Knowledge-Based Systems 287 (2024) 111432

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Evolutionary channel pruning for real-time object detection

Changcai Yang [a,b,c,1], Zhijie Lin [a,1], Ziyang Lan [a], Riqing Chen [a,b], Lifang Wei [a,b], Yizhang Liu [d,*]

[a] College of Computer and Information Sciences, Fujian Agriculture and Forestry University, Fuzhou 350002, China
[b] Center for Agroforestry Mega Data Science, School of Future Technology, Fujian Agriculture and Forestry University, Fuzhou 350002, China
[c] Key Laboratory of Smart Agriculture and Forestry (Fujian Agriculture and Forestry University), Fujian Province University, Fujian Agriculture and Forestry University, Fuzhou 350002, China
[d] School of Software Engineering, Tongji University, Shanghai 201804, China

ARTICLE INFO

Keywords:
Object detection
Channel pruning
Evolutionary
Channel information mixing convolution

ABSTRACT

Real-time object detection plays a crucial role in edge devices applications. Pruning methods are usually used to effectively eliminate redundant parameters of the object detection network so that it can detect objects efficiently. However, traditional pruning methods often result in a significant drop in accuracy, requiring time-consuming fine-tuning to restore the accuracy of the network. To address this issue, we propose an evolutionary channel pruning (ECP) method to reduce the redundant parameters in the network. Our proposed ECP method effectively reduces parameter redundancy and computation complexity in object detection networks while maintaining detection accuracy. Additionally, we introduce a novel Channel Information Mixing Convolution (CIMConv) that leverages more cost-effective operations to achieve higher accuracy and reduce the complexity associated with standard convolution. By applying our proposed ECP and CIMConv to the existing object detection methods, we achieve a superior balance between accuracy and complexity compared to state-of-the-art detectors. Notably, on challenging public datasets such as GTSDB, S^2TLD, TT100K, Wider Face, and Microsoft COCO, our proposed ECP substantially decrease the number of parameters and FLOPs of YOLOv5, simultaneously improving detection accuracy.

1. Introduction

Real-time object detection is a pivotal computer vision task [1–4] with wide-ranging applications, including autonomous driving [5,6] and multi-object tracking [7,8]. Currently, deep learning-based object detection algorithms have shown powerful capabilities [9], with the YOLO series emerging as one of the most popular frameworks. Notably, versions such as YOLOv5/6/7/8 [1,10–12], YOLOX [13], and PP-YOLOE [14] have made significant advancements in achieving a favorable trade-off between average precision (AP) and latency, making them widely applicable in real-world scenarios. Despite the significant progress achieved, it is inevitable that the convolutional layers of these models contain redundant channels. In addition, utilizing standard convolution (SC) ensures network accuracy but also introduces high complexity. These redundancies not only limit the detection speed of the network but may also lead to overfitting of the model [15].

Recently, researchers have proposed several efficient networks for object detectors, such as YOLOv6 [10], YOLOv7 [11], and YOLOv8 [12]. However, these structures are manually designed and cannot automatically determine the appropriate number of channels for each convolutional layer. Furthermore, the convolution operations used in these architectures are either excessively complicated or restrict the effective fusion of information across channels. With the emergence of Neural Architecture Search (NAS) [16], the discovery of superior network architectures has become more feasible. Nonetheless, it is important to note that NAS requires substantial computational resources and time costs. In addition, although depth-wise separable convolution (DSC) and group convolution (GC) alleviate the computational burden of convolutions, they constrain the inter-channel information mixing, resulting in inferior network accuracy. Compressing lightweight object detectors poses a significant challenge due to the relatively low redundancy of parameters and operations in real-time object detection networks and excessive compression can lead to a decrease in accuracy.

Previous works primarily focus on improving the accuracy of object detection networks by proposing various modules, resulting in a large number of parameters. They often overlook the importance of reducing network complexity. In this paper, we propose a novel evolutionary

* Corresponding author.
E-mail addresses: changcaiyang@gmail.com (C. Yang), 1211193011@fafu.edu.cn (Z. Lin), lzy121ly@163.com (Z. Lan), riqing.chen@fafu.edu.cn (R. Chen), weilifang1981@163.com (L. Wei), lyz8023lyp@gmail.com (Y. Liu).
[1] These authors contributed equally to this work.

https://doi.org/10.1016/j.knosys.2024.111432
Received 10 September 2023; Received in revised form 6 January 2024; Accepted 20 January 2024
Available online 27 January 2024
0950-7051/© 2024 Elsevier B.V. All rights reserved.

channel pruning (ECP) technique to tackle the problem of suboptimal channel allocation in convolutional layers. Our proposed ECP method can efficiently identify excellent network architectures by adaptively removing redundant channels in convolutional layers. Specifically, we adopt the evolutionary search algorithm to rapidly determine the appropriate number of channels in convolutional layers and evaluate sub-nets with Adaptive Batch Normalization. Our proposed ECP method enables fast and adaptive determination of channel numbers for each convolutional layer and removes the unimportant channels. Moreover, to alleviate the high complexity of SC, we introduce Adjacent Channel Convolution (ACC) derived from OGC [17], which significantly reduces the number of parameters and computations in convolutions. Based on ACC, we address the limitations of SC by introducing Channel Information Mixing Convolution (CIMConv), which not only reduces computational costs but also enhances information fusion across channels. By replacing some convolutions with CIMConv in the network, we achieve reduced complexity and improved accuracy. Additionally, we have observed that CIMConv performs better in convolutional layers with a larger number of channels. For instance, in YOLOv5, utilizing CIMConv in the SPPF module maximizes its effectiveness. Because our proposed ECP method and CIMConv are compatible, they can be added to the different state-of-the-art object detection networks to reduce the number of parameters and computations while improving the effectiveness. Extensive experiments on multiple datasets demonstrate that the object detection networks with our proposed ECP method and CIMConv (YOLOC), as shown in Fig. 1, significantly reduces the number of parameters, achieving a better balance between network accuracy and complexity.

The contributions of this paper are summarized in three folds. Firstly, we propose a novel ECP method that enables the quick identification of excellent network architectures by allocating appropriate channel numbers for convolutional layers. This approach effectively reduces the number of parameters and computational costs while maintaining detection accuracy. Secondly, we utilize a novel convolution operation ACC to construct a lightweight convolution module called CIMConv. By replacing SC with CIMConv, we not only reduce computational costs but also facilitate information mixing across channels. Thirdly, the proposed ECP and CIMConv can be added to the different state-of-the-art object detection. The proposed ECP and CIMConv could significantly reduce the parameter and computational complexity of the object detection networks while maintaining high accuracy.

2. Related works

2.1. Real-time object detectors

Despite achieving high accuracy, two-stage object detection algorithms [18–21] often suffer from issues such as high redundancy and slow inference speed. To address these problems, YOLO [1,10–14,22–25] has emerged as one of the most popular real-time object detection algorithms. YOLO performs object classification and localization in a single step, greatly enhancing detection efficiency and overcoming the limitations of two-stage detection algorithms. In addition to the YOLO series, other object detection algorithms such as SSD [26], FCOS [27,28], and EfficientNet [29] offer their own distinct advantages. Advanced real-time object detection algorithms typically demand both faster speed and higher accuracy. Consequently, these methods strive to achieve more efficient feature extraction and employ simpler operations to extract richer features. In this paper, we propose new optimization schemes to address the problem of redundant parameters and operations in object detection algorithms, with the goal of achieving faster speed and higher accuracy for real-time object detection.

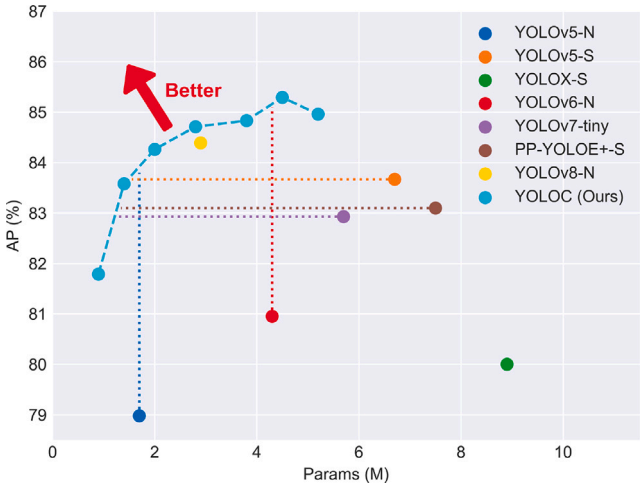


Fig. 1. Comparison with other real-time object detectors, our proposed methods achieve state-of-the-arts performance.

2.2. Network pruning

Network pruning is a promising technique to improve the efficiency of deep neural networks, and it can be categorized into two types: unstructured pruning [30,31] and structured pruning [32–40]. Unstructured pruning involves the removal of individual weights from the network, which leads to a significant reduction in the number of parameters. However, unstructured pruning often requires specialized hardware or libraries to achieve actual speedup. By contrast, structured pruning removes an entire group of parameters, such as an entire channel, which enables direct acceleration and makes it more widely applicable. The effectiveness of pruning algorithms depends on the choice of pruning criteria [32–36], sparsification methods [37,38], and guiding strategies [39,40]. Recent works have proposed various pruning criteria, including magnitude-based [32,34], geometric median-based [35], scaling factor-based [33], and rank-based [36]. Sparsification is essential to remove redundant parameters safely and avoid significant accuracy degradation. MetaPruning [41] is a promising approach that trains a network to predict the weight values of the pruned model. EagleEye [39] identifies the factors contributing to substantial accuracy degradation after pruning and proposes Adaptive Batch Normalization, which enables efficient evaluation of sub-net accuracy and guides the pruning algorithm in discovering optimal subnets. These approaches provide new insights into the design of efficient neural networks.

2.3. Lightweight network

AlexNet [42] shows impressive feature extraction capabilities, but its network complexity is extremely high. Subsequent networks including VGG [43], ResNet [44], and DarkNet [24,25] adopt the SC as a fundamental element in the architecture of their backbone networks. Although SC-based networks perform well, their large parameter sizes render them unsuitable for edge device deployment. In response, researchers develop various lightweight networks including MobileNet [45–47], ShuffleNet [48,49], GhostNet [50], and GSNet [51], etc. MobileNet uses DSC to replace SC, albeit with a certain degree of accuracy reduction. ShuffleNet introduces GC, which mixes channel information within the group to mitigate accuracy reduction. GhostNet achieves enhanced feature extraction by sequentially employing both SC and DSC. RepVGG uses a multi-branch structure during network training to ensure model accuracy, and converts it to a single-branch structure during inference, thereby accelerating the inference speed. More recently, variants of these lightweight networks

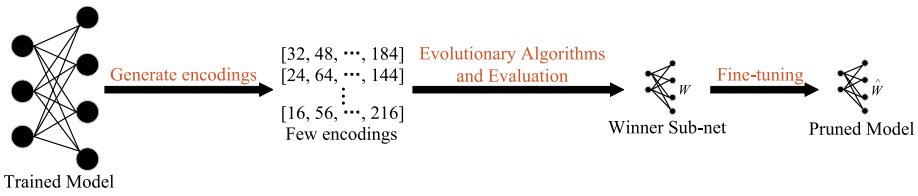


Fig. 2. The process of our proposed evolutionary channel pruning method.

emerge, such as GhostNetV2 [52] and RepGhost [53]. GhostNetV2 builds upon GhostNet by adding attention modules. RepGhost implements a hardware-efficient Ghost module via re-parameterization.

3. Methods

In this section, we first present a new evolutionary pruning method to significantly reduce the number of parameters and computational complexity while maintaining high accuracy. Then, we describe the proposed channel information mixing method. Finally, we give the implementation details.

3.1. Evolutionary channel pruning

The object detection network are typically large and complex, often containing a significant number of redundant parameters. To effectively eliminate these redundancies, pruning methods are commonly employed. However, the traditional pruning method often results in substantial accuracy degradation, requiring time-consuming fine-tuning to restore the network's accuracy. Moreover, fine-tuning and evaluating a vast number of sub-nets with varying pruning rates is impractical due to the high computational cost. Therefore, there is a need for a more efficient pruning method that can identify and remove redundant parameters while maintaining accuracy.

The previous channel pruning method [39] involved enumerating a large number of sub-nets, followed by fine-tuning and selecting the best-performing model. However, our proposed evolutionary channel pruning approach adopts an evolutionary algorithm with a small set of sub-nets to efficiently search the optimal sub-net, as shown in Fig. 2. A sub-net is represented as an array where each element indicates the number of channels in a convolution layer. In our evaluation technique, we adjust the statistics of the BN layer instead of fine-tuning it for each sub-net, which greatly reduces the time required. We only fine-tune the top sub-nets obtained by the evolutionary algorithm.

Specifically, to handle the excessive number of sub-nets, we propose an evolutionary algorithm-based method to determine the optimal number of channels for each convolution layer. Our proposed method is more specific than the enumeration method [39] and can quickly converge to obtain excellent sub-nets. To overcome the time-consuming fine-tuning process, we adopt Adaptive Batch Normalization [39], which utilizes small batches of data to adjust the BN layer's mean and variance in the network. Compared with traditional pruning methods, the proposed evolutionary channel pruning method is more reliable and efficient that can quickly evaluate sub-nets and identify networks with high accuracy.

3.1.1. Search for the number of channels

We construct an encoding $[c_1, c_2, \ldots, c_l]$ using the number of channels in each convolution layer of the network, where c_i denotes the number of channels in the ith convolution layer. In this way, every encoding represents a sub-net. However, the number of possible encodings is extremely large, making it impractical to enumerate them all. To efficiently search for high-accuracy pruning networks, we use evolutionary search, which only requires a small set of encodings. By evaluating these encodings and generating new ones, we can approach the optimal network more purposefully and converge quickly.

Given a pruning convolutional neural network under the constraints of the number of parameters or FLOPs, the problem can be formulated as

$$\left[c_1, c_2, \ldots, c_l\right]^* = \arg\min_{c_1, c_2, \ldots, c_l} \mathcal{L}\left(\mathcal{A}\left(c_1, c_2, \ldots, c_l; w\right)\right), \text{ s.t. } \mathcal{F} < C, \tag{1}$$

where $\mathcal{L}$ is the loss function; $\mathcal{A}$ is the convolutional neural network model; l is the number of convolution layers of the network, and w is the parameter value of the network. $\mathcal{F}$ is the number of parameters or FLOPs corresponding to the sub-net after determining the number of channels. C is the constraints of the number of parameters or FLOPs. Our aim is to find the network with the minimum loss under the constraints. The process of our proposed evolutionary channel pruning method is shown in Fig. 3. We start by generating n random encodings that satisfy the constraints and evaluate their corresponding sub-nets for accuracy. Subsequently, the k sub-nets that exhibit the highest accuracy are selected and subjected to mutation and crossover to obtain new sub-nets. During the mutation process, some values of an encoding are randomly changed, while crossover combines the values of two encodings to create a new one. The sub-nets whose encodings do not satisfy the parameter or computation constraints are removed from the candidate sub-nets. By repeatedly selecting the k sub-nets with the highest accuracy and generating new encodings through mutation and crossover, we can efficiently obtain a pruning network that meets the constraints while maintaining high accuracy. Our approach is more efficient than exhaustively enumerating a large number of sub-nets.

3.1.2. Adaptive batch normalization

The BN layer in a convolutional neural network is used to normalize the features. In this paper, the output of the BN layer is defined as

$$y = \gamma \frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}} + \beta, \tag{2}$$

where γ and β are trainable scale and bias terms, respectively. ϵ is a term with small value to avoid zero division. Given the batchsize N_{bs}, the statistics μ and σ^2 in the BN layer can be calculated as

$$\begin{aligned} \hat{\mu} &= E[X] = \frac{1}{N_{bs}} \sum_{i=1}^{N_{bs}} x_i, \\ \hat{\sigma}^2 &= D[X] = \frac{1}{N_{bs} - 1} \sum_{i=1}^{N_{bs}} \left(x_i - \hat{\mu}\right)^2. \end{aligned} \tag{3}$$

In the training phase, μ and σ^2 can be calculated by moving mean and variance:

$$\mu_t = m\mu_{t-1} + (1 - m)\hat{\mu}, \quad \sigma_t^2 = m\sigma_{t-1}^2 + (1 - m)\hat{\sigma}^2, \tag{4}$$

where m is the momentum coefficient, and t is the iteration number. The final values μ_T and σ_T^2 are used in testing phase when the total number of iterations is T.

After pruning, the statistics μ and σ^2 used in the BN layer are obtained from the original network structure. However, since the network structure changes after pruning, these statistics μ and σ^2 are no longer appropriate and reliable. As illustrated in Fig. 4(a), in the original network, the values of the three input nodes are A, B, and C, with coefficients of 1, 2, and 3, respectively. Therefore, the output of this layer is $A+2B+3C$. In the BN layer, assuming that the current value of μ being used is $A+2B+3C$, and then the result of $x-\mu$ is 0. However,

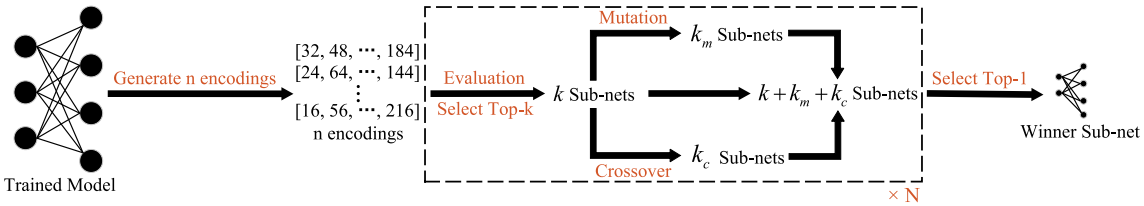


Fig. 3. The process of searching for the number of channels with the evolutionary algorithm.

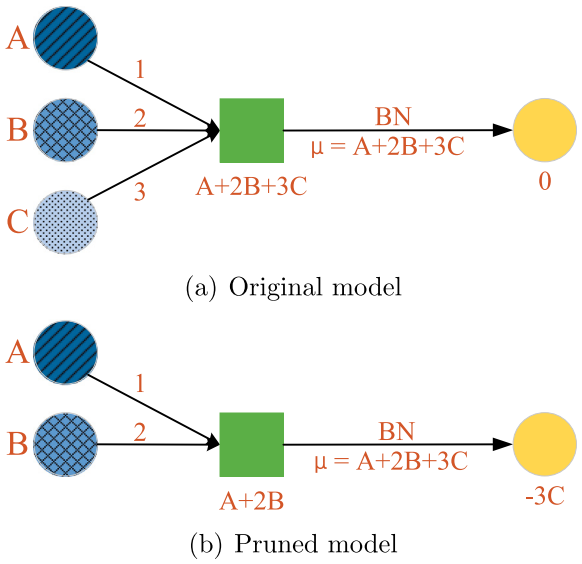


Fig. 4. Illustration of the differences of network output before and after pruning.

Algorithm 1 Evolutionary Channel Pruning Algorithm

Input: Number of Random Encodings: n, Constraints: C, Number of Excellent Sub-nets: k, Number of Mutation: k_m, Number of Crossover: k_c, Number of Iterations: N, Trained Model: $\mathcal{A}$.
Output: Model after Pruning: $\hat{\mathcal{A}}$.

$\mathcal{G}_n = \{\mathcal{A}_1, \mathcal{A}_2, \dots, \mathcal{A}_n\} = \text{Random}(\mathcal{A})$, s.t. $\mathcal{F}_i = \text{FLOPs}(\mathcal{A}_i) < C$;
$\mathcal{G}_{topk} = \text{Topk}(\mathcal{G}_n)$;
$j = 0$;
while $j < N$ **do**
  $\mathcal{G}_{k_m} = \{\mathcal{A}_{m1}, \mathcal{A}_{m2}, \dots, \mathcal{A}_{mk_m}\} = \text{Mutation}(\mathcal{G}_{topk})$, s.t. $\mathcal{F}_{mi} = \text{FLOPs}(\mathcal{A}_{mi}) < C$;
  $\mathcal{G}_{k_c} = \{\mathcal{A}_{c1}, \mathcal{A}_{c2}, \dots, \mathcal{A}_{ck_c}\} = \text{Crossover}(\mathcal{G}_{topk})$, s.t. $\mathcal{F}_{ci} = \text{FLOPs}(\mathcal{A}_{ci}) < C$;
  $\mathcal{G}_{k+k_m+k_c} = \mathcal{G}_{topk} \bigcup \mathcal{G}_{k_m} \bigcup \mathcal{G}_{k_c}$;
  $\mathcal{G}_{topk} = \text{Topk}(\mathcal{G}_{k+k_m+k_c})$;
  $j = j + 1$;
end while
$\hat{\mathcal{A}} = \text{Top1}(\mathcal{G}_{topk})$;
return $\hat{\mathcal{A}}$;

after removing one of the nodes, as shown in Fig. 4(b), the input of the BN layer becomes $A+2B$. If the original μ is still used, then $x-\mu$ will be $-3C$, and the output results will differ significantly from the original. This is the primary reason for the significant drop in accuracy observed after pruning.

Since the statistics of the BN layer are not suitable for the pruned network, recalculating these statistics is necessary. To this end, we freeze all parameters in the network and update μ and σ^2 using a small batch of data based on Eq. (4), without performing backpropagation. This step can be performed in a much shorter time than fine-tuning. Once the BN layer statistics are adjusted, the model's accuracy can be largely recovered.

3.1.3. Our proposed evolutionary channel pruning

Our proposed evolutionary channel pruning method is summarized in Algorithm 1. It can be iteratively applied, as illustrated in Fig. 5. This iterative approach not only reduces the network to the desired size, but also helps to mitigate the drastic drop in accuracy that may occur when a large number of parameters are removed at once.

3.2. Channel information mixing

3.2.1. Adjacent channel convolution

SC typically involves a large number of parameters, which makes it challenging to deploy in resource-constrained settings. To address this problem, researchers often use variants such as DSC or GC, which reduce the number of parameters. However, as the number of parameters reduces, these approaches have difficulty giving truly satisfactory results. In this paper, we introduce a new convolutional operation called ACC to overcome the limitations of DSC and GC. Fig. 6 provides visual illustrations of SC, DSC, GC, and ACC. SC can give highly accurate results due to its ability to mix information across channels. However, DSC involves independent convolution operations for each channel, while GC limits information interaction to within a group. But the proposed ACC not only ensures interaction between channels but is also less complex than SC in generating the same number of feature maps. OGC [17] unifies these convolutions, and ACC has the lowest complexity among the convolutions capable of interacting channel information in different groups. ACC has higher complexity compared to DSC, but it gives a significant improvement in accuracy. In addition, ACC mixes information on more channels with similar complexity as GC, which combines every two channels.

3.2.2. CIMConv

In a typical convolutional neural network, spatial information flows gradually to channel information as the backbone network processes image data. However, when spatial information is compressed and channel information is expanded, some semantic information is inevitably lost. SC keeps all input channels connected to the output feature map, but DSC completely disconnects the output feature map from the other input channels. Inspired by the structure of GhostNet [50], we propose CIMConv based on ACC, as depicted in Fig. 7. CIMConv aims to retain as many of the connections between the output feature map and input channels as possible with a low time complexity by using ACC. These connections are more efficient than those obtained using GC. CIMConv combines SC and ACC to generate the output feature map, where SC is used for half of the output feature map and ACC is used for the other half. To enhance the information interaction across channels, we introduce Channel Shuffle, which interleaves the two parts of the feature maps. Regarding the activation function, although SiLU can lead to improved performance, ReLU is more efficient for inference during model deployment.

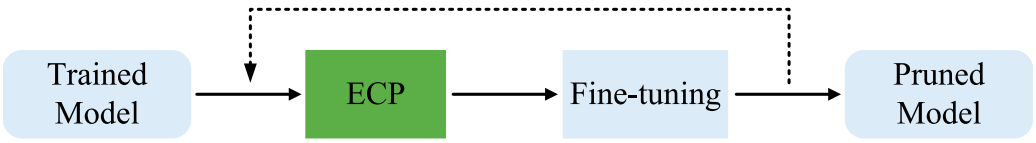


Fig. 5. Illustrations of the iterative approach to pruning.

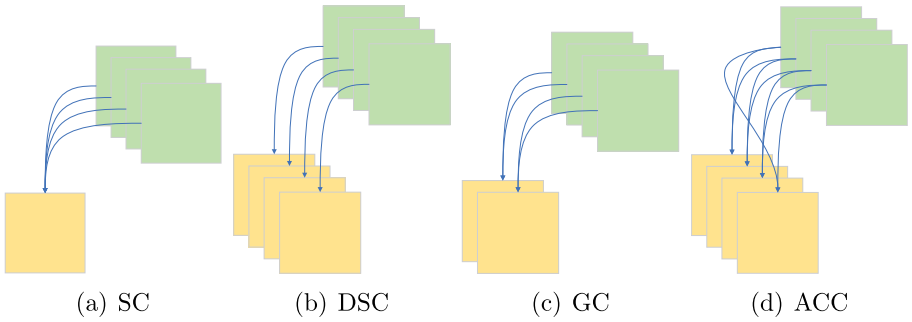


Fig. 6. Illustrations of various convolution operations.

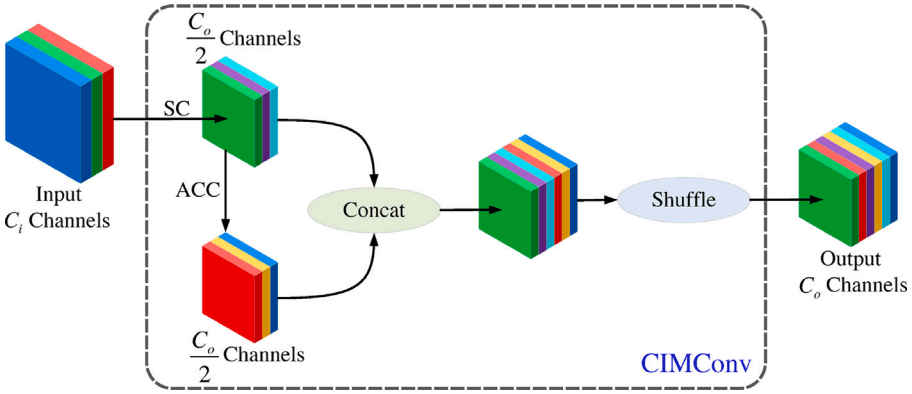


Fig. 7. The structure of the CIMConv module.

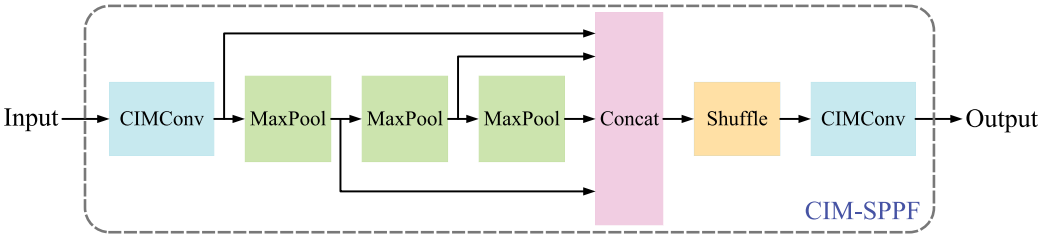


Fig. 8. The structure of the CIM-SPPF module.

The time complexities of SC, DSC, GC, and CIMConv are as follows:

$$\begin{aligned} Time_{SC} &\sim O\left(W \cdot H \cdot K_w \cdot K_h \cdot C_i \cdot C_o\right), \\ Time_{DSC} &\sim O\left(W \cdot H \cdot K_w \cdot K_h \cdot 1 \cdot C_o\right), \\ Time_{GC} &\sim O\left(W \cdot H \cdot K_w \cdot K_h \cdot \frac{C_i}{g} \cdot C_o\right), \\ Time_{CIMConv} &\sim O\left[W \cdot H \cdot K_w \cdot K_h \cdot \frac{C_o}{2}\left(C_i + 2\right)\right], \end{aligned} \tag{5}$$

where W and H represent the width and height of the output feature map, respectively. K_w and K_h represent the width and height of the convolution kernel. C_i and C_o denote the number of input and output feature maps, respectively. Note that in GC, the number of groups is represented by the g.

If CIMConv replaces all convolutions in the network, it significantly increases the depth of the network, adversely affecting the inference speed. Additionally, a larger number of channels facilitate better information interaction between channels. As the network deepens, the channels become smaller but more in terms of quantity. Therefore, using CIMConv at this stage is a better choice. The analysis in Eq. (5) also demonstrates that replacing SC with CIMConv is more effective when there are more channels.

3.3. Implementation details

We apply our proposed evolutionary channel pruning method and CIMConv to YOLOv5-S and call the version with about 30% of the original FLOPs YOLOC-N and that with about 80% of the original FLOPs YOLOC-S. In addition, the SPPF of YOLOv5 contains the largest

Table 1
The structure of a pruned network (Compress the FLOPs of YOLOv5-S to a model that is approximately 80% of the original.). The value before "/" is the pruned output channels, and the value after "/" is the original output channels.

backbone	
0	Conv(32/32)
1	Conv(64/64)
2	C3(32/32, 32/32, 64/64) Bottleneck(32/32, 32/32)
3	Conv(120/128)
4	C3(64/64, 48/64, 128/128) Bottleneck(56/64, 64/64, 64/64, 64/64)
5	Conv(248/256)
6	C3(128/128, 80/128, 232/256) Bottleneck(112/128, 128/128, 96/128, 128/128, 112/128, 128/128)
7	Conv(312/512)
8	C3(256/256, 216/256, 496/512) Bottleneck(136/256, 256/256)
9	SPPF(248/256, 512/512)
head	
10	Conv(232/256)
11	Upsample
12	Concat
13	C3(128/128, 120/128, 184/256) Bottleneck(72/128, 128/128)
14	Conv(80/128)
15	Upsample
16	Concat
17	C3(64/64, 64/64, 104/128) Bottleneck(64/64, 64/64)
18	Conv(120/128)
19	Concat
20	C3(128/128, 88/128, 224/256) Bottleneck(80/128, 128/128)
21	Conv(192/256)
22	Concat
23	C3(256/256, 208/256, 304/512) Bottleneck(168/256, 256/256)
24	Detect(27/27, 27/27, 27/27)

number of channels in the network. Therefore, it is the ideal location to replace SC with CIMConv in the SPPF. Since pooling operations at different scales produce features with varying receptive fields, we employ another Channel Shuffle operation to further enhance their mixing. Based on this, we propose the CIM-SPPF module, as depicted in Fig. 8. The proposed module has fewer parameters, lower computational cost than SPPF, and enables a more thorough mixing of information across channels.

We use the Pytorch framework to implement the models. In the context of the evolutionary search algorithm, a set of hyperparameters is designated for sub-net optimization. Specifically, the number of randomly generated encodings n is 200. The number of excellent sub-nets selected during each iteration k is 40. In addition, the number of new sub-nets generated using mutation and crossover is 100 for both k_m and k_c. The number of iterations N is 20.

In Table 1, we provide a pruned network that contains each layer of the network, which follows the official configurations of YOLOv5. In convolutional layers and modules containing convolutions, we specify the number of output channels. It can be seen that the network performs less pruning in the shallow layers to preserve more information, while in the deeper layers, only essential information is retained.

4. Experiments

To verify our proposed evolutionary channel pruning method and CIMConv, we compare the results obtained by YOLOC against those from state-of-the-art object detection algorithms on GTSDB [54], S^{2}TLD [55], TT100K [56], Wider Face [57], and Microsoft COCO [58] datasets and perform the ablation experiments.

4.1. Datasets

We conduct experiments using five datasets, namely GTSDB [54], S^{2}TLD [55], TT100K [56], Wider Face [57], and Microsoft COCO [58]. The GTSDB dataset, which consists of 600 training images and 300 testing images, has been widely used for traffic sign detection. It is divided into four categories: prohibitory, mandatory, danger, and other. The S^{2}TLD dataset encompasses a total of 5786 images, comprising 1222 images with a resolution of 1080 × 1920 pixels and 4564 images with a resolution of 720 × 1280 pixels. The dataset is categorized into four distinct classes, namely red, yellow, green, and off. Subsequently, we randomly partitioned the dataset into training and testing sets in accordance with a ratio of 7:3, respectively. The TT100K dataset comprises 9176 images, encompassing 6105 training images and 3071 testing images. Given the significant variance in data distribution among diverse categories in this dataset, 45 categories with sufficiently large data are selected for our experiments. The Wider Face dataset includes 12876 training images and 3226 testing images of a single category, i.e., face. The Microsoft COCO dataset is a vast object detection dataset. We use 118287 images as the training set and 5000 images as the test set.

4.2. Evaluation metrics

The evaluation metrics include $AP_{0.5}$, AP, $F1$, the number of parameters (Params), floating-point operations (FLOPs), and frame per second (FPS). $AP_{0.5}$ is the average precision for all categories with accuracy evaluation IoU thresholds at 0.5. Meanwhile, AP is the average precision with the IoU threshold taken in steps of 0.05 from 0.5 to 0.95. $F1$ is the harmonic mean of Precision and Recall.

4.3. Results on GTSDB

In this section, we evaluate the performance of our YOLOC-N and YOLOC-S on the GTSDB dataset, and compare them with other famous object detectors including Faster R-CNN [20], SSD [26], Cascade R-CNN [21], YOLOv5 [1], YOLOX [13], YOLOv6 [10], YOLOv7 [11], PP-YOLOE+ [14], and YOLOv8 [12]. In the experiments, we re-train all the networks. Table 2 gives the values of quality of AP, $AP_{0.5}$, Params, FLOPs and FPS for the object detection methods. From Table 2, we note that our proposed YOLOC-N and YOLOC-S achieve the best complexity-accuracy trade-off. Compared to YOLOv5-N, YOLOC-N increases by 2.81% AP, while requiring fewer parameters. YOLOC-S achieves the best performance in terms of AP and $AP_{0.5}$ and outperforms the baseline YOLOv5-S on all metrics. YOLOC-S is more efficient, with 3.0M fewer parameters, 3.7G fewer FLOPs, and 129.9 higher FPS compared to PPYOLOE+-S, yet achieves a 2.19% higher AP. We also present a typical detection result of the baseline model and our method in Fig. 9. It can be seen that our YOLOC-S can reduce missed detections and false detections in complex conditions, which demonstrates the effectiveness of the proposed method.

4.4. Results on S^{2}TLD

To validate the robustness of our proposed method, we conduct experiments on the traffic light dataset S^{2}TLD. The comparative results with other state-of-the-art methods are presented in Table 3. As observed from the table, our model YOLOC-S achieves the highest accuracy among all models while having a lower parameter count and computational cost. Specifically, it outperforms the baseline model YOLOv5-S by 1.96% on AP, 1.74% on $AP_{0.5}$ and 12.8 on FPS, and has 2.5M fewer parameters, 3.9G fewer FLOPs, and 12.8 higher FPS compared to YOLOv5-S. On the other hand, YOLOC-N demonstrates superior accuracy even with the lowest parameter count. It has 0.9M fewer parameters than YOLOv5-N, and achieves higher AP and $AP_{0.5}$ by 0.35% and 0.60%, respectively. Fig. 10 illustrates the better detection results obtained from our proposed method, which further validates its superiority.

Table 2
Comparison on GTSDB dataset. The best results are boldfaced. The second-best results are underlined.

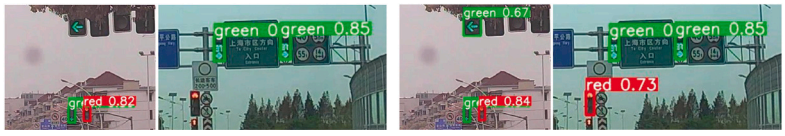
Model	Size	AP(%)	$AP_{0.5}$(%)	Params(M)	GFLOPs	FPS
Faster R-CNN [20]	800 × 1333	82.44	96.82	33.1	110.2	27.0
SSD [26]	640 × 640	68.14	92.84	22.9	137.5	44.2
Cascade R-CNN [21]	800 × 1333	83.90	97.50	69.2	202.1	22.5
YOLOv5-N [1]	640 × 640	78.98	96.04	1.7	**4.2**	**208.3**
YOLOX-S [13]	640 × 640	80.00	92.81	8.9	26.8	51.8
YOLOv6-N [10]	640 × 640	80.95	97.16	4.3	11.1	158.2
YOLOv7-tiny [11]	640 × 640	82.93	97.56	5.7	13.1	131.6
PP-YOLOE+-S [14]	640 × 640	83.10	**97.77**	7.5	16.4	55.3
YOLOv8-N [12]	640 × 640	84.39	96.43	2.9	8.1	178.6
YOLOv5-S [1]	640 × 640	83.67	96.70	6.7	15.8	169.5
YOLOC-N	640 × 640	81.79	96.44	**0.9**	4.9	**208.3**
YOLOC-S	640 × 640	**85.29**	**97.77**	4.5	12.7	185.2

(a) YOLOv5-S (b) YOLOC-S

Fig. 9. Traffic sign detection results on the GTSDB dataset. YOLOC-S reduces missed detections and false detections.

Table 3
Comparison on S^{2}TLD dataset. The best results are boldfaced. The second-best results are underlined.

Model	Size	AP (%)	$AP_{0.5}$ (%)	Params (M)	GFLOPs	FPS
Faster R-CNN [20]	800 × 1333	56.77	88.66	33.1	110.2	29.2
SSD [26]	640 × 640	37.72	69.64	22.9	137.5	50.4
Cascade R-CNN [21]	800 × 1333	60.01	93.22	69.2	202.1	25.4
YOLOv5-N [1]	640 × 640	59.73	90.64	1.7	**4.2**	**204.1**
YOLOX-S [13]	640 × 640	51.78	88.03	8.9	26.8	63.5
YOLOv6-N [10]	640 × 640	57.49	89.55	4.3	11.1	147.5
YOLOv7-tiny [11]	640 × 640	58.11	91.06	5.7	13.1	158.7
PP-YOLOE+-S [14]	640 × 640	54.27	87.92	7.5	16.4	68.2
YOLOv8-N [12]	640 × 640	62.48	91.66	2.9	8.1	181.8
YOLOv5-S [1]	640 × 640	61.08	91.81	6.7	15.8	172.4
YOLOC-N	640 × 640	60.08	91.24	**0.8**	5.0	200.0
YOLOC-S	640 × 640	**63.04**	**93.55**	4.2	11.9	185.2



(a) YOLOv5-S (b) YOLOC-S

Fig. 10. Traffic light detection results on the S^{2}TLD dataset. YOLOC-S reduces missed detections.

Table 4
Comparison on TT100K dataset. The best results are boldfaced. The second-best results are underlined.

Model	Size	AP (%)	$AP_{0.5}$ (%)	Params (M)	GFLOPs	FPS
Faster R-CNN [20]	640 × 1333	49.65	69.40	33.5	67.8	47.7
SSD [26]	640 × 640	48.00	75.58	28.1	146.5	54.2
Cascade R-CNN [21]	640 × 1333	55.63	72.26	69.3	120.3	42.9
YOLOv5-N [1]	640 × 640	56.00	76.52	1.7	**4.3**	**192.3**
YOLOX-S [13]	640 × 640	61.22	80.91	9.0	26.9	64.5
YOLOv6-N [10]	640 × 640	58.75	77.88	4.3	11.1	158.5
YOLOv7-tiny [11]	640 × 640	62.43	82.69	5.8	13.5	128.2
PP-YOLOE+-S [14]	640 × 640	56.81	74.96	7.7	16.9	65.0
YOLOv8-N [12]	640 × 640	59.54	78.04	2.9	8.1	181.8
YOLOv5-S [1]	640 × 640	63.79	83.78	6.8	16.2	172.4
YOLOC-N	640 × 640	62.53	83.46	**0.8**	5.1	**192.3**
YOLOC-S	640 × 640	**64.37**	**84.48**	4.6	13.0	181.8

(a) YOLOv5-S (b) YOLOC-S

Fig. 11. Traffic sign detection results on the TT100K dataset. YOLOC-S reduces missed detections.

Table 5
Comparison on Wider Face dataset. The best results are boldfaced. The second-best results are underlined.

Model	Size	AP (%)	$AP_{0.5}$ (%)	Params (M)	GFLOPs	FPS
Faster R-CNN [20]	736 × 1333	34.24	62.04	33.0	90.2	39.0
SSD [26]	640 × 640	27.07	52.56	22.5	136.9	54.0
YOLOv5-N [1]	640 × 640	35.83	68.22	**1.7**	**4.2**	**196.1**
YOLOX-S [13]	640 × 640	**41.21**	72.26	8.9	26.8	67.2
YOLOv6-N [10]	640 × 640	34.78	65.29	4.3	11.1	152.2
YOLOv7-tiny [11]	640 × 640	37.85	71.84	5.7	13.1	149.3
PP-YOLOE+-S [14]	640 × 640	33.63	59.17	7.5	16.3	74.6
YOLOv8-N [12]	640 × 640	37.44	67.74	2.9	8.1	172.4
YOLOv5-S [1]	640 × 640	39.91	<u>73.76</u>	6.7	15.8	166.7
YOLOC-N	640 × 640	37.74	70.94	**1.7**	<u>5.0</u>	<u>192.3</u>
YOLOC-S	640 × 640	<u>40.36</u>	**74.29**	4.5	12.7	178.6

(a) YOLOv5-S (b) YOLOC-S

Fig. 12. Face detection results on the Wider Face dataset. YOLOC-S reduces missed detections.

4.5. Results on TT100K

To further validate the efficacy of the proposed method, experiments are conducted on a distinct traffic sign dataset, TT100K. The experimental results of our methods and other detection approaches are presented in Table 4. From Table 4, we can see that our YOLOC-S outperforms other methods in terms of AP and $AP_{0.5}$, while requiring fewer parameters and FLOPs than SSD, YOLOv5-S, YOLOX-S, YOLOv7-tiny, and PP-YOLOE+-S. Specifically, YOLOC-S outperforms YOLOv7-tiny by 1.94% and 1.79% in terms of AP and $AP_{0.5}$, respectively, while reducing the number of parameters and FLOPs by 1.2M and 0.5G, respectively. Moreover, even with mere 0.8M parameters, YOLOC-N performs better on AP and $AP_{0.5}$ than SSD, YOLOv5-N, YOLOX-S, YOLOv6-N, YOLOv7-tiny, PP-YOLOE+-S, and YOLOv8-N. Fig. 11 presents detection results on the TT100K dataset. This experiment further validates the effectiveness of our proposed method.

4.6. Results on Wider Face

To validate the effectiveness and robustness of our proposed method, we conduct a comprehensive comparison with state-of-the-art algorithms on the Wider Face dataset. The experimental results are presented in Table 5. From Table 5, we can see that YOLOC-S achieves satisfactory experimental results while having a smaller number of parameters and computational costs. Compared to YOLOX-S, YOLOC-S achieves comparable results but only requires half of its number of parameters and FLOPs. Especially in terms of FPS, YOLOC-S surpasses it by 111.4. Additionally, YOLOC-S outperforms SSD, YOLOv5-S, YOLOv7-tiny, and PP-YOLOE+-S on AP while requiring fewer parameters and FLOPs. Similarly, YOLOC-N not only achieves higher accuracy than SSD, YOLOv6-N, PPYOLOE+-S, and YOLOv8-N, but also requires fewer parameters and FLOPs. Specifically, YOLOC-N achieves a reduction of 1.2M parameters and 3.1G FLOPs compared to YOLOv8-N, while showing an improvement of 0.3%, 3.2% and 19.9 on AP, $AP_{0.5}$ and FPS, respectively. Fig. 12 provides some detection results. These results demonstrate the effectiveness and efficiency of our proposed method for object detection on the Wider Face dataset.

4.7. Results on microsoft COCO

To fully validate the efficacy of our proposed method, we compare YOLOC-S with several state-of-the-art detectors on the Microsoft COCO dataset. From Table 6, it can be observed that YOLOC-S achieves the highest accuracy among competitors. YOLOC-S not only improves on AP by 0.26% and $AP_{0.5}$ by 0.43%, but also improves on FPS by 11.1, based on the reduced number of parameters and computational cost of the baseline YOLOv5-S. YOLOC-S outperforms YOLOv8-N in terms of AP and $AP_{0.5}$ respectively by 0.33% and 5.00% while having similar inference speeds. Additionally, YOLOC-S outperforms YOLOv7-tiny on all metrics. Compared with YOLOv6-N, YOLOC-S is faster and more accurate. This experiment shows the advantages of our proposed method on large-scale datasets as well.

4.8. Ablation experiments on GTSDB

Table 7 gives the results of the ablation experiments conducted on the GTSDB dataset. We choose YOLOv5-S as the baseline, and the efficacy of the Evolutionary Channel Pruning and CIM-SPPF modules is demonstrated by their incorporation. The highest-performing results are achieved when both parts are employed simultaneously.

Table 6
Comparison on Microsoft COCO dataset. The best results are boldfaced. The second-best results are underlined.

Model	Size	AP (%)	$AP_{0.5}$ (%)	Params (M)	GFLOPs	FPS
YOLOv5-N [1]	640 × 640	28.04	46.18	**1.8**	**4.5**	**185.2**
YOLOv6-N [10]	640 × 640	36.20	51.83	4.3	11.1	138.9
YOLOv7-tiny [11]	640 × 640	37.45	55.22	5.9	13.7	153.8
YOLOv8-N [12]	640 × 640	37.32	52.56	3.0	8.7	175.4
YOLOv5-S [1]	640 × 640	37.39	57.13	6.9	16.4	161.3
YOLOC-S	640 × 640	**37.65**	**57.56**	5.4	13.2	172.4

Table 7
Effects of Channel Pruning (CP) and CIM-SPPF (C-S) in our method. This experiment uses YOLOv5-S as the baseline. Results are reported on GTSDB dataset. The best results are boldfaced. The second-best results are underlined.

CP	C-S	AP (%)	$AP_{0.5}$ (%)	$F1$ (%)	Params (M)	GFLOPs	FPS
		83.67	96.70	95.55	6.7 (0.000%)	15.8 (0.000%)	62.5
✓		83.94	96.95	95.79	4.7 (−29.85%)	12.9 (−18.35%)	68.5
	✓	84.64	97.31	96.30	6.4 (−4.480%)	15.5 (−1.900%)	64.5
✓	✓	**85.29**	**97.77**	**96.69**	**4.5 (−32.84%)**	**12.7 (−19.62%)**	**69.9**

Table 8
Effects of SC, GC, GhostNet method and CIMConv. Results are reported on the GTSDB dataset. The best results are boldfaced. The second-best results are underlined.

Convolutional-block methods	AP (%)	$AP_{0.5}$ (%)	$F1$ (%)	Params (M)	GFLOPs	FPS
SC	84.36	96.30	95.61	6.7	15.8	62.5
GC	83.12	95.92	95.26	**6.1**	**15.3**	**65.8**
GhostNet method	83.92	96.84	95.88	6.4	15.5	64.5
CIMConv	**84.64**	**97.31**	**96.30**	6.4	15.5	64.5

Table 9
Effects of Shuffle in CIM-SPPF (C-S) and CIMConv. Results are reported on the GTSDB dataset. The best results are boldfaced. The second-best results are underlined.

Shuffle(C-S)	Shuffle(CIMConv)	AP (%)	$AP_{0.5}$ (%)	$F1$ (%)	Params (M)	GFLOPs	FPS
		83.87	96.09	95.36	6.4	15.5	64.5
✓		84.43	95.86	95.94	6.4	15.5	64.5
	✓	84.37	96.93	**96.52**	6.4	15.5	64.5
✓	✓	**84.64**	**97.31**	96.30	6.4	15.5	64.5

Table 10
The results of different compression rates on the GTSDB dataset. The best results are boldfaced. The second-best results are underlined.

Model	AP (%)	$AP_{0.5}$ (%)	Params (M)	GFLOPs	FPS(CPU)	FPS(GPU)
YOLOv5-S	83.67	96.70	6.7 (0.000%)	15.8 (0.000%)	6.8 (× 1.00)	116.3 (× 1.00)
YOLOC	84.96	97.27	5.2 (−22.39%)	14.0 (−11.39%)	7.0 (× 1.03)	117.6 (× 1.01)
	85.29	**97.77**	4.5 (−32.84%)	12.7 (−19.62%)	7.3 (× 1.07)	122.0 (× 1.05)
	84.83	96.87	3.8 (−43.28%)	11.1 (−29.75%)	7.8 (× 1.15)	126.6 (× 1.09)
	84.71	97.07	2.8 (−58.21%)	9.7 (−38.61%)	8.5 (× 1.25)	129.9 (× 1.12)
	84.26	96.84	2.0 (−70.15%)	8.2 (−48.10%)	9.6 (× 1.41)	133.3 (× 1.15)
	83.58	96.98	1.4 (−79.10%)	6.5 (−58.86%)	10.8 (×1.59)	135.1 (× 1.16)
	81.79	96.44	**0.9 (−86.57%)**	**4.9 (−68.99%)**	**12.2 (× 1.79)**	**138.9 (× 1.19)**

These experiments results demonstrate that the proposed method yields significant improvements on the evaluation metrics of AP, $AP_{0.5}$ and $F1$ by 1.62%, 1.07% and 1.14%, respectively, when compared to the YOLOv5-S baseline. In addition, the number of parameters and FLOPs of the proposed method are 32.84% and 19.62% less than the baseline, respectively.

4.8.1. Evolutionary channel pruning

The proposed evolutionary channel pruning method is used to reduce the number of parameters of YOLOv5-S. Table 7 shows that the proposed pruning method improves the performance of the YOLOv5-S baseline by 0.27%, 0.25%, and 0.24% in terms of AP, $AP_{0.5}$, and $F1$, respectively, while the proposed pruning method achieves a significant reduction of 29.85% and 18.35% in the number of parameters and FLOPs, respectively. These results validate the effectiveness of the proposed channel pruning method in enhancing the performance of real-time object detection.

4.8.2. CIM-SPPF

The original SPPF in YOLOv5-S is replaced with CIM-SPPF, leading to improved channel-level information mixing and reduced model complexity. Empirical results indicate that the use of CIM-SPPF leads to higher AP, $AP_{0.5}$, and $F1$ scores for YOLOv5-S, with improvements of 0.97%, 0.61% and 0.75%, respectively. Additionally, the number of parameters is reduced by 4.48%, and the amount of FLOPs is reduced by 1.90%. As presented in Table 7, the CIM-SPPF module outperforms the original SPPF in channel-level information mixing.

4.8.3. CIMConv

We evaluate various convolution methods applied to the SPPF while incorporating the Channel Shuffle operation to demonstrate the effectiveness of the proposed CIMConv. The results, as presented in Table 8, indicate that CIMConv outperforms the other three methods in terms of AP, $AP_{0.5}$, and $F1$. Notably, CIMConv has lower parameters and FLOPs than SC. To further validate the effectiveness of using Shuffle in CIM-SPPF and CIMConv, we compare models both with and without

the Shuffle operation in CIMConv and CIM-SPPF. As shown in Table 9, the experimental results demonstrate a noteworthy enhancement in the accuracy of the models when the Shuffle operation is introduced. We think that the Shuffle operation effectively enhances the network's capacity to learn diverse feature representations by redistributing information within the feature map. To summarize, the Shuffle operation plays a pivotal role in CIMConv and CIM-SPPF, and its mechanism for reorganizing information significantly boosts the accuracy of these models.

4.8.4. Different levels of compression

Furthermore, we evaluate different levels of compression for YOLOv5-S and present the results in Table 10. It is evident that reducing the number of parameters by over 70% does not result in decreased accuracy. Moreover, with approximately 80% parameter reduction, our method achieves comparable performance to the original network. From the FPS metric, it can be observed that as the compression ratio increases, our method indeed improves the inference speed. On the CPU platform, our approach exhibits a 1.59-fold improvement in speed while maintaining comparable accuracy levels. Notably, accuracy exhibits an initial increase with increasing compression rates. However, after a certain point, the accuracy decreases as the compression rate becomes excessive. We discover the existence of redundant parameters in the network, and their removal has a regularizing effect that improves the accuracy. However, removing too many parameters results in the removal of critical parameters and, subsequently, a decline in accuracy.

5. Conclusion

In this paper, we propose a novel evolutionary channel pruning technique and a lightweight convolutional approach, CIMConv. Our proposed evolutionary channel pruning optimizes the number of channels in convolutional layers, while the CIMConv maintains low computational complexity and encourages channel information mixing during convolutional operations. To verify the effectiveness of our proposed evolutionary channel pruning and CIMConv, we add them to YOLOv5. The results indicate that our proposed optimization strategies can reduce the parameter count of YOLOv5-S by approximately 30% and FLOPs by approximately 20%, while improving accuracy. The optimized model, YOLOC, achieves state-of-the-art performance in balancing the accuracy and complexity of object detection networks. Our future work will focus on extending our optimization strategies to other networks and deploying our models on edge devices, while exploring more advanced optimization techniques.

CRediT authorship contribution statement

Changcai Yang: Writing – review & editing, Writing – original draft, Supervision, Methodology, Investigation, Funding acquisition, Data curation. **Zhijie Lin:** Writing – original draft, Software, Methodology, Data curation. **Ziyang Lan:** Software, Methodology. **Riqing Chen:** Writing – review & editing, Funding acquisition, Conceptualization. **Lifang Wei:** Writing – review & editing, Methodology, Investigation, Funding acquisition, Conceptualization. **Yizhang Liu:** Writing – review & editing, Writing – original draft, Supervision, Software, Methodology, Investigation, Data curation, Conceptualization.

Declaration of competing interest

The authors declare that they have no conflict of interest.

Data availability

The data that has been used is confidential.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grant 61702101, and 61802064, in part by the Fujian University Industry University Research Joint Innovation Project under Grant 2022H6006, in part by the Fujian Science and Technology Planning Project under Grant 2021S0007.

References

[1] G. Jocher, YOLOv5 by Ultralytics, 2020.
[2] Y. Liu, B.N. Zhao, S. Zhao, L. Zhang, Progressive motion coherence for remote sensing image matching, IEEE Trans. Geosci. Remote Sens. 60 (2022) 1–13.
[3] Y. Liu, Y. Li, L. Dai, C. Yang, L. Wei, T. Lai, R. Chen, Robust feature matching via advanced neighborhood topology consensus, Neurocomputing 421 (2021) 273–284.
[4] L. Dai, Y. Liu, J. Ma, L. Wei, T. Lai, C. Yang, R. Chen, MS2DG-Net: Progressive correspondence learning via multiple sparse semantics dynamic graph, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 8973–8982.
[5] D. Feng, C. Haase-Schütz, L. Rosenbaum, H. Hertlein, C. Glaeser, F. Timm, W. Wiesbeck, K. Dietmayer, Deep multi-modal object detection and semantic segmentation for autonomous driving: Datasets, methods, and challenges, IEEE Trans. Intell. Transp. Syst. 22 (3) (2020) 1341–1360.
[6] L. Wang, Z. Song, X. Zhang, C. Wang, G. Zhang, L. Zhu, J. Li, H. Liu, SAT-GCN: Self-attention graph convolutional network-based 3D object detection for autonomous driving, Knowl.-Based Syst. 259 (2023) 110080.
[7] Y. Zhang, C. Wang, X. Wang, W. Zeng, W. Liu, Fairmot: On the fairness of detection and re-identification in multiple object tracking, Int. J. Comput. Vis. 129 (2021) 3069–3087.
[8] Y. Zhang, P. Sun, Y. Jiang, D. Yu, F. Weng, Z. Yuan, P. Luo, W. Liu, X. Wang, Bytetrack: Multi-object tracking by associating every detection box, in: Proceedings of the European Conference on Computer Vision, 2022, pp. 1–21.
[9] W. Ma, T. Zhou, J. Qin, Q. Zhou, Z. Cai, Joint-attention feature fusion network and dual-adaptive NMS for object detection, Knowl.-Based Syst. 241 (2022) 108213.
[10] C. Li, L. Li, H. Jiang, K. Weng, Y. Geng, L. Li, Z. Ke, Q. Li, M. Cheng, W. Nie, et al., YOLOv6: A single-stage object detection framework for industrial applications, 2022, arXiv preprint arXiv:2209.02976.
[11] C.-Y. Wang, A. Bochkovskiy, H.-Y.M. Liao, YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors, 2022, arXiv preprint arXiv:2207.02696.
[12] G. Jocher, A. Chaurasia, J. Qiu, YOLO by Ultralytics, 2023.
[13] Z. Ge, S. Liu, F. Wang, Z. Li, J. Sun, Yolox: Exceeding yolo series in 2021, 2021, arXiv preprint arXiv:2107.08430.
[14] S. Xu, X. Wang, W. Lv, Q. Chang, C. Cui, K. Deng, G. Wang, Q. Dang, S. Wei, Y. Du, et al., PP-YOLOE: An evolved version of YOLO, 2022, arXiv preprint arXiv:2203.16250.
[15] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, J. Mach. Learn. Res. 15 (1) (2014) 1929–1958.
[16] Y. Chen, T. Yang, X. Zhang, G. Meng, X. Xiao, J. Sun, Detnas: Backbone search for object detection, Adv. Neural Inf. Process. Syst. 32 (2019).
[17] G. Li, M. Zhang, J. Zhang, Q. Zhang, OGCNet: Overlapped group convolution for deep convolutional neural networks, Knowl.-Based Syst. 253 (2022) 109571.
[18] R. Girshick, J. Donahue, T. Darrell, J. Malik, Rich feature hierarchies for accurate object detection and semantic segmentation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 580–587.
[19] R. Girshick, Fast R-CNN, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 1440–1448.
[20] S. Ren, K. He, R. Girshick, J. Sun, Faster R-CNN: Towards real-time object detection with region proposal networks, Adv. Neural Inf. Process. Syst. 28 (2015).
[21] Z. Cai, N. Vasconcelos, Cascade R-CNN: Delving into high quality object detection, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018, pp. 6154–6162.
[22] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, You only look once: Unified, real-time object detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 779–788.
[23] J. Redmon, A. Farhadi, YOLO9000: better, faster, stronger, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 7263–7271.
[24] J. Redmon, A. Farhadi, Yolov3: An incremental improvement, 2018, arXiv preprint arXiv:1804.02767.
[25] A. Bochkovskiy, C.-Y. Wang, H.-Y.M. Liao, Yolov4: Optimal speed and accuracy of object detection, 2020, arXiv preprint arXiv:2004.10934.
[26] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, A.C. Berg, Ssd: Single shot multibox detector, in: Proceedings of the European Conference on Computer Vision, 2016, pp. 21–37.

[27] Z. Tian, C. Shen, H. Chen, T. He, Fcos: Fully convolutional one-stage object detection, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 9627–9636.
[28] Z. Tian, C. Shen, H. Chen, T. He, Fcos: A simple and strong anchor-free object detector, IEEE Trans. Pattern Anal. Mach. Intell. 44 (4) (2020) 1922–1933.
[29] M. Tan, R. Pang, Q.V. Le, Efficientdet: Scalable and efficient object detection, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 10781–10790.
[30] S. Han, J. Pool, J. Tran, W. Dally, Learning both weights and connections for efficient neural network, Adv. Neural Inf. Process. Syst. 28 (2015).
[31] Y. Guo, A. Yao, Y. Chen, Dynamic network surgery for efficient dnns, Adv. Neural Inf. Process. Syst. 29 (2016).
[32] H. Li, A. Kadav, I. Durdanovic, H. Samet, H.P. Graf, Pruning filters for efficient convnets, 2016, arXiv preprint arXiv:1608.08710.
[33] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, C. Zhang, Learning efficient convolutional networks through network slimming, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 2736–2744.
[34] Y. He, G. Kang, X. Dong, Y. Fu, Y. Yang, Soft filter pruning for accelerating deep convolutional neural networks, 2018, arXiv preprint arXiv:1808.06866.
[35] Y. He, P. Liu, Z. Wang, Z. Hu, Y. Yang, Filter pruning via geometric median for deep convolutional neural networks acceleration, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 4340–4349.
[36] M. Lin, R. Ji, Y. Wang, Y. Zhang, B. Zhang, Y. Tian, L. Shao, Hrank: Filter pruning using high-rank feature map, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 1529–1538.
[37] H. Wang, C. Qin, Y. Zhang, Y. Fu, Neural pruning via growing regularization, 2020, arXiv preprint arXiv:2012.09243.
[38] Y. Zhang, M. Lin, M. Chen, Z. Xu, F. Chao, Y. Shen, K. Li, Y. Wu, R. Ji, Optimizing gradient-driven criteria in network sparsity: Gradient is all you need, 2022, arXiv preprint arXiv:2201.12826.
[39] B. Li, B. Wu, J. Su, G. Wang, Eagleeye: Fast sub-net evaluation for efficient neural network pruning, in: Proceedings of the European Conference on Computer Vision, 2020, pp. 639–654.
[40] S. Gao, F. Huang, W. Cai, H. Huang, Network pruning via performance maximization, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 9270–9280.
[41] Z. Liu, H. Mu, X. Zhang, Z. Guo, X. Yang, K.-T. Cheng, J. Sun, Metapruning: Meta learning for automatic neural network channel pruning, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 3296–3305.
[42] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, Commun. ACM 60 (6) (2017) 84–90.
[43] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, 2014, arXiv preprint arXiv:1409.1556.
[44] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.
[45] A.G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, H. Adam, Mobilenets: Efficient convolutional neural networks for mobile vision applications, 2017, arXiv preprint arXiv:1704.04861.
[46] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, L.-C. Chen, Mobilenetv2: Inverted residuals and linear bottlenecks, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018, pp. 4510–4520.
[47] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, et al., Searching for mobilenetv3, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 1314–1324.
[48] X. Zhang, X. Zhou, M. Lin, J. Sun, Shufflenet: An extremely efficient convolutional neural network for mobile devices, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018, pp. 6848–6856.
[49] N. Ma, X. Zhang, H.-T. Zheng, J. Sun, Shufflenet v2: Practical guidelines for efficient cnn architecture design, in: Proceedings of the European Conference on Computer Vision, 2018, pp. 116–131.
[50] K. Han, Y. Wang, Q. Tian, J. Guo, C. Xu, C. Xu, Ghostnet: More features from cheap operations, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 1580–1589.
[51] F. Yao, S. Wang, L. Ding, G. Zhong, L.B. Bullock, Z. Xu, J. Dong, Lightweight network learning with zero-shot neural architecture search for UAV images, Knowl.-Based Syst. 260 (2023) 110142.
[52] Y. Tang, K. Han, J. Guo, C. Xu, C. Xu, Y. Wang, GhostNetV2: Enhance cheap operation with long-range attention, 2022, arXiv preprint arXiv:2211.12905.
[53] C. Chen, Z. Guo, H. Zeng, P. Xiong, J. Dong, RepGhost: A hardware-efficient ghost module via re-parameterization, 2022, arXiv preprint arXiv:2211.06088.
[54] S. Houben, J. Stallkamp, J. Salmen, M. Schlipsing, C. Igel, Detection of traffic signs in real-world images: The German traffic sign detection benchmark, in: The 2013 International Joint Conference on Neural Networks, IJCNN, 2013, pp. 1–8.
[55] X. Yang, J. Yan, W. Liao, X. Yang, J. Tang, T. He, Scrdet++: Detecting small, cluttered and rotated objects via instance-level feature denoising and rotation loss smoothing, IEEE Trans. Pattern Anal. Mach. Intell. (2022).
[56] Z. Zhu, D. Liang, S. Zhang, X. Huang, B. Li, S. Hu, Traffic-sign detection and classification in the wild, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 2110–2118.
[57] S. Yang, P. Luo, C.-C. Loy, X. Tang, Wider face: A face detection benchmark, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 5525–5533.
[58] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, C.L. Zitnick, Microsoft coco: Common objects in context, in: Proceedings of the European Conference on Computer Vision, 2014, pp. 740–755.