Berger Scientfic Press

Innovation & Technology Advances, 2023, 1(2), 1-24
https://bergersci.com/index.php/ita/index
DOI: 10.61187/ita.v1i2.36

A review of neural network lightweighting techniques

Ziyi Gong, Huifu Zhang*, Hao Yang, Fangjun Liu, Fan Luo

Department of Computer Sciences and Engineering, Hunan University of Science and Technology, Xiangtan, Hunan, China
804250678@qq.com (Gong, Z.), yanghaohnust@163.com (Yang, H.), 1989036826@qq.com (Liu, F.), 22010501023@mail.hnust.edu.cn (Luo, F.)

*Correspondence: hfzhang@hnust.edu.cn

Abstract: The application of portable devices based on deep learning has become increasingly widespread, which has made the deployment of complex neural networks on embedded devices a hot research topic. Neural network lightweighting is one of the key technologies for applying neural networks to embedded devices. This paper elaborates and analyzes neural network lightweighting techniques from two aspects: model pruning and network structure design. For model pruning, a comparison of methods from different periods is conducted, highlighting their advantages and limitations. Regarding network structure design, the principles of four classical lightweight network designs are described from a mathematical perspective, and the latest optimization methods for these networks are reviewed. Finally, potential research directions for lightweight neural network pruning and structure design optimization are discussed.

Keywords: lightweighting techniques for neural networks; model pruning; network structure design; convolutional structure optimization

How to cite this paper: Gong, Z., Zhang, H., Yang, H., et al. A Review of Neural Network Lightweighting Techniques. Innovation & Technology Advances, 2023, 1(2), 1–24. https://doi.org/10.61187/ita.v1i2.36

1. Introduction

Deep learning differs significantly from traditional manual feature design. Convolutional neural networks (CNNs) employed in deep learning can automatically extract deep features of targets without the need for manual feature extraction. This characteristic greatly reduces the difficulty of applying image recognition [1-3]. Consequently, deep CNNs have become increasingly mature and successful in various fields such as military, transportation, and healthcare. However, in order to achieve higher accuracy, the depth of neural networks continues to increase, resulting in higher computational complexity and storage requirements. As performance demands escalate, efficiency becomes a primary concern in network design. Specifically, efficiency issues primarily involve model storage and prediction speed. Therefore, lightweighting techniques are needed to address efficiency concerns while maintaining accuracy [4-7].

The goal of model lightweighting is to address the inability of traditional neural networks to run on small-scale hardware in terms of storage space and energy consumption. To achieve this goal, optimization techniques such as network structure design and model compression are primarily employed to reduce storage requirements, improve execution speed, and maintain the accuracy of traditional neural networks [8-10]. In recent years, the research direction of lightweight neural networks has been continuously expanding, requiring ongoing exploration, comparison, and updates. It is worth noting that excellent lightweight network models often possess multifunctionality, and the optimization trends have become diverse, no longer limited to a single model compression algorithm or the replacement of lightweight modules. Therefore, a comprehensive summary of optimization methods for lightweight neural network architectures is necessary [11-13].

This paper provides a comprehensive review of classical compression algorithms and network structures for neural networks. Firstly, it elaborates on model pruning algo-

rithms and analyzes and summarizes recent research advancements based on these algorithms. Model pruning encompasses structured pruning and unstructured pruning, with structured pruning covering techniques such as convolutional kernel pruning and channel pruning [14]. Next, it analyzes some classical lightweight neural networks from the perspectives of lightweight module design and convolutional structure optimization, and summarizes the latest research achievements based on these network structures. Finally, it discusses the prospects and challenges of lightweight neural networks and provides a comprehensive conclusion [15-17].

The paper is organized as follows: Section II introduces two pruning algorithms for model pruning and analyzes their advantages and disadvantages. Section III presents the ideas and methods of network structure design, summarizing four lightweight network structures that have been applied and improved in recent years. It also analyzes the characteristics and performance of these network structures. Section IV discusses the future development trends and challenges of lightweight neural networks. Section V provides a comprehensive summary of the work conducted in this paper.

2. Model pruning

Model pruning is one of the most commonly used methods in compressing neural network models. Its primary objective is to reduce computational complexity and model size by removing unimportant neurons in the neural network. Model pruning algorithms can be categorized into two types: unstructured pruning and structured pruning. The classification of model pruning methods [18] is illustrated in **Figure 1**. The distinction lies in whether the entire node or convolutional kernel is removed all at once.

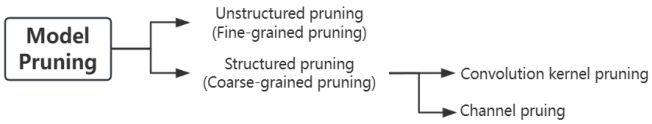

Figure 1. Classification methods for model pruning.

In unstructured pruning algorithms, each element of every convolutional kernel is considered, and the parameter information with zero values in the kernels is removed. This pruning method takes into account each parameter in the network model, allowing for more fine-grained pruning. In contrast, structured pruning algorithms employ a coarse-grained pruning approach by directly removing the structured information of entire convolutional kernels. This can effectively reduce the size of the model and improve its performance. Specifically, kernel pruning refers to the removal of a group of convolutional kernels in a convolutional layer, while channel pruning refers to the removal of entire channels in a convolutional layer. This subdivision provides a clearer description of the different ways in which structured pruning can be performed.

2.1. Unstructured pruning

Unstructured pruning does not adhere to specific geometric shapes or constraints when removing the parameter information with zero values in the convolutional kernels. **Figure 2** illustrates the process of unstructured pruning, demonstrating the fine-grained pruning approach.

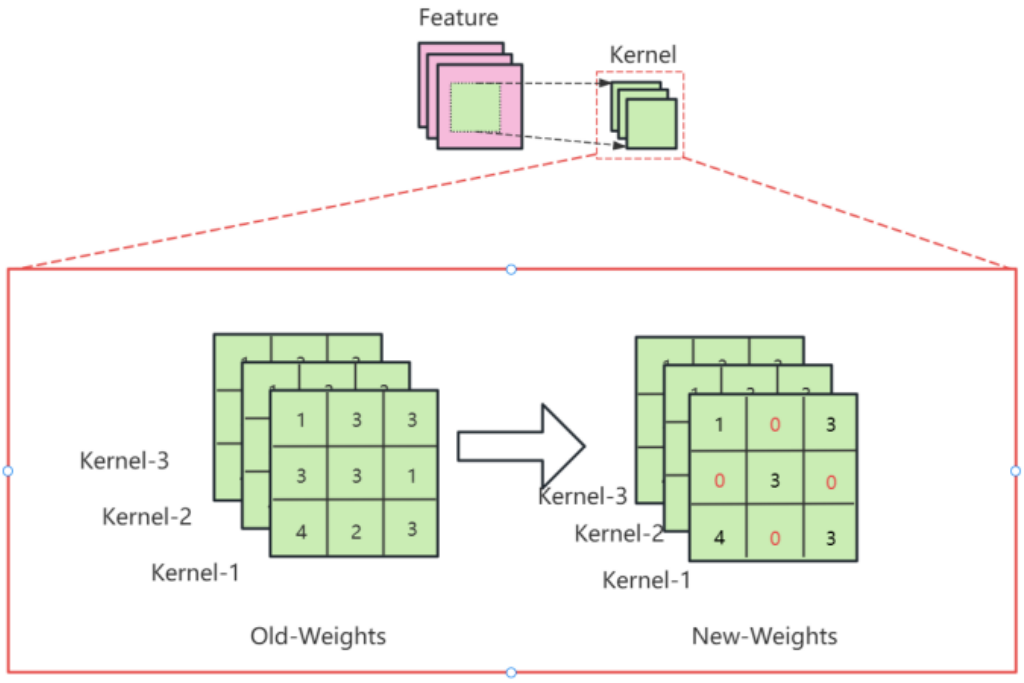

Figure 2. Fine-grained pruning process.

This paper investigates the representative works and recent advances in unstructured pruning algorithms. The earliest pruning algorithm can be traced back to the Optimal Brain Damage (OBD) algorithm proposed in reference [19], which belongs to the category of single-weight pruning algorithms within unstructured pruning. The OBD algorithm utilizes the Hessian matrix based on the loss function to calculate parameter weights and prunes the parameters with lower weights. However, the OBD algorithm simplifies the calculation of the Hessian matrix by ignoring the off-diagonal terms, which is a hypothetical simplification. Subsequently, reference [20] studied the off-diagonal terms of the Hessian matrix and discovered that the assumption made by the OBD algorithm is invalid in many cases. To overcome the limitations of the OBD algorithm, reference [19] proposed the Optimal Brain Surgeon (OBS) method, which utilizes all second-order derivative information of the error function for network pruning without the need for retraining. Both the OBD and OBS algorithms share a similar drawback, which is the high computational complexity involved in computing and updating the significance of all parameters in each iteration. To address this issue, reference [21] proposed a method that uses the minimum contribution variance as the pruning criterion. If a parameter's output remains almost the same before and after bias, its contribution is considered insignificant, these parameters, which have the smallest contribution variances on the training set, can be removed.

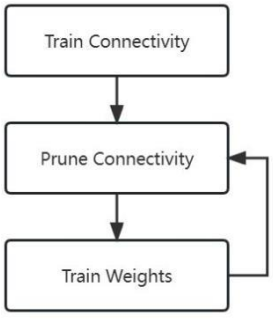

Figure 3. The three main steps of the pruning process.

In addition, reference [22] proposed a method that directly constructs a weight saliency matrix and performs sorting to select insignificant redundant nodes for pruning. Furthermore, in reference [23], a method based on learning weight connectivity importance for pruning was introduced. This method consists of three steps, as shown in **Figure 3**. Through an iterative process of connection pruning and weight training, it can reduce the storage and computational requirements by an order of magnitude while maintaining accuracy.

2.2 Structured pruning

In contrast to unstructured pruning algorithms, structured pruning algorithms target entire structures (such as convolutional kernels or channels) rather than individual parameters. By removing entire structures at once without the need to individually compute parameters, structured pruning algorithms have significantly lower complexity compared to unstructured pruning algorithms. Therefore, structured pruning algorithms have become an important direction in pruning algorithm research. This paper categorizes structured pruning into two types: convolutional kernel pruning and channel pruning, and discusses them [34-36].

2.2.1 Convolution kernel pruning

Convolutional kernel pruning is a coarse-grained pruning method characterized by the simultaneous pruning of connected input channels in the subsequent layer when pruning a specific convolutional kernel in a convolutional layer. This method effectively reduces the number of parameters in the model by removing convolutional kernels with lower importance. **Figure 4** illustrates the process of convolutional kernel pruning.

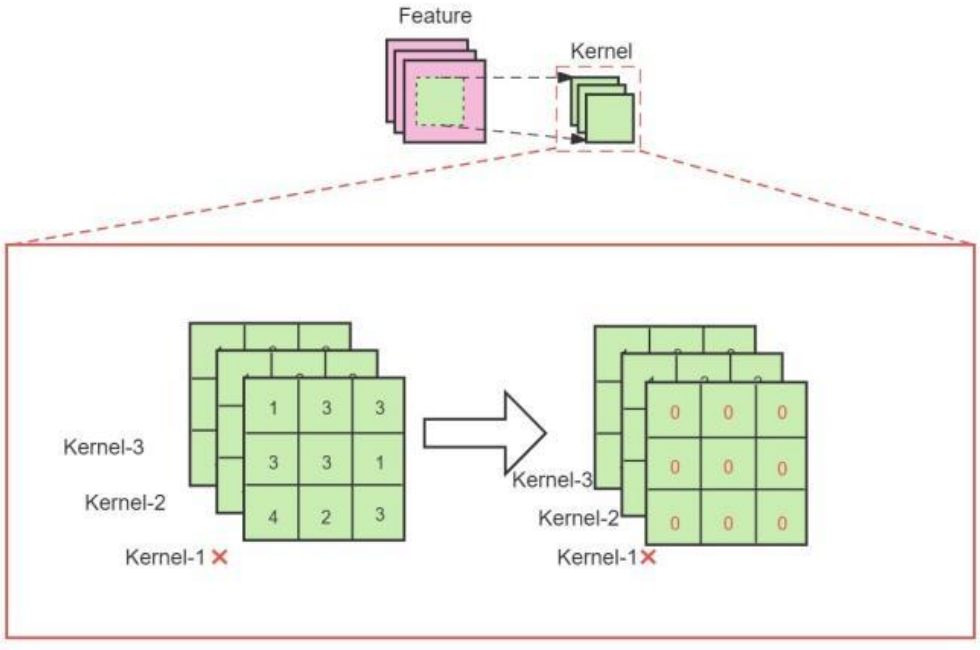

Figure 4. Illustration of the convolutional kernel pruning process.

In reference [37], an algorithm based on global search and convolutional kernel saliency is proposed. In reference [37], an algorithm based on global search and convolutional kernel saliency is proposed. The algorithm utilizes the Taylor expansion criterion to expand the objective function and identifies the convolutional kernel that causes the least change in the objective function as the salient kernel. It then replaces the salient kernel

with zero values. Reference [38] improves upon the aforementioned algorithm by introducing consistent extension capabilities to any layer in the network, eliminating the need for sensitivity analysis on each layer. Reference [39] introduces the ThiNet pruning algorithm, which establishes a one-to-one relationship between the current layer's convolutional kernels and the next layer's input channels through convolutional computations. This relationship is utilized to explore the saliency of the input channels for the next layer's convolutional kernels. In reference [40], a novel pruning method for convolutional kernels is proposed, known as Filter Pruning via Geometric Median (FPGM). The central idea of the geometric median, as described in reference [41], is as follows: Given a set of points with a quantity of $n\left(a^{(1)},\cdots,a^{(n)}\right)$, for each point $a^{(i)} \in \mathbb{R}^d$, find a point $x^* \in \mathbb{R}^d$ such that the sum of their Euclidean distances is minimized. This is shown in Equation (1). This pruning method is able to satisfy both the requirements of a larger paradigm deviation for the filter and a smaller minimum criterion for the filter. And its usefulness and advantages are verified on two image classification benchmarks. Reference [42] introduces an approximate Oracle convolutional kernel pruning algorithm. This algorithm prunes the kernels by randomly masking them and calculates the cumulative change in the output of the next layer to search for the least significant kernels. Furthermore, reference [43] proposes an end-to-end joint pruning method that can simultaneously prune convolutional kernels and other structures. By employing generative adversarial learning techniques, this method effectively addresses the optimization problem. Reference [44] presents a dynamic pruning algorithm. This algorithm dynamically predicts the saliency of the next layer's convolutional channels during the training process and skips channels with lower saliency. This dynamic nature allows different input images to flexibly skip different channels based on their characteristics. Moreover, reference [45] introduces the meta-convolutional kernel pruning algorithm, which considers the relationships between convolutional kernels and constructs a meta-pruning framework to adaptively select appropriate pruning methods when the distribution of kernels changes. Reference [46] proposes a meta-learning pruning algorithm. This algorithm first trains a pruning network using random structure sampling and utilizes meta-learning to predict the accuracy of the pruned network. It can search for pruned networks under different constraints without requiring manual intervention and does not require fine-tuning during the search process.

$$x^* = \arg\min f(x) \; where \; f(x) \cdot \sum_{i \in [1.n]} x - a^{(i)}{}_2, x \in \mathbb{R}^d \tag{1}$$

Reference [47] proposes a method called overall global pruning, which uses the idea of pruning convolutional kernels to address the limitations of amplitude-based methods when pruning fully connected layers. Furthermore, in reference [48], a novel method named Collaborative Channel Pruning (CCPrune) is introduced. This method effectively assesses the significance of channels by incorporating the weights of convolutional layers and the scaling factors of Batch Normalization (BN) layers. Moreover, reference [49] introduces a method known as Global Filter Importance-based Adaptive Pruning (GFI-AP). This approach assigns importance scores to each convolutional kernel by evaluating how effectively the network learns the mapping from input to output using the dataset. This enables a comprehensive comparison among the kernels. Reference [50] proposes a method for dynamically removing redundant convolutional kernels by embedding manifold information of all instances into the pruned network space. By aligning the manifold information between the recognition complexity and feature similarity of images in the training set with the pruned subnetwork, it maximizes the utilization of redundancy within the given network structure. Reference [51] introduces a novel method for pruning convolutional kernels, utilizing feature map ranking (HRank) for exploration and developing a mathematical formula for kernel pruning.

2.2.2 Channel pruning

Channel pruning is a method used to prune redundant channels in feature maps, without considering the impact of convolutional kernel weights. It is particularly effective in cases where there is a significant amount of redundancy in the feature maps. The process of channel pruning is illustrated in **Figure 5**. By pruning redundant channels, model compression can be achieved. One of the advantages of channel pruning is that it does not rely on sparse convolutional computation libraries or specialized hardware, yet it can still achieve high compression rates.

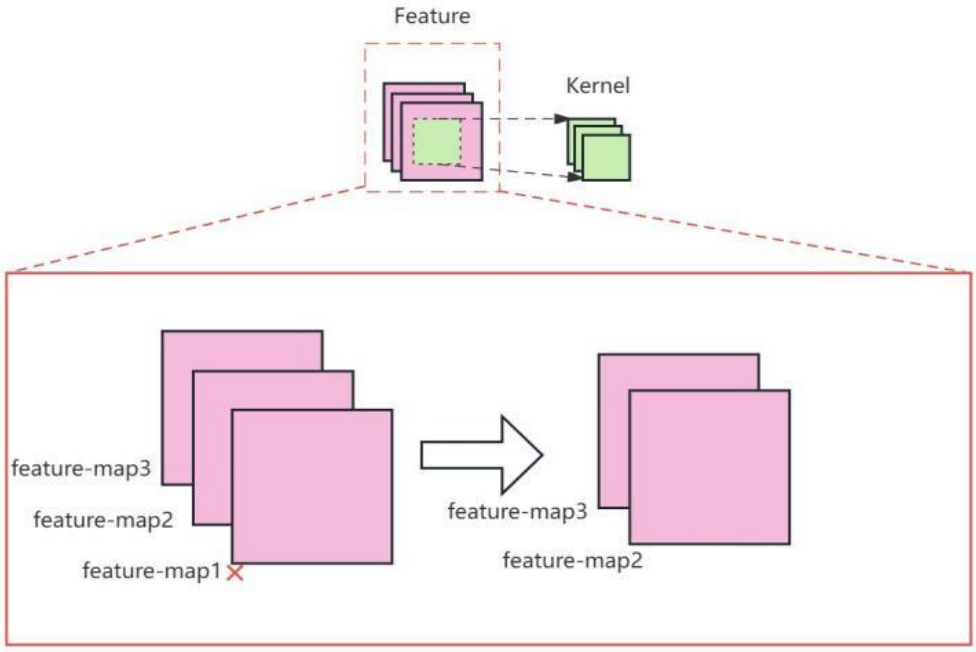

Figure 5. Illustration of the channel pruning process.

Reference [52] proposes a method based on eliminating low-activity channels, which reduces the computational operations between each convolutional kernel and channels that do not contribute significantly to the model's predictions. This method effectively reduces the computational load without significantly impacting the model's performance. Similarly, at the channel level, reference [53] introduces a channel pruning method based on LASSO regularization and linear least squares. This method first identifies and removes redundant convolutional kernels and their corresponding feature maps, reducing the model's parameter count and computational complexity. Then, the remaining network is reconstructed to restore the model's predictive ability. Building upon the work in reference [53], reference [54] argues for the necessity of jointly pruning neurons across the entire neural network based on a unified objective. By considering the relationships between different neurons and their contributions to the overall network performance during the pruning process, a unified objective ensures that the pruned network maintains good predictive performance during the retraining phase. Reference [55] proposes a network slimming method, which is a commonly used pruning algorithm for many large-scale networks. The core idea is to introduce a scaling factor γ for each channel and establish an objective function as shown in Equation (2):

$$L = \sum_{(x,y)} l\left(f\left(x,w\right),y\right) + \lambda \sum_{y\in\Gamma} g\left(\gamma\right) \tag{2}$$

Equation (2) where x and y are the input and output of the feature map respectively, w is the weight, $g(\gamma)$ is the penalty term, γ is the scaling factor, and λ is the balance factor. The joint optimisation of the regular term of the scaling factor γ and the weight loss function automatically identifies and removes unimportant channels to improve the computational speed of the network. The network thinning process is shown in **Figure 6**.

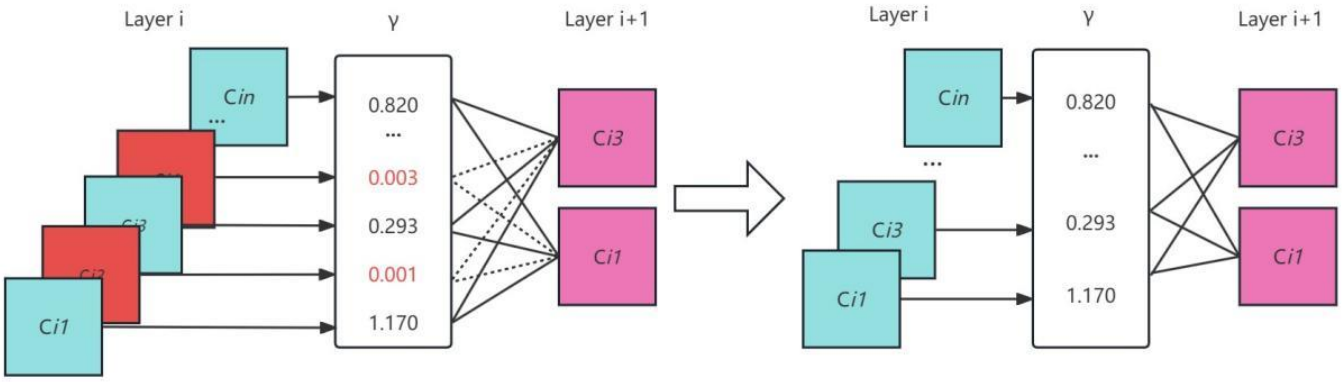

Figure 6. Illustration of the network slimming process.

Reference [56] presents a more general and effective improvement to the method proposed in reference [55]. Instead of directly using the parameters of the Batch Normalization (BN) layer, this approach introduces additional scale factors to enhance the method's applicability. Reference [57] proposes a discriminative-aware channel pruning method for pre-trained models. This method introduces an additional channel-aware loss function, which is combined with the classification loss function, and incorporates reconstruction error. It utilizes the $L_{2,0}$ norm to iteratively induce sparsity in channel pruning and parameter optimization. Reference [58] challenges the effectiveness of norm-based calculations and proposes a norm-independent channel pruning technique. This method employs an end-to-end random training approach, enforcing the constant output of certain channels and then adjusting the biases of the affected layers to eliminate these constant channels, achieving channel pruning. Furthermore, in reference [59], an optimal thresholding (OT) method is proposed. This method aims to prune channels with layer-correlated thresholds, optimally separating important channels from negligible ones. By utilizing OT, most unimportant channels are pruned to achieve high sparsity while minimizing performance degradation. In reference [60], researchers attempt to determine the channel configuration for pruning models through random search. Experimental results demonstrate the effectiveness of this simple strategy compared to other channel pruning methods. Existing methods often treat the pruning rate as a hyperparameter and overlook the sensitivity of different convolutional layers. Reference [61] introduces a sensitivity-based channel pruning method, measuring it using second-order sensitivity. The underlying concept involves the selective pruning of insensitive filters while preserving the sensitive ones. This is achieved by quantifying the sensitivity of a convolutional kernel through the summation of sensitivities of its individual weights. Additionally, the method incorporates layer sensitivity by considering Hessian eigenvalues, thereby automating the process of determining the optimal pruning rate for each layer.

2.3. Analysis and discussion

By analyzing pruning algorithms from different periods, including the latest ones, we can observe significant advantages of unstructured pruning. The most notable advantage is its ability to directly zero out or trim a large number of parameters, resulting in a highly sparse model that does not significantly affect model accuracy. Additionally,

unstructured pruning can modify parameters based on the underlying logic of different hardware, leading to improved acceleration. However, unstructured pruning also has noticeable drawbacks. Firstly, due to its consideration of the impact of individual neurons on the network, unstructured pruning algorithms can be computationally intensive. Secondly, simply applying unstructured pruning does not directly accelerate sparse matrix computations, as the size of the pruned matrix remains unchanged. This means that sparse matrix multiplication and other computations are still required, which may not yield substantial acceleration on certain hardware. Moreover, unstructured pruning algorithms may rely on specific software or hardware implementations, limiting their flexibility and portability across different deep-learning frameworks. In contrast, structured pruning algorithms have advantages in these aspects. Structured pruning reduces computational complexity, simplifies sparse matrix computations, and is easier to use across different deep learning frameworks. Consequently, recent research has been inclined towards employing structured pruning algorithms for model pruning [62-67].

Structured pruning algorithms have advantages in terms of hardware acceleration and prediction accuracy because they consider a more comprehensive set of factors. Compared to unstructured pruning, structured pruning can achieve model compression and acceleration by pruning entire convolutional kernels or channels. However, structured pruning algorithms also have some limitations. Firstly, in convolutional kernel pruning algorithms, the relationships between kernels are often overlooked. Kernels sometimes work together in a coordinated manner to achieve accurate predictions. Pruning based solely on the individual significance of each kernel may not lead to the optimal pruning results. Secondly, for new models, one-time pruning with structured pruning algorithms often struggles to maintain the same level of accuracy as the original model. Therefore, algorithm-level optimizations are needed to achieve better accuracy preservation. Additionally, conventional structured pruning algorithms require manual configuration of pruning thresholds and other hyperparameters, which limits the automation of the algorithm. As a result, fully automated learning modes cannot be realized [68-71].

To sum up, structured pruning algorithms have significant advantages over unstructured pruning in terms of hardware acceleration and prediction accuracy. However, there are still challenges to address. For example, it is crucial to consider the relationships between convolutional kernels and optimize the algorithms at the algorithmic level to achieve better accuracy preservation. Additionally, the level of automation in the algorithms needs to be further improved to facilitate a more convenient and efficient model pruning process. These are important directions in current research to further enhance the effectiveness of structured pruning algorithms.

3. Network Architecture Design

The design of lightweight network architectures aims to reduce model complexity and decrease computational resource consumption by optimizing the network architecture [72-74]. The goal of this design is to create more efficient network structures that achieve model size compression, faster runtime, and reduced training difficulty. In the network architecture design, this paper discusses how to achieve model lightweight through two aspects: lightweight module design and convolutional structure optimization.

3.1 Lightweight module design

The design of lightweight modules aims to reduce model complexity by creating compact and efficient network modules. These modules often employ specific structures and operations to minimize the number of parameters and computational requirements. Additionally, lightweight module design adopts a modular approach, breaking down the network into smaller modules and constructing the entire network by combining these modules. This modular design enhances the flexibility and scalability of the network.

3.1.1 Fire module

The structure of the Fire module consists of two sub-layers: the squeeze layer and the expand layer. The squeeze layer utilizes a 1×1 convolutional kernel, while the expand layer employs both 1×1 and 3×3 convolutional kernels. **Figure 7** illustrates the structure of the Fire module.

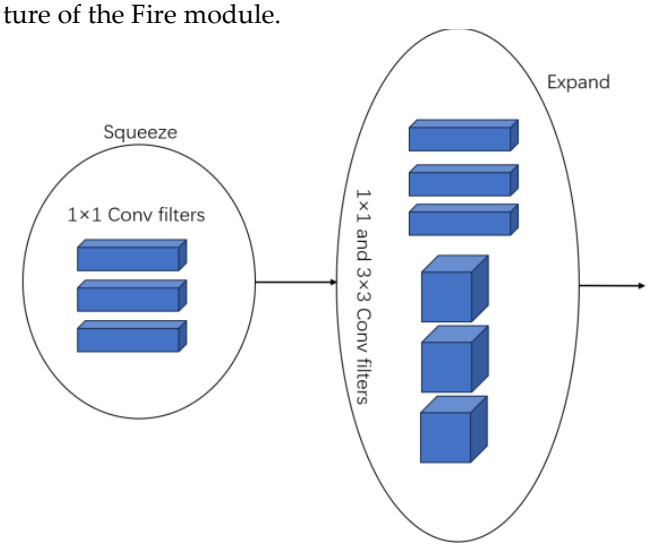

Figure 7. Fire module schematic

To reduce the number of network parameters, the Fire module utilizes the design of the squeeze and expand layers. In the expand layer, a 1×1 convolutional kernel is used instead of a 3×3 convolutional kernel to decrease the number of 3×3 convolutional kernels. Simultaneously, the squeeze layer employs a 1×1 convolutional kernel to limit the output channel count. This design strategy was applied in the classic SqueezeNet [75], where the Fire module serves as its core module. Compared to AlexNet [76], the SqueezeNet network constructed by stacking Fire modules reduces the number of parameters by 50 times while maintaining comparable accuracy. In addition to SqueezeNet, a novel neural network architecture called SqueezeNext is introduced in reference [77]. SqueezeNext combines the design principles of SqueezeNet and tensor decomposition. By decomposing the convolutional operation into an additive step followed by separable convolutional operations, SqueezeNext achieves a 3.2 times faster speed than SqueezeNet without sacrificing accuracy. Reference [78] presents an optimized SqueezeNet-based squeeze network model (OSQN-DNN) for unmanned aerial vehicle (UAV) aerial image classification. This model uses OSQN as the feature extractor and applies the Coyote Optimization Algorithm (COA) [79] to optimize the hyperparameter selection in the SqueezeNet model, significantly improving the overall classification performance. Experimental results demonstrate that the OSQN-DNN model achieves better accuracy and runtime inference time compared to SqueezeNet on the benchmark UCM dataset.In reference [80], an enhanced version of the SqueezeNet convolutional neural network is introduced. This improved model incorporates data preprocessing techniques such as data normalization and the Synthetic Minority Over-sampling Technique (SMOTE). Furthermore, the continuous wavelet transform is utilized to generate spectrograms, which are then employed for training and testing the modified SqueezeNet model. The results demonstrate that this enhanced SqueezeNet model achieves a remarkable accuracy of 90%. In reference [81], a novel approach combining the Aquila Sine Cosine Algorithm (ASCA) with the SqueezeNet model is presented. This integration aims to reduce both the training time and computational complexity of the detection process. By leveraging the ASCA technique, the weights of Deep Convolutional Neural Networks (DCNN) and SqueezeNet

are updated, resulting in improved efficiency. Experimental results demonstrate the superior performance of this combined model.

3.1.2 Ghost module

This paper visualizes the process of neural networks extracting features from data, as shown in **Figure 8**. During the feature extraction process, many features are similar. These redundant feature maps increase the computational burden of the network, and removing them would significantly degrade the model's recognition performance. To address this issue, Ghost provides a method to generate a large number of similar feature maps at a smaller computational cost. The core idea of the Ghost module is to generate multiple feature maps by sharing convolutional kernels, thereby reducing the number of parameters and computations. By leveraging inexpensive linear transformation operations (Cheap), the Ghost module can extract rich feature representations while maintaining a smaller computational cost. This makes the Ghost module highly applicable in lightweight network designs.

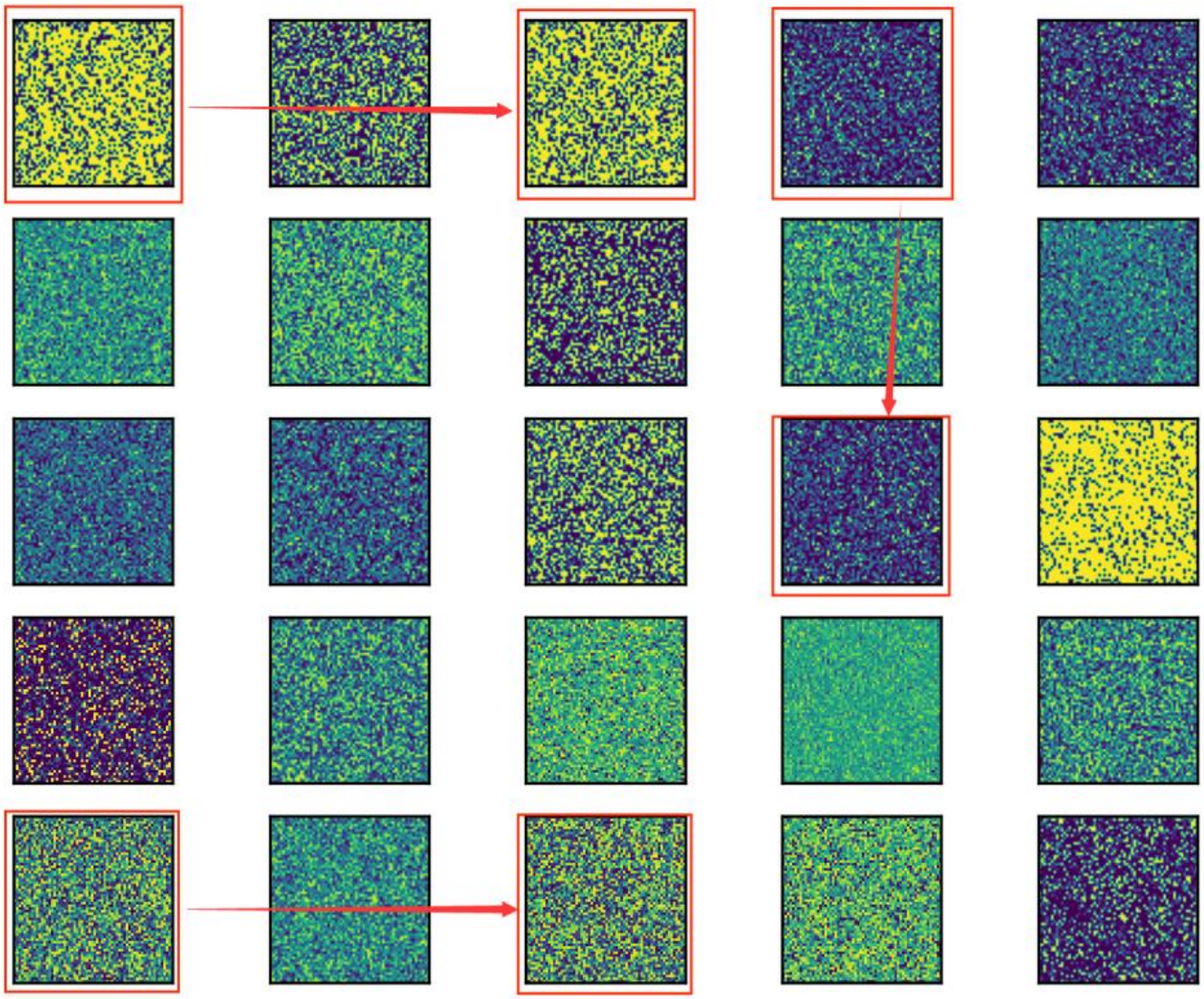

Figure 8. Neural network visualization feature map.

Let's analyze the principle behind Ghost in reducing computations from a theoretical perspective, as shown in **Figure 9.** Assuming an input feature map has a channel count of c we use m ordinary convolutional kernels size of $k \times k$ to generate m intermediate feature

maps. Each intermediate feature map is then transformed into s "ghost" feature maps through a series of linear operations (Cheap), combined with the identity mapping of the previous m intermediate feature maps, resulting in n output feature maps. In the Ghost module, the number of identity mappings is $m=\frac{n}{s}$, and the number of linear operations is $m\times(s-1)$. Let's assume the kernel size for each linear operation is $d\times d$. Therefore, the number of parameters for identity mappings in the Ghost module is $m\times c\times k\times k$, and the number of parameters for linear operations is $(s-1)\times m\times d\times d$. The magnitude of $d\times d$ is similar to $k\times k$ ($k\approx d$). Typically, the channel count of the input feature map c is much larger than the number of feature maps generated by linear operations s. On the other hand, the number of parameters for generating n output feature maps using ordinary convolution is $n\times c\times k\times k$.

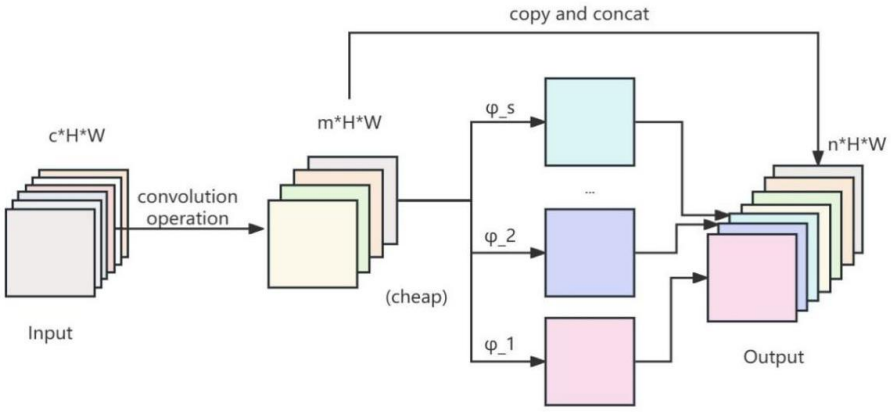

Figure 9. Ghost Convolution Principle.

Next, we further analyze the memory and computational benefits brought by using Ghost convolution through mathematical derivation, as shown in Equation (3). Through theoretical analysis, replacing ordinary convolution with Ghost convolution can reduce the number of convolutional parameters while obtaining the same number of feature maps, effectively reducing the model parameter count by approximately s-fold.

$$r_c=\frac{n\times c\times k\times k}{\frac{n}{s}\times c\times k\times k+(s-1)\times\frac{n}{s}\times d\times d}\approx\frac{s\times c}{s+c-1}\approx s \tag{3}$$

GhostNet [82] is a lightweight neural network based on the MobileNetV3 [83] architecture. It replaces ordinary convolutions in MobileNetV3 with Ghost modules, forming Ghost bottlenecks, and builds GhostNet upon this foundation. Experimental results have demonstrated that compared to other lightweight neural network architectures such as MobileNet series and ShuffleNet series, GhostNet achieves higher accuracy in ImageNet classification tasks while having comparable parameter and computational counts. In reference [84], researchers applied GhostNet to the backbone network of YoloV4, resulting in an improved network called Ghostnet-YoloV4. This enhanced network efficiently extracts features and significantly reduces the number of real-time counting operations. Through field testing in nursery plots, this method not only effectively overcomes noise interference in large field environments but also meets the computational requirements of low-configured management system embedded mobile devices. The counting and measurement accuracy both exceed 92%. To further enhance the performance of lightweight image recognition models, reference [85] introduces L-GhostNet. This model integrates

group convolution learning and an improved Channel Attention (CA) method into GhostNet. Experimental results show that compared to GhostNet, L-GhostNet achieves slightly higher accuracy across various datasets while reducing computational cost by over 44% and parameter count by over 33%. It also provides a 26% increase in frames per second (FPS). When compared to commonly used lightweight network models, such as MobileNets and ShuffleNets, operating at the same FLOP (Floating Point Operations) level, L-GhostNet demonstrates superior performance. L-GhostNet achieves the lowest FLOPs, highest accuracy, and fewer parameters, showcasing its exceptional overall performance. Furthermore, reference [86] proposes a CBAM-GhostNet-SSD network, which introduces Ghost modules and Efficient Channel Attention (ECA) mechanism to the SSD object detection algorithm. By dynamically allocating parameters and changing the weights of detection regions, this method improves the model's performance. To enhance the recognition accuracy of small objects, the CBAM module is also introduced. Compared to the original SSD network, the CBAM-GhostNet-SSD network reduces parameter and computational counts by 98.23% and achieves a 14.5% increase in mAP.

3.2 Convolutional structure optimization

Convolutional structure optimization aims to reduce computational resource consumption by optimizing the convolutional layers. Common optimization methods include the use of lightweight convolutional operations such as depthwise separable convolution and grouped convolution. These lightweight convolutional operations maintain lower computational complexity while still preserving a certain level of prediction accuracy.

3.2.1 Group convolution

Grouped convolution first appeared in AlexNet and was designed to address limited hardware resources. It allows for parallel computation on two GPUs, with their results subsequently fused. Grouped convolution divides the input feature map into groups based on channels and applies convolutional operations to each group individually. The results of the grouped convolution are then concatenated along the channel dimension to obtain the final output feature map. This operation has a lightweight effect, reducing computational complexity.

Assuming that the input feature map is $H \times W \times N$, the size of the convolution kernel is $K \times K$, and M convolution kernels are used to perform the convolution operation, the output is $H' \times W' \times M$. The computational amount of the standard convolution is $K \times K \times M \times H' \times W'$. When the input feature channels are divided into G groups in the grouped convolution, the calculation amount is $\frac{1}{G}$ of the standard convolution.

Therefore, grouped convolution reduces the computational burden by dividing the input feature channels into multiple groups. This is particularly useful in scenarios with limited hardware resources. However, it is important to note that grouped convolution also introduces some information loss since there is no direct interaction between channels within each group, resulting in inadequate fusion of information across feature maps [87-89]. Therefore, when choosing the number of groups, a balance between computational efficiency and model performance needs to be considered. ShuffleNetV1 [90] proposed a channel shuffle method to address the limitation of information exchange between groups in grouped convolution. In order to maintain the recognition accuracy, a uniform and random shuffling is performed on the feature maps of grouped convolution, as shown in **Figure 10**. The purpose of this channel shuffle operation is to ensure that the input information for the next grouped convolution comes from different groups. ShuffleNet achieved a 13-fold speed improvement compared to AlexNet while maintaining accuracy. However, ShuffleNetV2 [91] pointed out that evaluating model performance solely based on parameter count and floating-point operations (FLOPs) is inaccurate. Based on experimental observations, the actual runtime of a model depends not only on computational

operations but also on factors such as memory read/write, GPU parallelism, and file I/O. Therefore, ShuffleNetV2 proposed four guidelines to improve model efficiency: (1) Use convolutions with the same number of input and output channels. This minimizes memory read/write and communication overhead. (2) Reduce the use of grouped convolution. Although grouped convolution reduces computational burden, it also introduces information isolation. Therefore, ShuffleNetV2 suggests minimizing the use of grouped convolution to improve information exchange and overall performance. (3) Reduce network branches. Branch operations in the network increase computational and communication overhead. (4) Minimize element-wise operations.

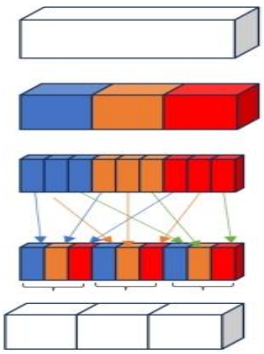

Figure 10. Channel shuffle.

ShuffleNetV2 is an improvement over ShuffleNetV1 based on the four guidelines mentioned earlier. Its structure is shown in **Figure 11**. In the residual branch of ShuffleNetV2, a 1×1 convolution with the same number of input and output channels is introduced to meet the requirement of guideline (1). Simultaneously, the use of grouped convolution is abandoned to comply with guideline (2). Finally, the feature addition operation is replaced with the channel concatenation (Concat) operation, aligning with the guideline (4). These improvements not only enhance the runtime speed but also improve the accuracy. Experimental results show that ShuffleNetV2 achieves a 63% speed improvement compared to ShuffleNetV1.

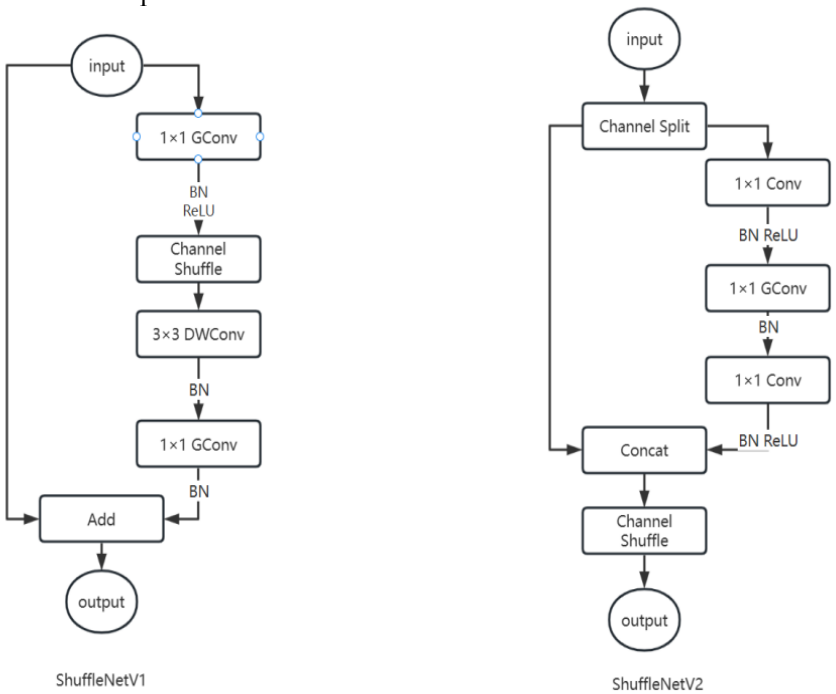

Figure 11. Comparison of ShuffleNetV1 and V2 structures.

In the latest research on ShuffleNet, a lightweight network called (2+1) D Distilled ShuffleNet is proposed in [92] for human action recognition using an unsupervised distillation learning paradigm. This network extracts knowledge from the teacher network through distillation techniques without the need for labeled data. On the UCF86 and HMDB4 datasets, (2+1) D Distilled ShuffleNet achieves better accuracy and inference runtime than other state-of-the-art distilled networks. Furthermore, reference [93] presents a lightweight garbage classification model known as Garbage Classification Network (GCNet), which builds upon ShuffleNetV2 with three notable enhancements: the incorporation of the Parallel Mixed Attention Mechanism (PMAM), utilization of a novel activation function, and the application of transfer learning. Experimental findings indicate that GCNet achieves an outstanding average accuracy of 97.9% on a custom dataset, showcasing a significant improvement of nearly 10% compared to ShuffleNetV2, while maintaining a similar number of model parameters. Furthermore, in [94], a recognition of individuals (RE) network called ShuffleNet-Triplet is proposed for individual cow recognition. This network utilizes ShuffleNetV2 for feature extraction to reduce the number of parameters and strengthen the network's ability to distinguish similar individuals by organically combining the triplet loss and cross-entropy loss. BNNeck is also introduced to reduce conflicts between the two loss functions. Experimental results show that ShuffleNet-Triplet achieves a 6.88% improvement in average accuracy compared to the ShuffleNetV2 model.

3.2.2 Deep separable convolution

Deep separable convolution is a convolutional operation that consists of two steps: depthwise convolution and pointwise convolution. This convolution operation effectively reduces computational complexity and model parameters. In the depthwise convolution step, each channel of the input features is convolved separately, as shown in **Figure 12(a)**. The purpose of this step is to capture features while preserving their spatial information. Next, in the pointwise convolution step, the output of the depthwise convolution is convolved with a 1×1 convolutional kernel, as shown in **Figure 12(b)**. The goal of pointwise convolution is to fuse information from different channels by convolving the set of output feature maps from the depthwise convolution with a 1×1 kernel. This process generates the final output feature map.

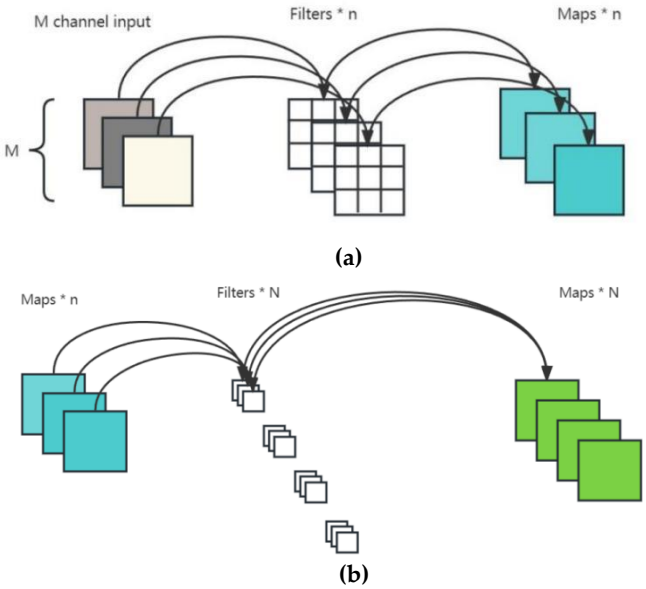

Figure 12. Depth-separable convolution principle. **(a)** Depth-Wise convolution; **(b)** Point-Wise convolution.

Such a combination can make the model achieve the effect of lightweight, here we assume an input feature map height and width of H and W, the number of channels is M, the output feature map height and width are unchanged, the number of channels is N. The number of standard convolution kernels is N, the size is $K \times K \times M$. Then the standard convolution computation is $K \times K \times M \times N \times H \times W$. The computation of the depth convolution in the depth-separable convolution is $K \times K \times M \times 1 \times H \times W$, and the computation of the point-by-point convolution is $1 \times 1 \times M \times N \times H \times W$. The ratio C_r obtained by comparing the total computation of the depth-separable convolution with that of the standard convolution is shown in in Equation (4):

$$C_r = \frac{K \times K \times M \times 1 \times H \times W + 1 \times 1 \times M \times N \times H \times W}{K \times K \times M \times N \times H \times W} = \frac{1}{N} + \frac{1}{K^2} \tag{4}$$

This ratio C_r can be used to measure the extent of computational reduction achieved by using deep separable convolution compared to standard convolution. Similarly, the ratio of parameter count between deep separable convolution and standard convolution is $\frac{1}{N} + \frac{1}{K^2}$. In Equation (4), the size of the convolutional kernel is usually 3×3, and the computational complexity and parameter count of deep separable convolution are approximately $\frac{1}{9}$ to $\frac{1}{8}$ of standard convolution.

The MobileNet series is a collection of lightweight network models based on deep separable convolution. These models aim to reduce computational complexity and parameter count while maintaining good performance, particularly in tasks such as image classification. MobileNetV1 [95] was the first model in this series, which replaced traditional convolutional operations with deep separable convolution, resulting in a significant reduction in computational complexity and parameter count. In the ImageNet classification task, MobileNetV1 achieved comparable performance to traditional network models such as GoogleNet and VGG-16 while reducing the model parameters by nearly 30 times. MobileNetV2 [96] further improved upon MobileNetV1 by introducing linear bottleneck structures and inverted residual blocks. The linear bottleneck structure combines depthwise convolution and pointwise convolution, enhancing both speed and accuracy. The inverted residual structure improves information flow and feature propagation. MobileNetV3 incorporates neural architecture search (NAS) to automatically obtain optimal network parameters and introduces the SE attention mechanism to enhance feature interaction between channels. This architecture demonstrated better performance than MobileNetV1 and MobileNetV2 in the ImageNet classification task. The latest research on the MobileNet series includes Mobile-Former [97], which combines MobileNet with the Transformer architecture to create a lightweight framework. By leveraging the advantages of MobileNet in local processing and Transformer in global interaction through a bidirectional bridge connection, this structure achieves bidirectional fusion of local and global features. In the ImageNet classification task, it achieved a 3.3% improvement in accuracy compared to MobileNetV1 while reducing computational complexity by 17%. A-MobileNet [98] introduces attention modules and parameter optimizations to the MobileNetV1 model, demonstrating better performance on FERPlus and RAF-DB datasets compared to other models. Additionally, BM-Net [99] is a lightweight network composed of a bilinear structure and MobileNet-V3, specifically designed for analyzing breast cancer whole-slide images (WSI). Experimental results show its significant potential in breast cancer WSI detection.

3.3 Analysis and discussion

From the performance comparison of various lightweight networks summarized below (**Table 1**), it can be seen that GhostNet exhibits the best overall performance among

these lightweight neural networks. It achieves the highest accuracy while maintaining relatively low computational complexity. The SqueezeNet series has higher computational complexity, poor scalability, and relatively lower recognition accuracy, but it has a smaller parameter count. The ShuffleNet series has lower computational complexity and utilizes channel shuffling operations to better leverage the information from each channel, thereby improving network accuracy. The MobileNet series, although not as impressive as other lightweight neural networks in experimental data, has widely applied the concept of deep separable convolution in many large-scale networks that require lightweight designs [100-102]. Each classic lightweight network model has its unique advantages, and there is currently no single network that can perfectly balance speed and accuracy. This provides us with a direction for future research, which is to explore better lightweight network models that achieve a better balance between speed and accuracy [103-105].

Table 1. Comparison table of four lightweight network model families.

Model	FLOPs/M	Params/M	Accuracy/%	DataSets
SqueezeNet[75]	837	1.20	Top-1:57.5 Top-5:80.3	ImageNet
SqueezeNext[77]	228	0.74	Top-1:58.58 Top-5:82.09	ImageNet
GhostNet[82]	141	5.2	Top-1:73.9 Top-5:91.4	ImageNet
ShuffleNetV1(g=3)[90]	140	2.4	Top-1:67.4	ImageNet
ShuffleNetV2[91]	142	-	Top-1:69.4	ImageNet
MobileNetV1[95]	569	4.2	Top-1:70.6	ImageNet
MobileNetV2[96]	300	3.4	Top-1:72.0	ImageNet
MobileNetV3[83]	219	5.4	Top-1:73.8	ImageNet

Movements and performance of these four classic network models in recent years. However, it can be observed that these papers mainly focus on applications in different domains, with less emphasis on innovative network structure design or lightweight module improvements. From the "Improvements" column in **Table 2**, it can be seen that some papers attempt to improve accuracy by introducing different attention mechanisms. However, this often increases the complexity of the network model, making it challenging to achieve lightweight goals. Other papers replace the backbone network of large models with these four classic lightweight networks to achieve lightweight models, but this often significantly reduces the recognition accuracy of the models.

However, only a few papers [106-121] have conducted further optimization research based on lightweight network architectures. For example, in the MicroNet model proposed in [121], the concept of microfactorized convolution is introduced. This method decomposes the convolution matrix into low-rank matrices to integrate sparse connections into the convolution, resulting in significant performance improvements at low FLOP states, surpassing existing techniques. Therefore, to promote the development of lightweight network computations, our research direction should focus more on innovative network structures and improvements in lightweight modules. Such efforts will reduce the complexity of models while maintaining good performance, and further drive the advancement of lightweight network computations.

Table 2. Improvement and performance of the latest lightweight networks.

Models	Improvements	Performance
Modified SqueezeNet[122]	Improved SqueezeNet architecture by reducing the number of Fire modules and increasing the number of pooling layers	Improved precision by 28% and recall by 20% compared to the original model SqueezeNet

OSQN-DNN[78]	1) use the Coyote Optimization Algorithm (COA) to optimize the hyperparameters involved in the SqueezeNet model. 2) using DNN models as classifiers to assign appropriate class labels.	Compared to the existing high caliber DLPSO algorithm, the accuracy is 1% higher and the runtime is reduced by 0.13s.
ASCA-SqueezeNet[81]	The weights of SqueezeNet are updated using the Aquila Sine Cosine Algorithm (ASCA).	Compared with SqueezeNet, the test accuracy is improved by 6.2% and the computation time is reduced by 1s.
CBAM-GhostNet-SSD[86]	1) Introduce the Ghost module into the SSD network and add the ECA attention mechanism. 2) Add CBAM module to the network.	Compared to Ghost-SSD, mAP is up 1% and FPS is up 3 frames/s.
GhostNet-YOLOv4[84]	1) replace the backbone network of YOLOv4 with GhostNet. 2) replacing the normal convolutional blocks of PANet in YOLOv4 with depth-separable convolutional blocks.	Nearly 4% improvement in mAP for identifying nursery saplings.
L-GhostNet[85]	1) Improved the Ghost module in GhostNet. 2) Introduced improved CA attention mechanism (p-CA)	Compared to the original model GhostNet, it reduces the amount of computation by 44% and the number of parameters by 33%, and improves the FPS by 26%.
D Distilled ShuffleNet[90]	1) distill learning using ShuffleNet as a student network. 2) knowledge extraction from two teacher networks.	Compared to the ResNet-18 backbone, the accuracy is improved by 0.7% on the UCF01 dataset, the amount of parameters is reduced by almost 30%, and the amount of computation is reduced by almost 60%.
GCNet[93]	1) Added CBAM attention mechanism. 2) replaced the Relu activation function with FRelu. 3) Used transfer learning.	Compared with the original model ShuffleNetV2, the accuracy is improved by 4.5%, and the number of parameters and computation are reduced by nearly 8%.
ShuffleNet-Triplet[94]	The triple loss function and the cross-entropy loss are calculated separately using the BNNeck structure, and then the two are combined.	Nearly 3% improvement in recognition of individual cows compared to the original model ShuffleNetV2.
Mobile-Former[97]	Enabling two-way cross-talk combines MobileNet and Transform.	The accuracy is 1.3% higher than MobileNetV3 and the computation is reduced by 17.4%.
A-MobileNet[98]	1) Introduce CBAM attention mechanism in MobileNetV1. 2) Combine center loss and softmax loss to optimize model parameters.	Accuracy is about 3% higher than MobileNetV1 on the RAF-DB dataset.
BM-Net[99]	Replaced MobileNetV3's classifier with a bilinear structure.	On the BACH-challenged Part B WSI segmentation dataset, the average accuracy is improved by 1% compared to MobileNetV3.

4. Challenges and Prospects

Currently, intelligent mobile devices are moving in the direction of edge computing and lightweight development. A key research focus at present is how to minimize model latency and storage space while maintaining neural network model accuracy to the greatest extent possible.

Most existing methods in model pruning algorithms eliminate redundant connections or neurons in the network. However, these low-level pruning methods pose non-structural risks. Irregular memory access patterns during computer operations can also impede further acceleration of the network [123-125]. In contrast, structurally pruned net-

works have smaller model sizes, faster execution speeds, and reduced storage space requirements, making them more suitable for deployment on computationally limited mobile devices. Among them, convolutional kernel pruning is one of the hot research topics in structured pruning. Most convolutional kernel pruning algorithms typically involve three classic steps: pre-training, pruning, and fine-tuning. However, in [126], various algorithmic evaluations were conducted on multiple network structures, revealing that training small target models from random initialization can achieve identical or even better model performance than classical three-step pruning algorithms. Additionally, training models from scratch can also achieve equivalent or better performance than fine-tuning models. This indicates that in pruning algorithms, it is more important to find suitable network structures rather than considering how to preserve important weights within existing structures. Furthermore, most current convolutional kernel pruning algorithms only prune in a single dimension such as depth, width, or resolution, which may lead to excessive loss in a particular dimension and reduce accuracy, while the compression rate of the model may not be significantly high. Exploring pruning from multiple dimensions could potentially yield better results, and this is an avenue worth investigating in the future [127]. Therefore, future research should focus on designing appropriate network architectures and exploring multidimensional pruning methods that prioritize achieving high efficiency pruning algorithms while maintaining model accuracy. This will facilitate the deployment of lightweight neural network models on computationally limited mobile devices.

However, there are still some challenges in pruning algorithms. For example, the evaluation systems and metrics used to assess the importance of weights and the performance of pruning algorithms are often oversimplified. Therefore, it is crucial to propose effective methods for measuring the impact of pruning on models, which remains a challenge in model pruning algorithms. Researching these challenges in pruning methods and proposing algorithms with superior performance holds great potential for development. Additionally, exploring algorithms that do not rely on manually designed hyperparameters is a promising direction. Currently, research in this area is relatively limited but holds significant importance for the advancement of pruning algorithms.

Significant progress has been made in network architecture design, primarily focusing on designing lighter modules and optimizing convolutional structures. Currently, there are two main approaches to designing lightweight models: (1) Improving existing lightweight modules based on specific requirements. (2) Designing traditional modules that meet the desired criteria and then replacing the convolutions in these modules with lightweight convolutions, adjusting the structural relationships between modules using existing lightweight functional structures or tools such as activation functions. Most lightweight networks adopt the first approach, which has achieved noticeable results in terms of model lightweighting. However, this approach has reached a plateau in terms of the degree of model lightweighting, making it difficult to achieve further breakthroughs. In contrast, the second approach is more challenging because it heavily relies on the designer's expertise and prior knowledge of deep learning. Designers need to possess a wealth of prior knowledge, such as how to design structures that resemble sparsely connected connections between human neurons in principle, and how to control the influence of prior knowledge while improving performance metrics like latency, computation speed, and storage space by altering the network structure.

Currently, reinforcement learning-based neural network architecture search is the mainstream approach for network architecture design. This method uses a reinforcement learning controller to search and generate network structures within the search space, eliminating the need for extensive manual effort. This is a key reason for its rapid development. However, reinforcement learning-based neural network architecture search methods tend to focus too much on improving model accuracy while neglecting the limitations of underlying hardware devices. The resulting models often have high hardware

requirements and are challenging to deploy on embedded devices. Therefore, lightweight network architecture design still faces some challenges. Future research directions include finding a balance between model accuracy and hardware requirements and considering the limitations of underlying hardware devices during the design process. It is essential to develop methods that can optimize both the performance of the model and its compatibility with resource-constrained devices.

Indeed, whether it is model pruning or network architecture design, the goal is not only to reduce the complexity of convolutional neural networks but also to maintain or even improve the original accuracy. With the rapid development of deep learning, emerging technologies such as Graph Neural Networks (GNNs) [128-130], and the integration of neural networks with Transformers (e.g., Vision Transformers or ViTs) [131-133], have gradually matured and gained recognition in the academic community. However, how to lightweight these models and apply them to real-world industrial applications while ensuring post-application security is a significant challenge for model compression and acceleration techniques [134,135].

5. Conclusion

This paper summarizes the methods of model pruning and network architecture design in lightweight neural networks. Regarding model pruning methods, a comparison and analysis of structured pruning and unstructured pruning algorithms are presented, highlighting their characteristics. Generally, unstructured pruning exhibits irregularity and requires specific hardware to leverage its advantages, while structured pruning offers more pruning options and is more easily applicable to general-purpose hardware. Currently, structured pruning algorithms are commonly used. In terms of network architecture design, this paper provides an overview of four lightweight neural networks: SqueezeNet, GhostNet, ShuffleNet, and MobileNet. The mathematical principles behind their ability to achieve model lightweight are explained, and their performance on the ImageNet dataset is compared and analyzed. Based on existing work, it can be observed that for model pruning, there is a need to establish a comprehensive evaluation metric system to measure algorithm performance effectively. Regarding network architecture design, further exploration is required to develop methods that can accelerate computation speed and reduce storage space while maintaining the accuracy of the original model as much as possible.

References

1. Ge, D., Li, H., Zhang, L., et al. Survey of lightweight neural network. Journal of Software, 2020, 31: 2627-2653.
2. Kumari, A., Sharma, N. A Review on Convolutional Neural Networks for Skin Lesion Classification. International Conference on Secure Cyber Computing and Communications. IEEE, 2021. https://doi.org/10.1109/icsccc51823.2021.9478151
3. Bouguettaya, A., Kechida, A., TABERKIT, A. M. A survey on lightweight CNN-based object detection algorithms for platforms with limited computational resources. International Journal of Informatics and Applied Mathematics, 2019, 2(2): 28-44.
4. Jinlin M A, Yu Z, Ziping M A, et al. Research Progress of Lightweight Neural Network Convolution Design. Journal of Frontiers of Computer Science and Technology, 2022, 16(3): 512-528. https://doi.org/10.3778/j.issn.1673-9418.2107056
5. Shen, X., Yi, B., Liu, H., et al. Deep variational matrix factorization with knowledge embedding for recommendation system, IEEE Transactions on Knowledge and Data Engineering, 2019, 33(5): 1906-1918. https://doi.org/10.1109/tkde.2019.2952849
6. Li, Z., Li, H., Meng, L. Model Compression for Deep Neural Networks: A Survey. Computers, 2023, 12(3): 60. https://doi.org/10.3390/computers12030060
7. Zeng, Y., Xiong, N., Park, J. H., et al. An emergency-adaptive routing scheme for wireless sensor networks for building fire hazard monitoring. Sensors, 2010, 10(6): 6128-6148. https://doi.org/10.3390/s100606128
8. Li, Y., Liu, J., & Wang, L. Lightweight network research based on deep learning: A review. In 2018 37th Chinese control conference (CCC), IEEE, July, 2018. https://doi.org/10.23919/chicc.2018.8483963
9. Zheng, M., Tian, Y., Chen, H., et al. Lightweight network research based on deep learning. International Conference on Computer Graphics, Artificial Intelligence, and Data Processing (ICCAID 2021). SPIE, 2022, 12168: 333-338. https://doi.org/10.1117/12.2631211

10. Xiao, Y., Tian, Z., Yu, J., et al. A review of object detection based on deep learning. Multimedia Tools and Applications, 2020, 79: 23729-23791. https://doi.org/10.1007/s11042-020-08976-6
11. Wang, C., Huang, K., Yao, Y., et al. Lightweight deep learning: An overview. IEEE Consumer Electronics Magazine, 2022. https://doi.org/10.1109/MCE.2022.3181759
12. Kang, L., Chen, R., Xiong, N., et al. Selecting hyper-parameters of Gaussian process regression based on non-inertial particle swarm optimization in Internet of Things. IEEE Access, 2019, 7: 59504-59513. https://doi.org/10.1109/access.2019.2913757
13. Zhao, J., Huang, J., Xiong, N. An effective exponential-based trust and reputation evaluation system in wireless sensor networks. IEEE Access, 2019, 7: 33859-33869. https://doi.org/10.1109/access.2019.2904544
14. Yao, J., Li, P., Kang, X., et al. A pruning method based on the dissimilarity of angle among channels and filters. 2022 IEEE 34th International Conference on Tools with Artificial Intelligence (ICTAI). IEEE, 2022: 528-532. https://doi.org/10.1109/ictai56018.2022.00084
15. Cong, S., Zhou, Y. A review of convolutional neural network architectures and their optimizations. Artificial Intelligence Review, 2023, 56(3): 1905-1969. https://doi.org/10.1007/s10462-022-10213-5
16. Hu, W., Fan, J., Du, Y., et al. MDFC–ResNet: an agricultural IoT system to accurately recognize crop diseases. IEEE Access, 2020, 8: 115287-115298. https://doi.org/10.1109/ACCESS.2020.3001237
17. Huang, S., Zeng, Z., Ota, K., et al. An intelligent collaboration trust interconnections system for mobile information control in ubiquitous 5G networks. IEEE transactions on network science and engineering, 2020, 8(1): 347-365. https://doi.org/10.1109/tnse.2020.3038454
18. Anwar, S., Hwang, K., Sung, W. Structured pruning of deep convolutional neural networks. ACM Journal on Emerging Technologies in Computing Systems (JETC), 2017, 13(3): 1-18. https://doi.org/10.1145/3005348
19. LeCun, Y., Denker, J., Solla, S. Optimal brain damage. Advances in neural information processing systems, 1989, 2: 598-605. https://doi.org/http://dx.doi.org/
20. Hassibi, B., Stork, D. Second order derivatives for network pruning: Optimal brain surgeon. Advances in neural information processing systems, 1992, 5.
21. Thimm, G., Fiesler, E. Evaluating pruning methods. Proceedings of the International Symposium on Artificial neural networks. 1995: 20-25.
22. Srinivas, S., Babu, R. V. Data-free parameter pruning for deep neural networks. arXiv preprint arXiv:1507.06149, 2015. https://doi.org/10.5244/c.29.31
23. Han, S., Pool, J., Tran, J., et al. Learning both weights and connections for efficient neural network. Advances in neural information processing systems, 2015, 28. https://doi.org/10.48550/arXiv.1506.02626
24. Han, S., Mao, H., Dally, W. J. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. arXiv preprint , 2015. https://doi.org/10.48550/arXiv.1510.00149
25. Han, S., Liu, X., Mao, H., et al. EIE: Efficient inference engine on compressed deep neural network. ACM SIGARCH Computer Architecture News, 2016, 44(3): 243-254. https://doi.org/10.1109/isca.2016.30
26. Guo, Y., Yao, A., Chen, Y. Dynamic network surgery for efficient dnns. Advances in neural information processing systems, 2016, 29. https://doi.org/10.48550/arXiv.1608.04493
27. Hu, H., Peng, R., Tai, Y., et al. Network trimming: A data-driven neuron pruning approach towards efficient deep architectures. arXiv preprint, 2016. https://doi.org/10.48550/arXiv.1607.03250
28. Louizos, C., Welling, M., Kingma, D. P. Learning sparse neural networks through regularization. arXiv preprint, 2017. https://doi.org/10.48550/arXiv.1712.01312
29. Ye, M., Gong, C., Nie, L., et al. Good subnetworks provably exist: Pruning via greedy forward selection. International Conference on Machine Learning. PMLR, 2020: 10820-10830. https://doi.org/10.48550/arXiv.2003.01794
30. Frankle, J., Carbin, M. The lottery ticket hypothesis: Finding sparse, trainable neural networks. arXiv preprint, 2018. https://doi.org/10.48550/arXiv.1803.03635
31. Wang, C., Zhang, G., Grosse, R. Picking winning tickets before training by preserving gradient flow. arXiv preprint, 2020. https://doi.org/10.48550/arXiv.2002.07376
32. Zhang, T., Ye, S., Zhang, K., et al. StructADMM: A Systematic, High-Efficiency Framework of Structured Weight Pruning for DNNs. 2018. https://doi.org/10.48550/arXiv.1807.11091
33. Xue, W., Bai, J., Sun, S., et al. Hierarchical Non-Structured Pruning for Computing-In-Memory Accelerators with Reduced ADC Resolution Requirement. 2023 Design, Automation & Test in Europe Conference & Exhibition (DATE). IEEE, 2023: 1-6. https://doi.org/10.23919/date56975.2023.10136975
34. Laurent, C., Ballas, C., George, T., et al. Revisiting loss modelling for unstructured pruning. arXiv preprint, 2020. https://doi.org/10.48550/arXiv.2006.12279
35. Vahidian, S., Morafah, M., Lin, B. Personalized federated learning by structured and unstructured pruning under data heterogeneity. 2021 IEEE 41st international conference on distributed computing systems workshops (ICDCSW). IEEE, 2021: 27-34. https://doi.org/10.48550/arXiv.2105.00562
36. Chen, X., Zhu, J., Jiang, J., et al. Tight compression: compressing CNN model tightly through unstructured pruning and simulated annealing based permutation. 2020 57th ACM/IEEE Design Automation Conference (DAC). IEEE, 2020: 1-6. https://doi.org/10.1109/dac18072.2020.9218701

37. Molchanov, P., Tyree, S., Karras, T., et al. Pruning convolutional neural networks for resource efficient inference. arXiv preprint, 2016. https://doi.org/10.48550/arXiv.1611.06440
38. Molchanov, P., Mallya, A., Tyree, S., et al. Importance estimation for neural network pruning. Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2019: 11264-11272. https://doi.org/10.1109/cvpr.2019.01152
39. Luo, J., Wu, J., Lin, W. Thinet: A filter level pruning method for deep neural network compression. Proceedings of the IEEE international conference on computer vision. 2017: 5058-5066. https://doi.org/10.1109/ICCV.2017.541
40. Mondal, M., Das, B., Roy, S. D., et al. Adaptive CNN filter pruning using global importance metric. Computer Vision and Image Understanding, 2022, 222: 103511. https://doi.org/10.1016/j.cviu.2022.103511
41. Fletcher, P. T., Venkatasubramanian, S., Joshi, S. Robust statistics on Riemannian manifolds via the geometric median. 2008 IEEE Conference on Computer Vision and Pattern Recognition. IEEE, 2008: 1-8. https://doi.org/10.1109/CVPR.2008.4587747
42. Ding, X., Ding, G., Guo, Y., et al. Approximated oracle filter pruning for destructive cnn width optimization. International Conference on Machine Learning. PMLR, 2019: 1607-1616. https://doi.org/10.48550/arXiv.1905.04748
43. Lin, S., Ji, R., Yan, C., et al. Towards optimal structured cnn pruning via generative adversarial learning. Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2019: 2790-2799. https://doi.org/10.1109/cvpr.2019.00290
44. Gao, X., Zhao, Y., Dudziak, Ł., et al. Dynamic channel pruning: Feature boosting and suppression. arXiv preprint, 2018. https://doi.org/10.48550/arXiv.1810.05331
45. Wang, Y., Zhang, X., Hu, X., et al. Dynamic network pruning with interpretable layerwise channel selection. Proceedings of the AAAI conference on artificial intelligence. 2020, 34(04): 6299-6306. https://doi.org/10.1609/aaai.v34i04.6098
46. Liu, Z., Mu, H., Zhang, X., et al. Metapruning: Meta learning for automatic neural network channel pruning. Proceedings of the IEEE/CVF international conference on computer vision. 2019: 3296-3305. https://doi.org/10.1109/iccv.2019.00339
47. Li, H., Kadav, A., Durdanovic, I., et al. Pruning filters for efficient convnets. arXiv preprint, 2016. https://doi.org/10.48550/arXiv.1608.08710
48. Chen, Y., Wen, X., Zhang, Y., et al. CCPrune: Collaborative channel pruning for learning compact convolutional networks. Neurocomputing, 2021, 451: 35-45. https://doi.org/10.1016/j.neucom.2021.04.063
49. Mondal, M., Das, B., Roy, S. D., et al. Adaptive CNN filter pruning using global importance metric. Computer Vision and Image Understanding, 2022, 222: 103511. https://doi.org/10.1016/j.cviu.2022.103511
50. Tang, Y., Wang, Y., Xu, Y., et al. Manifold regularized dynamic network pruning. Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2021: 5018-5028. https://doi.org/10.1109/cvpr46437.2021.00498
51. Lin, M., Ji, R., Wang, Y., et al. Hrank: Filter pruning using high-rank feature map. Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2020: 1529-1538. https://doi.org/10.1109/cvpr42600.2020.00160
52. Polyak, A., Wolf, L. Channel-level acceleration of deep face representations. IEEE Access, 2015, 3: 2163-2175. https://doi.org/10.1109/access.2015.2494536
53. He, Y., Zhang, X., Sun, J. Channel pruning for accelerating very deep neural networks. Proceedings of the IEEE international conference on computer vision. 2017: 1389-1397. https://doi.org/10.1109/ICCV.2017.155
54. Yu, R., Li, A., Chen, C., et al. NISP: Pruning Networks Using Neuron Importance Score Propagation. IEEE, 2018. https://doi.org/10.1109/CVPR.2018.00958
55. Liu, Z., Li, J., Shen, Z.,et al. Learning Efficient Convolutional Networks through Network Slimming. IEEE, 2017. https://doi.org/10.1109/ICCV.2017.298
56. Huang, Z., Wang, N. Data-driven sparse structure selection for deep neural networks. Proceedings of the European conference on computer vision (ECCV). 2018: 304-320. https://doi.org/10.1007/978-3-030-01270-0_19
57. Zhuang, Z., Tan, M., Zhuang, B., et al. Discrimination-aware channel pruning for deep neural networks. Advances in neural information processing systems, 2018: 31. https://doi.org/10.48550/arXiv.1810.11809
58. Ye, J., Lu, X., Lin, Z., et al. Rethinking the smaller-norm-less-informative assumption in channel pruning of convolution layers. arXiv preprint, 2018. https://doi.org/10.48550/arXiv.1802.00124
59. Ye, Y., You, G., Fwu, J. K., et al. Channel pruning via optimal thresholding. Neural Information Processing: 27th International Conference, ICONIP 2020, Bangkok, Thailand, November 18–22, 2020, Proceedings, Part V 27. Springer International Publishing, 2020: 508-516. https://doi.org/10.1007/978-3-030-63823-8_58
60. Li, Y., Adamczewski, K., Li, W., et al. Revisiting random channel pruning for neural network compression. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2022: 191-201. https://doi.org/10.1109/CVPR52688.2022.00029
61. Yang, C., Liu, H. Channel pruning based on convolutional neural network sensitivity. Neurocomputing, 2022, 507: 97-106. https://doi.org/10.1016/j.neucom.2022.07.051
62. Liu, N., Ma, X., Xu, Z., et al. Autocompress: An automatic dnn structured pruning framework for ultra-high compression rates. Proceedings of the AAAI Conference on Artificial Intelligence. 2020, 34(04): 4876-4883. https://doi.org/10.1609/aaai.v34i04.5924
63. Zhou, Y., Zhang, Y., Liu, H., et al. A bare-metal and asymmetric partitioning approach to client virtualization. IEEE Transactions on Services Computing, 2012, 7(1): 40-53. https://doi.org/10.1109/TSC.2012.32
64. Wang, H., Fu, Y. Trainability preserving neural structured pruning. arXiv preprint arXiv:2207.12534, 2022.
65. Xiong, N., Han, W., Vandenberg, A. Green cloud computing schemes based on networks: a survey. Iet Communications, 2012, 6(18): 3294-3300. https://doi.org/10.1049/iet-com.2011.0293

66. Fang, G., Ma, X., Song, M., et al. Depgraph: Towards any structural pruning. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2023: 16091-16101. https://doi.org/10.1109/cvpr52729.2023.01544
67. Hanson, E., Li, S., Li, H., et al. Cascading structured pruning: enabling high data reuse for sparse DNN accelerators. Proceedings of the 49th Annual International Symposium on Computer Architecture. 2022: 522-535. https://doi.org/10.1145/3470496.3527419
68. Bhalgaonkar, S. A., Munot, M. V., Anuse, A. D. Pruning for compression of visual pattern recognition networks: a survey from deep neural networks perspective. Pattern recognition and data analysis with applications, 2022: 675-687. https://doi.org/10.1007/978-981-19-1520-8_55
69. Choudhary, T., Mishra, V., Goswami, A., et al. A comprehensive survey on model compression and acceleration. Artificial Intelligence Review, 2020, 53: 5113-5155. https://doi.org/10.1007/s10462-020-09816-7
70. Wang, J., Jin, C., Tang, Q., et al. Intelligent ubiquitous network accessibility for wireless-powered MEC in UAV-assisted B5G, IEEE Transactions on Network Science and Engineering, 2020, 8 (4): 2801-2813. https://doi.org/10.1109/TNSE.2020.3029048
71. Zhang, W., Zhu, S., Tang, J., et al. A novel trust management scheme based on Dempster–Shafer evidence theory for malicious nodes detection in wireless sensor networks, The Journal of Supercomputing, 2018, 74 (4): 1779-1801. https://doi.org/10.1007/s11227-017-2150-3
72. Simonyan, K., Zisserman, A. Very deep convolutional networks for large-scale image recognition. arXiv preprint, 2014. https://doi.org/10.48550/arXiv.1409.1556
73. Huang, G., Liu, Z., Pleiss, G., et al. Convolutional networks with dense connectivity. IEEE transactions on pattern analysis and machine intelligence, 2019, 44(12): 8704-8716. https://doi.org/10.1109/TPAMI.2019.2918284
74. Han, D., Kim, J., Kim, J. Deep pyramidal residual networks. Proceedings of the IEEE conference on computer vision and pattern recognition. 2017: 5927-5935. https://doi.org/10.1109/cvpr.2017.668
75. Iandola, F. N., Han, S., Moskewicz, M. W., et al. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size. arXiv preprint, 2016. https://doi.org/10.48550/arXiv.1602.07360
76. Krizhevsky, A., Sutskever, I., Hinton, G. E. Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems, 2012: 25. https://doi.org/10.1145/3065386
77. Gholami, A., Kwon, K., Wu, B., et al. Squeezenext: Hardware-aware neural network design. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. 2018: 1638-1647. https://doi.org/10.1109/CVPRW.2018.00215
78. MS, M., SS, S. R. Optimal Squeeze Net with Deep Neural Network-Based Arial Image Classification Model in Unmanned Aerial Vehicles. Traitement du Signal, 2022, 39(1): 275-281. https://doi.org/10.18280/ts.390128
79. Pierezan, J., Coelho, L. D. S. Coyote optimization algorithm: a new metaheuristic for global optimization problems. 2018 IEEE congress on evolutionary computation (CEC). IEEE, 2018: 1-8. https://doi.org/10.1109/CEC.2018.8477769
80. Bernardo, L. S., Damaševičius, R., Ling, S., et al. Modified squeezenet architecture for parkinson's disease detection based on keypress data. Biomedicines, 2022, 10(11): 2746. https://doi.org/10.3390/biomedicines10112746
81. Nirmalapriya, G., Maram, B., Lakshmanan, R., et al. ASCA-squeeze net: Aquila sine cosine algorithm enabled hybrid deep learning networks for digital image forgery detection. Computers \& Security, 2023, 128: 103155. https://doi.org/10.1016/j.cose.2023.103155
82. Han, K., Wang, Y., Tian, Q., et al. Ghostnet: More features from cheap operations. Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2020: 1580-1589. https://doi.org/10.1109/CVPR42600.2020.00165
83. Howard, A., Sandler, M., Chu, G., et al. Searching for mobilenetv3. Proceedings of the IEEE/CVF international conference on computer vision. 2019: 1314-1324. https://doi.org/10.1109/ICCV.2019.00140
84. Yuan, X., Li, D., Sun, P., et al. Real-Time Counting and Height Measurement of Nursery Seedlings Based on Ghostnet–YoloV4 Network and Binocular Vision Technology. Forests. 2022, 13(9):1459. https://doi.org/10.3390/f13091459
85. Chi, J., Guo, S., Zhang, H., et al. L-GhostNet: Extract Better Quality Features. IEEE Access, 2023, 11: 2361-2374. https://doi.org/10.1109/access.2023.3234108
86. Ke, X., Hou, W., Meng, L. Research on Pet Recognition Algorithm With Dual Attention GhostNet-SSD and Edge Devices. IEEE Access, 2022, 10: 131469-131480. https://doi.org/10.1109/ACCESS.2022.3228808
87. Wang, X., Kan, M., Shan, S., et al. Fully learnable group convolution for acceleration of deep neural networks. Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2019: 9049-9058. https://doi.org/10.1109/CVPR.2019.00926
88. Cohen, T., Welling, M. Group equivariant convolutional networks. International conference on machine learning. PMLR, 2016: 2990-2999. https://doi.org/10.48550/arXiv.1602.07576
89. Zhang, J., Zhao, H., Yao, A., et al. Efficient semantic scene completion network with spatial group convolution. Proceedings of the European Conference on Computer Vision (ECCV). 2018: 733-749. https://doi.org/10.1007/978-3-030-01258-8_45
90. Zhang, X., Zhou, X., Lin, M., et al. Shufflenet: An extremely efficient convolutional neural network for mobile devices. Proceedings of the IEEE conference on computer vision and pattern recognition. 2018: 6848-6856. https://doi.org/10.1109/CVPR.2018.00716
91. Ma, N., Zhang, X., Zheng, H., et al. Shufflenet v2: Practical guidelines for efficient cnn architecture design. Proceedings of the European conference on computer vision (ECCV). 2018: 116-131. https://doi.org/10.1007/978-3-030-01264-9_8
92. Vu, D. Q., Le, N. T., Wang, J. (2+ 1) D Distilled ShuffleNet: A Lightweight Unsupervised Distillation Network for Human Action Recognition. 2022 26th International Conference on Pattern Recognition (ICPR). IEEE, 2022: 3197-3203. https://doi.org/10.1109/icpr56361.2022.9956634

93. Chen, Z., Yang, J., Chen, L., et al. Garbage classification system based on improved ShuffleNet v2. Resources, Conservation and Recycling, 2022, 178: 106090. https://doi.org/10.1016/j.resconrec.2021.106090
94. Wang, Y., Xu, X., Wang, Z., et al. ShuffleNet-Triplet: A lightweight RE-identification network for dairy cows in natural scenes. Computers and Electronics in Agriculture, 2023, 205: 107632. https://doi.org/10.2139/ssrn.4227546
95. Howard, A. G., Zhu, M., Chen, B., et al. Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint, 2017. https://doi.org/10.48550/arXiv.1704.04861
96. Sandler, M., Howard, A., Zhu, M., et al. Mobilenetv2: Inverted residuals and linear bottlenecks. Proceedings of the IEEE conference on computer vision and pattern recognition. 2018: 4510-4520. https://doi.org/10.1109/CVPR.2018.00474
97. Chen, Y., Dai, X., Chen, D., et al. Mobile-former: Bridging mobilenet and transformer. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2022: 5270-5279. https://doi.org/10.1109/CVPR52688.2022.00520
98. Nan, Y., Ju, J., Hua, Q., et al. A-MobileNet: An approach of facial expression recognition. Alexandria Engineering Journal, 2022, 61(6): 4435-4444. https://doi.org/10.1016/j.aej.2021.09.066
99. Huang, J., Mei, L., Long, M., et al. Bm-net: Cnn-based mobilenet-v3 and bilinear structure for breast cancer detection in whole slide images. Bioengineering, 2022, 9(6): 261. https://doi.org/10.3390/bioengineering9060261
100. Zhang, K., Cheng, K., Li, J., et al. A channel pruning algorithm based on depth-wise separable convolution unit. IEEE Access, 2019, 7: 173294-173309. https://doi.org/10.1109/ACCESS.2019.2956976
101. Shen, Y., Fang, Z., Gao, Y., et al., Coronary arteries segmentation based on 3D FCN with attention gate and level set function, Ieee Access , 2019,7: 42826-42835. https://doi.org/10.1109/ACCESS.2019.2908039
102. Hung, K. W., Zhang, Z., Jiang, J. Real-time image super-resolution using recursive depthwise separable convolution network. IEEE Access, 2019, 7: 99804-99816. https://doi.org/10.1109/ACCESS.2019.2929223
103. Wang, G., Ding, H., Li, B., et al. Trident-YOLO: Improving the precision and speed of mobile device object detection. IET Image Processing, 2022, 16(1): 145-157. https://doi.org/10.1049/ipr2.12340
104. Wan, R., Xiong, N., Hu, Q., et al. Similarity-aware data aggregation using fuzzy c-means approach for wireless sensor networks, EURASIP Journal on Wireless Communications and Networking, 2019: 1-11. https://doi.org/10.1186/s13638-019-1374-8
105. Yang, S., Xing, Z., Wang, H., et al. Maize-YOLO: a new high-precision and real-time method for maize pest detection. Insects, 2023, 14(3): 278. https://doi.org/10.3390/insects14030278
106. Tan, M., Chen, B., Pang, R., et al. Mnasnet: Platform-aware neural architecture search for mobile. Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2019: 2820-2828. https://doi.org/10.48550/arXiv.1807.11626
107. Huang, G., Liu, S., Maaten, L. V., et al. Condensenet: An efficient densenet using learned group convolutions. Proceedings of the IEEE conference on computer vision and pattern recognition. 2018: 2752-2761. https://doi.org/10.1109/CVPR.2018.00291
108. Mehta, S., Rastegari, M., Caspi, A., et al. Espnet: Efficient spatial pyramid of dilated convolutions for semantic segmentation. Proceedings of the european conference on computer vision (ECCV). 2018: 552-568. https://doi.org/10.1007/978-3-030-01249-6_34
109. Mehta, S., Rastegari, M., Shapiro, L., et al. Espnetv2: A light-weight, power efficient, and general purpose convolutional neural network. Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2019: 9190-9200. https://doi.org/10.1109/CVPR.2019.00941
110. Gao, H., Wang, Z., Ji, S. Channelnets: Compact and efficient convolutional neural networks via channel-wise convolutions. Advances in neural information processing systems, 2018, 31. https://doi.org/10.1109/TPAMI.2020.2975796
111. Wang, R., Li, X., Ling, C. Pelee: A real-time object detection system on mobile devices. Advances in neural information processing systems, 2018, 31. https://doi.org/10.48550/arXiv.1804.06882
112. Zhang, T., Qi, G., Xiao, B., et al. Interleaved group convolutions. Proceedings of the IEEE international conference on computer vision. 2017: 4373-4382. https://doi.org/10.1109/ICCV.2017.469
113. Xie, G., Wang, J., Zhang, T., et al. Interleaved structured sparse convolutional neural networks. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018: 8847-8856. https://doi.org/10.48550/arXiv.1804.06202
114. Sun, K., Li, M., Liu, D., et al. Igcv3: Interleaved low-rank group convolutions for efficient deep neural networks. arXiv preprint, 2018. https://doi.org/10.48550/arXiv.1806.00178
115. Wu, B., Dai, X., Zhang, P., et al. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2019: 10734-10742. https://doi.org/10.1109/CVPR.2019.01099
116. Wan, A., Dai, X., Zhang, P., et al. Fbnetv2: Differentiable neural architecture search for spatial and channel dimensions. Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2020: 12965-12974. https://doi.org/10.1109/cvpr42600.2020.01298
117. Dai, X., Wan, A., Zhang, P., et al. Fbnetv3: Joint architecture-recipe search using predictor pretraining. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2021: 16276-16285. https://doi.org/10.1109/cvpr46437.2021.01601
118. Koonce, B. EfficientNet. Convolutional Neural Networks with Swift for Tensorflow: Image Recognition and Dataset Categorization, 2021: 109-123. https://doi.org/10.1007/978-1-4842-6168-2
119. Tan, M., Le, Q. Efficientnetv2: Smaller models and faster training. International conference on machine learning. PMLR, 2021: 10096-10106. https://doi.org/10.48550/arXiv.2104.00298

120. Ma, N., Zhang, X., Huang, J., et al. Weightnet: Revisiting the design space of weight networks. European Conference on Computer Vision. Cham: Springer International Publishing, 2020: 776-792. https://doi.org/10.1007/978-3-030-58555-6_46
121. Li, Y., Chen, Y., Dai, X., et al. Micronet: Improving image recognition with extremely low flops. Proceedings of the IEEE/CVF International conference on computer vision. 2021: 468-477. https://doi.org/10.48550/arXiv.2108.05894
122. Tsivgoulis, M., Papastergiou, T., Megalooikonomou, V. An improved SqueezeNet model for the diagnosis of lung cancer in CT scans. Machine Learning with Applications, 2022, 10: 100399. https://doi.org/10.1016/j.mlwa.2022.100399
123. Mishra, D., Singh, S. K., Singh, R. K. Deep architectures for image compression: a critical review. Signal Processing, 2022, 191: 108346. https://doi.org/10.1016/j.sigpro.2021.108346
124. Wang, Y., Fang, W., Ding, Y., et al. Computation offloading optimization for UAV-assisted mobile edge computing: a deep deterministic policy gradient approach, Wireless Networks, 2021, 27 (4): 2991-3006. https://doi.org/10.1007/s11276-021-02632-z
125. Veza, I., Afzal, A., Mujtaba, M. A., et al. Review of artificial neural networks for gasoline, diesel and homogeneous charge compression ignition engine. Alexandria Engineering Journal, 2022, 61(11): 8363-8391. https://doi.org/10.1016/j.aej.2022.01.072
126. Liu, Z., Sun, M., Zhou, T., et al. Rethinking the value of network pruning. arXiv preprint, 2018. https://doi.org/10.48550/arXiv.1810.05270
127. Wang, W., Chen, M., Zhao, S., et al. Accelerate cnns from three dimensions: A comprehensive pruning framework. International Conference on Machine Learning. PMLR, 2021: 10717-10726. https://doi.org/10.48550/arXiv.2010.04879
128. Zhou, J., Cui, G., Hu, S., et al. Graph neural networks: A review of methods and applications. AI open, 2020, 1: 57-81. https://doi.org/10.1016/j.aiopen.2021.01.001
129. Wu, Z., Pan, S., Chen, F., et al. A comprehensive survey on graph neural networks. IEEE transactions on neural networks and learning systems, 2020, 32(1): 4-24. https://doi.org/10.1109/TNNLS.2020.2978386
130. Scarselli, F., Gori, M., Tsoi, A. C., et al. The graph neural network model. IEEE transactions on neural networks, 2008, 20(1): 61-80. https://doi.org/10.1109/TNN.2008.2005605
131. Han, K., Wang, Y., Chen, H., et al. A survey on vision transformer. IEEE transactions on pattern analysis and machine intelligence, 2022, 45(1): 87-110. https://doi.org/10.1109/TPAMI.2022.3152247
132. Zhou, D., Kang, B., Jin, X., et al. Deepvit: Towards deeper vision transformer. arXiv preprint, 2021. https://doi.org/10.48550/arXiv.2103.11886
133. Khan, S., Naseer, M., Hayat, M., et al. Transformers in vision: A survey. ACM computing surveys (CSUR), 2022, 54(10s): 1-41. https://doi.org/10.48550/arXiv.2101.01169
134. Liang, W., Xie, S., Cai, J., et al. Novel private data access control scheme suitable for mobile edge computing. China Communications, 2021, 18(11): 92-103. https://doi.org/10.23919/jcc.2021.11.007
135. Liang, W., Li, Y., Xie, K., et al. Spatial-temporal aware inductive graph neural network for C-ITS data recovery. IEEE Transactions on Intelligent Transportation Systems, 2023, 24(8): 8431–8442. https://doi.org/10.1109/tits.2022.3156266