Multimedia Tools and Applications (2024) 83:48165–48236
https://doi.org/10.1007/s11042-023-17192-x

A survey of model compression strategies for object detection

Zonglei Lyu[1,2] · Tong Yu[1,2] · Fuxi Pan[1,2] · Yilin Zhang[1,2] · Jia Luo[1,2] · Dan Zhang[1,2] · Yiren Chen[1,2] · Bo Zhang[1,2] · Guangyao Li[1,2]

Received: 18 July 2022 / Revised: 19 May 2023 / Accepted: 18 September 2023 /
Published online: 2 November 2023

Abstract
Deep neural networks (DNNs) have achieved great success in many object detection tasks. However, such DNNS-based large object detection models are generally computationally expensive and memory intensive. It is difficult to deploy them to devices with low memory resources or scenarios with high real-time requirements, which greatly limits their application and promotion. In recent years, many researchers have focused on compressing large object detection models without significantly degrading their performance, and have made great progress. Therefore, this paper presents a survey of object detection model compression techniques in recent years. Firstly, these compression techniques were divided into six categories: network pruning, lightweight network design, neural architecture search (NAS), low-rank decomposition, network quantization, and Knowledge distillation (KD) methods. For each category, we select some representative state-of-the-art methods and compare and analyze their performance on public datasets. After that, we discuss the application scenarios and future directions of model compression techniques. Finally, this paper is further concluded by analyzing the advantages and disadvantages of six types of model compression techniques.

Keywords Deep learning · Deep neural networks · Object detection · Model compression

1 Introduction

In recent years, deep learning, as a branch of machine learning, has achieved excellent performance in several fields, obviously including the popular computer vision. Object detection, the most fundamental and important part of computer vision, has received extensive attention for several years. The purpose of object detection is to recognize and identify the categories and locations of objects that appear in an image. Due to the rapid development of deep learning techniques and GPU computing power, which has attracted an increasing interest in object detection, it has brought significant breakthroughs. Currently, most state-of-the-art models exploit deep learning-based networks as their backbones to extract features. To obtain better performance, neural networks

Extended author information available on the last page of the article

Table 1 The current DNNs

Method	Parameter			Computation		
	Size (M)	Conv (%)	Fc (%)	FLOPs (G)	Conv (%)	Fc (%)
AlexNet [274]	61	3.8	96.2	0.72	91.9	8.1
VGG-S [276]	103	6.3	93.7	2.6	96.3	3.7
VGG-16 [276]	138	10.6	89.4	**15.5**	99.2	0.8
GoogLeNet [4]	6.9	85.1	14.9	1.6	99.9	0.1
ResNet-18 [112]	5.6	100	0	1.8	100	0
ResNet-50 [112]	12.2	100	0	3.8	100	0

Boldface numbers indicate the methods with the highest floating-point operations (FLOPs)

are incorporated into larger and more complex structures. As shown in Table 1, the current DNNs mainly include AlexNet [274], Visual Geometry Group (VGG) [276], the residual network (ResNet) [112], GoogLeNet [4], etc. However, these DNN models suffer from massive computing and storage requirements. For example, VGG-16 has up to 138 million parameters and requires more than 500 MB of storage space. To detect an image with a resolution of 224×224, the model requires more than 15.5 billion floating point operations per second (FLOPs) and 93 MB of additional runtime memory to store intermediate outputs.

However, due to the limitation of practical application scenarios, models often need to be deployed on embedded development platforms with weak computing power, such as autonomous driving and intelligent patrol cars. Although deploying these models in the cloud can provide high computing performance and storage availability, the high network latency of the cloud will cause certain security risks in practical application scenarios. At the same time, with the continuous development of the Internet of things, edge computing, and autonomous robots [5–8], many practical computer vision applications not only require extremely high detection accuracy, but also put forward higher requirements for real-time detection capabilities. In summary, although complex models have better performance, they require higher storage space and computing resources, which is the reason why DNNS-based object detection models are difficult to be directly and effectively deployed on resource-limited hardware platforms. Specifically, in practical application scenarios, the limitations of current complex object detection models mainly come from three aspects:1) model size, 2) memory at runtime, and 3) number of computations. Therefore, how to make a DNN model achieve efficient real-time detection in application scenarios and hardware with limited computing resources is becoming increasingly important. Since 2014, many scholars have conducted a lot of research on model compression strategies to break through these limitations.

This survey provides a comprehensive review of deep learning-based object detection models and model compression techniques. The Brazilian Laboratory for Theory and Applications of Machine Learning provides an important reference on how to choose the right model compression method. The standard experimental benchmarks for different model compression methods for object detection tasks was provided, performing pruning, KD, and NAS on You Only Look Once (YOLO) v3 and comparing model performance [9]. In addition to that, most of the reviews in related fields review the object detection model and model compression technology separately, and there is a lack of comprehensive reviews for both. For example, Yu Cheng et al. [10] conducted a review

of DNN model compression and acceleration technologies in early 2020, and deeply analyzed the performance, related applications, advantages and disadvantages of model compression technologies. However, only four types of model compression technologies were focused on, and due to the rapid development of model compression field, many latest research results were missing. The few surveys that mention both object detection and model compression [11] focus on the acceleration of "detection pipeline", "detector backbone" and "numerical computation", and only mention network pruning and quantization, and lightweight network design. Therefore, the papers related to object detection and model compression were collected from the top conferences and journals contained in class A and part of class B of computer vision recommended by China Computer Federation (CCF) and Association for Computing Machinery (ACM), and a total of 252 papers were obtained after removing useless documents.

In this paper, we sort out the research status in this field according to time and technology type, and provide a comprehensive review of deep learning-based object detection models and model compression techniques. In Section 2, six available compression strategies and the research status of related teams and institutions in this field are discussed. Section 3 compares and summarizes the compression effects of various compression strategies on public datasets, which also contains the performance comparison between the original object detection model and the lightweight model. The future research directions and applicable scenarios of various model compression strategies are listed in Section 4. Finally, by analyzing the advantages and existing problems of each strategy, the paper is further concluded in Section 5.

The main contributions of this paper are as follows:

(1) This paper systematically reviews six categories of compression techniques, including network pruning, lightweight network design, NAS, low-rank decomposition, network quantization, and KD methods. A total of 252 papers were collected from 247 top-tier conferences and journals, encompassing the latest works that were absent from previous surveys.
(2) Unlike existing surveys that primarily focus on either object detection models or model compression techniques, this paper specifically concentrates on both object detection and image classification. This targeted approach provides comprehensive data statistics and summaries for all methods on public datasets. For instance, in Fig. 25, we compare the performance of original object detection models with lightweight models on the VOC 2007 dataset, which has not been covered in other surveys.
(3) In addition to a simple summary based on strategy classification, this paper goes beyond and provides an additional analysis of the research landscape of compression techniques from the perspectives of research institutions, timeline, and publishing organizations. Figures 6, 7, 9, among others, present this comprehensive view, assisting readers in gaining a more thorough and efficient understanding of the research progress in this field.

2 Model compression strategies

Since 2014, methods for compressing and accelerating DNN models have gradually emerged. From 2014 to 2021, after eight years of research on DNN model compression methods, the field can be divided into the following six research directions: lightweight

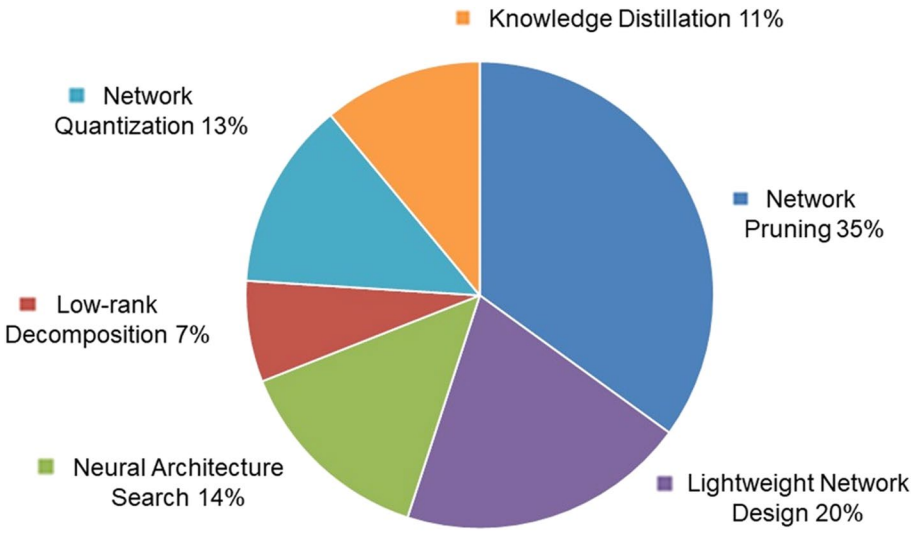


Fig. 1 Distribution of the research directions concerning model compression

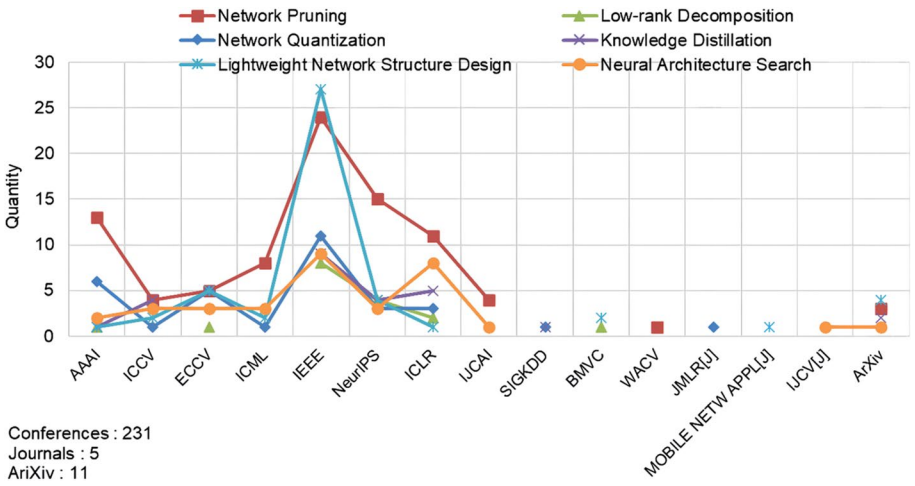


Fig. 2 The distribution of top conferences and journals papers in model compression research

network structure design, NAS, low-rank decomposition, network quantization and KD, and some methods that are combinations of each other. In this paper, a total of 252 papers are collected from the top conferences and journals contained in class A and part of class B of computer vision, as recommended by the CCF and ACM. The proportions of the research directions for model compression methods are shown in Fig. 1. It can be seen that network pruning and lightweight network structure design methods account for the largest proportions among the current model compression methods. In addition, the distribution of the top conference and journal papers in model compression research is shown in Fig. 2. Among them, 236 conference papers, 5 journal papers and 11 arXiv papers are included.

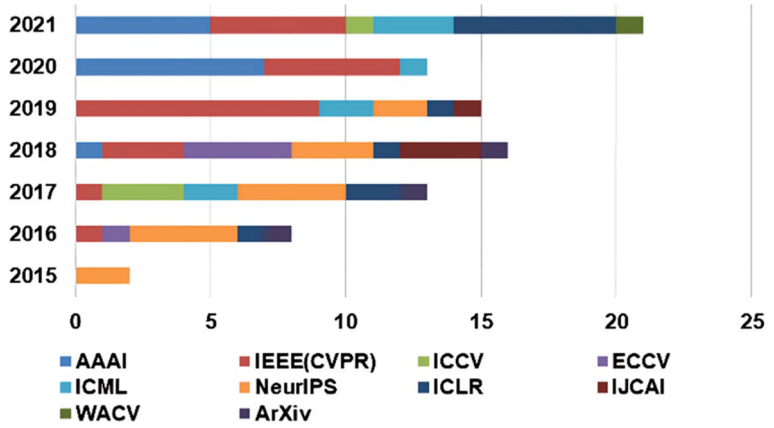
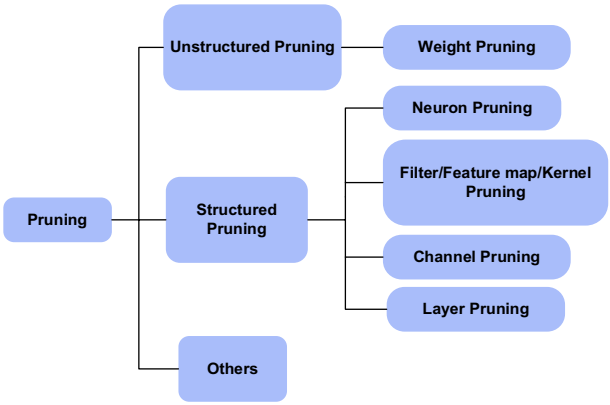


Fig. 3 Distribution of Network Pruning papers

Fig. 4 Classification of Network Pruning approaches

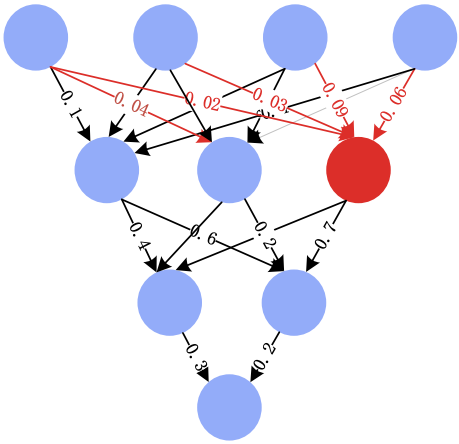


Research on model compression methods in the field of object detection has been a very important research hotspot at top conferences in recent years. In contrast, top journals have produced less research in this field.

2.1 Network pruning

The network pruning approach is one of the most extensive model compression methods in DNNs. Its main idea is to remove redundant parameters by designing different standards for judging whether parameters are important to make the network sparse and achieve compression. In fact, the history of model pruning can be traced back to 1989, and it was first proposed in [12]. Since then, many network pruning methods have been developed. The collection of the network pruning compression methods provided in top journals and conferences in the last 6 years is shown in Fig. 3. This paper counts 90 papers on network pruning, and we can find that network pruning-based compression methods have recently become increasingly popular.

Fig. 5 Weight pruning



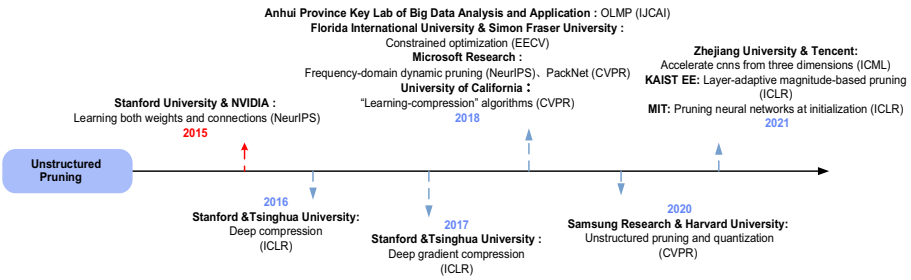


Fig. 6 The research situations of various research institutions and teams on regarding Unstructured Pruning

According to their different pruning approaches, network pruning methods can be divided into structured pruning and unstructured pruning. As shown in Fig. 4, unstructured pruning mainly includes unstructured weight pruning. The unstructured weight pruning can directly remove unimportant weights from each layer to reduce the number of model parameters. Structured pruning is implemented by deleting the unimportant structures contained in the whole DNN, these structures include convolution kernels, channels, filters, layers, and so on.

2.1.1 Unstructured pruning

The principle of unstructured weight pruning is shown in Fig. 5. The unimportant weight edges of the given network are pruned while minimally affecting the overall structure. As shown in Fig. 6, since 2015, various research institutions have achieved some excellent research results regarding unstructured pruning methods. Their achievements are introduced in turn below.

The Han Song team of Stanford University has produced many achievements in the field of model compression. In 2015, the Han Song team and NVIDIA team [13] jointly first proposed a method of pruning weights whose absolute values are less than a given threshold, and they then trained a sparse network to learn the remaining connection weight. Their model achieved a high sparsity rate at the weight level. Based on this

method, the team cooperated with Tsinghua University [14] to propose a three-stage pipeline-based deep compression method, which combines pruning, training quantization and Huffman coding to reduce the storage requirements of y the examined neural network while maintaining the original accuracy. This paper won the best paper award at the International Conference on Learning Representations (ICLR) 2016. To reduce the communication overhead, Lin et al. [15] proposed a deep gradient compression (DGC) method, which only selects a part of the most "important" gradient elements in each iteration for sending.

The method of pruning weights by providing thresholds for each layer is called layer-wise magnitude-based pruning (LMP). Many subsequent studies have been conducted based on this approach. In the past, the commonly used threshold adjustment method was manual operation [14]. Such strategies require much professional and expert knowledge. To facilitate the use of LMP by nonprofessional users, the Anhui Province Key Lab of Big Data Analysis and Application [16] proposed an approach called optimization based LMP (OLMP) to automatically adjust the pruning threshold of LMP. Consequently, the threshold adjustment problem was transformed into a constrained optimization problem, minimizing the size of the pruned model subject to an accuracy loss constraint. Similarly, other scholars have also transformed the compression problem into a constrained optimization problem [17–19]. In 2021, Lee et al. [20] proposed an importance score for global pruning by studying layered sparsity, namely, the layer-adaptive magnitude-based pruning (LAMP) score, and their approach did not require any hyperparameter tuning or heavy computation.

Many scholars are still exploring other methods of judging the importance of model weights instead of using their magnitude as the importance evaluation metric. The teams at Peking University and Microsoft Research [21] proposed a dynamic pruning scheme in the frequency domain by using spatial correlation. Carreira-Perpinán et al. of the University of California [22] proposed an algorithm called learning compression. They used the quadratic regularization term of a training model to periodically update the weight set and a Lagrange multiplier to find the most appropriate weight, not just the largest weight. Mallya et al. of Microsoft Research [23] proposed an algorithm called Packnet. They packaged "multiple tasks" into a single network by performing iterative pruning and network retraining to solve the catastrophic forgetting problem in weight pruning. The Samsung Research and Harvard University teams [24] proposed a sparse quantization neural network method for weight representation based on fine-grained and unstructured pruning. To speed up the training speed and prevent the loss of accuracy caused by the fine-tuning or modification of the network architecture during or after training, Frankle's team [25] at Massachusetts Institute of Technology (MIT) proposed random initialization pruning methods, which are different from magnitude-based pruning after training, randomly shuffling the pruned weights within each layer or sampling new initial values to preserve or improve accuracy.

Through the above discussion, we know that the research on unstructured pruning methods performed by various scholars and research institutions has focused on how to choose the weights to be pruned. Threshold pruning methods ranging from LMP [14] to dynamic weight selection via spatial correlation [21], and the recent random initialization-based pruning method [25] (performed to further sparsity the network) have achieved good results. On the whole, unstructured pruning, regardless of its structural position, has less impact on the overall model accuracy and can achieve a high sparsity rate. However, we should also pay attention to the irregular memory access caused by unstructured pruning. In addition, for unstructured pruning, the research of MIT Han Song team has provided many contributions.

2.1.2 Structured pruning

The concept of structural pruning was proposed based on the limitations of unstructured pruning. Structural pruning is realized by deleting the unimportant structures in a whole DNN and minimizing the reconstruction error. According to the granularity of structural pruning, structural pruning methods can be divided into neuron pruning, filter/feature map/kernel pruning, channel pruning and layer pruning. As shown in Fig. 7, from 2016 to the present, major research institutions and teams have put forward a lot of research results each year, among which the filter pruning method accounts for the largest proportion. This section introduces various structural pruning methods from the perspective of research institutions and teams according to their pruning granularity categories and the overall timeline.

(1) Neuron Pruning After a neural network receives an input, not all neurons in the network are activated, so the output value of those deactivated neurons can be set to zero through a certain criterion, which can reduce structural redundancy. Neuron pruning is relevant to weight pruning. Weight pruning focuses on the pruning of weight edges, while neuron pruning focuses on the pruning of the neurons pointed out by the weight edges. A large number of studies on neuron pruning have appeared since 2016.

Hong Kong University of Science and Technology and the SenseTime Technology team [26] proposed iterative alternate pruning and training, and the pruning of neurons with zero activation values. Guo et al. [27] proposed a dynamic remedy method to improve the threshold pruning technique, this approach considers that by dynamically changing the importance levels of the connections between neurons: the connections will be deleted when they are not important, and they will be remedied back when they are important. Improvements of the threshold pruning method are irreversible. The University of Maryland, Deepmind, IBM, Jingdong and other teams [28] jointly proposed applying feature

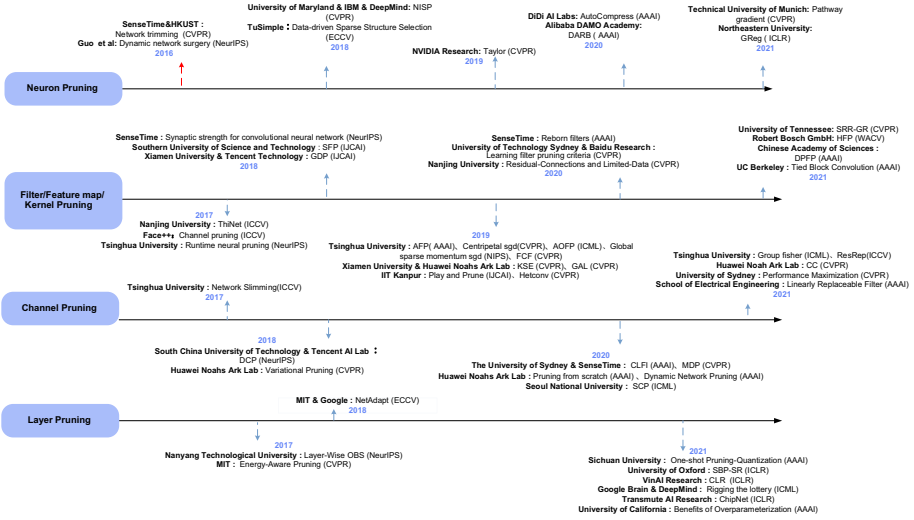


Fig. 7 The research situations of various research institutions and teams regarding Structured Pruning

ranking technology to measure the importance of each neuron in the "final response layer" (FRL) and removing the least important neurons. TuSimple Team [29] introduced a new parameter-scale factor to scale the output of a specific structure to achieve data-driven and end-to-end pruning. The NVIDIA team [30] repeatedly deleted neurons with small scores by estimating the contributions of neurons to the final loss. Khakzar [31] of the Technical University of Munich proposed a pathway selection method based on neuron contributions, and a feature attribution approach, (the "pathway gradient"), that reveals the input features associated with features encoded in the critical pathways. Wang [32] of Northeastern University proposed two algorithms that exploit regularization in a new fashion, where the penalty factor is uniformly raised to a large level, and they finally solved the two central problems regarding the pruning schedule and weight importance criterion of deep neural pruning.

Some scholars are also researching structured weight pruning. Last year, the DiDi AI research team [33] proposed an automatic structured pruning framework called AutoCompress based on the alternating direction method of multipliers (ADMM). Similarly, Ma et al. [34] also used the ADMM algorithm to obtain a sparse structure called partial convolution (PCONV). The Alibaba Dharma Academy team [35] proposed a new pruning method by using block-max weight masking (BMWM) and a density-adaptive regular block (DARB) by studying the inherent characteristics of irregular pruning weights.

The research on neuron pruning mainly focuses on the evaluation of neuron contribution rates and importance levels, and some studies have combined these topics with structural weight pruning. On the whole, few studies have examined neuron pruning.

(2) Filter/Feature Map/Kernel Pruning Convolutional neural networks (CNNs) usually have significant redundancy between different filters and feature maps. A feature map is an output of the network, and a filter is a network parameter. Feature maps and filters have corresponding relationships. When a feature map is subtracted, the filter connected to it is also removed. Some differences can be found between the filter pruning method and the neuron pruning method. The filters pruning method prunes CNN at the filter rather level than in terms of a single connection (parameter) or neuron. Filter pruning is the most important structured pruning method. The following introduces the research situation of this strategy from the perspectives of technology and research purposes.

Methods based on Sparse Constraint Regularization

Filter pruning methods have emerged in large numbers since 2016. To obtain sparse structures, many scholars have studied network parameter regularization methods based on group sparsity to penalize unimportant parameters. The University of Pittsburgh [36] used group least absolute shrinkage and selection operator (LASSO) regression for regularization specification to explore the sparsities of different structure levels, and the works in [37, 38] were similar. Yoon and Hwang [39] also studied group sparse regularization, by using the correlations between the features in the examined network for pruning. A team at the University of Maryland and NEC Labs America [40] proposed a structural sparse CNN regularization and acceleration method, which can greatly reduce the cost of computation by removing all the filters and their connected feature maps in the network that have little impact on the output accuracy. A team at the Skolkovo Institute of Science and Technology [41] used the online balanced descent (OBD) algo-

rithm [12] to regard the convolution operation as a matrix multiplication calculation, and used the group sparsity method to achieve sparse matrix multiplication and improve the calculation speed of their approach. Megvii Technology and Xi'an Jiaotong University [42] proposed a channel selection strategy based on the LASSO and a least-squares reconstruction algorithm to prune filters.

Many methods based on the sparse learning (SL) framework have also been proven to be effective [43–47] of Université de Sherbrooke proposed an SL framework, which allows learned and selected filters to be pruned in a CNN while considering the neuron budget. A University of Amsterdam team [48] used a Bayesian algorithm to construct sparse structures for pruning. However, the regularization based on pruning technique requires a sensitivity analysis for each layer and additional computations. To solve this problem, a team at NVIDIA [49] proposed a global rescaling method that depends on the standards of all layers and does not require sensitivity estimation.

Methods based on adaptive and dynamic pruning

Many scholars have proposed some strategies to achieve adaptive and dynamic filter pruning. In 2017, Lin et al. of Tsinghua University [50] considered that input samples should not be treated equally because of the different computational quantities required by different tasks. Therefore, they proposed a runtime neural pruning (RNP) framework that retains all the capabilities of the original network, and adaptively conducts pruning according to the input image and current feature map. In 2018, the Ding team of Tsinghua University proposed an auto-balanced filter pruning (AFP) method [51], which prunes some weak filters together with the corresponding feature maps. In the second year, they [52] also proposed a new momentum-based stochastic gradient descent (SGD) optimization method to reduce network complexity via dynamic pruning. Ji Rongrong's team of Xiamen University cooperated with Tencent Technology and other institutions [53] to propose a global and dynamic training algorithm to prune non-salient filters. A team at the University of Sydney and Baidu [54] proposed the learning filter pruning criterion (LFPC) by considering cross-layer filter distribution diversity, this approach adaptively selects the appropriate pruning criteria for different functional layers.

In 2021, Wang et al. [55] of the University of Tennessee proposed a layer-adaptive filter pruning approach based on structural redundancy reduction, which builds a graph for each convolutional layer of a CNN to measure the redundancy in each layer. This approach prunes unimportant filters in the most redundant layer(s) rather than the filters with the least importance across all layers. Peking University and Huawei's Noah's Ark Lab used the relationship between different inputs to propose a new paradigm that dynamically removes redundant filters by embedding the manifold information of all instances into the space of pruned networks (dubbed ManiDP) [56]. To overcome the problem that many extra hyperparameters and training epochs are caused by the existing filter pruning methods, Ruan [57] of the Chinese Academy of Sciences proposed a novel dynamic and progressive filter pruning scheme (DPFPS) that directly learns a structured sparsity network from scratch. In particular, the DPFPS specifically imposes a new structured sparsity inducing regularization upon the expected pruning parameters in a dynamically sparse manner.

Methods based on limited data and without retraining or fine-tuning

Considering that the current methods typically need to use the original data for retraining after pruning to compensate for the loss of accuracy, due to the data privacy problem, only a small part of the data may actually be obtained. The SenseTime team [58] proposed a data-driven method that uses a parameter class called synaptic strength to prune the connection between the input and output feature maps, and finally generate a nuclear-level sparse CNN. In 2020, SenseTime and Tsinghua University [59] jointly proposed a method for network compression when the number of training data is limited. Tang et al. of Nanjing University [60] proposed a network pruning method based on residual blocks and limited data to solve the problem of pruning residual connections and small-scale datasets. The Ding team of Tsinghua University [61] also proposed a method to safely remove redundant filters, but it is not ideal for large scale datasets because the model needs to be trained from scratch [62]. To solve this problem, the team further proposed approximated oracle filter pruning (AOFP) [63], which continuously searches for the least important filters in a binary search manner, performs pruning attempts by randomly masking out filters, accumulates the resulting errors, and finetunes the model via a multipath framework.

Most of the previous works involved "hard filter pruning", which means deleting the pruned filters directly. Therefore, this approach reduces the capacity of the original model and leads to performance degradation. In addition, after pruning the filters of the pretraining model, fine-tuning is needed. To solve this problem, Southern University of Science and Technology [64] proposed a soft filter pruning (SFP) method to accelerate the DNN inference process, so that each pruned filter can be updated during the training process of the model with little dependence on the pretraining model. To solve dimensional mismatch and retraining problems, the team cooperated with the Huawei Noah's Ark Lab and other institutions [65] to propose an end-to-end pruning method for heterogeneous redundant structures, and kernel sparsity and entropy (KSE) indicators to guide model compression, without iterative pruning and retraining. Lin et al. [66] also introduced a soft mask to learn a sparse soft mask in a label-free and end-to-end manner, and used a fast iterative shrinkage and thresholding algorithm (FISTA) to quickly and reliably remove the corresponding structures.

Others

Some scholars have studied pruning methods that can directly specify the accuracy of the model after pruning. The Pravendra Singh team at the IIT Kanpur Institute [67] proposed a play and prune (PP) framework consisting of an auto-balanced filter pruning (AFP) [51] module and a pruning rate controller (PRC) module. This method allows for directly specifying the desired error tolerance instead of conducting this step at the pruning level. Enderich [68] of Robert Bosch GmbH proposed the holistic filter pruning (HFP) method, which can specify accurate pruning rates for both numbers of parameters and multiplications, and they used the channelwise scaling factors of batch-normalization (BN) layers to calculate the model size.

Luo Jianhao's team at the National Key Software Technology Laboratory of Nanjing University [69] proposed a Thinet architecture based on filter pruning by using a greedy

algorithm. They considered filter pruning to be an optimization question and pointed out that whether a filter can be pruned depends on the input of the next layer rather than that of the current layer, they finally pruned the filter corresponding to the layer with the weakest input channel data in the next layer. This method has also been one of the main comparative approaches in follow-up research. Tsinghua University and the Tencent AI Lab also used optimization problems for pruning [70], they jointly proposed a factorized convolutional filter (FCF) structure, and updated it with an optimization method based on the ADMM to obtain a compact CNN model.

To solve the global filter importance ranking (GFIR) problem, Peking University's You et al. [71] proposed a global filter pruning algorithm called GateDecorator. He et al. [72] proposed a filter pruning method via the geometric median (FPGM), which compresses CNN models by pruning filters with redundancy, rather than those with "relatively less" importance. A team at the Skolkovo Institute of Science and Technology and Microsoft Research proposed Perforatedcnns [73], which uses different strategies to mask the activation values and employs adjacent valuesto represent the masked values to remove redundancy. The Pravendra Singh team at the IIT Kanpur Institute [74] proposed a new filter structure called hetconv based on heterogeneous kernel to reduce the number of FLOPs without incurring the loss of accuracy. In 2021, Wang et al. [75] of UC Berkeley proposed a tied block convolution (TBC), which shares the same thinning filter over equal channels blocks and produces multiple responses with a single filter, this approach can also be extended to group convolution and fully connected layers, and can be applied to various backbone networks and attention modules.

Through the study of the abovementioned filter, feature map, and kernel pruning method, it can be seen that the main techniques used in the research on filter pruning methods are the sparse constraint regularization method to penalize unimportant parameters, and cross-layer correlation, importance ranking and other mechanisms to evaluate importance and remove redundancy, and finally obtaining a sparse structure. In terms of research purposes, some studies mainly focus on filter pruning without fine-tuning and retraining, and pruning in the case of limited data. Some scholars specialize in adaptive and dynamic pruning. Generally, filter pruning methods are relatively excellent, and the global dynamic pruning may be the future trend.

(3) Channel Pruning Both filter pruning and channel pruning approaches are coarse-grained pruning methods, and their pruning techniques are similar. However, channel pruning can solve the dimensional mismatch problem caused by filter pruning. The main idea of channel pruning is to directly delete entire channels from the convolutional layer of a network. After pruning, the model can directly be trained to obtain higher compression ratio and a shorter computing time.

The team of Tsinghua University [76] proposed applying L1 regularization to channels by pushing the value of the BN scaling factor to zero to identify and delete insignificant convolution channels. In 2021, the team [77] also proposed a layer grouping algorithm to automatically find coupled channels, and derived a unified metric based on Fisher information to evaluate the importance of a single channel and coupled channels. Ding's team [78] at Tsinghua University proposed ResRep, a novel method for lossless channel pruning, which slims down a CNN by reducing the widths (numbers of output channels) of convolutional layers.

Hu et al. [79] proposed a channel pruning method based on genetic algorithm. According to the sensitivity of each channel layer, the network model is pruned layer by layer, and then fine-tuned. South China University of Technology [80] introduced an additional recognition perception loss to help select the channels that are truly helpful for recognition, and jointly optimized the reconstruction error. These methods remove the unimportant input feature mappings of the convolution layer and prevent dimensional mismatch. The team [81] of the Tencent AI Lab proposed a collaborative channel pruning (CCP) method to use the dependency relationship between channels to determine the pruning results to reduce the computational overhead with negligible performance degradation.

In recent years, Huawei Noah's Ark Lab has conducted excellent research on channel pruning methods. Zhao et al. [82] introduced a variational technique to estimate the saliency distribution of the parameter channels, and removed redundant channels from the original model. In 2020, they cooperated with Tsinghua University and Ant Financial [83] to use a simple binary search strategy to determine the channel number configuration of a pruning model, and start training and pruning directly from any initial weight. In the same year, the team [84] also proposed a model that explicitly establishes channel selections with discrete weights to encourage the use of more different weights and improve the interpretability of the model with the help of interpretable hierarchical channel selection in dynamic networks. In 2021, Huawei Noah's Ark Lab worked with Xiamen University [85] to propose a collaborative compression (CC) scheme, which joins channel pruning and tensor decomposition to compress CNN models by simultaneously learning the model sparsity and low-rank status.

Guo et al. [87] of the University of Sydney proposed a new layer-by-layer channel pruning method called channel pruning guided by classification loss and feature importance (CPLI), and a new strategy to suppress the influence of unimportant features, which are removed in the pruning stage. The team [88] also proposed multi-dimensional pruning (MDP), which considers multiple dimensional factors such as space, time, and channels for pruning. In 2021, another team including Gao et al. [89] at the University of Sydney considered the loss metric mismatch problem of pruning and proposed a novel channel pruning method for CNNs. At the same time, they introduced a corresponding episodic memory to update and collect sub-networks during the pruning process to solve the problem of catastrophic forgetting and the imbalanced distribution of subnetworks.

The Seoul University team [86] proposed that a model may converge to a poor local minimum after pruning, and channel pruning depends excessively on the absolute activation value in the corresponding channel. Therefore, they proposed a data-driven channel pruning algorithm using the random gradient descent method, which combines BN and a rectified linear unit (ReLU) for channel pruning without an additional fine-tuning process. Gao et al. [14] of the University of Pittsburgh proposed a discrete optimal structural pruning method to obtain compact CNN with strong resolution. Joo et al. [90] of the School of Electrical Engineering proposed a novel channel pruning method, a linearly replaceable filter (LRF), suggesting that a filter that can be approximated by the linear combination of other filters is replaceable. Moreover, weights compensation was proposed to support the LRF method by effectively reducing the output difference caused by removing filters via direct weight modification.

To reduces the risk of pruning important channels prematurely, another team including Hou et al. [91] propose a novel Channel Exploration methodology (CHEX) in 2022, which repeatedly prune and regrow the channels throughout the training process. They tackle the channel pruning problem via a well known column subset selection (CSS) formulation. The results show that compressed ResNet-50 model on ImageNet dataset achieves 76%

top1 accuracy with only 25% FLOPs of the original ResNet-50 model, outperforming previous state-of-the-art channel pruning methods.

Through the research on channel pruning methods, it can be seen that among these approaches, L1 regularization using the scaling factor of the BN layer, sensitivity and coupling the correlations of the channels in each layer to remove redundancy are most popular. In addition, the Huawei Noah's Ark Lab has performed more research on channel pruning methods in recent years, and they have focused on combining other methods such as search algorithms and tensor decomposition methods, with channel pruning methods to achieve further pruning effects. The proposed linear combination of other filters to achieve compression is a relatively novel filter pruning method in 2021. Most recently, a team of National Yang Ming Chiao Tung University proposes an accurate and efficient object detector capable of performing real-time inference under the hardware constraints of an edge device by leveraging structural pruning, feature distillation, and NAS [92]. Generally, the channel pruning methods have good compression effect and do not produce a sparse matrix, so special software is not required.

(4) Layer Pruning Layer pruning method is one of the deepest pruning methods, which directly prunes a layer in the network. The degree of compression of layer pruning is great, but the accuracy is relatively lower.

In recent years, some studies have focused on layer pruning methods. Dong et al. [93] of Nanyang Technology University proposed a new layer-wise pruning method for DNNs. The parameters of each individual layer are pruned independently based on the second order derivatives of a layer-wise error function with respect to the corresponding parameters, and the original prediction performance is restored through the reconstruction error and a retraining process. Yang et al. [94] in MIT used the method of energy perception to calculate the energy consumption of each layer, preferentially pruning the layers with high energy consumption. In addition, to prevent incorrect pruning, the weight that resulted in the greatest decrease in accuracy after pruning was retained. Subsequently, the team cooperated with Google [95] to propose Netadapt, which also takes hardware metrics (delay, energy consumption, etc.) as pruning evaluation criteria, uses empirical metrics for evaluation, and automatically iteratively prunes the pretraining network on a mobile platform until the imposed resource budget is met.

In the application of object detection, the Guangdong Key Laboratory of Modern Control Technology uses channel pruning and layer pruning to compress the YOLOv3-SPP garbage detection model, and the compressed model is directly deployed on the low-computing edge computing device carried by the robot to automatically classify and recycle garbage[279].

2.1.3 Other types of approaches

In 2021, some scholars conducted research on training perspectives, Google Brain and DeepMind [96] proposed a method to train sparse neural networks with a fixed parameter count and a fixed computational cost throughout training. This approach updates the topology of a sparse network during training by using parameter magnitudes and infrequent gradient calculations, this approach requires fewer FLOPs to achieve a given level of accuracy. Hayou et al. [97] of University of Oxford provided a comprehensive theoretical analysis of magnitude and gradient-based pruning for the initialization and training of

sparse architectures. Then, they introduced a simple rescaling trick to bring the pruned model back into the edge of chaos (EOC) regime, making the pruned neural network easily trainable.

Through a large number of experiments, Le et al. [98] of VinAI Research studied random pruning and learning rate restarting, and proposed that compared to traditional fine-tuning, in general, learning rate restarting is an efficient way to retrain pruned networks to recover performance drops caused by pruning. Liu et al. [99] investigates the effectiveness of random pruning for sparse training. Specifically, the paper adopted several random pruning methods(ERK, Uniform, Uniform+) and compare the results with non-random Pruning methods(SNIP,GraSP). They are conducted on different ResNet scaling models, with CIFAR-10/100 and ImageNet dataset under different sparsity ratios. Results show that random pruning can be quite effective for sparse training especially on larger or wider models. Hu [100] of Sichuan University proposed a novel one-shot pruning quantization (OPQ) approach, which analytically solves the compression allocation problem with only pre-trained weight parameters. During fine-tuning, the compression module is fixed and only the weight parameters are updated, without any complex iterative/manual optimization.

Tiwari et al. [101] presented ChipNet, a deterministic pruning strategy that employs a continuous Heaviside function and a novel crispness loss to identify a highly sparse network based on an existing dense network. For problems that involve training a small model from the beginning or training a large model first and then pruning, Chang et al. [102] of the University of California conducted systematic research and analysis, and finally comes to the conclusion that the effect of pruning using over-parameterized model is better.

Discussion According to whether the network structure is changed after pruning, network pruning can be divided into structured pruning and unstructured pruning. Unstructured pruning (weight pruning) does not change the original structure of the network, but the resulting sparse network needs to be supported by special hardware libraries or computing libraries. In contrast, structured pruning directly removes part of the network structure and is more suitable for general hardware acceleration. According to the granularity of pruning, structure pruning can be further divided into neuron pruning, kernel pruning, channel pruning and layer pruning. Neuron pruning is based on a single neuron, which is suitable for any type of layer, but the pruning effect is poor. Kernel pruning is based on convolution kernels and is usually applied to convolutional layers; Channel pruning is an extension of kernel pruning, which prunes multiple channels in a convolution kernel. Layer pruning is one of the most extreme structured pruning techniques applicable to some deeper neural networks where some layers can be considered useless or redundant.

2.2 Lightweight network design

Methods that directly compress existing network models can effectively construct lightweight network models (such approaches include network pruning, network quantization, low-rank decomposition and KD methods), but their performance largely depends on the utilized pretrained model. Without improving the network architecture, the accuracy cannot be further improved, so lightweight network design methods have emerged. The main idea of lightweight network design is to reduce the number of network structure parameters by directly designing a more compact convolution kernel or convolution method. As illustrated in Fig. 8, research on lightweight network structure design methods has been booming since 2015. The number of lightweight network design papers reached its peak starting

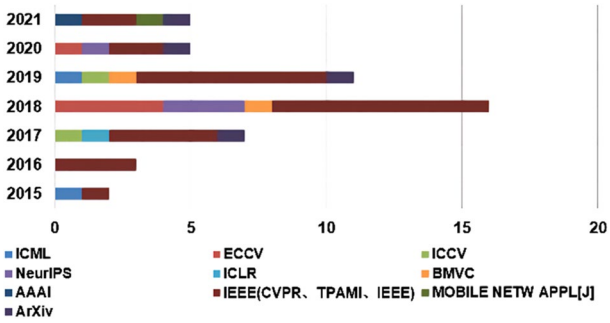


Fig. 8 Distribution of Lightweight Network Design papers

in 2017. As illustrated in Fig. 9, Google, Facebook, Microsoft Research, Megvii and Huawei have proposed many lightweight structure models with strong performance in the area of lightweight network design. These lightweight structure models are often proposed and improved in the form of series. In this paper, a total of 49 outstanding lightweight network designs are counted, and all of them have been published in top conferences and journals. The available lightweight network design methods are divided into model design method based on convolution structure and convolution operation. In addition, due to the excellent performance of the YOLO models in the field of object detection, many researchers have designed lightweight YOLO structures according to its unique structure, which are specifically introduced in 2.2.3.

2.2.1 Lightweight model design based on convolution structure

A common method is to construct a lightweight network to transform a convolution structure to reduce the number of network computations. In recent years, lightweight model design methods based on convolution structures have mainly converted traditional convolution operations, such as standard convolution, group convolution, pointwise convolution, dilated convolution and depth-wise separable convolution. These convolution structures are shown in Fig. 10. This section introduces the research status of lightweight models from the perspective of some different convolution structures and other methods.

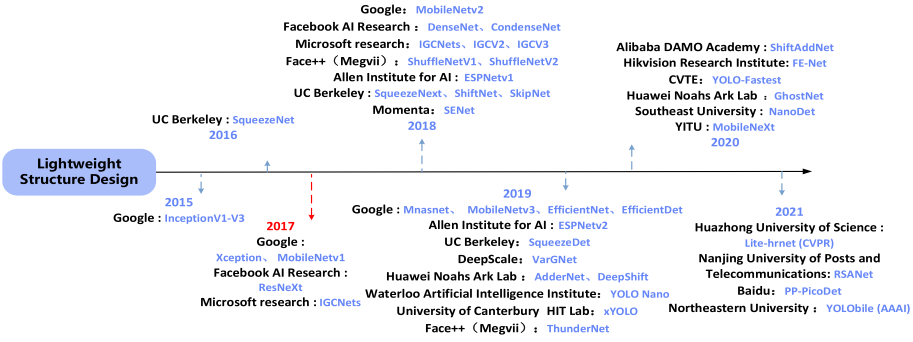


Fig. 9 The research situations of various research institutions and teams on Lightweight Network Design

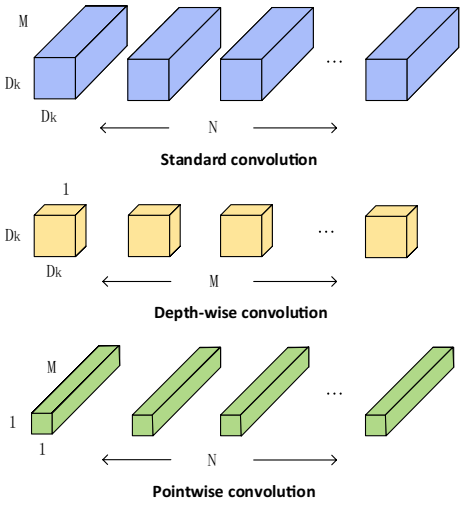


Fig. 10 Convolution operations

(1) Depth-wise Separable Convolution To reduce the number of model parameters, Google has made some excellent achievements in recent years by using the depth-wise separable convolution structure. Google first proposed the InceptionV1-V2 [103, 104] models in 2015. In the second year, InceptionV3 [105] used asymmetric convolution in a deeper position of the network, and the model parameters were reduced without affecting efficiency. To obtain a more lightweight network, Google improved InceptionV3 and proposed Xception [106] in 2017, which uses depth-wise separable convolution to replace the multidimensional convolution kernel feature response operation in the original InceptionV3, achieving better results.

From 2017 to 2018, Google's team successively proposed MobileNetV1-V2 [107, 108] lightweight network models. They proposed a lightweight network architecture MobileNetV1 by using depth-wise separable convolution instead of standard convolution. To improve the performance of MobileNetV1 so that it could perform better on mobile devices, MobileNetV2 was proposed by adding residual structure and a linear bottleneck structure on the basis of MobileNetV1. MobileNetV2 amplifies the given input from a low-dimensional feature map to a high-dimensional feature map, then uses the depth-wise convolution method to perform convolution operations, and finally uses a linear convolution to map the input to the low-dimensional space, which greatly reduces the number of convolution operation. In 2019, Google's team proposed a new mixed depth-wise convolution (MDconv), which mixes different kernel sizes in a convolution operation, and proposed a series of networks called MixNets [110] based on the search space of AutoML.

The team of Allen Institute for AI proposed ESPNetV2 [111] by adding depth-wise dilated separable convolutions to modify the ESPNetV1 model structure. ESPNetV2 achieved better accuracy and lower parameter quantity in image classification, semantic segmentation and object detection.

(2) Group Convolution Facebook AI Research combined ResNet [112] and Inception to propose ResNeXt [113] by using group convolution. Later, they cooperated with Cornell University and Tsinghua University to propose DenseNet [114], which achieved

better results and required fewer parameters due to the extreme use of feature maps. Since DenseNet's internal connections still exhibit considerable redundancy, the team and Tsinghua University research scholars cooperated to improve DenseNet and propose CondenseNet [115], which can reduce the video memory consumption and improve the speed of the model through using a learnable grouped convolution operation during training and pruning.

The Microsoft Research Institute proposed the interleaved group convolution network (IGCNets) [116] which was implemented by interleaved group convolutions in 2017. Interleaved group convolutions make the network wider, while the network parameters and computational complexity remain unchanged. In the second year, the team proposed IGCV2 [117] on the basis of IGCV1, and performed another interleaved group convolution for the second group convolution in the block. In the same year, the Microsoft Research team proposed IGCV3 based on the ideas of IGCV2 and a bottleneck [118] by combining low-rank convolution kernels and sparse convolution kernels to form dense convolution kernels, expanded the dimensionality of the input grouping features, reduced the dimensionality of output, and finally achieved compression. The core of the IGC series of networks is the ultimate application of grouped convolution, which decomposes conventional convolutions into multiple grouped convolutions and reduces the number of required parameters.

(3) Pointwise Convolution In the past, lightweight models based on discrete or group convolutions structures often did not make full use of 1×1 convolution, examples include the Xception and ResNeXt models. This kind of convolution is also called pointwise convolution, and it usually brings a large number of computations. To solve this problem, Megvii and Tsinghua University cooperated to use the group pointwise convolution calculation method to propose ShuffleNetV1 [119] in 2018. They used the pointwise convolution in place of the group convolution operation to reduce the calculation consumption, and used the channel shuffling method to solve the problem that the output channel can only be connected with some of the input channels. Subsequently, they proposed four guidelines for designing lightweight neural networks based on ShuffleNetV1. After carrying out a large number of experiments, they proposed ShuffleNetV2 [120] based on these four guidelines and improved the defects of ShuffleNetV1 by introducing channel splitting. In 2019, the Megvii team proposed ThunderNet [121] based on ShuffleNetV2 and Lighthead region-based CNN (RCNN), designed context enhancement module and spatial attention module to integrate local and global features and enhance network feature expression ability.

The University of Washington and Allen Institute for AI proposed a lightweight network called the efficient spatial pyramid network (ESPNet) [122] for semantic segmentation, which decomposes the standard convolution into 1×1 pointwise convolutions and a spatial pyramid of dilated convolutions. Both of themes are used to reduce the computational complexity of the network and resample the features of each effective receptive domain.

(4) Plug-and-Play Huawei Noah's Ark Lab has been keen to research plug-and-play lightweight network models in recent years. They proposed plug-and-play versatile filters with multiple channels or spatial versatile filters [123]. In 2020, the team proposed a new end-to-side neural network architecture called GhostNet [124], which uses a plug-and-play Ghost module to divide the original convolutional layer into two parts. The module first uses fewer convolution kernels to generate a small amount intrinsic feature map, and then uses simple linear change operation to further efficiently generate ghost feature map.

(5) Others Some lightweight networks use multiple convolutional structures for compression. Stanford University, DeepScale and UC Berkeley jointly proposed SqueezeNet [125], which was a relatively early and classic lightweight network that used the 1×1 convolution and grouped convolution methods, and then used the Fire module for parameter compression. Many subsequent works produced by of the team were conducted to optimize and improve this model. On the basis of SqueezeNet, they proposed SqueezeNext [126] by directly incorporating separation convolution to achieve improved parameter compression. By removing global average pooling and using the design idea of YOLO as reference, they added ConvDet structure to predict the characteristics extracted from the backbone network bounding box, and finally constructed SqueezeDet [127].

Many other scholars have also constructed lightweight structures by using channel features. In 2018, Gao et al. [128] proposed ChannelNet with three instances of channel-wise convolutions: group channel-wise convolutions, depth-wise separable channel-wise convolutions, and the convolutional classification layer, which reduce the number of model parameters. Momenta and the University of Chinese Academy of Sciences jointly proposed the squeeze-and-excitation network (SENet) [129] to obtain more channel features through modeling and introduced an attention mechanism between channels. The model was the champion of image classification task during the last ImageNet competition in 2017. In 2021, Yu et al. of Huazhong University of Science cooperated with Microsoft to apply efficient shuffle blocks to the high-resolution network HRNet [131], leading a lightweight network called Lite-HRNet [130]. They also introduced an efficient conditional channel weighting unit to replace the costly 1×1 convolution in shuffle blocks, and the weights were computed across channels and resolutions.

In recent years, a large number of models have been proposed, such as MobileNet, ShuffleNet and MobileNetV2. However, these models are heavily dependent on depth-wise separable convolution, which lacks effective implementation in most deep learning frameworks. Therefore, Weston University proposed an effective architecture called PeleeNet [132], which was constructed by conventional convolution. In 2020, Yitu and the National University of Singapore proposed a MobileNeXt [133] network architecture by using a novel SandGlass module group, which can be easily embedded into an existing network architecture to improve model performance.

In 2021, Zhou et al. [134] of Nanjing University of Posts and Telecommunications introduced a lightweight CNN, called residual semantic-guided attention network (RSANet). First, the lightweight convolutional network (LCNet) is used as backbone, and the Residual Semantic-guided attention feature pyramid network (RSAFPN) is used as detection head. In the LCNet, a constant channel module (CCM) was designed to save the memory access cost (MAC) and down sampling module (DSM) was designed to save the computational cost. In the RSAFPN, meanwhile, residual semantic-guided attention mechanism (RSAM) was employed to fuse the multi-scale features from LCNet to efficiently improve its detection performance. Han et al. [135] of the Chinese Academy of Sciences introduced versatile filters to construct efficient CNNs, which are widely used in various visual recognition tasks. By studying the applicability of the anchor-free strategy on lightweight object detection models, Yu et al. [109] of Baidu proposed a new family of real-time object detectors named PP-PicoDet, they enhanced the backbone structure and designed the lightweight structure of the neck, which improved the feature extraction ability of the network, and then improved label assignment strategy and loss function to make training more stable and efficient.

Lightweight structure design based on convolutional structure is a hot research topic for major companies and institutions. Lightweight structures designed by using traditional

grouped convolutions and deep separable convolutions are currently relatively mature and high-performing. In addition, the combination of hardware plug-and-play proposed by the Huawei Noah's Ark Lab is an attempt to combine lightweight model with hardware equipment. Some studies have also used combinations of channel features to design network structures, and we think this method may be extended to filters, feature maps and other structural features. The lightweight structure method designed by Yitu that can be embedded into an existing network framework is once and for all, and is the focus of future research. In general, lightweight structural design still has development prospects, as it saves additional compression steps and is more efficient and faster than traditional approaches.

2.2.2 Lightweight model design based on convolution operation method

To reduce the number of network operations, many researchers have optimized the convolution operation from the perspective of the convolution operation method, which can mainly be divided into two types: shift convolution operations and multiplication operations. shift convolution operations include standard shift, grouped shift, active shift and sparse shift operations. This section introduces the current research situation regarding lightweight model design based on convolution operation method in turn.

In order to solve the limitation of memory movement in shift operations, the UC Berkeley team first proposed an end-to-end training module which combines shift and pointwise convolutions [138]. In the same year, another team at the school used this principle to build a more compact and powerful network model called ShiftNet [139]. In 2019, Hangzhou Hikvision Research Institute proposed fully-exploited network (FE-Net) [140] based on ShiftNet, which improves the active shift layer by using sparse shift layer (SSL), eliminates meaningless shift operations and speeds up the running speed of models.

Aiming at the problems of high computational complexity and high energy consumption caused by a large number of intensive multiplication operations in traditional CNNs, some researchers have used other operations instead of multiplication operations. The Pengcheng Laboratory, Huawei Noah's Ark Lab and the School of EECS of Peking University proposed AdderNet [141], which uses addition instead of multiplication to greatly reduce the computational power consumption without reducing the accuracy of the model. Huawei Technologies put forward Deepshift [142], which utilizes bitwise shifts and sign flipping to replace multiplication operations, bitwise shift operations to replace absolute value multiplication operations, and sign flipping to replace positive and negative sign operations. Alibaba Damo Academy and other institutions jointly proposed ShiftAddNet [143], which uses the combination of addition operation and shift operation instead of multiplication.

2.2.3 Lightweight model based on YOLO model structure

In actual object detection projects, the trade-off between real-time and accuracy is very important, and the YOLO series models v2-v5 [144–147] are some of the most advanced object detection methods based on DNN, which have good performance in terms of speed and precision. The YOLO series models often include tiny versions when they are proposed. The lightweight idea of the tiny version is to greatly compress the parameters of the

network, reduce the convolutional layer, and make the network shallow and thin. In addition to these tiny versions of the YOLO series models, many researchers have constructed lighter weights based on the YOLO series models. The following introduces some excellent and typical lightweight models based on the YOLO model structure.

To achieve real-time object detection on laptops or mobile phones without GPUs, Huang et al. [148] proposed YOLO-LITE based on tiny-YOLOv2, which proved the shallow network has great potential in fast object detection without GPUs, and also proved that BN is unnecessary for shallow networks. Based on the single-stage object detection network architecture of the YOLO network series, researchers from the Waterloo Artificial Intelligence Institute AI Institute of the University of Waterloo and Darwin AI created YOLO Nano [149] by using a human–machine collaborative design strategy. HIT Lab NZ scholars at the University of Canterbury proposed xYOLO [150] based on YOLOv3-tiny. By using the normal convolutional layer and XNOR layer for training and recall, it achieved greater inference speed on desktop CPU and GPU. Researchers from CVTE put forward the YOLO-Fastest algorithm [151] by combining it with EfficientNet-lite, which is known as the strongest mobile lightweight neural network. It can be easily deployed on mobile devices, and its image classification effect is good and meets the requirements of all platforms. In addition, the original team that developed YOLOv4 proposed Scaled-YOLOv4 to scale of YOLOv4 [147]. In 2021, Cai et al. [153] of Northeastern University proposed YOLObile framework, a real-time object detection on mobile devices via compression-compilation co-design, and proposed a novel block-punched pruning scheme and GPU-CPU collaborative scheme to achieve improved computational efficiency on mobile devices. To deploy real-time object detection algorithms on UAV-friendly devices, Debojyoti B. Et al. used compressed convolutional techniques, transfer learning, backbone shrinkage, and scale prediction to reduce the number of learnable parameters in YOLOv5model. It is worth noting that in addition to reducing GPU memory consumption by 34%, this study also reduced energy consumption by 10 watts [154].

2.2.4 Other types of approaches

Many scholars study lightweight models. The University of California, Berkeley and Nanjing University jointly proposed SkipNet [155], which uses a gate module to determine whether to perform a certain layer of calculation to reduce the model size. Google Research used a set of fixed scaling factors to evenly scale each dimension of the network and built a high-precision lightweight image classification model called EfficientNet [156]. The team proposed EfficientDet [157] by using a bidirectional feature pyramid network (BiFPN) and Box prediction net to predict the features extracted from the backbone network.

In real-world environments, it's much common to use training data to fine-tune quantized networks for dealing with the performance drop induced by quantization errors. However, when training data is unavailable due to security, privacy, or confidentiality concerns, a popular method is to use zero-shot quantization to addresses such problems which represents neural networks by low-bit integer without accessing any of the real data. Choi et al. [158] propose AIT, a simple yet powerful technique for zero-shot quantization, which uses a KL distance loss only without a cross-entropy loss, and manipulates gradients to guarantee that a certain portion of weights are properly updated after crossing the rounding thresholds. Moreover, learning to synthesize data has emerged as a promising direction in zero-shot quantization. Zhong et al. [159] propose a novel zero-shot quantization method

(IntraQ) which well retain the intra-class heterogeneity in the synthetic images and also was observed to perform state-of-the-art.

Discussion: lightweight network design methods are mainly divided into "lightweight model design based on convolution structure" and "lightweight model design based on convolution operation method" and "lightweight model design based on YOLO model structure". The design of lightweight models based on convolutional structures is relatively simple and easy to implement and deploy. By choosing a simplified convolutional structure, the model can be greatly compressed, but the feature representation ability of the model will be sacrificed. Although the design of lightweight model based on convolution operation method is relatively complex and the convolution operation needs to be improved, it is easier to balance the expression ability and compression degree of the model. However, if the improved convolution operation is more complex, it will affect the training efficiency of the model. Lightweight model design based on YOLO model structure is mainly for object detection tasks. This improvement is based on the most popular object detection algorithm, mainly to take advantage of the real-time detection ability of the YOLO model itself.

2.3 NAS

As network structures becomes increasingly complex, the artificial model compression technology of neural network relies on heuristic and rule-based strategy, and its trial-and-error and time costs will be unbearable. Algorithm designers need to explore larger design spaces, and trade-off between model size, speed and accuracy, which is usually suboptimal and time-consuming. Therefore, the NAS method has been introduced in model compression to reduce the time and cost of model construction. NAS is an automatic design method of neural network, which consists of three parts: search space, evaluation method and search method. As shown in Fig. 11, the principle of NAS is to first give a set of candidate neural network structures called search space, then use the learned search strategy to build the optimal neural network architecture from the search space, and then use the performance evaluation strategy to measure the performance, and finally obtain the optimal neural network architecture through feedback-based iterative learning. In the field of lightweight model design, NAS has gradually become one of the most important research directions. As shown in Fig. 12, this paper lists the classic NAS compression methods published by major research institutions at the top conferences and journals in recent years. It illustrates be seen that the research on this method began to increase significantly since 2018. In this paper, a total of 34 papers on NAS for model compression are counted. At present, the mainstream NAS algorithm can be divided into the following aspects: reinforcement

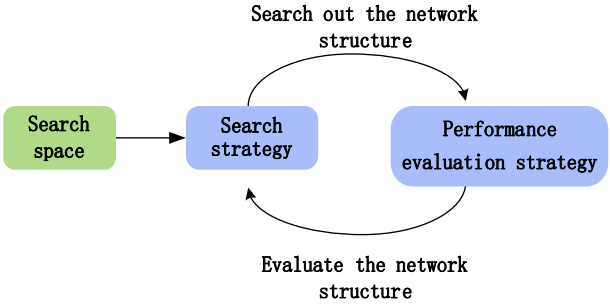


Fig. 11 The principle of NAS

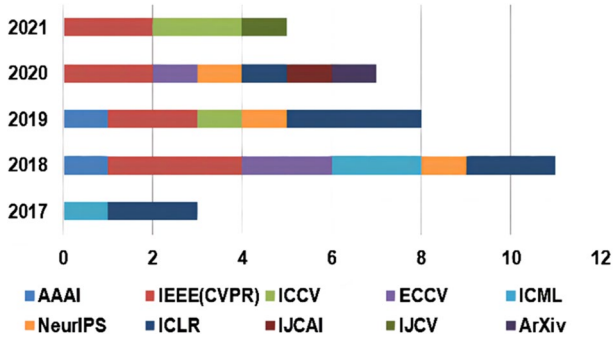


Fig. 12 Distribution of Light-weight NAS papers

learning methods (RL), neuro evolutionary methods, gradient based methods. This section is introduced according to these categories. The research situations of various research institutions and teams with respect to each aspect are shown in Fig. 13.

2.3.1 NAS methods based on reinforcement learning

With the rapid development of RL, it has begun to be applied in NAS. MIT and the University of Cambridge [160] proposed a NAS strategy based on reinforcement learning for the first time. This strategy models NAS as a Markov decision process, combines it with Q-learning optimization and experience playback method, and finally finds the optimal network architecture. Based on RL and block search space, Google team proposed NASNet [161], designed blocks with different architectures by using prior knowledge about characteristic problems, and constructed local search space of neural network architecture. The Chinese Academy of Sciences team and SenseTime [162] proposed a NAS method based on Q-learning algorithm in block-based search space. Shanghai Jiaotong University [163] used RL and path-level conversion operations to design an effective structure with complex path topologies. The team also considered RL agent as the meta-controller, and proposed an efficient architecture search (EAS) [164] framework based on the current network and reused its weights, saving considerable computational costs.

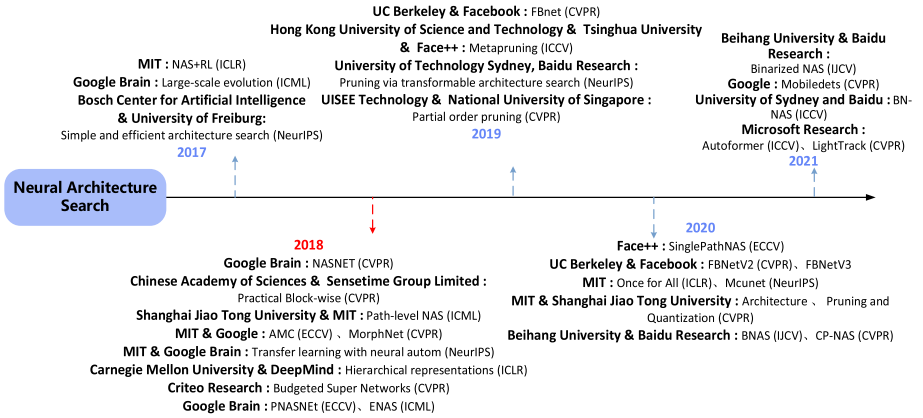


Fig. 13 The research situations of various research institutions and teams on NAS

To solve the limitations of manual heuristic methods and rule-based strategies, the MIT Han Song team and the Google team jointly proposed the AutoML model compression method (AMC) [165], which uses RL to automatically learn model compression strategies to search the design space, achieves a higher compression ratio than other approaches and maintains the performance of neural network. Many other scholars have followed up on this AutoML research. For example, Google and the Stanford University team [158] proposed a method to speed up networks by using knowledge transfer with AutoML in existing tasks. In 2019, they proposed an automatic NAS method MnasNet [137] for mobile neural network model design, which is takes the diversity of layers into account. Hong Kong University of Science and Technology, Tsinghua University, Megvii Technology and other teams [167] proposed a meta-pruning network for channel pruning, which uses the idea of AutoML to automatically prune and adjust the appropriate network structure under different constraints.

2.3.2 NAS methods based on evolutionary algorithm

With the development of evolutionary algorithms, many scholars have applied them to the NAS method. Google's team [168] used evolutionary algorithm to search neural network architecture automatically, and finally obtained the optimal image classification network architecture. In 2019, the team also used evolutionary algorithm to propose an image classifier called AmoebaNet [169], which surpassed the manual classifier for the first time. Carnegie Mellon University and DeepMind [170] proposed a search strategy based on evolutionary algorithm by designing a hierarchical search space. To reduce the weight coupling of the super network, the Megvii Research Institute used an evolutionary algorithm to propose a single path super network [171], which does not require fine-tuning.

2.3.3 NAS methods based on gradient

The gradient optimization algorithm is a method to obtain the optimal strategy by continuously approaching the optimal value. By using gradient descent technology, Sorbonne University and Crito Research Institute [172] proposed budget super networks (BSNs) to solve the computational cost, memory consumption and distributed computing cost problems. From 2019 to 2020, the feature balance network (FBNet) series of models [173–175] were proposed by UC Berkeley and Facebook, these are a series of lightweight networks that are completely based on NAS method. FBNet combines differentiable neural network search (DNAS) [176] and resource constraints, FBNetV2 adds channel and input resolution searches, and FBNetV3 uses accuracy prediction to perform fast NAS.

2.3.4 Other types of approaches

In addition to the methods mentioned above, the Google artificial intelligence team has conducted many research on other methods of NAS. They used the sequential model-based optimization (SMBO) strategy to propose PNASNET [177]. In the same year, they and the MIT team used a sparsifying regularizer to calculate the costs of neurons relative to target resources, and proposed an automated NAS method called MorphNet [178] that iteratively scales neural networks. Another team at Google proposed an efficient neural architecture search (ENAS) [179] method by searching the optimal subgraph in large computational graph to learn and discover NAS.

The team at Google combined the depth-wise separable convolution structure and the inverted residual with linear bottleneck structure, using the MnasNet [137] model based on the SE structure to propose MobileNetV3 [136], and using complementary search technology and NetAdapt algorithm to achieve computer vision tasks carried on the mobile phone. In 2021, Google cooperated with the University of Wisconsin-Madison [180] to propose an augmented search space family with building blocks based on regular convolutions, they called it Mobiledets, that NAS methods can substantially benefit from this enlarged search space to achieve better latency-accuracy trade-off on a variety of mobile devices.

To solve the DNAS problem of high memory consumption in a large candidate set, Han Song's team at MIT proposed proxylessNAS [181], which can directly learn the architecture for large-scale target tasks and target hardware platforms. The University of Illinois at Urbana-Champaign, ByteDance, and the Snap team collaborated to propose the Slimmable Neural Network [182], which dynamically adjusts the network width according to the device benchmark and resource constraints at runtime, and has good performance in object detection, instance segmentation, human key point detection and other tasks. Elsken et al. [183] team at the Center for Artificial Intelligence of Bosch proposed a new method based on a simple Hill-climbing process to automatically search for a CNN architecture with good performance. In 2021, Chen et al. [184] of the University of Sydney and Baidu proposed NAS with batch normalization (BN-NAS) to accelerate NAS, which can significantly reduce the time required by model training and evaluation by only training the BN parameters during the supernet training, thereby accelerating network convergence for NAS. Chen et al. [185] of Stony Brook University and Microsoft Research proposed a simple yet effective framework for efficient training of transformer supernets. Without extra fine-tuning or retraining steps, the trained supernet is able to produce thousands of high-quality transformers by inheriting weights from it directly.

Many researchers still combine NAS methods with other compression methods. In order to overcome the structural limitations of pruning networks, University of Technology Sydney and Baidu Research [186] proposed using NAS to directly search for networks with flexible channels and layer sizes. The National University of Singapore and UISEE Technology [187] used the search space of the partial-order hypothesis pruning architecture to automatically search for the structure with the best speed and accuracy performance. MIT's [188] team proposed once for all (OFA), which can train subnetworks under different hyperparameters by combining KD, network pruning, and NAS algorithms to search for suitable subnetworks according to different accuracy requirements and resource constraints. The team of MIT and Shanghai Jiaotong University [189] proposed an APQ algorithm that combines NAS, pruning strategy, and quantization strategy for efficient deep learning-based inference on resource-constrained hardware. The National University of Taiwan and MIT IBM Watson Artificial Intelligence Laboratory jointly proposed microcontroller units Net (MCUNet) [190], a framework that jointly designs the efficient neural architecture (TinyNAS) and the lightweight inference engine (TinyEngine), enabling ImageNet-scale inference on microcontrollers. Combining binary quantization can also achieve compression. In 2021, Beihang University and Baidu Research [191] introduced the partial channel connections for memory-efficient differentiable architecture search (PC-DARTS) strategy into the search of the binary network, and gradually deleted operations with lower probability during the search process to narrow the search space. Beihang University proposed a child parent search strategy, called CP-NAS [192], which uses a full-precision model (parent) to guide the search of binary model (child). Another team Yan

et al. [193] of Microsoft Research used NAS to design a lightweight object tracker, called LightTrack, which reformulates one-shot NAS specialized for object tracking, and introduces an effective search space. In addition, it can run in real-time on diverse resource-restricted platforms.

Discussion Many studies use NAS for model compression and acceleration. According to the method of solving the search problem, NAS methods are divided into reinforcement learning based, evolutionary algorithms and gradient based. NAS method based on reinforcement learning is very common, it can find a good neural network structure, but the training time is long. Although the NAS method based on evolutionary algorithm is easy to fall into local optimal solution, its search time is relatively short. At present, some scholars have used evolutionary algorithm to search and accelerate the network structure, and have achieved some excellent research results in image classification. The gradient-based NAS method to find the optimal value is more novel. It does not need to maintain the whole population like evolutionary algorithms, which can effectively reduce the computational cost of search, but it is still in the development stage. In addition, NAS methods are often used in combination with other model compression methods, which also illustrates their compatibility. The NAS method proposed by MIT in recent years, which can search for a special network structure by combining hardware performance with different precision levels and resource constraints, may be a hot spot for future development.

2.4 Network quantization

The main idea of network quantization is to convert floating-point operations into integer fixed-point operations, so as to compress the model size and accelerate network training. As shown in Fig. 14, Quantization model is a method that uses low-precision weights to approximate conventional precision (generally 32-bit floating-point) to store model weights. At present, INT8 quantization is mostly used in the industry. TensorRT, TensorFlow, PyTorch and many other deep learning frameworks have already enabled (or are enabling) quantization. Academic research on quantization pays more attention to how to reduce the accuracy loss caused by model quantization, which can be roughly divided into binary quantization, ternary quantization, cluster quantization, and quantization combined with other compression methods. As shown in Fig. 15, this paper lists the classic network quantization methods published by major research institutions at the top meeting in recent years. In this paper, a total of 33 papers about network quantization compression model methods are counted. The research situation of various research institutions and teams regarding each aspect are shown in Fig. 16.

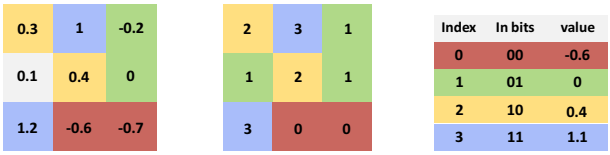


Fig. 14 Network Quantization

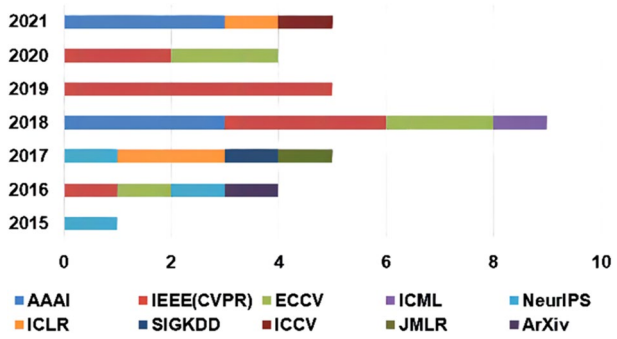


Fig. 15 Distribution of Network Quantization papers

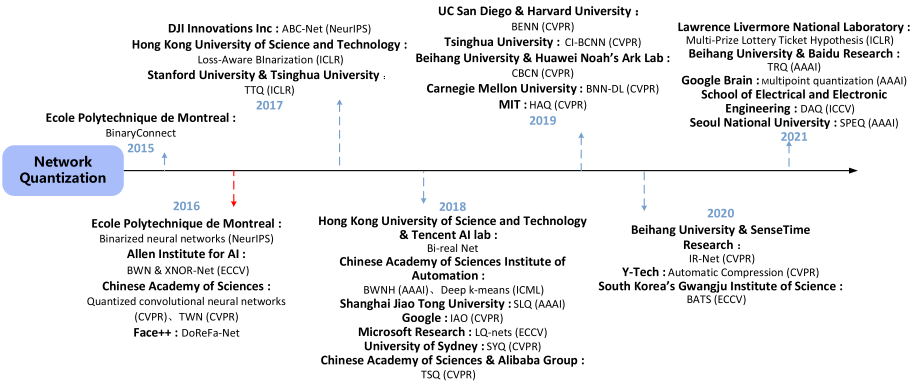


Fig. 16 The research situations of various research institutions and teams on Network Quantization

2.4.1 Binary quantization

Binarization means that the network parameters are limited to 1 or—1, which greatly reduces the storage space and memory space requirements of the given model, and converts the original multiplication operation into an addition or shift operation. Since 2015, quantization methods have been applied in compression. Courbariaux et al. [194] first proposed binary connection, a weight binarization scheme, and provided a training method for binarization parameters in forward and backward propagation processes. Subsequently, the team proposed a method for training binarized neural networks (BNN) [195], which uses binary weights and activation values to calculate parameter gradient in the training phase, quantifies weights and activation functions for the first-time during inference and the whole training period, and replaces most arithmetic operations with bit operations. The Allen Institute of Artificial Intelligence team proposed binary-weight-networks (BWN) and XNOR-Net [196]. They greatly improved the computational efficiency of floating-point numbers used in traditional convolutions by utilizing approximate convolutions of binary operations.

Shenzhen DJ-Innovations (DJI) [197] proposed a high-precision BNN called the accurate binary CNN (ABC-Net), which uses a linear combination of multiple binary weight bases to approximate full-precision weights, this was the first time that the accuracy of BNN can matching with the full-precision model on ImageNet. Hong Kong University of

Science and Technology [198] proposed a binarization algorithm called LAB that directly considers the influence of binarization weights on loss, and uses the approximate Newton algorithm for the diagonal Heisen approximation to obtain the appropriate binarization weights. Another team at the school and the Tencent Artificial Intelligence Laboratory jointly proposed Bi-Real Net [199] to compensate for the defects of XNOR-Net. They introduced shortcut connection and adopted second-order function to fit activation function sign, achieving binarization of network weight and activation output, while maintaining high inference accuracy. Hu et al. [200] team at the Institute of Automation of the Chinese Academy of Sciences proposed the BWNH method, which trains a binary weight network by hashing, and converts the problem of learning binary parameters into a hashing problem under inner product similarity. Diffenderfer et al. [201] proposed a method that simply prunes and quantizes randomly weighted full-precision neural networks to obtain a highly accurate BNN, and then proposed and verified the multi-prize lottery ticket hypothesis, that is, a sufficiently overparametrized randomly weighted network contains binary subnetworks, which can achieve high accuracy without any training.

Many scholars are still making improvements to binary networks. Xilinx Research Labs [202] proposed FINN, a framework for building fast and flexible field-programmable gate array (FPGA) accelerators by using a flexible heterogeneous streaming architecture, which enables efficient mapping of BNNs to hardware. The University of California San Diego and Harvard University [203] proposed binary ensemble neural network (BENN) to improve the performance of BNN at the cost of limited efficiency. Wang et al. [204] proposed a channel-wise interaction based binary convolutional neural network learning (CI-BCNN) method for efficient inference. Liu et al. [205] of Peking University of Aeronautics and Astronautics and Huawei Noah's Ark Laboratory designed the circulant binary convolutional network (CBCN) based on the redesign of binary filter, and enhanced the expression ability of binary convolution features by rotating the binary filter from multiple angles. Ding et al. [206] proposed a method of using distribution loss to explicitly regularize the activation flow, and developed a BNN distribution loss (BNN-DL) framework to systematically formulate the loss, so as to minimize the adverse effects caused by forward binarization and back propagation.

2.4.2 Ternary quantization

Ternary Quantization is to introduce 0 as the third threshold on the basis of binarization to reduce quantization error. The Han Song team of Stanford University and Tsinghua University proposed trained ternary quantization (TTQ) [207], a method that can reduce the precision of weights in neural networks to ternary values. This method improves the accuracy of some models on ImageNet but brings very little accuracy degradation. The Academy of Mathematics and Systems Science in Chinese Academy of Sciences [208] proposed a ternary weight network (TWN), which uses a weight that is different from the traditional 1 or weight average, they convert the weight into $\{w, 0, +w\}$.

To solve the problem of significant accuracy loss caused by the existing ternary neural network methods based on rule-of-thumb quantization methods with simple thresholding operations, Li et al. [209] of Beihang University and Baidu Research proposed an order residual framework called ternary residual quantization (TRQ), which recursively performs quantization on full-precision weights for a refined reconstruction by combining the binarized stem and residual parts, this method endows the quantizer with high flexibility and precision.

2.4.3 Cluster quantization

Some scholars have combined clustering and quantization methods to achieve compression. The National Laboratory of Patter Recognition Institute of Automation at the Chinese Academy of Sciences [210] proposed a quantitative CNN method, which extends the K-means clustering method to the volume layer, divides the weight matrix into many blocks, then obtains the codebook by clustering, and proposes an effective training scheme to suppress the multi-layer cumulative error after quantization. Subsequently, the team proposed a convolution compression method called deep K-means [211], which clusters the weights by K-means, records K clustering centers and weight allocation indicators, and finally realizes shared weight compression. Shanghai Jiao Tong University [212] proposed level quantization, which considers two quantization methods: single-level quantization (SLQ) and multi-level quantization (MLQ).

2.4.4 Other types of approaches

In addition to the previously described quantitative methods, many scholars have conducted other research and improvement on quantization. Megvii Technology [213] proposed DoReFa-Net, which quantifies the weight and activation value to 1-bit and 2-bit. Google Technologies has proposed a method integer-arithmetic-only (IAO) [214], which allows the associated network to use only integer calculations when inferring, and quantifies it on already compressed networks such as MobileNet. Microsoft Research [215] proposed learned quantization (LQ-nets), so that the quantizer could be jointly trained with the network to adaptively learn the optimal quantization bit width. The University of Sydney proposed a gradient-based symmetric quantization method (SYQ) [216], which designs binary or ternary weights and defines scaling factors with different granularities at multiple levels to estimate network weights. Wang et al. [217] proposed a two-step quantization (TSQ), which first uses a sparse quantization method to quantify the activation value, and then regards the quantization weight as a nonlinear least square regression problem. Qin et al. [218] proposed an information retention network (IR-Net) to retain the information of forward activation and backward gradient. Due to the discontinuity of the quantization function, it is difficult to calculate the gradient. Alibaba [219] converted low-bit quantization into an ADMM-based optimizable objective function to solve this problem. Hubara et al. [220] proposed a method to train quantized neural networks (QNNs) — neural networks with extremely low precision (e.g., 1-bit) weights and activations at runtime. In 2021, Liu et al. [221] of the University of Texas at Austin and Google Brain considered the post-training quantization problem, then proposed multipoint quantization, a quantization method that approximates a full-precision weight vector by using a linear combination of multiple low-bit numbers vectors. Kim et al. [222] of the School of Electrical and Electronic Engineering proposed a novel quantizer, called distance aware quantizer (DAQ), which is mainly consists of a distance aware soft rounding (DASR) and a temperature controller. The DASR approximates discrete rounding with the kernel soft argmax, allowing the training of quantization networks in an end-to-end manner to alleviate the gradient mismatch problem. The controller adaptively adjusts the temperature parameter in DASR according to the input, addressing the quantizer gap problem.

In the past two years, many achievements have been made in research combining quantization with other compression methods. In 2019, Wang et al. [223] at MIT proposed an automated hardware-aware automated quantization framework (HAQ), which uses RL to

specify specific quantization strategies according to the utilized hardware structures. In 2020, Y-Tech and Seattle AI Lab [224] jointly proposed automatic compression, which automatically combines pruning and quantization according to the size of the target model. Quantization can also be combined with NAS methods. The Samsung Artificial Intelligence Center [225] proposed a binary architecture search (BATS), which greatly reduces the accuracy gap between BNNs and their actual values through NAS. South Korea's Gwangju Institute of Science (GIST) proposed a binary network learning architecture called BNAS [226], which specifically designs a search space for binary networks. Boo et al. [227] of Seoul National University and Hanyang University studied the use of KD methods to improve the performance of quantization networks, and then proposed stochastic precision ensemble training for quantum DNNs (QDNNs) [228], which is a KD training scheme. Moreover, they also proposed cosine similarity as an essential loss function to effectively distill the knowledge of activation quantization in training.

Discussion Network quantization is a technique to quantize the parameters of neural networks into a low-bit representation, which mainly includes binary quantization, ternary quantization and cluster quantization. Binary quantization and ternary quantization are research hotspots in the field of quantization. In addition, some scholars have studied the combination of multiple quantizations approaches to achieve very obvious compression effects, but these methods also result in serious losses of precision. Therefore, many researchers have studied the problems of precision recovery and gradient mismatch, and have also achieved good results. In general, quantization can speed up the calculation speed and greatly reduce the size of the examined model. The combination of it with hardware equipment is also being studied by many research institutions. Quantization is a very important model compression method.

As shown in the table, a direct comparison of the three sub methods is shown. Binary quantization is suitable for scenarios that require very high computing and storage resources. It has low accuracy but very small computing speed and storage space. Ternary quantization is more accurate than binary quantization, but it requires more storage space and computation time. Cluster quantization is suitable for scenarios with relatively low computational and storage resource requirements, but requires some knowledge of clustering algorithms.

2.5 KD

KD was first proposed in 2006, who used a shallow network to simulate the output structure of a large network. The main idea of KD is to transfer the knowledge learned from a teacher network with a complex structure and huge parameters to a small student network with a relatively simple structure. The student network can achieve similar expression ability while saving time and space costs. According to the types of knowledge to be transferred, KD can be divided into three categories: output transfer, feature transfer and relation transfer. A total of 26 papers on KD are counted in this paper. As shown in Fig. 17, many studies have been conducted on KD by major research institutions in recent years. In terms of various available strategies, the research results of major institutions are shown in Fig. 18. Many studies have applied KD method to compress the tasks of object detection and semantic segmentation. The following focuses on object detection.

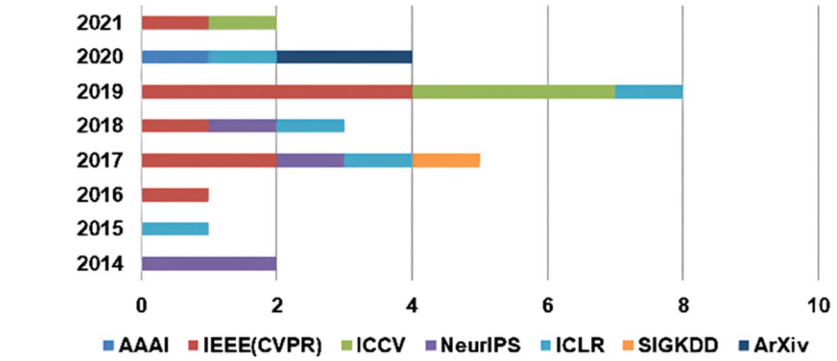


Fig. 17 Distribution of KD papers

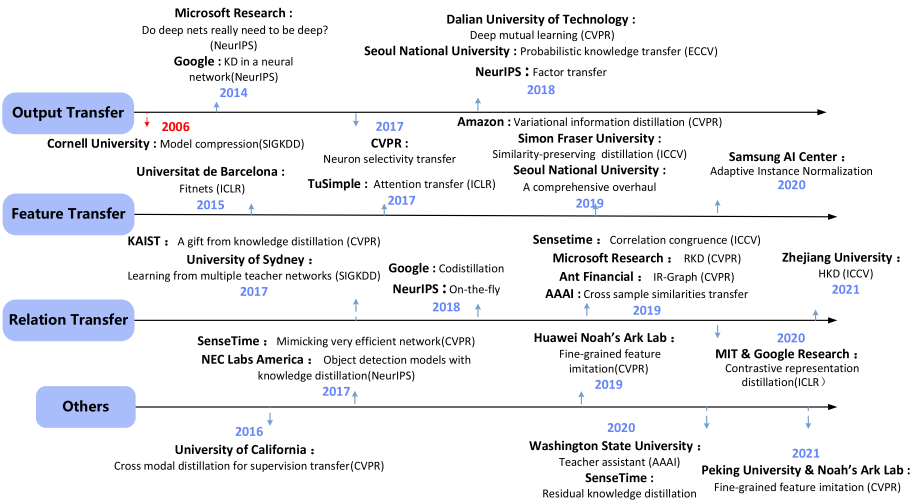


Fig. 18 The research situations of various research institutions and teams on KD

2.5.1 Output transfer

Output transfer uses network outputs as knowledge for student' network learning. In fact, it was not until 2014 that the relatively mature idea of KD was proposed. Instead of using hard targets, they used the final output of the teacher network to directly guide the learning process of the student' network, and proved that the knowledge capacity of shallow network is not smaller than that of deep network. In the same year, Google team first named this technology KD [229]. As shown in Fig. 19, such approaches first use hard targets to train the teacher model, and then use hard targets and soft targets of softmax layer output of teacher network for training. Finally, they send the softmax layer outputs of the two networks into total loss calculation to guide student' model training, so that the student' networks can achieve similar expression abilities and achieve compression.

In the traditional KD method, there is only a one-way transfer between teachers and students. Zhang et al. [230] proposed deep mutual learning method, which allows a group

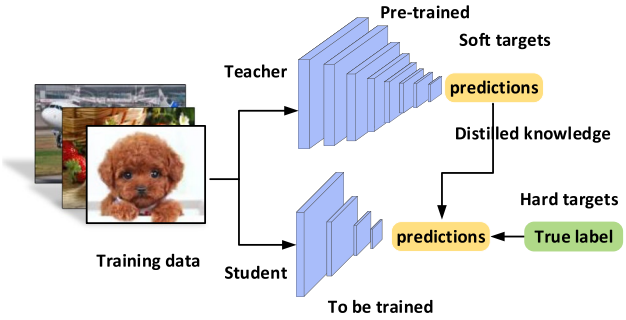


Fig. 19 The process of KD

of students to learn and teach each other according to the outputs of multiple networks. To enable students to better understand the teacher model, Kim et al. [231] of Seoul National University proposed utilizing convolution operation to encode the output of the teacher model and decode (translate) it to students, so as to achieve better learning effect.

Borui et al. [232] of Megvii Technology Limited provide a novel viewpoint that the potential of output KD has actually not been fully exploited, so they reformulate the classical KD loss into two parts, i.e., target class knowledge distillation (TCKD) and non-target class knowledge distillation (NCKD). Traditional KD method naturally suppresses the effectiveness of NCKD and limits the flexibility to balance these two parts. Thus, they present Decoupled Knowledge Distillation (DKD), further decouples TCKD and NCKD enabling TCKD and NCKD to play their roles more efficiently and flexibly, and controls the action of both by independent hyperparameters, resulting in a significant performance gains on a range of visual tasks.

2.5.2 Feature transfer

Feature transfer uses the hidden layer features of network learning as knowledge for students to learn. In 2015, Romero et al. [233] proposed Fitnet on the basis of Hinton, which trains the student network by using the hidden features obtained from the middle layer of the teacher network. Komodakis and Zagoruyko [234] proposed attention transfer (AT), which encourages student' networks to imitate the attention and sensitivity distributions (gradient) of teacher' networks for feature extraction. TuSimple [235] regards knowledge transfer as a domain adaptation problem, and implement knowledge transfer by matching the activation distribution of the feature maps of the teacher model and student model.

The Korea Advanced Institute of Science and Technology, Amazon and other institutions [2] proposed an information-theoretic framework for knowledge transfer that formulates knowledge transfer as maximizing the mutual information between the teacher and student networks. Tung et al. [236] proposed a method of maintaining similarity KD, so that the input sample pairs of similar (different) activations in the teacher network can produce the same (different) activations in the student network, thereby guiding the learning of the student network. Heo et al. [1] of Seoul National University proposed a new feature distillation method by designing the distillation loss as a synergy between teacher transformation, student transformation, distillation feature location and distance function. The Computer Vision Lab of the University of Nottingham and the Samsung Artificial Intelligence Center [237] proposed a new KD method based on feature statistics, which feed the learned statistical information back to the teacher through adaptive instance normalization

(on the condition of students), and allow the teacher network to reliably transmit the statistical information of student' learning through loss.

2.5.3 Relation Transfer

Relation transfer uses the relationships between the layers of network characteristics or the relationships of samples as knowledge for students to learn. Yim et al. [238] distilled by fitting the relationships between layers of teachers and students to let students learn the process of feature extraction from teachers. You et al. [239] integrated the knowledge of multiple teacher networks, used the information of the middle and output layers in the teacher network to standardize some layers in the student network, and finally trained a thin and deep student network. Similarly, the Google team [240] proposed the codistillation method to improve accuracy and speed up training.

Instance relationships can also be learned as a kind of knowledge. The National University of Defense Technology and Shangtang Technology [241] proposed a new distillation framework named correlation congruence KD (CCKD), which considers not only instance information but also correlation information between instances when transferring knowledge. Park et al. [242] introduced a novel approach called dubbed relational KD (RKD), which transfers mutual relations of data examples instead. Liu et al. [243] used the transformation of the distance between two samples in different layers for distillation. MIT and Google [244] proposed that more information can be obtained by training students to learn the teacher's description of data. It is critical for ufficeent student network learning to integrate both individual knowledge and relational knowledge while reserving their inherent correlation. Zhou et al. [245] of Zhejiang University proposed distilling the novel holistic knowledge based on an attributed graph constructed among instances.

2.5.4 Other types of approaches

In addition to the abovementioned conventional KD methods, some special KD methods have been researched. Some scholars have specially studied the KD method for object detection task. Gupta et al. [246] proposed a cross modal distillation method for supervision transfer, which transfers the knowledge learned from the RGB data set to a deep learning scenario. Li et al. [247] proposed using the feature map sampled by the region candidate frame to perform distillation, and L2 loss to reduce the differences between the local feature sampling results of large and small networks, to achieve certain results on detection tasks. The Communications & Media Research Laboratory of Korea Eletronics Telecommunications Research Instutue [248] also used KD method to improve the performance of lightweight mobile network single shot multibox detector(SSD) [273] model in object detection. Chen et al. [249] introduced the method of combining hint learning, soft targets and hard targets into object detection distillation simultaneously. By giving different weights to different loss functions, the whole network is trained, and the performance of multi-classification object detection network is improved. Wang et al. [250] introduced anchor points around the area as supervisory information to avoid the background noise problem during simulation. Chen et al. [251] of Peking University and Noah's Ark Lab proposed utilizing a large amount of unlabeled data in the wild to address the data-free KD problem, then in order to improve the performance of the student network, a data-free noise distillation algorithm was also proposed, which uses the student network to learn an

adaptation matrix to correct the label noise produced by the teacher network on the collected unlabeled images.

Some scholars put forward "assistant" to help learning. Mirzadeh et al. [252] introduced multi-step distillation method, using a medium-sized network as a teacher assistant to improve knowledge extraction ability and reduce the gap between primary school student' networks and teacher network. Tianjin University, the Chinese University of Hong Kong and SenseTime [253] proposed residual KD method. They used assistant to help students acquire more knowledge by learning the residual between teachers and students.

Discussion:The research on KD mainly includes relationship transfer and feature transfer. Output Transfer constrains the output of the student model to be similar to the teacher model. Feature Transfer constrains the intermediate features of the student model to be similar to those of the teacher model. And relation Transfer constraints the category relation of the student model is similar to that of the teacher model, and this relation can be obtained by computing the correlation matrix. Compared with Output Transfer and Feature Transfer, Relation Transfer can control the similarity between the student model and the teacher model more fine-grained. In addition, these three distillation methods can also be combined, such as the combination of Output Transfer and Feature Transfer, which can better preserve the details and features of the model. It should be noted that KD also has some limitations. When there is a large gap between the performance of the teacher model and the student model, the effect of KD will be limited. In addition, for some complex tasks, KD may need to be performed in multiple stages to achieve the best results. In addition, the combination of knowledge and relationships proposed by scholars at Zhejiang University is a new attempt. Some researchers used semi supervised learning mode to reduce the gap between students and teachers. We think that more research can be conducted in this area, The problem of KD without data and the wider application of KD the field of target detection instead of image classification may be future research focuses.

2.6 Low-rank decomposition

Since most of the calculations in a CNN are convolution computations, in essence, we can consider convolution computations to be matrix analysis. Therefore, the compression method of low-rank decomposition was introduced. The main idea is to decompose the convolution kernel through matrix analysis methods such as low-rank decomposition. Therefore, a large convolution kernel tensor is decomposed into a small convolution kernel to approximate the original weight matrix, reducing the computations and parameters required by the model to a certain extent. The compression method of low-rank decomposition has been rising in popularity since 2013. This paper collects a total of 17 papers on DNN model compression via low-rank decomposition. Figure 20 shows the popularity of low-rank decomposition research over the past 8 years. More studies on this method were conducted in 2018, while fewer studies have been conducted in recent years. The process of low-rank decomposition for the convolution kernel is shown in Fig. 21. The research situations of various research institutions and teams are shown in Fig. 22.

In 2013, the University of Oxford and Facebook [254] first theoretically explained the feasibility of applying matrix factorization to convolution kernels, and proposed that the remaining values could be accurately predicted by given part of the weight value of each feature. In the second year, the Skolkovo Institute of Science and Technology [255] proposed the CP decomposition method to compress the convolution kernel in the convolutional network. With the development of low-rank decomposition methods, Facebook

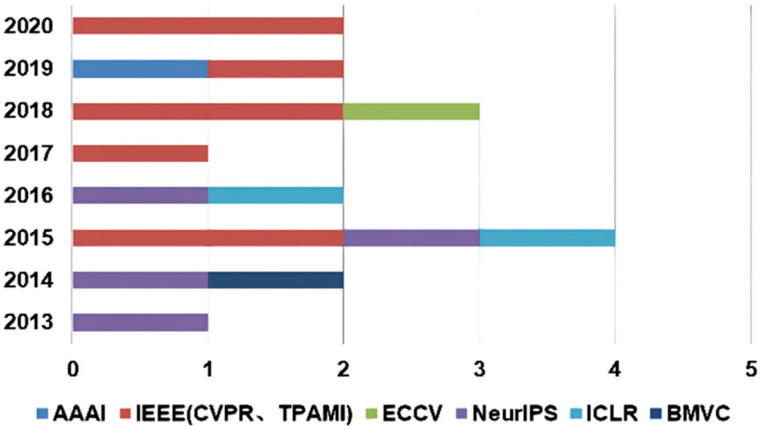


Fig. 20 Distribution of Low-rank Decomposition papers

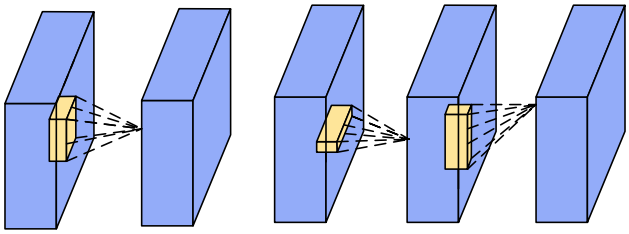


Fig. 21 The process of low-rank decomposition for the convolution kernel

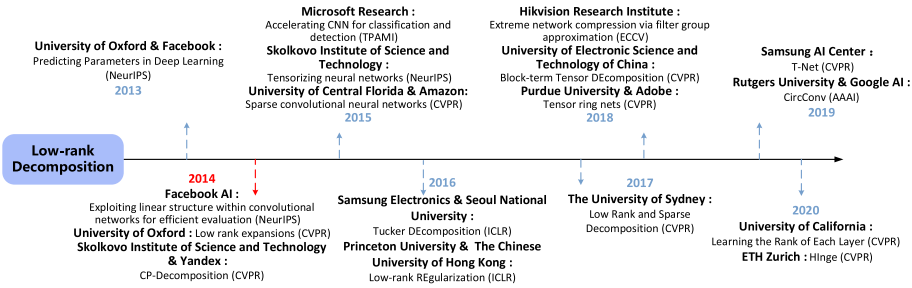


Fig. 22 The research situations of various research institutions and teams regarding low-rank decomposition

AI research [256] has proposed a variety of low-rank decomposition strategies, such as CP decomposition and cluster-based decomposition. Jaderberg et al. [257] used low-rank decomposition techniques to take advantage of the redundancy between different channels and filters to accelerate the trained network.

In addition to CP decomposition, Samsung Electronics [258] proposed a convolution kernel decomposition method based on Tucker decomposition. First, the rank was selected by using variational Bayesian matrix decomposition, then kernel tensor Tucker

decomposition was performed, and finally the model was adjusted. Microsoft Research [259] adopted the generalized singular value decomposition (GSVD) strategy, which achieved acceleration while the precision decreased slightly. The Skolkovo Institute of science and technology [260] used Tensor-train Decomposition algorithm to decompose the full connection layer to achieve network parameter compression. A team of the University of Central Florida [261] introduced LASSO sparsity constraint on the basis of tensor decomposition to further sparsify the convolution kernel after preliminary decomposition.

Princeton University and the Chinese University of Hong Kong team [262] found that a specific form of low-rank decomposition has a closed global optimal solution. They used the BN method to train a deep low-rank neural network, and tensor decomposition technology to remove convolution kernel redundancy. The University of Sydney team [263] proposed a unified framework for feature reconstruction by combining low-rank decomposition and sparsity. The Hikvision Research Institute [264] proposed a new low-rank decomposition method based on filter bank structure, which can significantly reduce redundancy while maintaining most of the feature representation. The University of Electronic Science and Technology team [265] proposed to decompose tensor by combining CP decomposition and Tucker decomposition. The Purdue University team [266] used Tensor-ring decomposition to decompose the weight tensor. SAIC-Cambridge added various low-rank constraints to the weight tensor to construct a fully parameterized CNN called T-Net, which has a single high-order and low-rank tensor [267]. The University of California [268] rebuilt the loss function by adding constraints to the tensor rank, and converted it into an optimization problem to achieve compression. Google Artificial Intelligence [269] proposed a new algorithm to convert non-circulant tensors into circulant tensors, making it possible for a new algorithm to implement circulant convolution on the pretrained non-circulant model. ETH Zurich [270] combined filter pruning and tensor decomposition for compression.

Recent research on low-rank decomposition methods has focused on breaking through the limitations of traditional low-rank decomposition methods in the past. The popular low-rank decomposition method singular value decomposition (SVD) ignores that the relative importance or contribution of each neuron in each layer is different. When it is used for model compression, it is mainly applied to the fully connected layer and only retains the most prominent components of the decomposition matrix. Sridhar Swaminathan et al. [271]at Bennett University reconsidered the importance of the input–output neurons of the layer, sparsified the SVD matrix to keep the unimportant neurons low-rank, and verified the effectiveness of this compression method for image recognition models. More significantly, existing tensor compression methods enforce a low-rank factorization structure on neural network weights, and have been effectively applied to the compression of some large neural networks, including VGG. But advanced backbone networks often used in object detection tasks, such as MobileNet, have taken advantage of this decomposition structure through depthwise separable convolutions. Pure tensor factorization becomes trivial at this point. A team from the University of California and Meta [272] combined low-rank tensor factorization with sparse pruning to capture complementary coarse and fine structure, respectively, and outperform sparse pruning or low-rank tensor factorization alone. This study is the first to consider the combination of low-rank tensor compression with sparse pruning. Alternatively, this is the first study to investigate low-rank + sparse weight compression for SOTA architectures that rely on efficient depthwise separable convolutions.

Low-rank decomposition is very suitable for model compression and acceleration, but the implementation of low-rank decomposition method is not easy and involves high

computational costs. The current method mainly involves layer by layer address approximation. The compression of global parameters and the retraining process after decomposition may be the focuses of future research.

3 Datasets and evaluation metrics

This section presents an overview of the Common Objects in Context (COCO) test-dev2015, the Pascal Visual Object Classes(VOC)2007, ImageNet Large Scale Visual Recognition Challenge (ILSVRC)2012, CIFAR10, and CIFAR100 public datasets.

3.1 Datasets

3.1.1 COCO test-dev2015

The COCO dataset is a large-scale object detection, keypoint detection, and captioning dataset published by Microsoft. It is one of the most challenging datasets available and a very general standard. It contains 91 classes of objects taken from complex everyday scenes, 328,000 images and 2,500,000 labels, with an average of 3.5 classes and 7.7 instances per image. Among them, 80 object categories and the corresponding instance labeling information are selected for the object detection task. Test-dev2015 is a subset of the test set published in 2015 in the COCO dataset, which has about 20 k images in total. The differences between the test-dev and validation sets are threefold: guaranteed consistent evaluation of test-dev using the evaluation server, test-dev cannot be used for training (annotations are private), and a leaderboard is provided for test-dev, allowing for comparison with the state-of-the-art. So the COCO test-dev2015 is the default test data for general testing, and latest publications often report their performance results in the Test-dev2015 dataset for fair and open comparison.

3.1.2 VOC 2007

The VOC dataset is a dataset of competitions that took place between 2005 and 2012 with the goal of object recognition in images. The labeled objects in this dataset include 20 categories including people, animals (such as cats, dogs, islands, etc.), transportation vehicles (such as cars, boats, airplanes, etc.), and furniture (such as chairs, tables, sofas, etc.). This dataset is also a commonly used dataset for object detection tasks and contains 11,530 images for training and validation, 27,450 calibrated objects of interest, and an average of 2.4 objects per image. The VOC 2007 dataset contains 9,963 labeled images with a total of 24,640 labeled objects Table 2.

3.1.3 ImageNet ILSVRC 2012

The ImageNet is a well-known dataset in computer vision. The dataset used in the ImageNet ILSVRC is a lightweight version of the ImageNet dataset. Many papers at the forefront of computer vision use the ImageNet ILSVRC 2012 dataset to test their models. ImageNet ILSVRC 2012 has a total of 1000 classes. There are 1,281,167 images in the training set with 732 to 1,300 images of each type, 50,000 images in the validation set with 50

Table 2 A Comparison of Sub Methods for Network Quantization

	Binary Quantization	Ternary Quantization	Cluster Quantization
Accuracy	minimum	better than the former	rely on clustering algorithms
Calculation speed	fastest	slower than the former	slowest
Storage space	least	Less than the former	most
Training effect	good	good	rely on clustering algorithms
Application	mobile terminal, embedded device	mobile terminal, embedded device	Server

images of each type, and 100,000 images in the test set with 100 images of each type. It is worth mentioning that the evaluation metrics of ImageNet are generally fixed, namely top1, top5.

3.1.4 CIFAR10 and 100

The CIFAR-10 and CIFAR-100 datasets are subsets of the Visual Dictionary(Teaching computers to recognize objects), Collected from google and various search engines by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton.

The CIFAR-10 dataset contains 60,000 color images with 32×32 resolution, in total 10 categories with 6000 images per class. There are 50,000 images in the training set and 10,000 in the test set. It's worth noting that these classes are completely mutually exclusive, with no overlap between "cars" and "trucks": "cars" includes cars, SUVs, etc., and "trucks" includes only big trucks, and neither includes pickup trucks.

The CIFAR-100 dataset is very similar to CIFAR-10 except that it has 100 classes, each class contains 600 images, and each class has 500 training images and 100 test images. The 100 classes in CIFAR-100 are divided into 20 superclasses. Each image is accompanied by a "fine" label (the class it belongs to) and a "coarse" label (the superclass it belongs to). The categories in CIFAR-100 are shown in Tables 3.

3.2 Metrics

3.2.1 FLOPs

FLOPs is the number of floating-point operations that can be performed per second. It reflects the demand of the model for hardware computing units, and is usually used to evaluate the computational complexity of a deep learning model or the demand for computing resources. Many studies choose FLOPs as a reference for the speed of the model. A higher value of FLOPs means that the model is more computationally demanding and requires more computing resources. The overall computational cost of the model is equal to the sum of the computational cost of each operator in the model. The computation amount of each operator is calculated in different ways. For convolution, FLOPs can be approximated as Eqs. (1), Where OC is the number of output channels, OH and OW are the height and width of the output feature map, KH and KW are the height and width of the convolution kernel, and IC is the number of input channels.

$$FLOPs_{Conv} = 2 \times OC \times OH \times OW \times KH \times KW \times IC \tag{1}$$

Table 3 The Categories of CIFAR-100

Superclass	Classes
aquatic	mammals beaver, dolphin, otter, seal, whale
fish	aquarium fish, flatfish, ray, shark, trout
flowers	orchids, poppies, roses, sunflowers, tulips
food	containers bottles, bowls, cans, cups, plates
fruit and vegetables	apples, mushrooms, oranges, pears, sweet peppers
household electrical devices	clock, computer keyboard, lamp, telephone, television
household	furniture bed, chair, couch, table, wardrobe
insects	bee, beetle, butterfly, caterpillar, cockroach
large	carnivores bear, leopard, lion, tiger, wolf
large man-made outdoor things	bridge, castle, house, road, skyscraper
large natural outdoor scenes	cloud, forest, mountain, plain, sea
large omnivores and herbivores	large omnivores and herbivores
medium-sized mammals	fox, porcupine, possum, raccoon, skunk
non-insect invertebrates	crab, lobster, snail, spider, worm
people	baby, boy, girl, man, woman
reptiles	crocodile, dinosaur, lizard, snake, turtle
small mammals	hamster, mouse, rabbit, shrew, squirrel
trees	maple, oak, palm, pine, willow
vehicles 1	bicycle, bus, motorcycle, pickup truck, train
vehicles 2	lawn-mower, rocket, streetcar, tank, tractor

3.2.2 Param

The number of parameters is the sum of the parameters in the model and is directly related to the amount of disk space required by the model. For embedded devices, limited Flash cannot deploy models that require large disk space. Therefore, the number of parameters is also a valuable comparison for model compression techniques, although the number of parameters does not directly affect the inference performance of the model.

3.2.3 AccTop-1

AccTop-1% represents the proportion of the class corresponding to the highest probability of the model prediction results that agrees with the true label. It is a commonly used evaluation metric for classification models, and it is also an indicator to measure the feature extraction ability of the backbone network of object detection models. It is calculated as Eqs. (2), the value of AccTop-1% ranges from 0 to 1, with higher values indicating better model performance. To some extent, is equivalent to accuracy, but AccTop-1% and AccTop-5% appear together to distinguish the performance of the model on different classes when there is a severe imbalance between the classes. AccTop-5% is the ratio of the top 5 classes that contain the true class in the final output probability vector.

$$AccTop - 1\% = (TP + TN)/(P + N) \tag{2}$$

3.2.4 AP

Average Precision (AP) is one of the commonly used metrics to evaluate the performance of object detection and image classification tasks. AP is used to measure the balance between precision and recall of the model on different categories. The value of AP ranges from 0 to 1, with higher values indicating better performance of the model. When calculating AP, it is also necessary to balance the categories according to the least categories in the dataset to avoid that some categories dominate the dataset and affect the overall evaluation results. The value of AP is equal to the area under the Precision-Recall (P-R) curve, which is calculated using the Eqs. (3):

$$AP = \int_0^1 p(r)dr \tag{3}$$

4 Performance comparison

In this section, we comprehensively summarize and compare the actual compression effects of various typical DNN compression methods on the COCO test-dev2015, VOC 2007, ImageNet ILSVRC 2012, CIFAR10, and CIFAR100 public datasets. The performance of a model is affected by many factors such as the size and scale of the input image, the feature extractor, the GPU architecture and performance, the number of proposals, the training method, the loss function, etc. Moreover, to reproduce the inevitable errors that may arise from experiments, we assume that the results published in the original paper represent the best experimental results obtained by the original authors, so as to make a fair performance comparison.

4.1 Performance comparison among lightweight structure design methods

In this section, we summarize the performance of some classic lightweight structure design models and original object detection model methods on ImageNet ILSVRC 2012 dataset, COCO test-dev2015 dataset, VOC 2007 dataset, and CIFAR-10 dataset. The evaluation indices include: model parameter size: Param; the accuracy of the first category consistent with the actual result: AccTop-1; the mean average precision: mAP; the average precision: AP; and the operation speed of the model: FLOPs.

The comparison methods include: VGG-16 [276], SSD [273], YOLOv2 [145], YOLOv3 [146], YOLOv4 [147], PeleeNet [132], ThunderNet [121], EfficientDet [157], GhostNet [124], MobileNetV1 [107], MobileNetV2 [108], MobileNetV3 [136], ESPNetV2 [111], ShuffleNetV1 [119], ShuffleNetV2 [120], Xception [106], AlexNet [274], GoogleNet [4], ResNet-153 [112], ResNeXt-101 [113], DenseNet [114], InceptionV1 [103], InceptionV2 [104], InceptionV3 [105], Inception-v4 [275], SqueezeNet [125], Squeezenext [126], SqueezeDet [127], CondenseNet [115], ChannelNets [128], NASNet-A [161], IGCV1 [116], IGCV2 [117], IGCV3 [118], EfficientNet [156], MixNets [110], ShiftNet [139], MobileNeXt [133], YOLO Nano [149], YOLO-LITE [148], VarGNetV1 [278], HRNetV1 [131], Lite-HRNet [130], MobileDets + MobileNetV2 SSDLite [180], YOLObile [153], RSANet [134], BN-NAS [184], and PP-PicoDet [109].Some method names are

abbreviations adapted from the original author's title, specifically in the references. The results are presented in Table 4 and Figs. 23, 24 and 25.

On the COCO test-dev 2015 dataset, it is clear that as a lightweight object detection model, YOLO-LITE has a very small param and FLOPs, but at the same time its AP drop is also very obvious. The key role of YOLO-LITE is to bring light model design implications, namely the ability of shallow networks for non-GPU real-time object detection applications, and batch normalization is not necessary for shallow networks. However, in general, the experimental effect of PP-PicoDet model is higher than that of all other methods. The model size is only 0.99 M, and the AP can reach 27.1%. The proposed method is more stable than other lightweight structures by enhancing the backbone structure and combining the label assignment strategy. On ImageNet ILSVRC 2012, most of the lightweight models since MobileNetV1-V3 have achieved amazing performance with a good balance between parameters and accuracy. On the CIFAR-10 dataset, Ghost-ResNet-56 can reduce the parameter size to 0.43 m while maintaining 92.7% accuracy. It can be seen that the combination of GhostNet [124] and ResNet can achieve excellent model compression

Table 4 Compression performance comparison among the lightweight model design methods on COCO test-dev 2015 dataset

Models	Param(M)	AP	FLOPs(B)
SSD	34.3	25.1	99.5
YOLOv2	50.7	21.6	17.5
YOLOv3	62.3	33	71
YOLOv4	64.36	38.2	35.5
YOLOv2-Tiny	-	23.7	5.41
YOLOv3-Tiny	8.86	16.6	5.62
YOLOv4-Tiny	6.06	21.7	6.96
YOLO-LITE	0.6	12.26	1
PeleeNet	5.98	22.4	1.39
ThunderNet	-	23.7	0.47
EfficientDet	3.9	33.8	2.5
IGCV3	4	22.2	0.21
GhostNet	-	26.6	0.164
MobileNetV1 SSDLite	5.1	22.2	1.3
MobileNetV2 SSDLite	4.3	22.1	0.8
MobileNetV2	-	26.7	0.3
MobileNetV3	-	26.4	0.22
ESPNetv2	-	21.9	0.7
ShuffleNetV1	-	20.8	0.14
ShuffleNetV2	-	22.7	0.15
Xception	-	23	0.15
HRNetV1	28.5	25.4	1.6
Lite-HRNet	1.1	33.1	0.45
MobileDets + MobileNetV2 SSDLite	3.17	21.8	-
YOLObile	4.59	31.6	3.95
RSANet	4.348	23.7	2.34
BN-NAS	-	33.32	2.7
PP-PicoDet	0.99	27.1	0.73

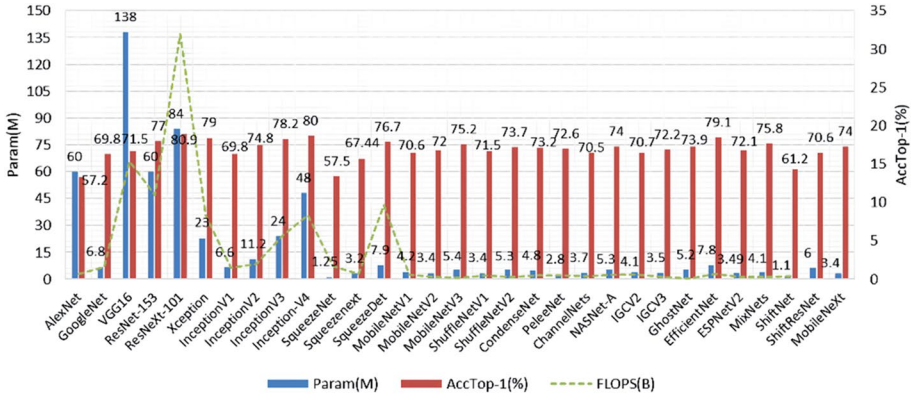


Fig. 23 Compression performance comparison among the lightweight model design methods on ImageNet ILSVRC 2012

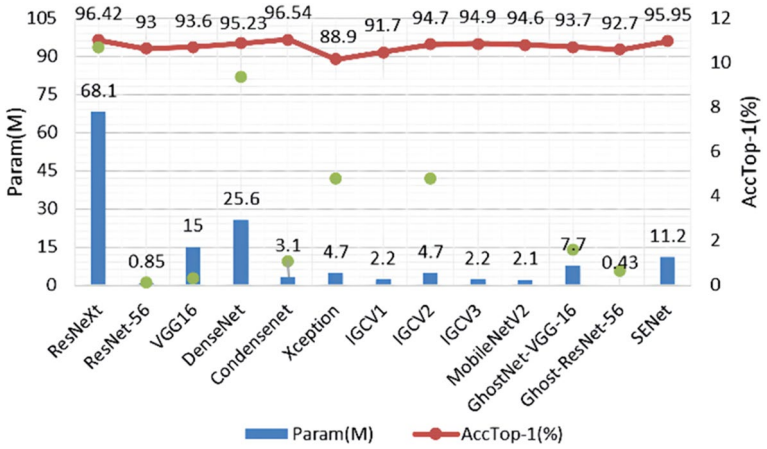


Fig. 24 Compression performance comparison among the lightweight model design methods on CIFAR-10 dataset

effects and build a multi-channel plug-and-play lightweight network structure. The performance comparison on VOC 2007 dataset is mainly for the lightweight object detection model and the original object detection model. It is surprising that most of the lightweight object detection models have reached the same level of accuracy as the original model. This is the goal that the cutting-edge research on compressed object detection models is still pursuing.

4.2 Performance comparison among NAS methods

In this section, we summarize the performance of some classic NAS methods on ImageNet ILSVRC 2012 dataset and CIFAR-10 dataset. The evaluation indices include: model parameter size: Param/M; the accuracy of the first category consistent with the actual

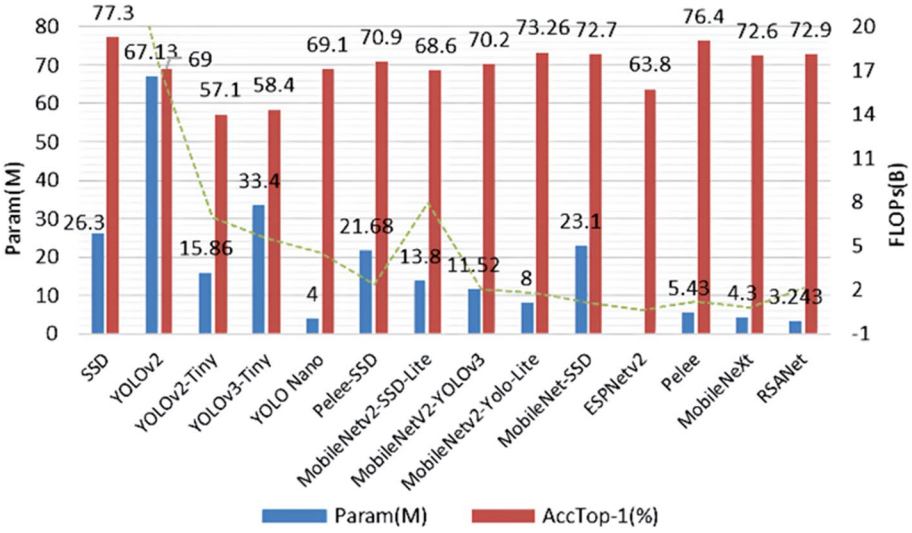


Fig. 25 Compression performance comparison among the lightweight model design methods on VOC 2007 dataset

result: $Acc_{Top\text{-}1}$%; the operation speed of the model: FLOPs; and search time: GPU days (the type of GPU is NVIDIA GTX 1080Ti). The results are shown in Tables 5 and 6.

The comparison methods include: ResNet152 [112], EfficientNetB0 [156], ResNeXt-101 [113], MnasNet [137], ProxylessNAS [181], DF [187], FBNetV1 [173], FBNetV2-L2 [174], FBNetV3-A [175], BNAS [226], SinglePathNAS [171], OFA [188], APQ [189], CP-NAS [192], PNASNET [177], NASNet [161], AmoebaNet [169], Baker [160], Zoph [161], EAS [164], Real [169], Zhong [162], Path-level NAS [163], Liu [170], DARTS [176], ENAS [179], Elsken [183], AutoFormer [185], BN-NAS [184], and LightTrack [193]. Some method names are abbreviations adapted from the original author's title, specifically in the references.

As shown in Tables 5 and 6, the best size of the NAS method can be compressed to 2.5 M, but the accuracy of this model is only 69.8%. Regarding the overall effect, the CP-NAS method has the best effect on CIFAR10 dataset, which has a size of 2.9 M and an accuracy of 96.35%. And its search time is relatively short. It can be seen that the CP-NAS method achieves high compression ratio and accuracy by using the full precision model (parent model) to guide the search of the binary model (child model). In addition, the accuracy of AutoFormer method is 99.1%. It can be seen that CPNAS approach achieves high compression and accuracy by using full precision model (parent model) to guide the search of the binary model (child model). AutoFormer methods of obtaining high-quality transformers from trained supernetworks are superior in accuracy.

4.3 Performance comparison among network quantization methods

In this section, we summarize some of the classic network quantization methods. We quantify ResNet-18 and ResNet34 on the ImageNet ILSVRC 2012 dataset and quantify VGG-Small and ResNet20 on the CIFAR10 dataset. The evaluation indices include: the number

Table 5 Comparison of compression performance among NAS methods on ImageNet ILSVRC 2012 dataset

Mothods	Param(M)	$Acc_{Top\text{-}1}$ (%)	FLOPs(M)
ResNet152	-	78.3	11,000
EfficientNetB0	5.3	77.3	390
ResNeXt-101	-	79.3	7800
ResNet-18	11.17	69.3	-
MnasNet	5.2	76.7	403
ProxylessNAS	4.1	74.6	320
DF	2.5	69.8	746
FBNetV1	4.5	74.9	375
FBNetV2	4.3	78.1	423
FBNetV3	3.7	78	343
DARTS	4.9	74.1	595
BNAS	6.2	71.3	-
SinglePathNAS	4.4	74.7	328
OFA	-	80	595
APQ	-	75.1	-
CP-NAS	12.5	64.3	-
PNASNET	5.1	74.2	588
NASNet	5.3	74	564
AmoebaNet	5.1	74.5	555
BN-NAS	5.4	75.67	470
LightTrack	-	77.6	-
AutoFormer	5.7	74.7	1300

Table 6 Comparison of compression performance among NAS methods on CIFAR10 dataset

Methods	Param(M)	$Acc_{Top\text{-}1}$(%)	GPU(Days)
Baker	11.18	93.08	100
Zoph	37.4	96.35	1800
EAS	23.4	95.77	10
NASNet	3.3	96.59	2000
Real	5.4	94.6	2600
Zhong	39.8	96.46	96
Path-level NAS	5.7	97.01	200
Liu	15.7	96.25	300
DARTS	3.1	97.06	1.5
BNAS	3.3	97.16	0.08
CP-NAS	2.9	96.35	0.1
NASNet-A	3.3	93.5	-
ENAS	4.6	97.35	0.5
PNASNET	3.2	96.46	225
AmoebaNet	3.2	96.59	3150
Elsken	19.7	96.66	1
AutoFormer	23	99.1	-

of model weight quantization bits: W-bit; the number of activation quantization bits: A-bit; model parameter size: Param; the accuracy of the first category consistent with the actual results: $Acc_{Top\text{-}1}$%. The results are shown in Tables 7 and 8.

The methods of comparison include: BinaryConnect [194], BNN [195], XNOR-Net [196], BWN [196], LAB [198], CI-BCNN [204], BNN-DL [206], TWN [208], LQ-Nets [215], TSQ [217], IR-Net [218], TTQ [207], Bi-Real [199], BWNH [200], BENN [203], CBCN [205], DoReFa-Net [213], IAO [214], SYQ [216], BATS [225], DAQ [222], PQMP

Table 7 Compression performance of various network quantization methods for ResNet-18 and ResNet34 on the ImageNet ILSVRC 2012 dataset

		ResNet-18		ResNet34	
Methods	W/A	AccTop-1(%)	Param(M)	AccTop-1(%)	Param(M)
Fullprecision	32/32	**69.6**	374	**73.3**	697 M
BNN	1/1	42.2	-	-	-
XNOR-Net	1/1	51.2	33.7	-	43.9
BWN	1/32	60.8	11.2	60.8	-
ABC-Net	1/32	62.8	11.2	-	-
	2/32	63.7	-	-	-
	1/1	42.7	11.2	52.4	21.8
Bi-Real	1/1	56.4	33.6	62.2	43.7
BWNH	1/32	64.3	11.2	-	-
BENN	1/1	61	-	-	-
CI-BCNN	1/1	59.9	33.5	64.93	43.5
CBCN	1/1	61.4	-	-	-
TWN	2/32	61.8	-	-	-
DoReFa-Net	1/4	**69.9**	-	-	-
	1/2	53.4	-	-	-
	2/2	62.6	-	-	-
IAO	8/8	74.9	-	-	-
LQ-Nets	1/2	62.6	-	66.6	-
	2/2	64.9	-	69.8	-
	3/3	68.2	-	71.9	-
SYQ	1/2	55.4	-	-	-
	1/8	62.9	-	-	-
IR-Net	1/32	66.5	-	70.4	21.8
	1/1	58.1	4.21	62.9	21.8
BATS	1/1	60.4	-	-	-
TTQ	2/32	66.6	-	-	-
DAQ	1/1	56.2	-	62.1	-
	1/2	64.6	-	69.4	-
	1/32	67.2	-	71.9	-
	2/32	69.8	-	-	-
PQMP	4/8	61.68	5.37	-	-
	4/4	65.89	5.41	-	-

Table 8 Compression performance of various network quantization methods for VGG-Small and ResNet20 on CIFAR10 dateset

Methods	W/A	VGG-SmalL AccTop-1(%)	ResNet20
Fullprecision	32/32	93.8	**92.1**
BinaryConnect	1/32	91.73	-
BNN	1/1	89.9	-
XNOR-Net	1/1	89.8	-
BWN	1/32	90.1	-
LAB	1/32	89.5	-
	1/1	87.7	-
CI-BCNN	1/1	92.5	91.1
BNN-DL	1/1	90.0	-
TWN	-	92.56	-
LQ-Nets	1/2	93.4	88.4
TSQ	3/2	93.5	-
IR-Net	1/1	90.4	85.4
TTQ [207]	2/32	-	91.13
DoReFa-Net	1/1	-	79.3
	1/32	-	90
DAQ	1/1	-	85.8
	1/32	-	91.2
MPT	1/1	91.9	-
	1/32	91.48	94.8

[221], and MPT [201]. Some method names are abbreviations adapted from the original author's title, specifically in the references.

The "W/A" in the Table 7 and 8 refers to the ratio of the total amount of parameter weights to the total amount of activation values in the model. A low "W/A" value means that the computational cost of the model is relatively low, that is, the model pays more attention to the calculation of activation values rather than the calculation of parameter weights in the calculation process. This is exactly what is pursued in network quantization. Based on this, we analyze various models obtained by compressing ResNet-18 and ResNet34 using network quantization methods on the ImageNet ILSVRC 2012 dataset. Compared with ResNet-18 "Fullprecision", the DoReFa-Net method has no loss of accuracy and provides an improvement after weighting, and the accuracies of IAO and DAQ are higher than before. The DoReFa-Net method uses low-bit wide gradients to train low-bit wide CNNs, and its accuracy is significantly higher than the BWN and TWN methods. For the quantization of ResNet34, there are few studies, but most of AccTop-1 also reach the level with "Fullprecision".

4.4 Performance comparison among KD methods

In this section, we summarize the performance of some classic KD methods on ImageNet ILSVRC 2012 dataset, CIFAR-10 dataset and CIFAR-100 dataset, using WideResNet (WRN), ResNet (RN) and other models as teacher and student models. The evaluation indices include: the accuracy of the first category consistent with the actual results:

AccTop-1%. (The size of the method model after KD is consistent with the student network). The results are shown in Tables 9, 10 and 11.

The methods of comparison include: KD [229], AT [234], OFD [1], RKD [242], CRD [244], AIN [237], FSP [238], FT [231], and SP [236]. Some method names are abbreviations adapted from the original author's title, specifically in the references.

As shown in Tables 9, 10 and 11, We summarize the student networks and teacher networks obtained with different architectures and the same architecture. In general, the general accuracy of the small student model obtained by the KD method is not much lower than that of the teacher network, and the accuracy of the AIN method only drops by 0.15%, which method is based on feature statistics can better transfer statistical information from teachers to students with less loss of accuracy.

4.5 Performance comparison among low-rank decomposition methods

In this section, we summarize some of the current classic low-rank decomposition methods. On the ImageNet, CIFAR10, CIFAR100 datasets, we use AlexNet, VGG-16, ResNet-32, and GoogLeNet as the original models to perform low-rank decomposition, and compare them with some pruning algorithms. The evaluation indices include: model parameter size: Param/M; the accuracy of the first category consistent with the actual result: $\text{Acc}_{\text{Top-1}}\%$; the accuracy of the fifth category consistent with the actual result: $\text{Acc}_{\text{Top-5}}\%$; and the operation speed of the model: FLOPS ($\times 10^9$). The results are shown in Tables 12, 13, 14 and 15.

The comparison methods include: Tai [262], Tucker [258], GreBdec [263], SVD [256], Guo [27], Han [14], neuron importance score propagation (NISP) [28], AFP [51], TT [260], and TRN [266]. Some method names are abbreviations adapted from the original author's title, specifically in the references.

Among the low-rank decomposition methods for VGG-16 and AlexNet [274] in Tables 12 and 13, the GreBdec method performs very well, compressing the size to 7.0% and 9.8% of the original size respectively, while the accuracy is slightly improved. The GreBdec method uses feature reconstruction and combines low-rank decomposition and sparsity to achieve better results than the traditional Tucker and Tai methods. Table 14 compares the compression effects of some low-rank decomposition methods of GoogLeNet.

Table 9 Compression performance among various KD methods on ImageNet dataset

Student (Param(M))	RN18 (11.69)	MobileNet (4.23)	RN18 (13.95)
Teacher (Param(M))	RN34 (21.80)	RN50 (25.56)	RN101
Student	70.04	70.13	70.83
Teacher	73.31	76.16	78.05
KD	70.68	70.68	71.43
AT	70.59	70.72	71.58
OFD	71.08	71.25	-
RKD	71.34	71.32	-
CRD	71.17	71.4	-
AIN	71.82	72.49	-
FSP	-	-	71.28
HKD + KD	-	69.45	-
DFND	61.75	-	-

Table 10 Compression performance among various KD methods on CIFAR-10 dataset

Student (Param(M))	VGG-13 (9.4)	RN-20 (0.27)	RN-20 (0.27)	NIN-thin (0.2)	WRN-16–1 (0.2)	WRN-16–1 (0.2)	WRN-16–2 (0.7)	WRN-16–2 (0.7)	WRN-40–2 (2.2)	RN-20 (0.27)	RN18 (11.69)
Teacher (Param(M))	WRN-46–4 (10)	WRN-40–1 (0.56)	RN-56 (0.85)	NIN-wide (1)	WRN-40–1 (0.6)	WRN-16–2 (0.7)	WRN-40–2 (2.2)	WRN-16–8 (11.0)	WRN-16–8 (11.0)	RN-164 (2.6)	RN34 (21.80)
Student	94.01	92.22	92.22	90.62	91.26	91.25	93.93	93.93	94.82	91.42	93.92
Teacher	95.56	93.16	93.61	92.72	93.49	93.79	**94.82**	95.76	95.76	93.43	94.85
AT	94.46	92.66	92.87	91.07	91.7	91.89	94.11	94.53	95.53	-	-
KD	94.29	92.91	92.81	91.45	91.52	92.32	94	94.38	95.14	91.12	94.34
FT	95.16	93.15	93.15	-	-	-	-	-	-	-	-
AT + KD	94.7	93	93.11	91.67	91.99	92.48	94.29	-	-	-	-
FT + KD	95.35	93.05	92.96	-	-	92.41	-	-	-	-	-
SP	-	-	-	-	91.87	92.48	94.48	94.66	95.45	-	-
OFD	-	-	-	-	-	92.28	94.3	-	-	-	-
RKD	-	-	-	-	-	92.51	94.41	-	-	-	-
AIN	-	-	-	-	-	92.9	94.67	-	-	-	-
DFND	-	-	-	-	-	-	-	-	-	-	94.02

Table 11 Compression performance among various KD methods on CIFAR-10 dataset

Student	WRN-16–2 (0.70 M)	WRN-16–4 (2.77 M)	WRN-10–10 (7.49 M)	RN-10 (0.34 M)	RN-18 (0.75 M)	RN-10 (4.95 M)	WRN-16–2 (0.70 M)	MobileNetV2 (2.37 M)	MobileNetV2 (2.37 M)
Teacher	WRN-40–4 (8.97 M)	WRN-40–4 (8.97 M)	WRN-16–10 (17.2 M)	RN-34 (1.39 M)	RN-50 (1.99 M)	RN-34 (21.33 M)	RN-34 (21.33 M)	RN-34 (21.33 M)	WRN-40–4 (8.97 M)
Student	72.22	75.68	73.75	67.6	70.21	74.2	72.22	68.36	68.36
Teacher	78.31	78.31	78.75	72.05	72.83	77.26	77.26	77.26	78.31
KD	73.32	77.17	76.63	68.05	72.4	75.79	73.32	68.65	68.4
AT	72.56	76.48	74.63	68.39	70.78	75.03	71.46	66.68	66.46
OFD	74.12	77.79	76.97	67.94	71.54	75.6	73.78	70.18	70.1
RKD	73.08	77.48	76.28	68.43	72.1	75.04	72.95	69.52	68.64
CRD	73.75	77.49	76.28	69.17	72.31	76.63	73.39	70.98	70.45
AIN	74.58	78.25	77.12	69.49	72.59	76.11	74.38	70.66	71.15

Table 12 Compressing VGG-16 by using low-rank decomposition methods on ImageNet dataset

Methods	Param	$Acc_{Top\text{-}1}$/%	$Acc_{Top\text{-}5}$/%	FLOPs
VGG-16	138	68.5	88.68	15.5
Tai	50.18	-	90.31	7.56
Tucker	126.6	-	89.4	3.14
Han	10.3	68.66	89.12	-
GreBdec	9.7	68.75	89.06	-
Han	15	68.15	-	-

Table 13 Compression effects among some low-rank decomposition methods for AlexNet on ImageNet dataset

Methods	Param	$Acc_{Top\text{-}1}$/%	$Acc_{Top\text{-}5}$/%	FLOPs
AlexNet [274]	61	57.22	80.27	0.72
Tai	12.2	-	79.66	0.39
Tucker	10.9	56.6	78.33	0.27
GreBdec	6	57.26	80.31	-
Denton	12.2	55.98	-	-
Guo	3.47	56.91	-	-
Han	6.8	57.23	80.33	-
NISP	-	55.87	79.86	0.31
AFP	-	56.17	79.53	0.15

Table 14 Compressing GoogLeNet by using low-rank decomposition methods on ImageNet dataset

Methods	Param	$Acc_{Top\text{-}1}$/%	$Acc_{Top\text{-}5}$/%	FLOPs
GoogLeNet	6.9	68.7	88.9	-
Tai	2.4	-	91.79	0.39
Tucker	4.7	-	88.66	0.27
GreBdec	1.5	67.3	88.11	-

Table 15 Compressing ResNet-32 by using low-rank decomposition methods on CIFAR10, CIFAR100 dataset

Method	Params	CIFAR10 $Acc_{Top\text{-}1}$/%	CIFAR100
ResNet-32	0.46	92.5	68.1
Tucker	0.09	87.7	57.8
TT	0.096	88.3	62.9
TRN	0.09	90.6	48.7

Although the accuracy can reach the same level as that of GoogLeNet, the compression ratio is also very limited due to the small size of GoogLeNet. Table 15 compares the compression effects of some low-rank decomposition methods of ResNet-32 on CIFAR10 and CIFAR100 datasets, and it can be seen that the TT method has the best comprehensive performance on the two datasets.

Table 16 Pruning AlexNet by using various pruning methods on ImageNet dataset

Methods	Param(M)	$Acc_{Top\text{-}1}(\%)$	FLOPs(G)
AlexNet [274]	61	57.22	0.72
Han(2015b)	6.9	57.22	-
Han(2015a)	27	57.23	-
Zhang	2.9	-	-
Liu	0.3	57.48	-
Guo	3.4	56.91	-
PerforatedCNNs	30.5	57.22	0.28
GDP	61	55.44	0.26
AOFP	-	56.17	0.49
Dong	6.8	57.23	-

Table 17 Pruning VGG-16 by using various pruning methods on ImageNet dataset

Methods	Param(M)	$Acc_{Top\text{-}1}(\%)$	FLOPSs(B)
VGG-16	138	68.5	15.5
Han(2015b)	10.6	68.66	-
Han(2015a)	11.3	68.83	-
Hu	9.2	64.78	4.4
PerforatedCNNs	57.5	-	5.54
PFEC	-	53.9	12.4
ThiNet	8.32	67.34	9.34
CP	-	66.8	12.4
RNP	-	64.92	12.4
GDP	138	65.69	3.79
AOFP	2.04	53.7	-
Slimming	-	42.6	7.75
DCP	-	65.27	7.75
AMC	-	67.1	12.4
CC	8.35	68.81	7.37
OPQ	7.7	71.39	-

4.6 Performance comparison among network pruning methods

In this section, we summarize some of the current network pruning methods. On the ImageNet, CIFAR10, CIFAR100 datasets, we use AlexNet [274], VGG-16 [276], ResNet-50, ResNet-56, and GoogLeNet as the original models to perform network pruning. The evaluation indices include: model parameter size: Param/M; the accuracy of the first category consistent with the actual result: $Acc_{Top\text{-}1}\%$; and the operation speed of the model: FLOPs ($\times 10^9$). The results are shown in Tables 16, 17, 18, 19 and 20 and Figs. 26, 27 and 28.

The comparison methods include: Han(2015b) [14], Han(2015a) [13], Zhang [17], Liu [21], Guo [27], PerforatedCNNs [73], AOFP [63], Dong [93], PFEC [40], ThiNet [69], CP [42], RNP [50], GDP [53], Slimming [76], DCP [80], AMC [165], Lin [58], NISP [28], SSS [29], Taylor [30], SFP [64], KSE [65], GAL [66], C-SGD [52], GBN [71], CURL [59], Hu [79], MDP [88], DMC [3], MobileNet [107], Slimmable NN [182], TAS [186], Hinge [270], MetaPruning [167], Ghost-ResNet-50 [124], Versatile [123], AFP [51], PP

Table 18 Pruning ResNet-50 by using various pruning methods on CIFAR10 dataset

Methods	Param(M)	$Acc_{Top\text{-}1}(\%)$	FLOPs(G)
ResNet-50	25.56	72.88	7.72
Han(2015a)	22.26	72.87	-
ThiNet	8.66	68.42	2.2
CP	-	73.3	2.0
Lin	5.89	72.26	
SFP	-	75.1	2.9
GDP	-	68.67	1.35
DCP	12.38	71.82	3.41
MetaPruning	-	76.2	3.0

Table 19 Pruning ResNet-50 by using various pruning methods on ImageNet dataset

Methods	Param(M)	$_{AccTop\text{-}1(\%)}$	FLOPs(B)
ResNet-50	25.5	76.15	4.09
SFP	-	74.61	2.38
GDP	-	71.19	1.88
C-SGD	-	75.54	1.81
AOFP	-	75.63	2.58
He et al	-	72.3	2.73
DMC	-	75.35	1.84
TAS	-	76.2	2.31
Hinge	-	74.7	1.9
CCP	-	75.21	1.88
FPGM	-	74.83	1.94
MetaPruning	-	69.92	1.03
SRR-GR	-	75.76	2.29
Group fisher	-	76.42	2.04
Greg	3.26	76.27	-
GSM	2.1	74.3	-
RigL	-	74.6	-
ResRep	-	76.15	1.86
DPFPS	-	75.55	2.2
OPQ	6.59	76.41	-
Han et al	-	76.15	-
SBP-SR-0.8	-	69.75	-

[67], Wang [84], CCP [241], CLFI [87], CC [85], OPQ [100], NPPM [89], SRR-GR [55], HFP [68], ACTD [19], Greg [32], CLR [98], ResRep [78], LRF [90], DPFPS [57], Group fisher [77], GSM [51], RigL [96], TBC [75], SBP-SR-0.8 [97], LAMP [20], and L-Versatile [135]. Some method names are abbreviations adapted from the original author's title, specifically in the references.

At present, the pruning effects of the network pruning methods are relatively better. Many methods exhibit precision losses of less than 0.5% and the model sizes can be

Table 20 Pruning ResNet-56 by using various pruning methods on CIFAR10 dataset

Methods	Param(M)	$Acc_{Top\text{-}1}(\%)$	FLOPs(G)
ResNet-56	0.85	93.57	0.13
NISP-56	0.49	93.01	0.17
PFEC	0.73	93.06	0.094
ThiNet	0.43	92.98	0.065
CP	0.425	91.8	0.065
SFP	-	93.35	0.062
AFP-G	-	92.94	0.051
KSE	0.38	93.23	0.0624
GAL	0.75	90.36	0.049
PP	-	93.09	0.041
C-SGD	-	93.44	0.06085
GBN	0.39	93.34	0.052
Slimming	0.425	78.74	0.065
He et al	-	90.8	0.062
DCP	0.43	93.59	0.065
Zhao	0.58	92.26	0.1
SCP	-	93.23	0.0515
DMC	-	93.69	0.065
WM MobileNet	0.43	93.24	0.065
AMC	0.425	91.9	0.065
TAS	-	93.69	0.059
Hinge	0.41	93.69	0.065
CCP	-	93.42	0.062
FPGM	-	93.49	0.062
CLFI	-	94.7	0.045
NPPM	-	93.4	0.065
SRR-GR	-	93.75	0.06
CC	0.44	93.64	0.062
HFP	0.425	93.57	0.057
ACTD	-	93.76	0.062
Greg	-	93.18	-
CLR	-	92.41	-
ResRep	-	93.57	0.061
LRF	0.31	93.73	0.049
DPFPS	0.452	93.2	0.61

reduced by more than 90%, such as the AOFP, SCP, and Lin methods. The model size of OPQ method is 7.7 M, and the accuracy reaches 71.39%, which is 2.89% higher than the original model. In addition, CC method also performs very well, the model size of CC method is 8.35 M, and the accuracy reaches 68.81%, and its FLOPs are reduced by more than 50%. It can be seen that the overall effect of combining multiple compression methods is better, such as CC method, which combines channel pruning and tensor decomposition, and OPQ method, which combines pruning and quantization.

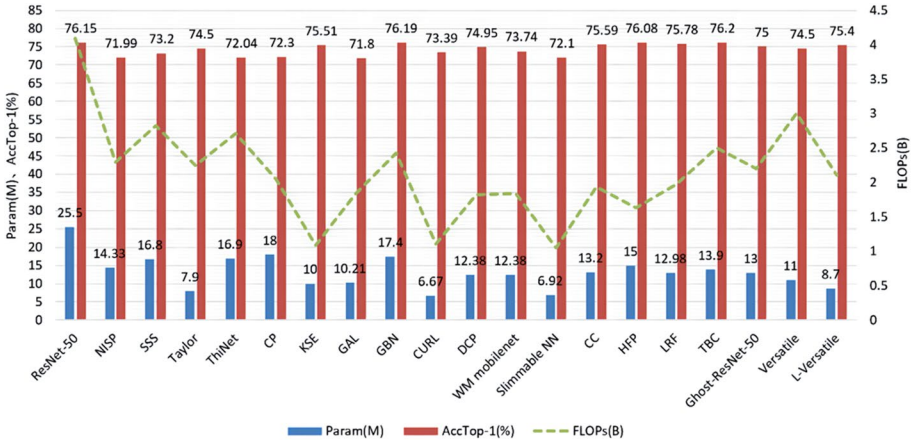


Fig. 26 Pruning ResNet-50 by using various pruning methods on ImageNet dataset

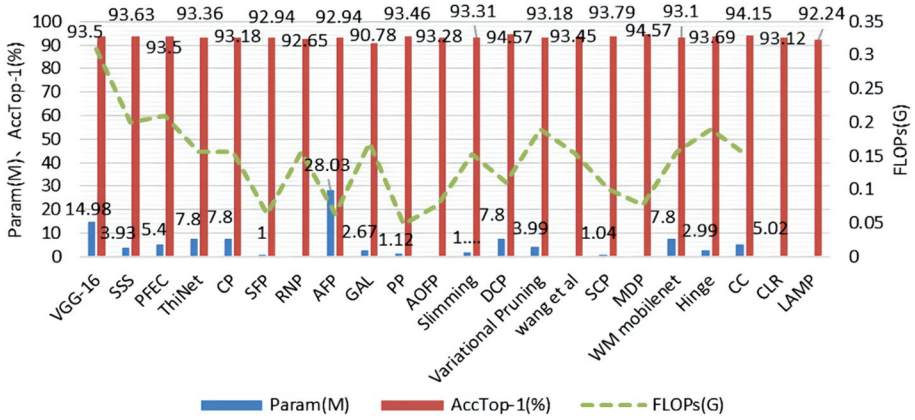


Fig. 27 Pruning VGG-16 by using various pruning methods on CIFAR10 dataset

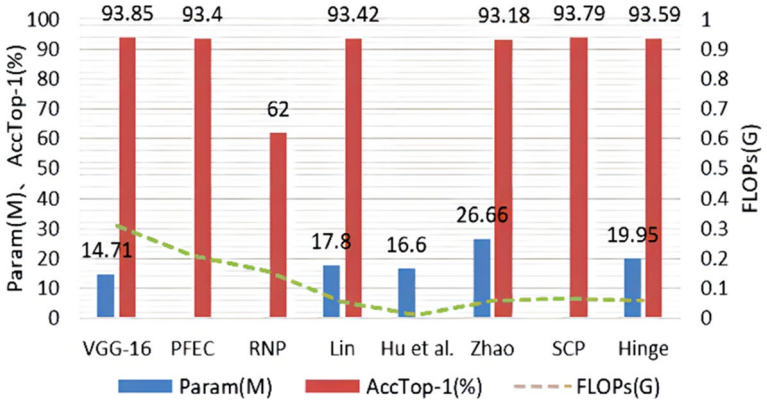


Fig. 28 Pruning VGG-16 by using various pruning methods on CIFAR100 dataset

5 Discussion and future directions

In this section, the impact of different model compression strategies on the model is analyzed, and the application scenarios and future development directions of different strategy adaptations are discussed.

Despite the advantages and limitations of different model compression techniques, they share a common objective of reducing model parameters or computational complexity to accelerate inference speed. However, most model compression techniques lead to a decline in model accuracy, highlighting the crucial balance between compression and accuracy. The effect of network pruning on accuracy depends on pruning strategies and levels, as excessive pruning inevitably results in a significant drop in accuracy. The accuracy of lightweight architecture design relies on specific design choices and requires extensive expert experience. Network quantization has made notable progress, such as post-training quantization and quantization-aware training, which mitigate the impact on accuracy and enable compressed models to approach the accuracy level of the original models. NAS falls within the domain of automated machine learning and can yield models that achieve a balance between accuracy and compression ratio in certain cases, although the search process is often time-consuming. KD generally achieves comparable accuracy to that of the teacher model, necessitating the selection of an appropriate teacher model along with proper data augmentation and model complexity control. Low-rank decomposition requires careful selection of decomposition strategies to mitigate the impact and maintain a reasonable level of accuracy in the model.

It is worth noting that the exact impact of these techniques may vary depending on specific implementations, datasets, and application scenarios. Based on the characteristics of models compressed by each technique, we discuss the future development directions and application scenarios of the six compression methods in Table 21. Models pruned through network pruning exhibit fewer parameters, making them suitable for devices with low computational memory and storage capacity. Lightweight architecture design is applicable to real-time end-to-end embedded platforms with high time constraints. Network quantization is more suitable for devices with high inference speed and low computational memory requirements. NAS is suitable for mobile devices with specific storage and accuracy requirements. KD performs well in scenarios with small or no datasets. Low-rank decomposition, due to its computationally expensive decomposition operations and the need for extensive model training to achieve convergence, is primarily used for compressing fully connected layers.

6 Conclusion

The development of lightweight models makes neural networks smaller and more efficient, and makes it possible to apply them in a wider range of fields. On the one hand, lightweight neural networks help with the application of deep learning systems in virtual reality, augmented reality, smart security and smart wearable devices; on the other hand, the real-time performance of lightweight neural networks is of great significance for online learning and automatic driving technology.

In this paper, we first introduced the research background of neural network compression technology. Second, we introduced the current compression strategies and the corresponding research work in detail, and then compared their compression effects. Finally,

Table 21 Discussion and future directions

Methods	Future directions	Application scenarios
Network Pruning	Combine network pruning with other compression methods to achieve further compression	Devices with low computing memory sizes and storage capacities
Lightweight Structure Design	1. Consider the combination of lightweight structure design and network architecture search 2. Explore more application tasks and scenarios	End-to-end embedded platforms with high real-time requirements
Network Quantization	1. Improve the expression ability of the network and tap the training potential of the network 2. Increase the acceleration research on actual hardware devices	Devices with high inference speeds and low computational memory sizes
NAS	1. Apply neural network search methods to semantic segmentation and other tasks 2. Increase the research on search spaces	Mobile devices with specific storage and accuracy requirements
KD	1. Expand to other fields such as object detection 2. Combine the intermediate feature layer and use different forms of knowledge	Small or no datasets
Low-Rank Decomposition	Study how to simplify the calculation operations required for decomposition	Fully connected layer compression

Table 22 The advantages and existing problems of six strategies

Methods	Advantages	Existing problems
Network Pruning	Small number of parameters and small loss of precision	Needs to retrain and fine-tune the recovery accuracy after pruning sometimes; may lead to non-uniform sparsity patterns that can impact hardware acceleration
Lightweight Structure Design	Simple to train the network; small network parameters	Designing high-performance models requires strong design skills; weak generalization ability
Network Quantization	Significantly speeds up bit operations; small network parameters	High implementation difficulty and unstable accuracy; needs special compiler and hardware support
NAS	Excellent performance in device adaptation problems; structural hyperparameter tuning instead of manual design	Difficult to design the search space; Computationally expensive and time-consuming; Lack of interpretability
KD	Better training effects on small-scale networks	High requirements for dataset scale and training time without a pretrained teacher model
Low-Rank Decomposition	Applied during and after training; reduction in training time	Requires many calculations and a larger number of retraining steps (because each layer must be readjusted after decomposition)

we summarized the current problems and future research directions of various compression strategies.

As shown in Table 22, we have summarized the highlights and existing problems of the six strategies. It can be seen that, in general, each method has its own unique advantages. If we can combine various methods to develop strengths and avoid weaknesses, we should be able to achieve better results. For example, it can be seen that pruning and lightweight structure design methods are widely used in the field of object detection. NAS method can be used to search for a better lightweight network structure; According to the structural characteristics, combining pruning with KD technology and using the original model to guide the learning of pruning model can further improve the performance. In addition, the use of quantization to complete real-time object detection tasks on hardware devices has reached a relatively mature stage. Supporting model transformation tools and platforms are the research hotspots of major internet companies for intelligent front-end object detection devices. Model training acceleration has been given attention. How to incorporate hardware performance, and co design with hardware for compression is the trend of model compression in object detection.

Through reading this article, readers can attain a more comprehensive and thorough understanding of neural network compression technology. We believe that in the future, lightweight neural network technology will continue to develop and play an important role in the intelligence of our lives.

Acknowledgements This work was supported by the Fundamental Research Funds for Central Universities of the Civil Aviation University of China (Grant No. 3122021088).

Data availability All data analyzed during this study are available in, COCO test-dev2015, VOC 2007, ImageNet ILSVRC 2012, CIFAR10, and CIFAR100, the public datasets. These datasets were derived from the following public domain resources:

http://cocodataset.org;
https://pjreddie.com/projects/pascal-voc-dataset-mirror/;
https://www.image-net.org;
http://www.cs.toronto.edu/~kriz/cifar.html

References

1. Heo B, Kim J, Yun S, Park H, Kwak N, Choi JY (2019) A comprehensive overhaul of feature distillation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision 2019:1921–1930. https://doi.org/10.1109/ICCV.2019.00201
2. Ahn S, Hu SX, Damianou A, Lawrence ND, Dai Z (2019) Variational information distillation for knowledge transfer. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition 2019:9163–9171. https://doi.org/10.1109/CVPR.2019.00938
3. Gao S, Huang F, Pei J, Huang H (2020) Discrete model compression with resource constraint for deep neural networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition 2020:1899–1908. https://doi.org/10.1109/CVPR42600.2020.00197

4. Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D et al (2015) Going deeper with convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015:1–9. https://doi.org/10.1109/CVPR.2015.7298594
5. Feroz MdA, Sultana M, Hasan MdR, Sarker A, Chakraborty P, Choudhury T (2022) Object detection and classification from a real-time video using SSD and YOLO models. In Computational Intelligence in Pattern Recognition, Advances in Intelligent Systems and Computing 1349:37–47. https://doi.org/10.1007/978-981-16-2543-5_4
6. Vulli A, Srinivasu PN, Sashank MSK, Shafi J, Choi J, Ijaz MF (2022) Fine-tuned DenseNet-169 for breast cancer metastasis prediction using FastAI and 1-cycle policy. Sensors (Basel, Switzerland) 22(8):2988. https://doi.org/10.3390/s22082988
7. Basha SM, Ahmed ST, Al-Shammari NK (2022) A study on evaluating the performance of robot motion using gradient generalized artificial potential fields with obstacles. In Computational Intelligence in Data Mining, Systems and Technologies 281:113–125. https://doi.org/10.1007/978-981-16-9447-9_9
8. Ahmed S, Guptha N, Fathima A, Ashwini S (2021) Multi-view feature clustering technique for detection and classification of human actions. In: Proceedings of the First International Conference on Advanced Scientific Innovation in Science, Engineering and Technology, ICASISET. https://doi.org/10.4108/eai.16-5-2020.2304034
9. de Aguiar SA, Barros RC (2021) Model compression in object detection. In: Int Joint Conf Neural Netw (IJCNN) 2021:1–8.10. https://doi.org/10.1109/IJCNN52387.2021.9533792
10. Cheng Y, Wang D, Zhou P, Tao Z (2017) A survey of model compression and acceleration for deep neural networks. arXiv preprint arXiv:1710.09282. https://doi.org/10.48550/arXiv.1710.09282
11. Zou Z, Shi Z, Guo Y, Ye J (2019) Object detection in 20 years: a survey. Proceedings of the IEEE 111:257–276. https://doi.org/10.1109/JPROC.2023.3238524
12. LeCun Y, Denker JS, Solla SA (1990) Optimal brain damage. In: advances in neural information processing systems 2:598–605
13. Han S, Pool J, Tran J, Dally W (2015a) Learning both weights and connections for efficient neural network. Adv Neural Inf Proces Syst 1:1135–1143
14. Han S, Mao H, Dally WJ (2015b). Deep compression: compressing deep neural networks with pruning, trained quantization and huffman coding. In Proceedings of International Conference on Learning Representations, 2016. https://doi.org/10.48550/arXiv.1510.00149
15. Lin Y, Han S, Mao H, Wang Y, Dally WJ (2017) Deep gradient compression: reducing the communication bandwidth for distributed training. In Proceedings of International Conference on Learning Representations, 2018. https://doi.org/10.48550/arXiv.1712.01887
16. Li G, Qian C, Jiang C, Lu X, Tang K (2018) Optimization based layer-wise magnitude-based pruning for DNN compression. IJCAI 330:2383–2389
17. Zhang T, Ye S, Zhang K, Tang J, Wen W, Fardad M, Wang Y (2018) A systematic dnn weight pruning framework using alternating direction method of multipliers. Proceedings of the European Conference on Computer Vision (ECCV) 11212:191–207. https://doi.org/10.1007/978-3-030-01237-3_12
18. Chen C, Tung F, Vedula N, Mori G (2018) Constraint-aware deep neural network compression. Proceedings of the European Conference on Computer Vision (ECCV) 11212:400–415. https://doi.org/10.1007/978-3-030-01237-3_25
19. Wang W, Chen M, Zhao S, Chen L, Hu J, Liu H, Liu W (2021) ACTD Accelerate cnns from three dimensions: a comprehensive pruning framework. International Conference on Machine Learning, vol 2021. PMLR, pp 10717–10726
20. Lee J, Park S, Mo S, Ahn S, Shin J (2020) Layer-adaptive sparsity for the magnitude-based pruning. In International Conference on Learning Representations, 2021. https://doi.org/10.48550/arXiv.2010.07611
21. Liu Z, Xu J, Peng X, Xiong R (2018) Frequency-domain dynamic pruning for convolutional neural networks. Proceedings of the 32nd International Conference on Neural Information Processing Systems. pp 1051–1061
22. Carreira-Perpinán MA, Idelbayev Y (2018) "Learning-compression" algorithms for neural net pruning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2018:8532–8541. https://doi.org/10.1109/CVPR.2018.00890
23. Mallya A, Lazebnik S (2018) Packnet: Adding multiple tasks to a single network by iterative pruning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2018:7765–7773. https://doi.org/10.1109/CVPR.2018.00810
24. Kwon SJ, Lee D, Kim B, Kapoor P, Park B, Wei GY (2020) Structured compression by weight encryption for unstructured pruning and quantization. In Proceedings of the IEEE/CVF Conference

on Computer Vision and Pattern Recognition 2020:1906–1915. https://doi.org/10.1109/CVPR42600.2020.00198
25. Frankle J, Dziugaite GK, Roy DM, Carbin M (2020) Pruning neural networks at initialization: why are we missing the mark? In Proceedings of International Conference on Learning Representations 535:6377–6389. https://doi.org/10.5555/3495724.3496259
26. Hu H, Peng R, Tai YW, Tang CK (2016). Network trimming: a data-driven neuron pruning approach towards efficient deep architectures. arXiv:1607.03250. https://arxiv.org/abs/1607.03250
27. Guo Y, Yao A, Chen Y (2016) Dynamic network surgery for efficient dnns. Advances in Neural Information Processing Systems 29:1387–1395. https://doi.org/10.48550/arXiv.1608.04493
28. Yu R, Li A, Chen CF, Lai JH, Morariu VI, Han X, Davis LS (2018) Nisp: pruning networks using neuron importance score propagation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp 9194–9203. https://doi.org/10.1109/CVPR.2018.00958
29. Huang Z, Wang N (2018) Data-driven sparse structure selection for deep neural networks. In Proceedings of the European conference on computer vision (ECCV) 11220:317–334. https://doi.org/10.1007/978-3-030-01270-0_19
30. Molchanov P, Mallya A, Tyree S, Frosio I, Kautz J (2019) Importance estimation for neural network pruning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition,pp 11264–11272. https://doi.org/10.1109/CVPR.2019.01152
31. Khakzar A, Baselizadeh S, Khanduja S, Rupprecht C, Kim ST, Navab N (2021) Neural response interpretation through the lens of critical pathways. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp 13523-13533. https://doi.org/10.1109/CVPR46437.2021.01332
32. Wang H, Qin C, Zhang Y, Fu Y (2020) Neural pruning via growing regularization. In International Conference on Learning Representations 2021. https://doi.org/10.48550/arXiv.2012.09243
33. Liu N, Ma X, Xu Z, Wang Y, Tang J, Ye J (2020) AutoCompress An automatic DNN structured pruning framework for ultra-high compression rates. In Proceedings of the AAAI Conference on Artificial Intelligence 34(04):4876–4883
34. Ma X, Guo FM, Niu W, Lin X, Tang J, Ma K, Wang Y (2020) Pconv: the missing but desirable sparsity in dnn weight pruning for real-time execution on mobile devices. In Proceedings of the AAAI Conference on Artificial Intelligence 34(04):5117–5124
35. Ao R, Tao Z, Yuhao W, Sheng L, Peiyan D, Yen-kuang C,.et al (2020). DARB: a density-adaptive regular-block pruning for deep neural networks. In Proceedings of the AAAI Conference on Artificial Intelligence 34(04):5495–5502. https://doi.org/10.1609/aaai.v34i04.6000
36. Wen W, Wu C, Wang Y, Chen Y, Li H (2016) Learning structured sparsity in deep neural networks. In Advances in Neural Information Processing Systems, pp 2082–2090
37. Zhou H, Alvarez JM, Porikli F (2016) Less is more: Towards compact cnns. European Conference on Computer Vision, vol 9908. Springer, Cham, pp 662–677. https://doi.org/10.1007/978-3-319-46493-0_40
38. Alvarez JM, Salzmann M (2016) Learning the number of neurons in deep networks. In Advances in Neural Information Processing Systems, pp 2270–2278
39. Yoon J, Hwang SJ (2017) Combined group and exclusive sparsity for deep neural networks. In International Conference on Machine Learning (PMLR) 70:3958–3966
40. Li H, Kadav A, Durdanovic I, Samet H, Graf HP (2017) Pruning filters for efficient convnets. In: ICLR 2017: International conference on learning representations 2017. https://doi.org/10.48550/arXiv.1608.08710
41. Lebedev V, Lempitsky V (2016) Fast convnets using group-wise brain damage. Proc IEEE Conf Comput Vis Pattern Recognit 2016:2554–2564. https://doi.org/10.1109/CVPR.2016.280
42. He Y, Zhang X, Sun J (2017) Channel pruning for accelerating very deep neural networks. In: Proceedings of the IEEE International Conference on Computer Vision 2017:1398–1406. https://doi.org/10.1109/ICCV.2017.155
43. Kingma DP, Salimans T, Welling M (2015) Variational dropout and the local reparameterization trick. In Advances in Neural Information Processing Systems (NIPS) 28:2575–2583
44. Louizos C, Welling M, Kingma DP (2017) Learning sparse neural networks through L0 regularization. arXiv:1712.01312. https://doi.org/10.48550/arXiv.1712.01312
45. Molchanov D, Ashukha A, Vetrov D (2017) Variational dropout sparsifies deep neural networks. In International Conference on Machine Learning (PMLR) 70:2498–2507
46. Neklyudov K, Molchanov D, Ashukha A, Vetrov D (2017) Structured bayesian pruning via log-normal multiplicative noise. In Advances in Neural Information Processing Systems, pp 6778–6787

47. Lemaire C, Achkar A, Jodoin PM (2019) Structured pruning of neural networks with budget-aware regularization. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition 2019:9108–9116. https://doi.org/10.1109/CVPR.2019.00932
48. Louizos C, Ullrich K, Welling M (2017) Bayesian compression for deep learning. In: Proceedings of the 31st International Conference on Neural Information Processing Systems 30:3290-3300. https://doi.org/10.48550/arXiv.1705.08665
49. Molchanov P, Tyree S, Karras T, Aila T, Kautz J (2016) Pruning convolutional neural networks for resource efficient inference. In: Proceedings of International Conference on Learning Representations 2017. https://doi.org/10.48550/arXiv.1611.06440
50. Lin J, Rao Y, Lu J, Zhou J (2017) Runtime neural pruning. In: Proceedings of the 31st International Conference on Neural Information Processing Systems, 30:2178–2188
51. Ding X, Ding G, Han J, Tang S (2018) Auto-balanced filter pruning for efficient convolutional neural networks. In: Proceedings of the AAAI Conference on Artificial Intelligence 32(1):6797–6804. https://doi.org/10.1609/aaai.v32i1.12262
52. Ding X, Ding G, Zhou X, Guo Y, Han J, Liu J (2019) Global sparse momentum sgd for pruning very deep neural networks. Adv Neural Inf Processing Syst 32:8867
53. Lin S, Ji R, Li Y, Wu Y, Huang F, Zhang B (2018) Accelerating convolutional networks via global & dynamic filter pruning. In IJCAI, pp 2425–2432
54. He Y, Ding Y, Liu P, Zhu L, Zhang H, Yang Y (2020) Learning filter pruning criteria for deep convolutional neural networks acceleration. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition 2020:2009–2018. https://doi.org/10.1109/CVPR42600.2020.00208
55. Wang Z, Li C, Wang X (2021) Convolutional neural network pruning with structural redundancy reduction. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition 2021:14913–14922. https://doi.org/10.1109/CVPR46437.2021.01467
56. Tang Y, Wang Y, Xu Y, Deng Y, Xu C, Tao D, Xu C (2021) Manifold regularized dynamic network pruning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition 1:5018–5028
57. Ruan X, Liu Y, Li B, Yuan C, Hu W (2021) DPFPS: dynamic and progressive filter pruning for compressing convolutional neural networks from scratch. Proc AAAI Conf Artif Intell 35(3):2495–2503
58. Lin C, Zhong Z, Wu W, Yan J (2018) Synaptic strength for convolutional neural network. Adv Neural Inf Processing Syst 2018:10149–10158
59. Luo JH, Wu J (2020) Neural network pruning with residual-connections and limited-data. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition 2020:1458–1467. https://doi.org/10.48550/arXiv.1911.08114
60. Tang Y, You S, Xu C, Han J, Qian C, Shi B, Zhang C (2020) Reborn filters: pruning convolutional neural networks with limited data. In Proc AAAI Conf Artif Intell 34(04):5972–5980. https://doi.org/10.1609/aaai.v34i04.6058
61. Ding X, Ding G, Guo Y, Han J (2019) Centripetal sgd for pruning very deep convolutional networks with complicated structure. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition 2019:4943–4953. https://doi.org/10.1109/CVPR.2019.00508
62. Liu Z, Sun M, Zhou T, Huang G, Darrell T (2018) Rethinking the value of network pruning. In Proceedings of International Conference on Learning Representations 2019
63. Ding X, Ding G, Guo Y, Han J, Yan C (2019) Approximated oracle filter pruning for destructive cnn width optimization. In International Conference on Machine Learning 1:1607–1616 (PMLR)
64. He Y, Kang G, Dong X, Fu Y, Yang Y (2018) Soft filter pruning for accelerating deep convolutional neural networks. International Joint Conference on Artificial Intelligence (IJCAI) 2234–2240. https://doi.org/10.24963/ijcai.2018/309
65. Li Y, Lin S, Zhang B, Liu J, Doermann D, Wu Y, ... Ji R (2019) Exploiting kernel sparsity and entropy for interpretable CNN compression. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition 2019:2800–2809. https://doi.org/10.1109/CVPR.2019.00291
66. Lin S, Ji R, Yan C, Zhang B, Cao L, Ye Q, Doermann D (2019) Towards optimal structured cnn pruning via generative adversarial learning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition 2019:2790–2799. https://doi.org/10.1109/CVPR.2019.00290
67. Singh P, Verma VK, Rai P, Namboodiri VP (2019a) Play and prune: adaptive filter pruning for deep model compression. In IJCAI, pp 3460–3466. https://doi.org/10.24963/ijcai.2019/480

68. Enderich L, Timm F, Burgard W (2021) Holistic filter pruning for efficient deep neural networks. In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, pp 2596–2605. https://doi.org/10.1109/WACV48630.2021.00264
69. Luo JH, Wu J, Lin W (2017) Thinet: A filter level pruning method for deep neural network compression. In Proceedings of the IEEE international conference on computer vision 2017:5058. https://doi.org/10.1109/ICCV.2017.541
70. Li T, Wu B, Yang Y, Fan Y, Zhang Y, Liu W (2019) Compressing convolutional neural networks via factorized convolutional filters. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition 2019:3977–3986. https://doi.org/10.1109/CVPR.2019.00410
71. You Z, Yan K, Ye J, Ma M, Wang P (2019) Gate decorator: global filter pruning method for accelerating deep convolutional neural networks. Adv Neural Inf Proces Syst 32(191):2133–2144
72. He Y, Liu P, Wang Z, Hu Z, Yang Y (2019) Filter pruning via geometric median for deep convolutional neural networks acceleration. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition 2019:4340–4349. https://doi.org/10.1109/CVPR.2019.00447
73. Figurnov M, Ibraimova A, Vetrov DP, Kohli P (2016) Perforatedcnns: acceleration through elimination of redundant convolutions. In Adv Neural Inf Processing Syst 29:947–955
74. Singh P, Verma VK, Rai P, Namboodiri VP (2019b) Hetconv: heterogeneous kernel-based convolutions for deep cnns. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition 2019:4835–4844. https://doi.org/10.1109/CVPR.2019.00497
75. Wang X, Stella XY (2021) Tied Block Convolution: Leaner and Better CNNs with Shared Thinner Filters. Proc AAAI Conf Artif Intell 35(11):10227–10235
76. Liu Z, Li J, Shen Z, Huang G, Yan S, Zhang C (2017) Learning efficient convolutional networks through network slimming. In Proceedings of the IEEE International Conference on Computer Vision 2017:2736–2744. https://doi.org/10.1109/ICCV.2017.298
77. Liu L, Zhang S, Kuang Z, Zhou A, Xue JH, Wang X,... Zhang W, (2021) Group fisher pruning for practical network compression. In International Conference on Machine Learning 139:7021–7032
78. Ding X, Hao T, Tan J, Liu J, Han J, Guo Y, Ding G (2021) ResRep: lossless cnn pruning via decoupling remembering and forgetting. In Proceedings of the IEEE/CVF International Conference on Computer Vision 2021:4490–4500. https://doi.org/10.1109/ICCV48922.2021.00447
79. Hu Y, Sun S, Li J, Wang X, Gu Q (2018) A novel channel pruning method for deep neural network compression. arXiv preprint arXiv:1805.11394. https://doi.org/10.48550/arXiv.1805.11394
80. Zhuang Z, Tan M, Zhuang B, Liu J, Guo Y, Wu Q, Zhu J (2018) Discrimination-aware channel pruning for deep neural networks. In Advances in Neural Information Processing Systems 31:883–894.
81. Peng H, Wu J, Chen S, Huang J (2019) Collaborative channel pruning for deep networks. In International Conference on Machine Learning 1:5113–5122 (PMLR)
82. Zhao C, Ni B, Zhang J, Zhao Q, Zhang W, Tian Q (2019) Variational convolutional neural network pruning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition 2019:2775-2784. https://doi.org/10.1109/CVPR.2019.00289
83. Wang Y, Zhang X, Xie L, Zhou J, Su H, Zhang B, Hu X (2020a) Pruning from scratch. In Proceedings of the AAAI Conference on Artificial Intelligence 34(07):12273–12280
84. Wang Y, Zhang X, Hu X, Zhang B, Su H (2020b) Dynamic Network Pruning with Interpretable Layerwise Channel Selection. Proc AAAI Conf Artif Intell 34(04):6299–6306
85. Li Y, Lin S, Liu J, Ye Q, Wang M, Chao F,... Ji R (2021) Towards compact cnns via collaborative compression. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition 2021:6434–6443. https://doi.org/10.1109/CVPR46437.2021.00637
86. Kang M, Han B (2020) Operation-aware soft channel pruning using differentiable masks. In International Conference on Machine Learning 119:5122–5131
87. Guo J, Ouyang W, Xu D (2020) Channel pruning guided by classification loss and feature importance. Proc AAAI Conf Artif Intell 34(07):10885–10892
88. Guo J, Ouyang W, Xu D (2020b) Multi-dimensional pruning: a unified framework for model compression. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition 2020:1505–1514. https://doi.org/10.1109/CVPR42600.2020.00158
89. Gao S, Huang F, Cai W, Huang H (2021) Network pruning via performance maximization. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition 2021:9266–9276. https://doi.org/10.1109/CVPR46437.2021.00915
90. Joo D, Yi E, Baek S, Kim J (2021) Linearly Replaceable Filters for Deep Network Channel Pruning. Proc AAAI Conf Artif Intell 35(9):8021–8029

91. Hou Z, Qin M, Sun F, Ma X, Yuan K, Xu Y, Chen Y-K, Jin R, Xie Y, Kung S-Y, (2022) CHEX: channel exploration for cnn model compression. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition 2022:12277–12288. https://doi.org/10.1109/CVPR52688.2022.01197
92. Poliakov E et al (2022) Model compression via structural pruning and feature distillation for accurate multi-spectral object detection on edge-devices. 2022 IEEE International Conference on Multimedia and Expo (ICME). pp 1–6. https://doi.org/10.1109/ICME52920.2022.9859994
93. Dong X, Chen S, Pan SJ (2017) Learning to prune deep neural networks via layer-wise optimal brain surgeon. Adv Neural Inf Processing Syst 30:4857–4867.
94. Yang TJ, Chen YH, Sze V (2017) Designing energy-efficient convolutional neural networks using energy-aware pruning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2017:6071–6079. https://doi.org/10.1109/CVPR.2017.643
95. Yang TJ, Howard A, Chen B, Zhang X, Go A, Sandler M, Adam H (2018) Netadapt: platform-aware neural network adaptation for mobile applications. In Proceedings of the European Conference on Computer Vision (ECCV) 11214:289–304. https://doi.org/10.1007/978-3-030-01249-6_18
96. Evci U, Gale T, Menick J, Castro PS, Elsen E (2020) Rigging the lottery: making all tickets winners. In International Conference on Machine Learning 119(276):2943–2952
97. Hayou S, Ton JF, Doucet A, Teh YW (2020) Robust pruning at initialization. arXiv preprint arXiv:2002.08797. https://doi.org/10.48550/arXiv.2002.08797
98. Le DH, Hua BS (2020) Network pruning that matters: a case study on retraining variants. arXiv preprint arXiv:2105.03193. https://doi.org/10.48550/arXiv.2105.03193
99. Liu S, Chen T, Chen X, Shen Li, Mocanu D, Wang Z, Pechenizkiy M (2022) The unreasonable effectiveness of random pruning: return of the most naive baseline for sparse training. arXiv preprint arXiv:2202.02643. https://doi.org/10.48550/arXiv.2202.02643
100. Hu P, Peng X, Zhu H, Aly MMS, Lin J (2021) OPQ: compressing deep neural networks with one-shot pruning-quantization. In Proceedings of the Thirty-Fifth AAAI Conference on Artificial Intelligence (AAAI-21), Vancouver, VN, Canada 35(9):7780–7788
101. Tiwari R, Bamba U, Chavan A, Gupta D (2020) ChipNet: Budget-aware pruning with heaviside continuous approximations. In International Conference on Learning Representations 2021. arXiv preprint arXiv:2102.07156. https://doi.org/10.48550/arXiv.2102.07156
102. Chang X, Li Y, Oymak S, Thrampoulidis C (2021) Provable Benefits of Overparameterization in Model Compression: From Double Descent to Pruning Neural Networks. Proc AAAI Conf Artif Intell 35(8):6974–6983
103. Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D,... Rabinovich A, (2015) Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition 2015:1–9. https://doi.org/10.1109/CVPR.2015.7298594
104. Ioffe S, Szegedy C (2015) Batch normalization: accelerating deep network training by reducing internal covariate shift. In International conference on machine learning 37:448–456
105. Szegedy C, Vanhoucke V, Ioffe S, Shlens J, Wojna Z (2016) Rethinking the inception architecture for computer vision. In Proc IEEE Conf Comput Vis Pattern Recognit 2016:2818–2826. https://doi.org/10.1109/CVPR.2016.308
106. Chollet F (2017) Xception: Deep learning with depthwise separable convolutions. In Proc IEEE Conf Comput Vis Pattern Recognit 2017:1800–1807. https://doi.org/10.1109/CVPR.2017.195
107. Howard AG, Zhu M, Chen B, Kalenichenko D, Wang W, Weyand T,... Adam H (2017) Mobilenets: efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861. https://doi.org/10.48550/arXiv.1704.04861
108. Sandler M, Howard A, Zhu M, Zhmoginov A, Chen LC (2018) Mobilenetv 2: inverted residuals and linear bottlenecks. In Proc IEEE Conf Comput Vis Pattern Recognit 2018:4510–4520. https://doi.org/10.1109/CVPR.2018.00474
109. Yu G, Chang Q, Lv W, Xu C, Cui C, Ji W,... Ma Y (2021) PP-PicoDet: a better real-time object detector on mobile devices. arXiv preprint arXiv:2111.00902. https://doi.org/10.48550/arXiv.2111.00902
110. Tan M, Le QV (2019b) Mixconv: mixed depthwise convolutional kernels. In British Machine Vision Conference (BMVC), 33(116):1–13. https://dx.doi.org/10.5244/C.33.116
111. Mehta S, Rastegari M, Shapiro L, Hajishirzi H (2019) Espnetv2: a light-weight, power efficient, and general purpose convolutional neural network. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition 2019:9182–9192. https://doi.org/10.1109/CVPR.2019.00941

112. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In Proc IEEE Conf Comput Vis Pattern Recognit 2016:770–778. https://doi.org/10.1109/CVPR.2016.90
113. Xie S, Girshick R, Dollár P, Tu Z, He K (2017) Aggregated residual transformations for deep neural networks. In Proc IEEE Conf Comput Vis Pattern Recognit 2017:5987–5995. https://doi.org/10.1109/CVPR.2017.634
114. Huang G, Liu Z, Van Der Maaten L, Weinberger KQ (2017) Densely connected convolutional networks. In Proc IEEE Conf Comput Vis Pattern Recognit 2017:2261–2269. https://doi.org/10.1109/CVPR.2017.243
115. Huang G, Liu S, Van der Maaten L, Weinberger KQ (2018) Condensenet: An efficient densenet using learned group convolutions. In Proc IEEE Conf Comput Vis Pattern Recognit 2018:2752–2761. https://doi.org/10.1109/CVPR.2018.00291
116. Zhang T, Qi GJ, Xiao B, Wang J (2017) Interleaved group convolutions. In Proceedings of the IEEE international conference on computer vision 2017:4383–4392. https://doi.org/10.1109/ICCV.2017.469
117. Xie G, Wang J, Zhang T, Lai J, Hong R, Qi GJ (2018) Interleaved structured sparse convolutional neural networks. In Proc IEEE Conf Comput Vis Pattern Recognit 2018:8847–8856. https://doi.org/10.1109/CVPR.2018.00922
118. Sun K, Li M, Liu D, Wang J (2018) Igcv3: interleaved low-rank group convolutions for efficient deep neural networks. In British Machine Vision Conference (BMVC). https://doi.org/10.48550/arXiv.1806.00178
119. Zhang X, Zhou X, Lin M, Sun J (2018) Shufflenet: an extremely efficient convolutional neural network for mobile devices. In Proc IEEE Conf Comput Vis Pattern Recognit 2018:6848–6856. https://doi.org/10.1109/CVPR.2018.00922
120. Ma N, Zhang X, Zheng HT, Sun J (2018) Shufflenet v2: practical guidelines for efficient cnn architecture design. In Proceedings of the European conference on computer vision (ECCV) 11218: 122–138. https://doi.org/10.1007/978-3-030-01264-9_8
121. Qin Z, Li Z, Zhang Z, Bao Y, Yu G, Peng Y, Sun J (2019) Thundernet: towards real-time generic object detection on mobile devices. In Proceedings of the IEEE/CVF International Conference on Computer Vision 2019:6717–6726. https://doi.org/10.1109/ICCV.2019.00682
122. Mehta S, Rastegari M, Caspi A, Shapiro L, Hajishirzi H (2018) Espnet: efficient spatial pyramid of dilated convolutions for semantic segmentation. In Proceedings of the european conference on computer vision (ECCV) 11214:561–580. https://doi.org/10.1007/978-3-030-01249-6_34
123. Wang Y, Xu C, Xu C, Xu C, Tao D (2018) Learning versatile filters for efficient convolutional neural networks. In Advances in Neural Information Processing Systems 31:1608–1618
124. Han K, Wang Y, Tian Q, Guo J, Xu C, Xu C (2020) Ghostnet: More features from cheap operations. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition 1580–1589
125. Iandola FN, Han S, Moskewicz MW, Ashraf K, Dally WJ, Keutzer K (2016). SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size. arXiv preprint arXiv:1602.07360. https://doi.org/10.48550/arXiv.1602.07360
126. Gholami A, Kwon K, Wu B, Tai Z, Yue X, Jin P, Keutzer K (2018). Squeezenext: hardware-aware neural network design. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp 1638–1647. https://doi.org/10.1109/CVPRW.2018.00215
127. Wu B, Iandola F, Jin PH, Keutzer K (2017) Squeezedet: unified, small, low power fully convolutional neural networks for real-time object detection for autonomous driving. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp 446–454. https://doi.org/10.1109/CVPRW.2017.60
128. Gao H, Wang Z, Ji S (2018) Channelnets: compact and efficient convolutional neural networks via channel-wise convolutions. In Advances in Neural Information Processing Systems 43(8):2570–2581. https://doi.org/10.1109/TPAMI.2020.2975796
129. Hu J, Shen L, Sun G (2018) Squeeze-and-excitation networks. In Proc IEEE Conf Comput Vis Pattern Recognit 42(8):2011–2023. https://doi.org/10.1109/TPAMI.2019.2913372
130. Yu C, Xiao B, Gao C, Yuan L, Zhang L, Sang N, Wang J (2021) Lite-hrnet: a lightweight high-resolution network. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition 2021:10435–10445. https://doi.org/10.1109/CVPR46437.2021.01030
131. Wang J, Sun K, Cheng T, Jiang B, Deng C, Zhao Y,... Xiao B, (2020) Deep high-resolution representation learning for visual recognition. IEEE Trans Pattern Anal Mach Intell 43(10):3349–3364. https://doi.org/10.1109/TPAMI.2020.2983686

132. Wang RJ, Li X, Ling CX (2018) Pelee: a real-time object detection system on mobile devices. In Adv Neural Inf Proces Syst 43(10):3349–3364. https://doi.org/10.1109/TPAMI.2020.2983686
133. Zhou D, Hou Q, Chen Y, Feng J, Yan S (2020) Rethinking bottleneck structure for efficient mobile network design. European Conference on Computer Vision 12348:680–697. https://doi.org/10.1007/978-3-030-58580-8_40
134. Zhou Q, Wang J, Liu J, Li S, Ou W, Jin X (2021) RSANet: towards real-time object detection with residual semantic-guided attention feature pyramid network. Mobile Networks and Applications 26(1):77–87
135. Han K, Wang Y, Xu C, Xu C, Wu E, Tao D (2021) Learning versatile convolution filters for efficient visual recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence
136. Howard A, Sandler M, Chu G, Chen LC, Chen B, Tan M,... Adam H (2019). Searching for mobilenetv3. In Proceedings of the IEEE/CVF International Conference on Computer Vision 2019:1314–1324. https://doi.org/10.1109/ICCV.2019.00140
137. Tan M, Chen B, Pang R, Vasudevan V, Sandler M, Howard A, Le QV (2019) Mnasnet: platform-aware neural architecture search for mobile. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition 2018:2815–2823. https://doi.org/10.1109/CVPR.2019.00293
138. Wu B, Wan A, Yue X, Jin P, Zhao S, Golmant N,... Keutzer K, (2018) Shift: A zero flop, zero parameter alternative to spatial convolutions. In Proc IEEE Conf Comput Vis Pattern Recognit 2017:9127–9135. https://doi.org/10.1109/CVPR.2018.00951
139. Yan Z, Li X, Li M, Zuo W, Shan S (2018) Shift-net: image inpainting via deep feature rearrangement. In Proceedings of the European conference on computer vision (ECCV) 11218:3–19. https://doi.org/10.1007/978-3-030-01264-9_1
140. Chen W, Xie D, Zhang Y, Pu S (2019) All you need is a few shifts: designing efficient convolutional neural networks for image classification. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition 2019:7234–7243. https://doi.org/10.1109/CVPR.2019.00741
141. Chen H, Wang Y, Xu C, Shi B, Xu C, Tian Q, Xu C (2020) AdderNet: do we really need multiplications in deep learning? In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition 2019:1465–1474. https://doi.org/10.1109/CVPR42600.2020.00154
142. Elhoushi M, Chen Z, Shafiq F, Tian YH, Li JY (2019) Deepshift: towards multiplication-less neural networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 2359–2368. https://doi.org/10.1109/CVPRW53098.2021.00268
143. You H, Chen X, Zhang Y, Li C, Li S, Liu Z,... Lin Y (2020) Shiftaddnet: a hardware-inspired deep network. In Proceedings of the 34th International Conference on Neural Information Processing Systems (NIPS'20) 233:2771–2783.
144. Redmon J, Divvala S, Girshick R, Farhadi A (2016) You only look once: unified, real-time object detection. In Proc IEEE Conf Comput Vis Pattern Recognit 2015:779–788. https://doi.org/10.1109/CVPR.2016.91
145. Redmon J, Farhadi A (2017) YOLO9000: better, faster, stronger. In Proc IEEE Conf Comput Vis Pattern Recognit 2016:6517–6525. https://doi.org/10.1109/CVPR.2017.690
146. Redmon J, Farhadi A (2018) Yolov3: an incremental improvement. arXiv preprint arXiv:1804.02767. https://doi.org/10.48550/arXiv.1804.02767
147. Bochkovskiy A, Wang CY, Liao HYM (2020) Yolov4: optimal speed and accuracy of object detection. arXiv preprint arXiv:2004.10934. https://doi.org/10.48550/arXiv.2004.10934
148. Huang R, Pedoeem J, Chen C (2018) YOLO-LITE: a real-time object detection algorithm optimized for non-GPU computers. In 2018 IEEE International Conference on Big Data (Big Data), pp 2503–2510. https://doi.org/10.1109/BigData.2018.8621865
149. Wong A, Famuori M, Shafiee MJ, Li F, Chwyl B, Chung J (2019) Yolo nano: a highly compact you only look once convolutional neural network for object detection. 2019 Fifth Workshop on Energy Efficient Machine Learning and Cognitive Computing - NeurIPS Edition (EMC2-NIPS), pp 22–25. https://doi.org/10.1109/EMC2-NIPS53020.2019.00013
150. Barry D, Shah M, Keijsers M, Khan H, Hopman B (2019). XYOLO: a model for real-time object detection in humanoid soccer on low-end hardware. In 2019 International Conference on Image and Vision Computing New Zealand (IVCNZ), pp 1–6. https://doi.org/10.1109/IVCNZ48456.2019.8960963
151. Qiuqiu D (2020) Yolo-fastest: yolo universal object detection model combined with efficientnet-lite. https://github.com/dog-qiuqiu/Yolo-Fastest
152. Wang CY, Bochkovskiy A, Liao HYM (2020) Scaled-YOLOv4: Scaling cross stage partial network. In 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) 2021:13024–13033. https://doi.org/10.1109/CVPR46437.2021.01283

153. Cai Y, Li H, Yuan G, Niu W, Li Y, Tang X et al (2021) YOLObile: real-time object detection on mobile devices via compression-compilation co-design. In Proceedings of the AAAI Conference on Artificial Intelligence 35(2):955–963. https://doi.org/10.1609/aaai.v35i2.16179
154. Biswas, Debojyoti et al. (2022) Improving the energy efficiency of real-time DNN object detection via compression, transfer learning, and scale prediction. 2022 IEEE International Conference on Networking, Architecture and Storage (NAS), pp. 1–8. https://doi.org/10.1109/NAS55553.2022.9925528
155. Wang X, Yu F, Dou ZY, Darrell T, Gonzalez JE (2018) Skipnet: learning dynamic routing in convolutional networks. In Proceedings of the European Conference on Computer Vision (ECCV) 11217:420–436. https://doi.org/10.1007/978-3-030-01261-8_25
156. Tan M, Le Q (2019) Efficientnet: rethinking model scaling for convolutional neural networks. In International Conference on Machine Learning 97:6105–6114
157. Tan M, Pang R, Le QV (2020) Efficientdet: scalable and efficient object detection. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition 2020:10781–10790. https://doi.org/10.1109/CVPR42600.2020.01079
158. Choi K, Lee H, Hong D, Yu J, Park N, Kim Y, Lee J (2022) It's all in the teacher: zero-shot quantization brought closer to the teacher. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition 2022:8301–8311. https://doi.org/10.1109/CVPR52688.2022.00813
159. Zhong Y, Lin M, Nan G, Liu J, Zhang B, Tian Y, Ji R (2021) IntraQ: learning synthetic images with intra-class heterogeneity for zero-shot network quantization. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) 2022:12329–12338
160. Baker B, Gupta O, Naik N, Raskar R (2016) Designing neural network architectures using reinforcement learning. In Proceedings of International Conference on Learning Representations 2017. https://doi.org/10.48550/arXiv.1611.02167
161. Zoph B, Vasudevan V, Shlens J, Le Q (2017) Learning transferable architectures for scalable image recognition. IEEE/CVF Conference on Computer Vision and Pattern Recognition 2018:8697–8710
162. Zhong Z, Yan J, Wu W, Shao J, Liu CL (2018) Practical block-wise neural network architecture generation. In Proc IEEE Conf Comput Vis Pattern Recognit 2018:2423–2432
163. Cai H, Yang J, Zhang W, Han S, Yu Y (2018) Path-level network transformation for efficient architecture search. In International Conference on Machine Learning 80:678–687
164. Cai H, Chen T, Zhang W, Yu Y, Wang J (2018) Efficient architecture search by network transformation. In Proceedings of the AAAI Conference on Artificial Intelligence 32(1):2787–794
165. He Y, Lin J, Liu Z, Wang H, Li LJ, Han S (2018) Amc: automl for model compression and acceleration on mobile devices. In Proceedings of the European Conference on Computer Vision (ECCV) 11211:815–832. https://doi.org/10.1007/978-3-030-01234-2_48
166. Wong C, Houlsby N, Lu Y, Gesmundo A (2018) Transfer learning with neural automl. In Adv Neural Inf Proces Syst 31:8366–8375
167. Liu Z, Mu H, Zhang X, Guo Z, Yang X, Cheng KT, Sun J (2019) Metapruning: meta learning for automatic neural network channel pruning. In Proceedings of the IEEE/CVF International Conference on Computer Vision 2019:3296–3305
168. Real E, Moore S, Selle A, Saxena S, Suematsu YL, Tan J et al (2017) Large-scale evolution of image classifiers. In International Conference on Machine Learning 70:2902–2911
169. Real E, Aggarwal A, Huang Y, Le QV (2019) Regularized evolution for image classifier architecture search. In Proceedings of the AAAI conference on artificial intelligence 33(01):4780–4789
170. Liu H, Simonyan K, Vinyals O, Fernando C, Kavukcuoglu K (2017) Hierarchical representations for efficient architecture search. arXiv preprint arXiv:1711.00436, 2017. https://doi.org/10.48550/arXiv.1711.00436
171. Guo Z, Zhang X, Mu H, Heng W, Liu Z, Wei Y, Sun J (2020) Single path one-shot neural architecture search with uniform sampling. In European Conference on Computer Vision 12361:544–560. https://doi.org/10.1007/978-3-030-58517-4_32
172. Veniat T, Denoyer L (2018) Learning time/memory-efficient deep architectures with budgeted super networks. In Proc IEEE Conf Comput Vis Pattern Recognit 2019:3492–3500
173. Wu B, Dai X, Zhang P, Wang Y, Sun F, Wu Y, Keutzer K (2019) Fbnet: hardware-aware efficient convnet design via differentiable neural architecture search. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition 2020:10734–10742
174. Wan A, Dai X, Zhang P, He Z, Tian Y, Xie S, Gonzalez JE (2020) Fbnetv2: differentiable neural architecture search for spatial and channel dimensions. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition 2020:12965–12974

175. Dai X, Wan A, Zhang P, Wu B, He Z, Wei Z, Gonzalez JE (2020) FBNetV3: joint architecture-recipe search using neural acquisition function. 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) 2021:16271–16280
176. Liu H, Simonyan K, Yang Y (2018) Darts: differentiable architecture search. arXiv preprint arXiv:1806.09055. https://doi.org/10.48550/arXiv.1806.09055
177. Liu C, Zoph B, Neumann M, Shlens J, Hua W, Li LJ, et al (2018) Progressive neural architecture search. In Proceedings of the European conference on computer vision (ECCV) 11205:19–35. https://doi.org/10.1007/978-3-030-01246-5_2
178. Gordon A, Eban E, Nachum O, Chen B, Wu H, Yang TJ, Choi E (2018) Morphnet: fast & simple resource-constrained structure learning of deep networks. In Proc IEEE Conf Comput Vis Pattern Recognit 2018:1586–1595. https://doi.org/10.1109/CVPR.2018.00171
179. Pham H, Guan M, Zoph B, Le Q, Dean J (2018) Efficient neural architecture search via parameters sharing. In International Conference on Machine Learning 80:4095–4104
180. Xiong Y, Liu H, Gupta S, Akin B, Bender G, Wang Y, et al (2021). Mobiledets: searching for object detection architectures for mobile accelerators. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition 2021:3825–3834. https://doi.org/10.1109/CVPR46437.2021.00382
181. Cai H, Zhu L, Han S (2018) Proxylessnas: direct neural architecture search on target task and hardware. In Proceedings of International Conference on Learning Representations 2019. https://doi.org/10.48550/arXiv.1812.00332
182. Yu J, Yang L, Xu N, Yang J, Huang T (2018) Slimmable neural networks. In Proceedings of International Conference on Learning Representations 2019. https://doi.org/10.48550/arXiv.1812.08928
183. Elsken T, Metzen JH, Hutter F (2017) Simple and efficient architecture search for convolutional neural networks. In Proceedings of International Conference on Learning Representations 2018. https://doi.org/10.48550/arXiv.1711.04528
184. Chen B, Li P, Li B, Lin C, Li C, Sun M, et al (2021). Bn-nas: neural architecture search with batch normalization. In Proceedings of the IEEE/CVF International Conference on Computer Vision 2021:307–316. https://doi.org/10.1109/ICCV48922.2021.00037
185. Chen M, Peng H, Fu J, Ling H (2021). Autoformer: searching transformers for visual recognition. In Proceedings of the IEEE/CVF International Conference on Computer Vision 2021:12270–12280. https://doi.org/10.1109/ICCV48922.2021.01205
186. Dong X, Yang Y (2019) Network pruning via transformable architecture search. In Advances in Neural Information Processing Systems
187. Li X, Zhou Y, Pan Z, Feng J (2019) Partial order pruning: for best speed/accuracy trade-off in neural architecture search. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition 2019:9145–9153. https://doi.org/10.1109/CVPR.2019.00936
188. Cai H, Gan C, Wang T, Zhang Z, Han S (2019) Once-for-all: train one network and specialize it for efficient deployment. In Proceedings of International Conference on Learning Representations 2020. https://doi.org/10.48550/arXiv.1908.09791
189. Wang T, Wang K, Cai H, Lin J, Liu Z, Wang H, ... Han S (2020). Apq: joint search for network architecture, pruning and quantization policy. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition 2020:2075–2084. https://doi.org/10.1109/CVPR42600.2020.00215
190. Lin J, Chen WM, Lin Y, Cohn J, Gan C, Han S (2020) Mcunet: tiny deep learning on iot devices. In Adv Neural Inf Proces Syst 33:11711–11722
191. Chen H, Zhang B, Zheng X, Liu J, Ji R, Doermann D, Guo G (2021) Binarized neural architecture search for efficient object recognition. Int J Comput Vision 129(2):501–516
192. Zhuo LA, Zhang B, Chen H, Yang L, Chen C, Zhu Y, Doermann D (2020) CP-NAS: child-parent neural architecture search for binary neural networks. In IJCAI 144:1033–1039. https://doi.org/10.24963/ijcai.2020/144
193. Yan B, Peng H, Wu K, Wang D, Fu J, Lu H (2021) LightTrack: finding lightweight neural networks for object tracking via one-shot architecture search. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition 2021:15175–15184. https://doi.org/10.1109/CVPR46437.2021.01493
194. Courbariaux M, Bengio Y, David JP (2015) Binaryconnect: training deep neural networks with binary weights during propagations. In Adv Neural Inf Proces Syst 28:3123–3131

195. Hubara I, Courbariaux M, Soudry D, El-Yaniv R, Bengio Y (2016) Binarized neural networks. In Proceedings of the 30th International Conference on Neural Information Processing Systems 29:4114–4122
196. Rastegari M, Ordonez V, Redmon J, Farhadi A (2016) Xnor-net: imagenet classification using binary convolutional neural networks. In European conference on computer vision 9908:525–542. https://doi.org/10.1007/978-3-319-46493-0_32
197. Lin X, Zhao C, Pan W (2017). Towards accurate binary convolutional neural network. In Proceedings of the 31st International Conference on Neural Information Processing Systems 30:344–352
198. Hou L, Yao Q, Kwok JT (2016) Loss-aware binarization of deep networks. In Proceedings of International Conference on Learning Representations 2017. https://doi.org/10.48550/arXiv.1611.01600
199. Liu Z, Wu B, Luo W, Yang X, Liu W, Cheng KT (2018) Bi-real net: enhancing the performance of 1-bit cnns with improved representational capability and advanced training algorithm. In Proceedings of the European Conference on Computer Vision (ECCV) 11219:722–737. https://doi.org/10.1007/978-3-030-01267-0_44
200. Hu Q, Wang P, Cheng J (2018) From hashing to cnns: training binary weight networks via hashing. In Proceedings of the AAAI Conference on Artificial Intelligence 32(1):3247–3254
201. Diffenderfer J, Kailkhura B (2020) Multi-prize lottery ticket hypothesis: finding accurate binary neural networks by pruning a randomly weighted network. In International Conference on Learning Representations 2021.arXiv preprint arXiv: 2103.09377. https://doi.org/10.48550/arXiv.2103.09377
202. Umuroglu Y, Fraser NJ, Gambardella G, Blott M, Leong P, Jahre M, Vissers K (2017) Finn: a framework for fast, scalable binarized neural network inference. In Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, pp 65-74. https://doi.org/10.1145/3020078.3021744
203. Zhu S, Dong X, Su H (2019) Binary ensemble neural network: more bits per network or more networks per bit? In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition 2019:4918-4927. https://doi.org/10.1109/CVPR.2019.00506
204. Wang Z, Lu J, Tao C, Zhou J, Tian Q (2019) Learning channel-wise interactions for binary convolutional neural networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition 43(10):3432-3445. https://doi.org/10.1109/TPAMI.2020.2988262
205. Liu C, Ding W, Xia X, Zhang B, Gu J, Liu J,... Doermann D (2019). Circulant binary convolutional networks: enhancing the performance of 1-bit dcnns with circulant back propagation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition 2019:2691–2699. https://doi.org/10.1109/CVPR.2019.00280
206. Ding R, Chin TW, Liu Z, Marculescu D (2019) Regularizing activation distribution for training binarized deep networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition 2019:11408–11417. https://doi.org/10.1109/CVPR.2019.01167
207. Zhu C, Han S, Mao H, Dally WJ (2016) Trained ternary quantization. arXiv preprint arXiv:1612.01064. https://doi.org/10.48550/arXiv.1612.01064
208. Li F, Zhang B, Liu B (2016) Ternary weight networks. arXiv preprint arXiv:1605.04711. https://arxiv.org/abs/1605.04711
209. Li Y, Ding W, Liu C, Zhang B, Guo G (2021) TRQ: ternary neural networks with residual quantization. Proc AAAI Conf Artif Intell 35(10):8538–8546
210. Wu J, Leng C, Wang Y, Hu Q, Cheng J (2016) Quantized convolutional neural networks for mobile devices. In Proc IEEE Conf Comput Vis Pattern Recognit 2016:4820–4828
211. Wu J, Wang Y, Wu Z, Wang Z, Veeraraghavan A, Lin Y (2018) Deep k-means: re-training and parameter sharing with harder cluster assignments for compressing deep convolutions. In International Conference on Machine Learning 80:5363–5372
212. Xu Y, Wang Y, Zhou A, Lin W, Xiong H (2018) Deep neural network compression with single and multiple level quantization. In Proceedings of the AAAI Conference on Artificial Intelligence 32(1):335–442
213. Zhou S, Wu Y, Ni Z, Zhou X, Wen H, Zou Y (2016) Dorefa-net: training low bitwidth convolutional neural networks with low bitwidth gradients. arXiv preprint arXiv:1606.06160. https://arxiv.org/abs/1606.06160
214. Jacob B, Kligys S, Chen B, Zhu M, Tang M, Howard A,... Kalenichenko D, (2018) Quantization and training of neural networks for efficient integer-arithmetic-only inference. In Proc IEEE Conf Comput Vis Pattern Recognit 2018:2704–2713
215. Zhan D, Yang J, Ye D, Hua G (2018) Lq-nets: Learned quantization for highly accurate and compact deep neural networks. In Proceedings of the European conference on computer vision (ECCV). 2018:365–382

216. Faraone J, Fraser N, Blott M, Leong PH (2018) Syq: Learning symmetric quantization for efficient deep neural networks. In Proc IEEE Conf Comput Vis Pattern Recognit 2018:4300–4309
217. Wang P, Hu Q, Zhang Y, Zhang C, Liu Y, Cheng J (2018) Two-step quantization for low-bit neural networks. In Proc IEEE Conf Comput Vis Pattern Recognit 2018:4376–4384
218. Qin H, Gong R, Liu X, Shen M, Wei Z, Yu F, Song J (2020) Forward and backward information retention for accurate binary neural networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020:2250–2259
219. Leng C, Dou Z, Li H, Zhu S, Jin R (2018) Extremely low bit neural network: Squeeze the last bit out with admm. In Proceedings of the AAAI Conference on Artificial Intelligence 32(1):466–3473
220. Hubara I, Courbariaux M, Soudry D, El-Yaniv R, Bengio Y (2017) Quantized neural networks: training neural networks with low precision weights and activations. J Machine Learning Res 18(1):6869–6898
221. Liu X, Ye M, Zhou D, Liu Q (2021) Post-training quantization with multiple points: mixed precision without mixed precision. Proc AAAI Conf Artif Intell 35(10):8697–8705
222. Kim D, Lee J, Ham B (2021) Distance-aware quantization. In Proceedings of the IEEE/CVF International Conference on Computer Vision. 2021:5271–5280
223. Wang K, Liu Z, Lin Y, Lin J, Han S (2019) Haq: Hardware-aware automated quantization with mixed precision. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition 2019:8612–8620
224. Yang H, Gui S, Zhu Y, Liu J (2020) Automatic neural network compression by sparsity-quantization joint learning: A constrained optimization-based approach. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition 2020:2178–2188
225. Bulat A, Martinez B, Tzimiropoulos G (2020) Bats: binary architecture search. In European Conference on Computer Vision 12368:309–325
226. Kim D, Singh KP, Choi J (2020) Learning architectures for binary networks. In European conference on computer vision. 12357:575–591
227. Boo Y, Shin S, Choi J, Sung W (2021) Stochastic precision ensemble: self-knowledge distillation for quantized deep neural networks. Proc AAAI Conf Artif Intell 35(8):6794–6802
228. Zhao C, Gao X (2021) QDNN: deep neural networks with quantum layers. Quantum Machine Intell 3:1–9
229. Hinton G, Vinyals O, Dean J (2015) Distilling the knowledge in a neural network. In: Proc Adv Neural Inf Processing Syst Workshop 27:2014
230. Zhang Y, Xiang T, Hospedales TM, Lu H (2018) Deep mutual learning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2018:4320–4328
231. Kim J, Park S, Kwak N (2018) Paraphrasing complex network: network compression via factor transfer. In Advances in Neural Information Processing Systems. 2018:2765–2774
232. Zhao B, Cui Q, Song R, Qiu Y, Liang J (2022) Decoupled knowledge distillation. In Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition. 2022:11953–11962
233. Romero A, Ballas N, Kahou SE, Chassang A, Gatta C, Bengio Y (2014) Fitnets: hints for thin deep nets. In Proceedings of International Conference on Learning Representations 2015. https://doi.org/10.48550/arXiv.1412.6550
234. Komodakis N, Zagoruyko S (2017) Paying more attention to attention: improving the performance of convolutional neural networks via attention transfer. In ICLR
235. Huang Z, Wang N (2018) Like what you like: knowledge distill via neuron selectivity transfer. In Proceedings of International Conference on Learning Representations 2019. https://doi.org/10.48550/arXiv.1707.01219
236. Tung F, Mori G (2019) Similarity-preserving knowledge distillation. In Proceedings of the IEEE/CVF International Conference on Computer Vision. 2019:1365–1374
237. Yang J, Martinez B, Bulat A, Tzimiropoulos G (2020) Knowledge distillation via adaptive instance normalization. arXiv preprint arXiv:2003.04289. https://arxiv.org/abs/2003.04289
238. Yim J, Joo D, Bae J, Kim J (2017) A gift from knowledge distillation: fast optimization, network minimization and transfer learning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017:4133–4141.
239. You S, Xu C, Xu C, Tao D (2017) Learning from multiple teacher networks. In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2017:1285–1294
240. Anil R, Pereyra G, Passos A, Ormandi R, Dahl GE, Hinton GE (2018). Large scale distributed neural network training through online distillation. In Proceedings of International Conference on Learning Representations 2018. https://doi.org/10.48550/arXiv.1804.03235

241. Peng B, Jin X, Liu J, Li D, Wu Y, Liu Y, ... Zhang Z (2019) Correlation congruence for knowledge distillation. In Proceedings of the IEEE/CVF International Conference on Computer Vision. 2019:5007–5016
242. Park W, Kim D, Lu Y, Cho M (2019) Relational knowledge distillation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019:3967–3976
243. Liu Y, Cao J, Li B, Yuan C, Hu W, Li Y, Duan Y (2019) Knowledge distillation via instance relationship graph. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019:7096–7104
244. Tian Y, Krishnan D, Isola P (2019) Contrastive representation distillation. In Proceedings of International Conference on Learning Representations, 2020. https://arxiv.org/abs/1910.10699
245. Zhou S, Wang Y, Chen D, Chen J, Wang X, Wang C, Bu J (2021) Distilling holistic knowledge with graph neural networks. In Proceedings of the IEEE/CVF International Conference on Computer Vision. 2021:10387–10396
246. Gupta S, Hoffman J, Malik J (2016) Cross modal distillation for supervision transfer. In Proc IEEE Conf Comput Vis Pattern Recognit 2016:2827–2836. https://doi.org/10.1109/CVPR.2016.309
247. Li Q, Jin S, Yan J (2017) Mimicking very efficient network for object detection. In Proceedings of the ieee conference on computer vision and pattern recognition. 2017:6356–6364
248. Ko Jong-gook, Yoo Wonyoung (2020) Knowledge distillation based compact model learning method for object detection. International Conference on Information and Communication Technology Convergence (ICTC) 2020:1276–1278
249. Chen G, Choi W, Yu X, Han T, Chandraker M (2017) Learning efficient object detection models with knowledge distillation. In Proceedings of the 31st International Conference on Neural Information Processing Systems 2020(30):742–51
250. Wang T, Yuan L, Zhang X, Feng J (2019) Distilling object detectors with fine-grained feature imitation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition 2019:4933–4942
251. Chen H, Guo T, Xu C, Li W, Xu C, Xu C, Wang Y (2021) Learning student networks in the wild. 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2021:6424–6433
252. Mirzadeh SI, Farajtabar M, Li A, Levine N, Matsukawa A, Ghasemzadeh H (2020) Improved knowledge distillation via teacher assistant. Proc AAAI Conf Artif Intell 34(04):5191–5198
253. Gao M, Shen Y, Li Q, Loy CC (2020) Residual Knowledge Distillation. arXiv preprint arXiv: 2002.09168. https://arxiv.org/abs/2002.09168
254. Denil M, Shakibi B, Dinh L, Ranzato M, de Freitas N (2013) Predicting parameters in deep learning. Adv Neural Inf Processing Syst. Lake Tahoe 26:2148–2156
255. Lebedev V, Ganin Y, Rakhuba M, Oseledets I, Lempitsky V (2015). Speeding-up convolutional neural networks using fine-tuned cp-decomposition. In International Conference on Learning Representations. https://doi.org/10.48550/arXiv.1412.6553
256. Denton EL, Zaremba W, Bruna J, LeCun Y, Fergus R (2014) Exploiting linear structure within convolutional networks for efficient evaluation. Neural information processing systems foundation (NIPS). 2014: 1269–1277
257. Jaderberg M, Vedaldi A, Zisserman (2014) Speeding up Convolutional Neural Networks with Low Rank Expansions. In British Machine Vision Conference. 2014: 1–13
258. Kim Y, Park E, Yoo S, Choi T, Yang L, Shin D (2015) Compression of deep convolutional neural networks for fast and low power mobile applications. CoRR. https://arxiv.org/abs/1511.06530
259. Zhang X, Zou J, He K, Sun J (2015) Accelerating very deep convolutional networks for classification and detection. IEEE Trans Pattern Anal Mach Intell 38(10):1943–1955
260. Novikov A, Podoprikhin D, Osokin A, Vetrov D (2015) Tensorizing neural networks. In Advances in Neural Information Processing Systems. 28:442–450
261. Liu B, Wang M, Foroosh H, Tappen MF, Pensky M (2015) Sparse convolutional neural networks. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2015:806–814
262. Tai C, Xiao T, Zhang Y, Wang X (2015) Convolutional neural networks with low-rank regularization. arXiv preprint arXiv:1511.06067. https://arxiv.org/abs/1511.06067
263. Yu X, Liu T, Wang X, Tao D (2017) On compressing deep models by low rank and sparse decomposition. Proc IEEE Conf Comput Vis Pattern Recognit 2017:7370–7379. https://doi.org/10.1109/CVPR.2017.15
264. Peng B, Tan W, Li Z, Zhang S, Xie D, Pu S (2018) Extreme network compression via filter group approximation. In Proceedings of the European Conference on Computer Vision (ECCV) 11212:300–316. https://doi.org/10.1007/978-3-030-01237-3_19
265. Ye J, Wang L, Li G, Chen D, Zhe S, Chu X, Xu Z (2018) Learning compact recurrent neural networks with block-term tensor decomposition. In Proc IEEE Conf Comput Vis Pattern Recognit 9378–9387. https://doi.org/10.1109/10.1109/CVPR.2018.00977

266. Wang W, Sun Y, Eriksson B, Wang W, Aggarwal V (2018) Wide compression: tensor ring nets. In Proc IEEE Conf Comput Vis Pattern Recognit 9329–9338. https://doi.org/10.1109/CVPR.2018.00972
267. Kossaifi J, Bulat A, Tzimiropoulos G, Pantic M (2019) T-net: parametrizing fully convolutional nets with a single high-order tensor. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition 2019:7822–7831. https://doi.org/10.1109/CVPR.2019.00801
268. Idelbayev Y, Carreira-Perpinán MA (2020) Low-rank compression of neural nets: learning the rank of each layer. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition 2020:8046–8056. https://doi.org/10.1109/CVPR42600.2020.00807
269. Liao S, Yuan B (2019) CircConv: a structured Convolution with low complexity. In Proc AAAI Conf Artif Intell 33(01):4287–4294
270. Li Y, Gu S, Mayer C, Gool LV, Timofte R (2020) Group sparsity: the hinge between filter pruning and decomposition for network compression. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition 2020:8015–8024. https://doi.org/10.1109/CVPR42600.2020.00804
271. Swaminathan S, Garg D, Kannan R, Andres F (2020) Sparse low rank factorization for deep neural network compression. Neurocomputing 398:185–196. https://doi.org/10.1016/j.neucom.2020.02.035
272. Hawkins C, Yang H, Li M, Lai L, Chandra V (2021) Low-rank+sparse tensor compression for neural networks. arXiv preprint arXiv:2111.01697. https://doi.org/10.48550/arXiv.2111.01697
273. Liu W, Anguelov D, Erhan D, Szegedy C, Reed S, Fu CY, Berg AC (2016). Ssd: single shot multibox detector. In Computer Vision – ECCV 2016 9905:21–37. https://doi.org/10.1007/978-3-319-46448-0_2
274. Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. In Adv Neural Inf Proces Syst 60(6):84–90
275. Szegedy C, Ioffe S, Vanhoucke V, Alemi A (2017) Inception-v4, inception-resnet and the impact of residual connections on learning. Proc AAAI Conf Artif Intell 31(1):4278–4284
276. Simonyan K, Zisserman A (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556. https://doi.org/10.48550/arXiv.1409.1556
277. Zhang T, Qi G, Xiao B, Wang J (2017) Interleaved Group Convolutions for Deep Neural Networks. arXiv preprint arXiv:1707.02725. https://doi.org/10.48550/arXiv.1707.02725
278. Zhang Q, Li J, Yao M, Song L, Zhou H, Li Z, Meng W, Zhang X, Wang G (2019) VarGNet: variable group convolutional neural network for efficient embedded computing. arXiv preprint arXiv:1907.05653. https://doi.org/10.48550/arXiv.1907.05653
279. Yang K, Jiao Z, Liang J, Lei H, Li C, Zhong Z (2022) An application case of object detection model based on Yolov3-SPP model pruning. IEEE Int Conf Artif Intell Comput App (ICAICA) 2022:578–582

Authors and Affiliations

Zonglei Lyu[1,2] · Tong Yu[1,2] · Fuxi Pan[1,2] · Yilin Zhang[1,2] · Jia Luo[1,2] · Dan Zhang[1,2] · Yiren Chen[1,2] · Bo Zhang[1,2] · Guangyao Li[1,2]

✉ Zonglei Lyu
zllv@cauc.edu.cn

Tong Yu
2021051030@cauc.edu.cn

Fuxi Pan
2019051023@cauc.edu.cn

Yilin Zhang
2020051019@cauc.edu.cn

Jia Luo
2020051011@cauc.edu.cn

Dan Zhang
2019052031@cauc.edu.cn

Yiren Chen
2020052039@cauc.edu.cn

Bo Zhang
2020051029@cauc.edu.cn

Guangyao Li
2019053075@cauc.edu.cn

[1] College of Computer Science and Technology, Civil Aviation University of China, Tianjin 300300, China

[2] Key Laboratory of Smart Airport Theory and System, Civil Aviation University of China, Tianjin, China