6.2 million, the model size is compressed to 23 MB, the mobile inference speed reaches 25 FPS, and mAP decreases by only 2.3%, meeting the requirements of real-time detection.

5.3.2 Embedded device optimization

The optimization of embedded devices realizes YOLOv4 lightweight mainly through model compression and acceleration. For example, depthwise separable convolution is used to replace standard convolution to reduce the amount of computation, and channel pruning is combined to remove redundant parameters, reducing the model volume by 78%. At the same time, INT8 quantization is introduced to compress the weight precision, the inference speed is increased by 2.3 times, real-time detection of 30FPS is realized on the Raspberry Pi 4B, and the mAP only drops by 1.7%, balancing accuracy and deployment efficiency.

6. Conclusion and Prospect

6.1 Research Summary

Existing research indicates that lightweight optimization strategies for the YOLOv4 model primarily revolve around three core directions: model pruning, quantization compression, and network structure refinement. For model pruning,

advanced channel pruning schemes have effectively reduced the parameter count of YOLOv4 while preserving detection accuracy, and dynamic pruning methods have further enhanced pruning efficiency. In quantization, mixed-precision quantization is widely adopted to lower model storage overhead and computational latency. Regarding structural optimization, integrating MobileNetv3 into the YOLOv4 backbone reduces the total number of parameters by 83%, making the model well-suited for mobile deployment. Additionally, improved feature fusion modules enable real-time inference on edge devices. Nevertheless, current approaches still struggle with the trade-off between lightweight design and detection accuracy, particularly the issue of feature degradation in complex scenes. Going forward, more efficient lightweight frameworks should be explored by combining techniques such as knowledge distillation and adaptive quantization [8][9].

6.2 Future Research Directions

Future research can explore the optimization of lightweight YOLOv4 models from multiple dimensions. First, model structure innovation is the core, such as developing more efficient lightweight backbone networks and reducing redundant computation through cross-layer feature reuse or dynamic sparse convolution. Second, combining adaptive dynamic inference techniques to dynamically adjust the computational path for inputs of varying complexity, balancing real-time performance and accuracy. In addition, hardware–software co-optimization needs further breakthroughs, such as designing dedicated quantization strategies or operator fusion schemes for edge computing chips to improve deployment efficiency. Meanwhile, exploring multimodal lightweight methods, such as fusing sensor data from infrared, LiDAR, and other sources, to enhance detection robustness via lightweight feature fusion modules without excessively increasing the number of parameters. Finally, research on the collaborative mechanism of knowledge distillation and data augmentation in the lightweight process is also worthy of attention, especially how to maintain model generalization ability in few-shot scenarios. Future studies should more precisely resolve the contradiction between accuracy loss and computational efficiency while compressing the model scale [10].

6.2.1 More efficient lightweight methods

Replacing the original backbone network with GhostNet constitutes one of the efficient solutions for lightweighting YOLOv4. GhostNet generates redundant feature maps through the Ghost module, which drastically cuts down the parameter count. Relevant literature demonstrates that this scheme reduces the model's computational load (FLOPs) by 52%, compresses the number of parameters by 48%, incurs merely a 0.8% accuracy drop, and elevates the detection speed by 67%. It realizes real-time inference on edge devices while preserving the model's target recognition performance.

6.2.2 Cross-platform adaptation optimization

Cross-platform adaptation optimization improves the compatibility of YOLOv4 by adjusting the model structure and introducing an efficient inference framework. For example, the TensorRT acceleration framework is adopted to convert the model to the ONNX standard format and then deploy it to the NVIDIA Jetson platform; the computational delay is reduced through FP16 quantization and layer fusion technology, and the inference speed is increased by about 30% under the ARM architecture, realizing efficient real-time detection of edge devices.

References

[1] Li J Q. (2023). Research on post-disaster personnel detection algorithm based on deep learning[D]. Qinhuangdao: Yanshan University. https://doi.org/10.27440/d.cnki.gysdu.2023.002145.

[2] Zhu H Y. (2024). Research on power target detection algorithm for inspection images of transmission lines[D]. Qingdao: Qingdao University of Science and Technology. https://doi.org/10.27264/d.cnki.gqdhc.2024.000893.

[3] Chen Z L, Chen Y R, Li Q Z, Zhong R, Zhu Y K. Research on indirect monocular pose estimation method for spacecraft based on deep learning[J]. Flight Control and Detection, in press.

[4] Zhang L H. (2022). Research on target detection application based on RK3399Pro and YOLO[D]. Beijing: University of Chinese Academy of Sciences (Institute of Optoelectronics, Chinese Academy of

Sciences). https://doi.org/10.27543/d.cnki.gkgdk.2022.000046.

[5] Zhang X H. (2025). Method of infrared-visible image fusion and target detection in complex sea surface scenes[D]. Guangzhou: Guangdong University of Technology. https://doi.org/10.27029/d.cnki.ggdgu.2025.003872.

[6] Jiang F, Xu Z T, Cui D Y, Yao H W, Zhang H Y, Han Y. (2025). Application of large artificial intelligence models in the prevention and control of forest and grass pests[J]. World Forestry Research, 38(06):43-48. https://doi.org/10.13348/j.cnki.sjlyyj.2025.0101.y.

[7] Feng K Y. (2022). Sparse optimization and search of deep neural network structures[D]. Xi'an: Xidian University. https://doi.org/10.27389/d.cnki.gxadu.2022.003449.

[8] Wang X W, Li X, Zhu Z H, Qin T, Zhu H, Luo A Q. Multi-class weed detection model based on improved YOLO11n[J]. Journal of Tianjin University of Technology, in press.

[9] Yang Y C. (2022). Research on model compression method based on quantization[D]. Guangzhou: South China University of Technology. https://doi.org/10.27151/d.cnki.ghnlu.2022.002479.

[10] Wang Q, Wang Z Q, Liu H, Liang M, Zhang Y C, Lin Y. (2025). Evolution law, application efficiency and future prospect of YOLO series models in wheat ear detection[J]. Hubei Agricultural Sciences, 64(12):218-227. https://doi.org/10.14088/j.cnki.issn0439-8114.2025.12.037.

