Artificial Intelligence Review (2023) 56:1905–1969
https://doi.org/10.1007/s10462-022-10213-5

A review of convolutional neural network architectures and their optimizations

Shuang Cong[1] · Yang Zhou[1]

Published online: 22 June 2022

Abstract
The research advances concerning the typical architectures of convolutional neural networks (CNNs) as well as their optimizations are analyzed and elaborated in detail in this paper. This paper proposes a typical approach to classifying CNNs architecture based on modules in order to accommodate more new network architectures with multiple characteristics that make them difficult to rely on the original classification method. Through the pros and cons analysis of diverse network architectures and their performance comparisons, six types of typical CNNs architectures are analyzed and explained in detail. The CNNs architectures intrinsic characteristics is also explored. Moreover, this paper provides a comprehensive classification of network compression and accelerated network architecture optimization algorithms based on the mathematical principle of various optimization algorithms. Finally, this paper analyses the strategy of NAS algorithms, discusses the applications of CNNs, and sheds light on the challenges and prospects of the current CNNs architecture and its optimizations. The explanation of the advantages brought by optimizing different network architecture types, the basis for constructively choosing appropriate CNNs in specific designs and applications are provided. This paper will help the readers to choose constructively appropriate CNNs in specific designs and applications.

Keywords Machine learning · Convolutional neural network · Network architecture

1 Introduction

Machine learning (ML), which was first proposed at the 1952 Dartmouth Conference in Canada, is a tremendously widely applied interdisciplinary and frontier branch of computer science. After over 60 years of development, numerous sub-fields of ML have been derived, such as deep learning, computer vision, and speech recognition. Deep learning is considered as a novel research field in ML, whose concept was published in a seminar paper by Hinton and Salakhutdinov (2006). The paper introduced an efficient and unsupervised greedy layer-wise training algorithm based on deep belief network (DBN). Subsequently, a surging number of deep learning networks, including neural

✉ Shuang Cong
scong@ustc.edu.cn

1 Department of Automation, University of Science and Technology of China, Hefei 230027, China

networks (NN), have begun to attract widespread attention. Typical deep learning models include DBNs, stacked auto-encoder networks and convolutional neural networks (CNNs), etc. Theoretical research has demonstrated that when the function represented by deeper network architecture is expressed through shallow network architecture, its computing unit increases exponentially (Håstad and Goldmann 1991). The potential of such function expression fully suggests broad prospect of the deep learning networks. Accordingly, focusing on the demands of computer vision tasks, a special type of neural network architectures, i.e. CNNs, have emerged. CNNs are considered as one of the best techniques for learning image content, which have achieved remarkable efficiency in image recognition, image segmentation, object detection and other related fields (Punyani et al. 2020; Wang et al. 2021). The networks were originally inspired by the neural mechanisms of vision system, whose architecture was inspired by the neurobiological experiments conducted on cat visual cortex cells in 1962 by Hubel and Wiesel (Hubel and Wiesel 2009, 1962). In 1989, LeCun et al. proposed the first multi-layer CNNs named ConvNet based on Neocognitron, its predecessor (Gi 1989; Fukushima and Miyake 1982). It defined the most basic framework of CNNs, namely convolution layer, pooling layer and fully connected (FC) layer, which successfully addressed the problems related to the recognition of handwritten digits and postal codes. Later in 1998, LeCun et al. improved the architecture of ConvNet (LeCun et al. 1998) by combining the convolution layer with the downsampling layer, and named it LeNet-5, which was the prototype of modern CNNs.

From the late 1990s to the early 2000s, people, however, showed little interest in deep learning due to the limitations of computer performance. In particular, far greater attention was paid to the support vector machines (SVMs) than to the CNNs, although there were still researchers who optimized the CNNs architecture. Simard et al. modified the CNNs architecture in 2003 (Simard et al. 2003), which exhibited better efficiency than SVMs on the Mixed National Institute of Standards and Technology (MNIST) database, a handwriting benchmark. In 2010, Li's team at Stanford University constructed a large image database called ImageNet, containing millions of tagged images. Based on this database, the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) was held annually for performance evaluation and scoring of various models. After over 10 years of stagnation, AlexNet emerged and won the 2012-ILSVRC championship, which marked a major turning point in CNNs performance.

Currently, human improvements to the CNNs architectures are no longer limited to attaining better efficiency by enhancing recognition rate. More advanced architectures and modules will enable the CNNs to have better adaptability and transplantation capability. As early as the 1990s, research protocol for optimizing neural network architectures has been proposed. However, since most of the NNs at that time were shallow, the demand for architectural optimization was not high. Moreover, given the insufficient computer capacity at that time, few breakthrough has been made in the architecture optimization of CNNs. With the shift of ML application to the mobile and embedded devices, the research of CNNs architecture optimization has become ever popular. Network architectures with higher operating efficiency and lower memory usage have received general attention. For instance, modern smartphones are increasingly performing object recognition (Montremerlo et al. 2008), medical equipment acquisition and patient data analysis (Lee and Verma 2013; Alzubaidi et al. 2018, 2020) in real time through image recognition operations, robots and self-driving vehicles. Compared with GPUs or computer clusters, these devices are designed for low power consumption and portability, which generally have rather small RAMs.

Over the past few years, researchers have investigated on the typical architectures of CNNs and their optimization schemes. Gu et al. reviewed the typical CNNs models emerged from 2012 to 2015 and their basic components (Gu et al. 2018). Similarly, there are also some well-known studies which discussed different CNNs algorithms and their applications (LeCun et al. 2010; Srinivas et al. 2016; Alzubaidi et al. 2021). Khan et al. classified various recent CNNs architectures according to the design pattern of processing units (Khan et al. 2020). Reviews by Zhang et al. (2019) and Choudhary et al. (2020) discussed the CNNs classification based on acceleration techniques. Recently, there still requires comprehensive summarization of various CNNs architectures and optimization methods. Meanwhile, the ML research direction is tending to expand, which necessitates continuous integration, updating, comparison and exploration. It is worth noting that the classification logic based on basic characteristics of CNNs can hardly be applied to more new proposed network architectures by the previous CNNs architecture classification method. Currently, more mature and advanced CNNs architectures are usually multi-featured, and the trend of network optimization is more diverse and no longer limited to basic network characteristics. Comparison and analysis of different reviews are summarized in Table 1.

The contribution of this paper are the analysis and summary of the research achievements for the CNNs architectures and their optimizations in recent years by means of the performance comparisons and elaboration detailedly. A typical approach to classifying CNNs architecture based on modules are proposed. An overview of network compression and accelerated network architecture optimization algorithms is made. The strategy of NAS algorithms and the application of CNNs are discussed. The challenges and prospects of the CNNs architecture and its optimizations are provided. The proposed six types of typical CNNs architectures are: basic network architecture, convolution splitting, cross-layer connection idea, depthwise separable convolution, object detection CNNs and transformer encoder. Through the pros and cons analysis of diverse network architectures and their performance comparisons, six types of typical CNNs architectures are analyzed and explained in detail. The similarities and differences in respective architectures and characteristics are thoroughly explored. The optimization techniques for network architectures are classified into four types: network pruning, tensor decomposition, network quantization and knowledge transfer. Among them, the network pruning techniques include fine-grained pruning, vector-level pruning, kernel-level pruning, group-level pruning and filter-level pruning. The tensor decomposition techniques include singular value decomposition, Tucker decomposition, CP decomposition, block term decomposition and tensor-train decomposition. As for the network quantization techniques, they include binary quantization, ternary quantization, multi-bit quantization and Hash quantization. Additionally, the various algorithms of NAS and their discrepancies are analyzed from three orientations: search space, search strategy and performance evaluation strategy, and some latest algorithms of NAS are summarized on this basis. Based on mathematical principles of various optimization algorithms, this paper provides a comprehensive classification of network compression and accelerated network architecture optimization algorithms. Finally, the paper analyses the strategy of NAS algorithms, reviews recent development of CNNs applications in the fields of video object segmentation, medical imaging and face recognition and illustrates the challenges and prospects of the CNNs architectures and their optimizations.

The rest of this paper is arranged as follows: Sect. 2 introduces the CNNs architectures. Section 3 overviews various classifications of existing CNNs architecture features. Section 4 summarizes relevant research ideas and methods for CNNs architecture optimization. Section 5 discusses the strategies and partial recent advances of NAS algorithms.

Table 1 Comparison and analysis of different reviews

Reviews	Years	Research Targets			Contribution
		Architecture	Compression	Applications	
LeCun et al. (2010)	1989–2009	Yes	No	No	Summarized new unsupervised learning algorithms, new non-linear stages, and applications to object detection and vision navigation for off-road mobile robots
Srinivas et al. (2016)	1962–2016	Yes	No	No	General principles behind CNNs; Various modifications to suit different problems
Zhang et al. (2019)	1985–2018	No	Yes	No	CNNs acceleration and compression algorithms are summarized, and hardware-oriented acceleration methods were organized
Choudhary et al. (2020)	1962–2019	Yes	Yes	No	Discussed compression techniques for feed-forward NN and popular efficient CNNs architectures
Khan et al. (2020)	1959–2020	Yes	No	Yes	Summarized the history of CNNs development and the evolution of CNNs architecture; propose a taxonomy of architecture based on basic features
Alzubaidi et al. (2021)	1962–2021	Yes	No	Yes	Described the concepts, theory, and state-of-the-art architectures of CNNs
Ours	1959–2021	Yes	Yes	Yes	Typical approach to CNNs architecture and network compression classification; Summary of architectures or algorithms based on comparison and analysis; Discusses the strategy of NAS algorithms and the application of CNNs

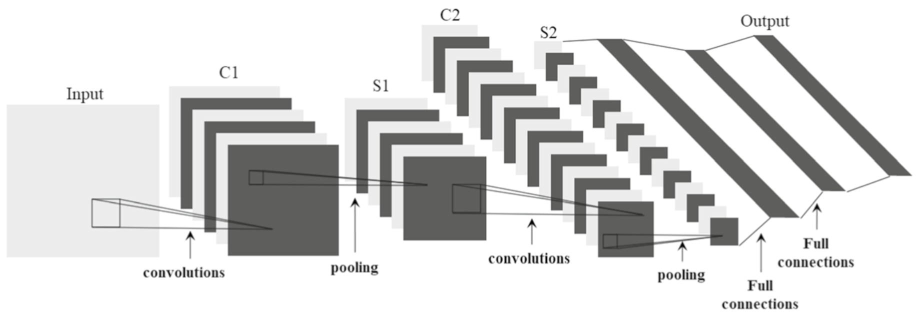


Fig. 1 Basic architecture of LeNet-5

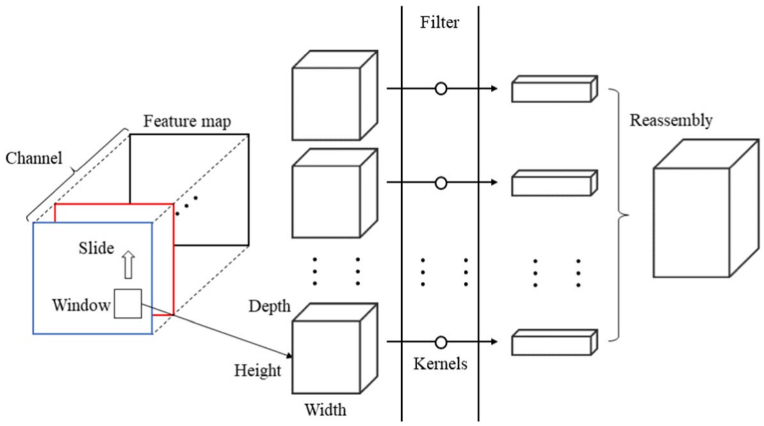


Fig. 2 Working diagram of convolution layer

Section 6 concludes with some applications of CNNs. Section 7 clarifies the future development direction of CNNs, as well as the current challenges.

2 CNNs architectures

Architecture of a typical CNNs usually includes the alternation between convolution and pooling layers. In the case of basic LeNet-5 architecture shown in Fig. 1, the CNNs architectures consist of four parts: the input layer, the convolution layer, the pooling layer, the FC layer and the output layer. CNNs architectures play an important role in the design of neural network architectures, since a more reasonable network architecture can enhance the fitting effect between layers or reduce redundant computations in the network, which usually signifies that it can bring more superior performance.

2.1 Convolution layer

Convolution layer comprises a set of convolution kernels, whose working diagram is displayed in Fig. 2. The process is aims to slide a predefined fixed-size window on the feature map, extract adjacent feature tiles at various positions step by step, and perform tensor product of each feature tile with the learned weight matrix convolution kernel, followed by reorganization of obtained vector space to acquire a new tensor.

The mathematical formula between the input and output of convolution operation is:

$$x_j^l = f^l\left(u_j^l\right) \tag{1}$$

$$u_j^l = \sum_{i \in M_j} x_i^{l-1} * k_{ij}^l + b_j^l \tag{2}$$

where x_j^l represents the output of jth channel in the convolution layer l, $f^l(.)$ stands for the activation function of convolution layer, u_j^l represents the net activation of jth channel in the convolution layer l, M_j represents the feature map subset sampled by sliding window in the feature map, and k_ij^l, b_j^l respectively denote the convolution kernel matrix of convolution layer l and the bias of feature map.

2.2 Pooling layer

Pooling layer, which is also known as the down-sampling layer, generally includes max-pooling, mean-pooling and stochastic pooling. Max-pooling takes the maximum value of feature points in the neighborhood, mean-pooling takes the mean value of feature points in the neighborhood, while stochastic pooling takes the value of random feature points in the neighborhood. The mathematical formula between the input and output of pooling operation is:

$$x_j^l = f^l\left(u_j^l\right) \tag{3}$$

$$u_j^l = \alpha_j^l \cdot f_{pooling}^l\left(x_j^{l-1}\right) + b_j^l \tag{4}$$

where x_j^l and u_j^l respectively denote represent the output and net activation of jth channel in the pooling layer l, $f^l(\cdot)$ stands for the activation function connected to pooling layer l, α_j^l denotes the weight coefficient of pooling layer l, and $f_{pooling}^l$ represents the pooling function of pooling layer l. The pooling layer (blocks $S1$ and $S2$ in Fig. 1) reduces the number of elements processed in the feature map through down-sampling, and introduces a hierarchical spatial filter structure by gradually increasing the window of continuous convolution layer. The convolution layer can significantly reduce the network parameters and prevent the network overfitting.

2.3 Activation function

Activation functions play a decision-making role in the CNNs, which are conducive to learning nonlinear complex patterns. They are primarily applied to introduce nonlinear factors into the neural networks for enhancing their fitting ability. Table 2 details the typical activation function types in CNNs, including sigmoid, tanh, rectified linear unit (ReLU) (Nair and Hinton 2010) and its variants [Leakly ReLU (Maas et al. 2013), Parametric ReLU (He et al. 2015b), Randomized ReLU (Xu et al. 2015), Exponential

Table 2 Activation function summary

Function	Definition	Parameter of α
Sigmoid	$f(x) = \frac{1}{1+e^{-x}}$	–
Tanh	$f(x) = \frac{2}{1+e^{-2x}} - 1$	–
ReLU	$f(x) = \begin{cases} x & x \geqslant 0 \\ 0 & x < 0 \end{cases}$	–
LeakyReLU	$f(x) = \begin{cases} x & x \geqslant 0 \\ \alpha x & x < 0 \end{cases}$	$\alpha \in (0, 1)$
PReLU	$f(x) = \begin{cases} x & x \geqslant 0 \\ \alpha x & x < 0 \end{cases}$	Learned parameter
RReLU	$f(x) = \begin{cases} x & x \geqslant 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$	Uniform(a, b)
ELU	$f(x) = \begin{cases} x & x \geqslant 0 \\ x & x < 0 \end{cases}$	Predefined parameter
Switch	$f(x) = x.sigmoid(\alpha x)$	Learned parameter
Maxout	$f(x) = \max\limits_{j\in[1,k]} x^T W + b_{ij}\ W \in R^{d\times m\times k}$	–

Linear Unit Clevert et al. (2015)], Switch and Maxout. Obviously, compared to functions like sigmoid and tanh, excessively large and small inputs do not make ReLU tend to be saturated. Thus, ReLU and its variants are superior to conventional activation functions such as sigmoid and tanh in overcoming the vanishing gradient problem (Gulcehre et al. 2014). Regarding the Maxout activation function, it avoids the neuron death and other problems by the basis of retaining the linearity and unsaturation advantages of the ReLU functions.

2.4 Fully connected layer

Fully connected (FC) layer is generally located at the last part of a CNNs, which is adopted for converting the two-dimensional (2D) feature information output by the previous layer into the one-dimensional (1D) classification information. It resembles the hidden layer of multilayer perceptrons (MLPs), where the output is obtained through weighted combination of FC neurons in the previous layer. The mathematical formula between the input and output operation of each neuron is:

$$x_j^l = f^l\left(u_j^l\right) \tag{5}$$

$$u_j^l = max(0, \sum_i y_i^l \cdot w_{ij}^l + b_j^l) \tag{6}$$

where x_j^l and u_j^l respectively represent the output and net activation of the jth channel in the FC layer l, $f^l(\cdot)$ stands for the activation function connected by the pooling layer l, w_{ij}^l and b_j are respectively the weight coefficient and bias of FC layer l, and max $(\cdot)$ denotes the largest number in the selected array range.

3 Typical CNNs network structure development

Architectures of CNNs have undergone over 30 years of research and continuous development. The primary impetus for their performance improvement comes from the design of advanced modules and extensive recognition capabilities. The advanced CNNs modules focus on improving, redesigning and modularizing the CNNs architectures, and thus the networks can be optimized in terms of depth, width and space utilization. Meanwhile, the more extensive recognition capabilities are responsible for attaining better network efficiency in derivative applications such as image and video recognition. Figure 3 displays a chronological summary of various CNNs architectures, classifications and improvement processes, which is based on the ideas proposed by various networks and their respective architectural characteristics. In this study, the development history of typical CNNs architectures are explored by classifying existing CNNs into basic network architecture, convolution splitting, cross-layer connection idea, depthwise separable convolution, object detection CNNs and transformer encoder.

3.1 Basic network architecture

The core idea of CNNs is to share local receptive fields and weights. LeNet, which was the first CNNs architecture proposed in 1998 by LeCun et al. (Hinton and Salakhutdinov 2006), is considered the prototype of CNN. It adopts an architecture where five convolution layers are directly connected to the pooling layer. Originally, LeNet was designed to recognize handwritten and printed characters, which achieved excellent efficiency. The currently common LeNet has been modified from the original model. Specifically, the nonlinear transformation was transferred from the down-sampling layer to the convolution layer, and the activation function of output layer was changed from the radial basis function to

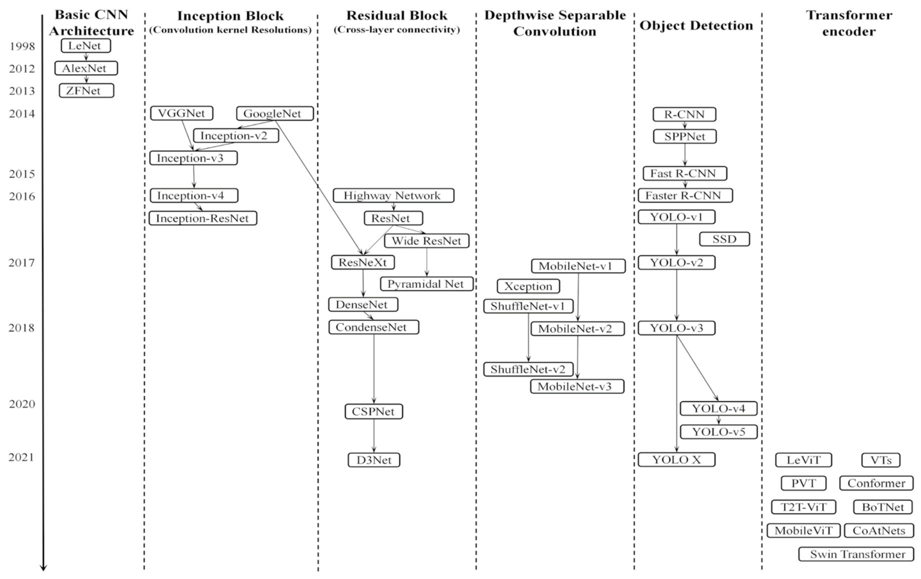


Fig. 3 Development diagram of convolutional neural network

the softmax function. Performance comparison and analysis of basic network architecture are summarized in Table 3.

Although the inception of deep learning is universally acknowledged to be marked by Hinton et al.'s proposal of the concept in 2006 (Hinton and Salakhutdinov 2006), it has received widespread attention from academia and industry since the success of AlexNet developed in 2012 by Krizhevesky et al. in image processing. Figure 4 presents the architecture diagram of AlexNet (Krizhevsky et al. 2012). The network consists of five convolution layers and three FC layers. Each convolution layer contains the activation function ReLU, as well as the local response normalization (LRN) and the pooling (down-sampling) processing.

AlexNet is regarded as the first deep CNNs architecture, which enhances the learning ability of CNNs by deepening its depth and applying numerous parameter optimization strategies. Compared with the previous CNNs models, AlexNet uses ReLU instead of the saturated nonlinear function tanh, which reduces the computational complexity of the model in alleviating the vanishing gradient problem. Besides, through overlapping pooling, it allows superposition of pooling windows, thereby effectively slowing down overfitting. To improve the CNNs performance, AlexNet performs LRN processing on certain layers. The computational formula for such normalization processing is:

$$b^i_{x,y} = a^i_{x,y} \Big/ \left(k + \alpha \sum_{j=max(0,i-n/2)}^{\min(N-1,i+n/2)} \right) \tag{7}$$

where $b^i_{x,y}$ denotes the LRN value calculated at the position (x, y) across convolution surface, $a^i_{x,y}$ denotes the value at the position (x, y) on convolution surface, N is the total number of convolution surfaces, n is the number of adjacent surfaces, and k, α, β all denote the parameters under adjustment.

Architectures of NNs, including CNNs, generally lack interpretability, which makes the architectural design a major challenge in the CNNs development. In 2013, Zeiler and Fergus proposed a deconvolutional neural network (DeconvNet), which is well-known as ZfNet (Zeiler and Fergus 2014). Through visualization technology, Zeiler et al. explored the implementation of internal feature layer. Figure 5 depicts the internal operating mechanism, where the left half shows a deconvolution layer, and the right half displays a convolution layer.

The deconvolution layer reconstructs an approximate version of convolution features from the next layer. For quantitative visualization of the network performance, DeconvNet adopts deconvolution and depooling operations. It is of note that depooling is unachievable in theory. The authors approximately realized depooling by recording the position of maximum value in each pooling region through the variable switch conversion. Such reverse mapping projects the output of convolution layer back to a visually perceptible image pattern, thereby explaining the internal features learned in each layer at the neuron level.

3.2 Convolution splitting

The universal approximation theorem proposed by Csĵi points out that a single hidden layer is sufficient to approximate any function, albeit at the cost of multiplying many neurons, indicating that the theorem almost do not hold in practice (Csáji 2001). In 2013, based on experience, Bengio et al. demonstrated based on experience that with increase in the network depth, the network can better approximate objective function

Table 3 Performance comparison and analysis of basic network architecture

Name	Year	Weight (M)	Accuracy MNIST	(% *Top*5) ImageNet	Contribution	Gaps
LeNet	1998	0.06	99.2	–	The first CNNs architecture	Low-level performance Poor efficiency
Alex	1998	60	–	83.6	Introduced ReLU, Dropout and overlap	Large convolution kernel size
ZfNet	1998	60	–	83.6	Architectural visualization	Visual processing consumes extra performance

through numerous nonlinear mappings and improved feature representations (Bengio 2013). Additionally, the outstanding performance of AlexNet has also proven that the performance of the network can be improved by increasing its depth. However, with the increasing number of network layers, the ever-increasing computational burdens and insignificant performance improvements have made more advanced network architectures another major research direction. Accordingly, the idea of convolution splitting has emerged as a crucial solution to expand the network depth. Performance comparison and analysis of convolution splitting architecture are summarized in Table 4

Simonyan et al. were the first to propose a CNNs architecture design principle similar to convolution splitting, and named it VGG (Simonyan and Zisserman 2014). It won the second place in the 2014-ILSVRC competition. VGG can be regarded as a deepened version of AlexNet, which uses several consecutive 3×3 convolution kernels to replace the larger kernels $(11 \times 11, 7 \times 7, 5 \times 5)$ in the network. VGGNet divides the network into five segments, with every segment connecting multiple 3×3 convolution kernels together. There is a maximum pooling layer following each segment of convolution, and at last, three FC layers and a softmax layer are added.

Lin et al. proposed a nonlinear architecture Mlpconv in 2013, and stacked the Mlpconv layers to constitute a network in network (NiN) (Lin et al. 2013). Figure 6 illustrates the Mlpconv architecture, where an MLP is added behind the convolution kernel. Due to the strong fitting ability of MLP, the Mlpconv architecture enhances the nonlinear expressiveness of the network and the feature recognition capability of local perceptual field in comparison with the conventional CNNs convolution process. Meanwhile, the NiN also replaces the conventional CNNs FC layer with a global average pooling layer.

Inspired by the idea of replacing each NiN layer with the Mlpconv architecture, Szegedy et al. (2015) proposed to replace the conventional convolution layer with small block convolution in 2015, proposing a module architecture called Inception-v1, whose network name was GoogLeNet. GoogLeNet won the 2014-ILSVRC championship. Figure 7a displays the original architecture of the Inceptionv1 module, where several $1 \times 1, 3 \times 3$ and 5×5 convolutions are stacked together with 3×3 sized pooling operations for increasing the network width, as well as the ability to extract features of different sizes. Figure 7b presents the updated Inceptionv1 architecture. Since the original module significantly increases the computational burden, the updated Inceptionv1 module adds 1×1 convolution layers before the $3 \times 3, 5 \times 5$ convolutions and after the pooling layer, which are applied respectively to compress the numbers of input and output image channels. GoogLeNet introduces a new concept of Inception block into the CNNs, which integrates multi-scale convolution and transformation through the splitting, transformation and merging

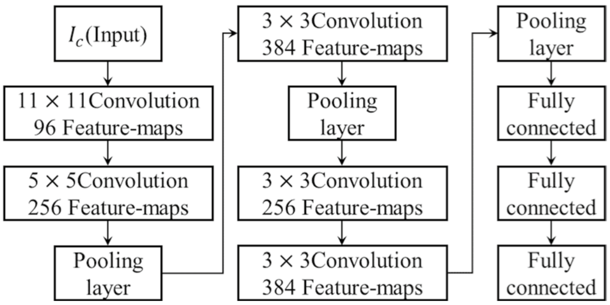

I_c(Input)
11 × 11Convolution 96 Feature-maps
5 × 5Convolution 256 Feature-maps
Pooling layer
3 × 3Convolution 384 Feature-maps
Pooling layer
3 × 3Convolution 256 Feature-maps
3 × 3Convolution 384 Feature-maps
Pooling layer
Fully connected
Fully connected
Fully connected

Fig. 4 AlexNet architecture

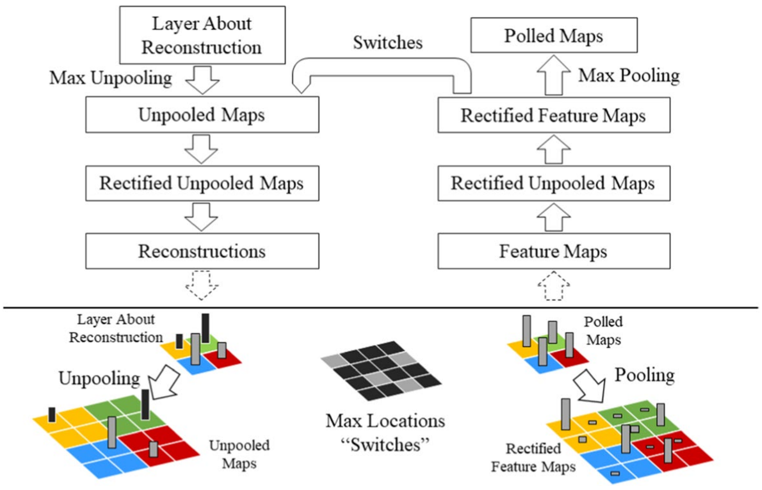


Fig. 5 Schematic of ZfNet operating mechanism

approaches. This enables the CNNs to attain high accuracy while lowering the computational cost.

In the same year, Ioffe et al. proposed the Inceptionv2 (also known as BN-Inception (Ioffe and Szegedy 2015)] architecture by making improvements to the Inceptionv1 (Ioffe and Szegedy 2015). As shown in Fig. 8a, the Inceptionv2 splits the large convolution kernel (5×5) in the original Inceptionv1 module into two small convolution kernels (3×3). Its greatest contribution, compared with the Inceptionv1, is the introduction of the batch normalization (BN) concept. With BN, the output of a certain network node is processed to make it approximate normal distribution with a mean of 0 and a variance of 11, thereby alleviating the vanishing gradient and gradient explosion problems in backpropagation.

By borrowing the convolution kernel decomposition idea of VGGNet, Szegedy et al. (Ioffe and Szegedy 2015) proposed the Inceptionv3 architecture in 2016, aiming to reduce the computational cost of deeper networks without affecting generalization. The Inceptionv3 introduces asymmetric convolution kernels, and Fig. 8b describes its strategy for convolution kernel decomposition. It decomposes an $n \times n$ convolution kernel into $n \times 1$ and $1 \times n$. Such operation further reduces the number of network parameters. Based on the

Table 4 Performance comparison and analysis of convolution splitting architecture

Name	Year	Accuracy (% *Top*5) ImageNet	Contribution	Gaps
VGG	2014	92.7	Replace large size with small size kernel	Excessive computation in the fully connected layer
GoogLeNet	2015	93.3	Introducing convolution splitting	Inefficient block structure
Inceptionv2	2015	95.1	Introduction of the batch normalization	Optimizable convolutional layers
Inceptionv3	2015	96.5	Introduced asymmetric convolution kernels	Complex architectural design hard-to-adjust hyperparameters
Inceptionv4	2016	95.9	Modular network construction	Expensive computational costs
Inception-ResNet	2016	96.5	Integration of Inception and residual modules	Optimizable multi-size feature extraction

calculations, the use of $1 \times 3, 3 \times 1$ asymmetric stacked convolutions reduces the parameter number by 28% than decomposing the 3×3 convolution kernel into two 2×2 convolution kernels. Compared with Inceptionv2, the Inceptionv3 also replaces the 7×7 and 5×5 large filters with the $1 \times 7, 7 \times 1$ and $1 \times 5, 5 \times 1$ filter stacks. Additionally, the Inceptionv3 applies auxiliary classifiers to accelerate the convergence of CNNs training, which achieves a performance improvement of a 04% top-1 accuracy.

In the same year, Szegedy et al. put forward a series of local network architectures including Stem, Inception-A, Inception-B, Inception-C, Reduction-A and Reduction-B (Szegedy et al. 2017), which were integrated into the Inceptionv4 module. Besides, they proposed Inception-ResNet by incorporating residual learning into GoogLeNet. Experimentation revealed that the Inception-ResNet with residual connection had the same generalization ability as the ordinary Inceptionv4, albeit the increased depth and width. Moreover, the Inception-ResNet converged faster than the Inceptionv4, which more directly proves that the use of residual connections can significantly accelerate the Inception network training. Inceptionv4 and Inception-ResNet achieve faster training and better performance. Among them, the Inceptionv4 has 70 layers, and its error rate on ImageNet is 4.01. Besides, the The Inception-ResNet has 572 layers, and its error rate on ImageNet is 3.52%.

3.3 Cross-layer connection idea

Generally, conventional CNNs are constituted by connections between convolution, pooling and FC layers, where the inter-layer information is transferred only to the adjacent layers. Such architecture not only limits the fitting ability of CNNs, but also worsens the fitting effect as the network depth increases. With deepening of the network architecture, the gradient vanishing, explosion or degradation problems have become increasingly severe. The idea of cross-layer connection is an effective solution to the existing problems, which is well-know for allowing the networks to deliver information between non-adjacent layers. Performance comparison and analysis of convolution splitting architecture are summarized in Table 5

Srivastava et al. proposed a deep CNNs called Highway Networks in 2015 (Srivastava et al. 2015), which achieves unimpeded information flow across various layers by

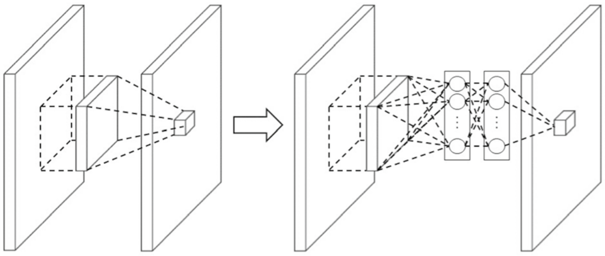


Fig. 6 MLP network architecture

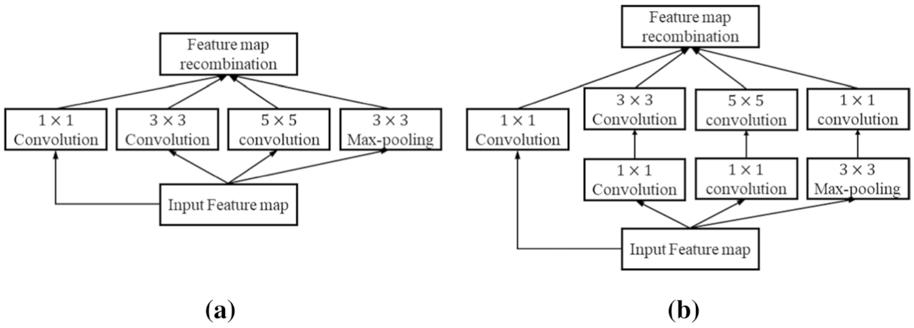


Fig. 7 Inceptionv1 architecture: **a** Original Inceptionv1 architecture, **b** updated Inceptionv1 architecture

introducing new cross-layer connections and assigning two gated units in the layers. The gating mechanism is inspired by the long short-term memory (LSTM)-based recurrent neural networks (RNNs) (Mikolov et al. 2010; Sundermeyer et al. 2012). By combining the information of the lth layer and its previous *l*–*k* layers, the information aggregation is accomplished to generate a regularization effect, thereby facilitating the gradient-based training of deep networks. In this way, a stochastic gradient descent (SGD) algorithm can be used to train a network with over 100 layers or even up to 900 layers.

Some networks also encounter a degradation problem during training, which results in the quick saturation of accuracy, and the increasing error rate with deepening level. The error rate arising from such problem is attributed merely to the increased number of network layers, rather than overfitting (He and Sun 2015). In 2016, He et al. proposed ResNet by exploiting the bypassing paths applied in Highway Networks, and introduced the concept of residual learning to address the degradation problem (He et al. 2016). Figure 9 displays the architecture of ResNet residual blocks. ResNet inherits the cross-layer connection idea of Highway Networks. The difference refers to that its gating mechanism is unimpeded at all times rather than learnable, which helps greatly reduce the network complexity. Through shortcut connections, ResNet transfers the input across layers and adds it to the convolution result, and thus the underlying network is fully trained to significantly improve the accuracy. On the Microsoft Common Objects in Context (MS COCO) dataset, ResNet shows a 28% improvement.

Although the residual module improves the network expressiveness, it still has several shortcomings, such as the time-intensive training, the inactivation of many feature maps (feature reuse problem), as well as the gradient vanishing and explosion problems. Thus, some researchers believe that widening the network layers may provide more effective performance improvement than deepening the networks. To address the above problems, Wide

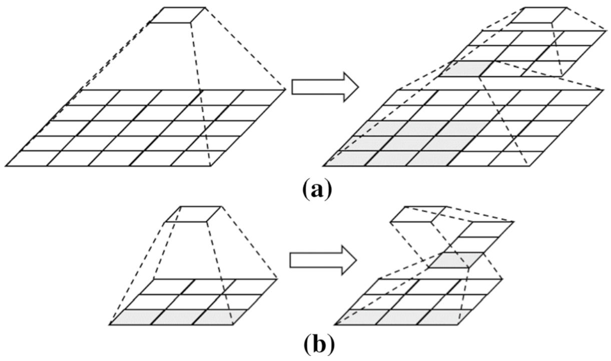


Fig. 8 Convolution kernel decomposition: **a** large convolution kernel (5 × 5) is split into two small convolution kernels (3 × 3), **b** asymmetric convolution kernel splitting

ResNet was proposed by Zagoruyko, Komodakis et al. in 2016, which utilizes the function of residual blocks by making ResNet wider instead of deeper (Zagoruyko and Komodakis 2016b). Besides, an additional factor k is introduced for controlling the network width. Empirical research has demonstrated that the Wide ResNet can achieve better training of deep networks, even though its parameter number is twice that of the ResNet.

Han et al. developed Pyramidal Net in 2017, which increases gradually the width of each residual unit in contrary to the drastic decrease of spatial width caused by the increasing depth in the ResNet (Han et al. 2017). Such strategy enables the pyramid network to cover all possible positions, rather than keeping the same spatial size within each residual block until down-sampling. In the pyramid network, the depth of feature maps is adjusted by a factor l and calculated according to the following formula:

$$f(x) = \begin{cases} 6 & l = 1 \\ \left\lfloor d_{l-1} + \frac{\lambda}{n} \right\rfloor & 2 \leqslant l \leqslant n+1 \end{cases} \tag{8}$$

where d_l denotes the number of dimensions of the lth residual unit, n signifies the total number of residual units, and λ is the step factor. Moreover, λ/n regulates the increase in depth. Obviously, the primary problem with the pyramidal network lies in the quadratic increases in both space and time with increasing width.

In 2017, Xie et al. pointed out that improving the model accuracy by conventional widening and deepening approaches can considerably increase the network complexity, especially when there are many design factors and hyperparameters. Hence, by drawing on the ideas of Inception module and residual network, Xie et al. introduced the concept of cardinality to ResNeXt, a new network architecture (Xie et al. 2017). As an additional dimension, cardinality refers to the size of transform set. Through experimentation, Xie et al. found through experimentation that increasing the cardinality can improve the classification accuracy while maintaining complexity. Moreover, compared with the increasing the depth and width, increasing the cardinality value achieves better accuracy improvement. Figure 10 illustrates the architectures of ResNet block and the ResNeXt block with cardinality = 32. ResNeXt further broadens the network architecture, improves recognition accuracy without increasing network complexity, and reduces the number of hyperparameters. According to experimental findings, the 101-layer ResNeXt (ResNeXt-101) can attain better accuracy than ResNet-200, but with only 50% of its complexity.

DenseNet is a CNNs architecture with dense connections, which draws on the quintessential idea of cross-layer connections in ResNet (Huang et al. 2017). As shown in Fig. 11, each layer in DenseNet is connected to every other layer in a feed-forward manner, thereby strengthening the effect of cross-layer deep convolution more thoroughly. The $H_l(\cdot)$ block in DenseNet is a composite function BN-ReLU-Conv, which is composed of three successive operations, namely, BN, ReLU and 3×3 convolution (Conv). The basic modules of DenseNet are called Dense Blocks, each of which takes the feature maps of all previous Dense Blocks as input. Unlike ResNet, which applies direct summation, the Dense Blocks merge multiple previous inputs in a concatenated manner. Such connection method allows more effective flows of information and gradient between networks, which significantly reduces the number of DenseNet parameters. Gao et al. modified the DenseNet to propose a lightweight network CondenseNet, which integrates such methods as learned group convolution, dense connection and pruning (Huang et al. 2018). The authors added a new index layer for implementing grouped convolution of subsequent convolution layers, and modified the input channel number of network convolution layers to grow exponentially. Moreover, the CondenseNet extends the application of dense connection mode adopted in the dense blocks by DenseNet to every layer of the network. Experimentation has shown that CondenseNet outperforms the cutting-edge lightweight networks MobileNet and ShuffleNet.

In 2019, Wang et al. considered that the network of the DenseNet has a large amount of gradient information being repeatedly used to update the weights of different Dense blocks, which means that the Dense blocks repeatedly learn the same information (Wang et al. 2020a). Considering that, they proposed a network structure called the CSPNet that maximizes the difference in the gradient combination by adding the partial transition layer. Figure 12 shows the network model of CSPNet structure combined with multiple CNNs structures. In 2021, Takahashi et al. introduced a densely connected multidilated (D3Net) network architecture for tackling the task of high-resolution dense prediction (Takahashi and Mitsufuji 2021), which combines multi-layer convolution with the DenseNet architecture to obtain exponentially growing perceptual fields at each layer.

3.4 Depthwise separable convolution

Some highly performing CNNs methods unavoidably bring huge computational costs, which often require support from the high-performance GPUs (Jia et al. 2013; Krizhevsky et al. 2012) or the highly optimized distributed CPU architectures (Vanhoucke et al. 2011). Despite the expansion of CNNs application towards the mobile terminal, most mobile devices do not have powerful computing power or huge memory space. Hence, research on lightweight network architectures is needed, which helps properly handle the above problems. Lightweight CNNs generally refer to smaller CNNs architectures obtained after compression and acceleration, whose features are as follows:

1. Small communication requirements with the server;
2. Fewer network parameters and low model data volume;
3. Suitable for deployment on memory-constrained devices.

Table 5 Performance comparison and analysis of cross-layer connection architecture (all inference are performed on ImageNet)

Name	Year	Parameters (M)	Accuracy (% $Top1$)	FLOPs	Contribution	Gaps
ResNet-10	2016	5.24	63.5	10.11B	Introduction of residual module	Feature information is dropped in the feed-forward process
ResNet-50		25.6	75.9	–		
ResNet-101		44.5	77.6	–		
ResNet-152		60.2	77.8	22.6B		
Wide ResNet	2016	68.9	78.1	–	Wider network architecture	Prone to overfitting
Pyramidal Net	2017	27.0	–	–	Gradual network widening	Expensive calculation costs
ResNeXt-50	2017	22.19	77.8	10.11B	Introduction of group convolution	Expensive calculation costs
DenseNet-201	2017	20.01	77.4	8.63B	Global cross-layer information flow	Repeated gradient information calculation
DenseNet-264		27.21	77.8	11.03B		
CSPResNet10	2019	2.73	65.3	1.905B	Maximizing the gradient difference of fusion	Optimizable multi-scale feature extraction
CSPResNeXt50		20.50	77.9	7.93B		
CSPDenseNet20		23.07	77.7	8.47B		

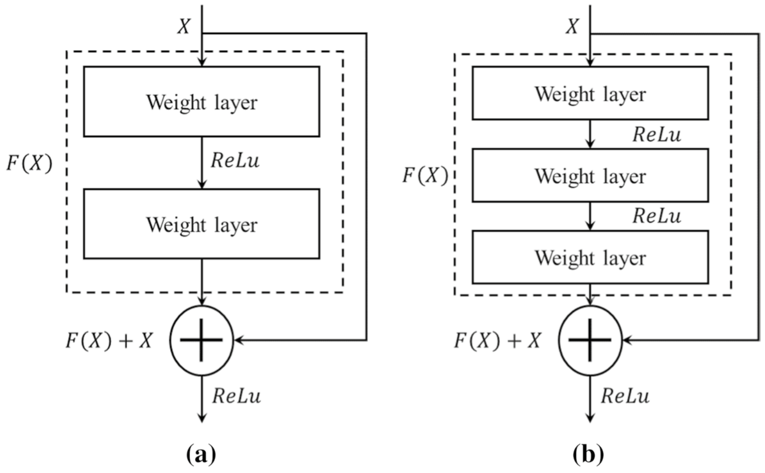
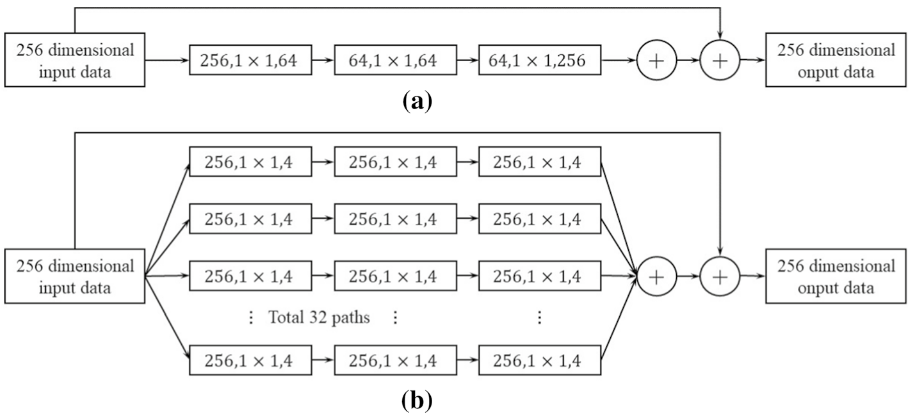


Fig. 9 Residual module architectures: **a** connection across two layers, **b** connection across three layers

In order to make the CNNs meet the demand while maintaining the performance, the architecture of depthwise separable convolution is proposed. Besides, performance comparison and analysis of depthwise separable convolution are summarized in Table 6.

In 2017, MobileNetv1 proposed by Howard et al. decomposes the conventional convolution process into two steps, namely the Depthwise convolution and the Pointwise convolution, which achieves substantial compression regarding both the model size and the computational burden. Lightweight networks constructed therefrom can run on embedded mobile devices (Howard et al. 2017). By combining separable convolutions, Sandler et al. put forward an inverted residual with linear bottleneck, based on which MobileNetv2 was constructed, with both speed and accuracy superior to the MobileNet (Sandler et al. 2018). Andrew et al. proposed MobileNetv3, showing the improvements in both network architecture and algorithm (Howard et al. 2019). They combined the platform-aware NAS and the NetAdapt for network architecture search. Meanwhile,



256 dimensional input data
256,1 × 1,64
64,1 × 1,64
64,1 × 1,256
256 dimensional onput data
(a)
256 dimensional input data
256,1 × 1,4
Total 32 paths
256 dimensional onput data
(b)

Fig. 10 Architecture of ResNet block: **a** block of ResNeXt, **b** block of ResNeXt with cardinality = 32

a modified activation function h-swish was put forward, which greatly reduces the computational burden while retaining the high accuracy of swish activation function. Compared with MobileNetv2, the accuracy of MobileNetv3 in ImageNet classification increases by 3.2%, and the latency is reduced by 15%. Moreover, at almost identical accuracies, the running speeds of MobileNetv3 on the MS COCO dataset and in the Cityscapes semantic segmentation task increase by 25% and 30%, respectively.

Xception, which can be considered an extreme Inception architecture, borrows the idea of deep separable convolution introduced in AlexNet (Chollet 2017). Figure 13 depicts the Xception module, which widens the original Inception, and uses a layer of 3×3 convolution kernel closely following 1×1 to replace different spatial dimensions $(1 \times 1, 3 \times 3, 3 \times 3)$, thereby adjusting the computational complexity. Initially, the input feature map is processed by a 1×1 convolution kernel. Then, convolution operation is performed on each channel of output feature map using a 3×3 convolution kernel. Finally, all the outputs are spliced together to obtain a new feature map. Despite fewer training parameters than the Inceptionv3, the Xception network has the same recognition accuracy and training speed as the Inceptionv3, which also has better performance on larger datasets.

Zhang et al. developed two novel operation methods, i.e. pointwise group convolution and channel shuffle, and named them ShuffleNet (Sandler et al. 2018). Acccording to the experiments on ImageNet classification and MS COCO object detection, ShuffleNet's top-1 error on ImageNet classification tasks was 7.8% lower than that of MobileNet under a computational budget of 40 MB, thus allowing the CNNs to run on an extremely limited number of mobile devices. In 2018, Ma et al. proposed a new architecture called ShuffleNetv2 (Ma et al. 2018). Ma et al. argued that: (1) ShuffleNetv1 makes extensive use of 1×1 group convolution to increase the memory access cost (MAC). (2) ShuffleNetv1 uses a bottleneck layer similar to ResNet, where the number of input and output channels are different, and the same channel size minimizes the memory access. The bottleneck layer is similar to ResNet, where the number of input and output channels are different, and the same channel size minimizes the memory access. (3) The multiplex structure in ShuffleNetv1 causes network fragmentation and reduces the parallelism of the model. (4) There are a large number of element-wise operators such as ReLU and Add in the short circuit connection. Therefore, Ma et al. introduced a new operation: channel split. Instead of Add elements, the outputs of two branches are concat together and the results are then channel shuffled. compared to ShuffleNetv2 is more accurate with the same complexity than ShuffleNetv1 and MobileNetv2.

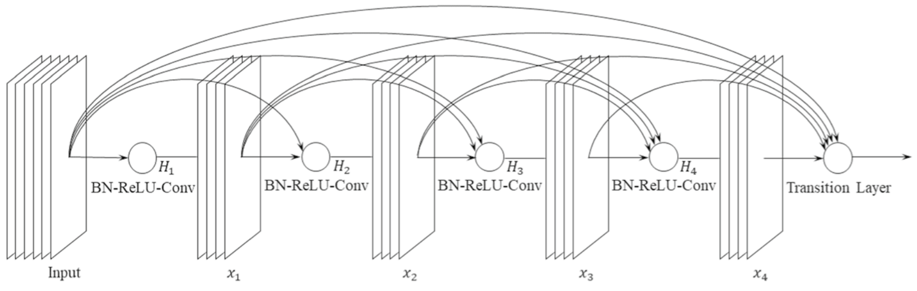


Fig. 11 DenseNet architecture

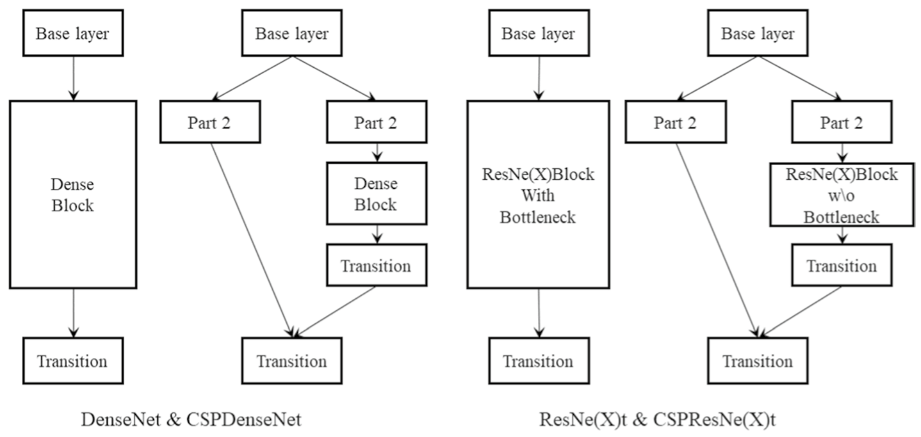


Fig. 12 DenseNet architecture

3.5 Object detection CNNs

Owing to the remarkable performance of CNNs in image classification problem, its application in the direction of object detection has been expected. Unlike the image classification problem, the object detection problem requires detection and localization of multiple specific objects from images. The majority of conventional object detection methods are based on image recognition. Over the past decade, the traditional machine vision field has often used feature descriptors to handle object recognition tasks. The most common of these feature descriptors include the scale-invariant feature transform (SIFT) (Lowe 2004) and the histogram of oriented gradient (HOG) (Dalal and Triggs 2005). Meanwhile, the emergence of AlexNet has made CNNs the mainstream method of object detection. Currently, the basic idea of CNN-based object detection is to propose candidate regions prior to classifying them with a CNN. The current section primarily describes the formation and development of object detection CNNs.

Inspired by AlexNet, Girshick et al. attempted to generalize AlexNet's ability in ImageNet object recognition to the PASCAL Visual Object Classes Challenge (VOC) in 2014, also proposing a model of regional CNNs (R-CNN) (Girshick et al. 2014). Figure 14 illustrates the object detection process with R-CNN. To achieve object detection, the input image goes through three modules, respectively, region proposals, feature extraction and region classification. Relevant ideas are:

1. In region proposals, selective search (Uijlings et al. 2013) is employed to extract region proposals (approximately 2000), which are combined with CNN.
2. Feature extraction is responsible for extracting eigenvectors with a fixed length of 4096 from each region proposal, and for calculating the features of each region proposal based on CNN.
3. In region classification, each box is classified and regressed by bounding box regression and SVMs.

Table 6 Performance comparison and analysis of depthwise separable convolution

Name	Year	Params	Accuracy (% $Top1$) ImageNet	Contribution	Gaps
MobileNetv1	2017	4.2 M	70.6	Replacing ordinary convolution with depthwise separable convolution	Invalid convolution kernels exist in Depthwise Convolution
ShuffleNetv1	2017	3.4 M	71.5	Introduction of pointwise group convolution and channel shuffle	Boundary effect generation
Xception	2017	22.8M	79.0	Combining convolution splitting and depthwise separable convolution	Expensive computational costs
MobileNetv2	2018	3.4 M	72.0	Introduction of inverted residual and linear bottleneck	Optimizable feature extraction efficiency
ShuffleNetv2	2018	2.3 M	69.4	Introduction of pointwise group convolution and channel shuffle	Significant feature loss
MobileNetv3	2019	5.4 M	75.2	Introduction of inverted residual and linear bottleneck	–

Such methodology not only enhances the efficiency of object detection, but also improves the detection accuracy. On VOC2007, the mean average precision (mAP) attained by DPM HSD increases to 66% from 34.3%.

Despite the high accuracy in object region detection, R-CNN still has obvious shortcomings. Most CNNs, including R-CNN, are limited to accepting fixed-size inputs. In fact, however, the convolution layers of CNNs can generate feature surfaces of arbitrary size. Such limitation is subject to the requirement for fixed-length inputs by the FC layers. Additionally, the training of R-CNN is a complicated and redundant process. Both SVMs and bounding box regression learning require extraction of features for each region proposal and their storage in the hard disk, which is computationally enormous and consumes the storage space. In 2015, He et al. (2015a) proposed a SPPNet model, whose architecture is presented in Fig. 15. The model adds a spatial pyramid pooling (SPP) layer between the last convolution layer and the first FC layer of a CNN.

The SPP consists of three layers in total. Depending on the resolution of spatial blocks partitioned by the input feature map, it can be divided into a low-resolution layer, a medium-resolution layer, and a high-resolution layer. The three layers partition the input feature map into 1, 4 and 16 spatial blocks, respectively. For feature maps of different sizes, the total number of finally output spatial blocks remains unchanged. Hence, after the connection to the last convolution layer, the SPP layer outputs fixed-length tensors and sends them to the FC layer for the classification of each region proposal. Compared with R-CNN, the testing time of SPPNet can be accelerated by 10×–100×. Obviously, the existing SPP layer allows CNNs inputs of different sizes to produce outputs of the same size, which breaks the limitation that the inputs of CNNs models all have fixed sizes.

Nevertheless, SPPNet still features complicated training and hard disk reading and writing processes. Besides, SPPNet cannot update the convolution layers before the SPP layer during object detection, limiting its recognition accuracy. In response to these problems, Girshick et al. developed Fast R-CNN in 2015, which introduces a special single-layer SPP called RoI pooling without restricting the input size of feature maps, thereby unifying the sizes of feature maps input to the FC layer (Girshick 2015). Figure 16 displays the architecture of Fast R-CNN, where the images and their region proposals are subjected to convolution and maximum pooling operations together as the input, thereby obtaining a shared feature map that contains region proposals. The RoI pooling extracts the region proposals in such feature map as fixed-length eigenvectors. After processing by the FC layer, these

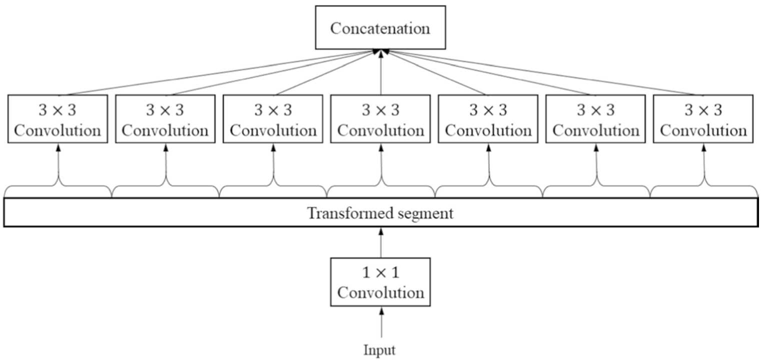


Fig. 13 Xnception module

eigenvectors are sent to two output layers, which are respectively responsible for outputting the softmax probability estimates (K object categories and 1 background category) and the bounding box locations (K object categories). Unlike R-CNN and SPPNet, which employs regression models like linear SVM for bounding box regression, the Fast R-CNN adopts softmax and bounding box regression. Apart from addressing the inability of convolution layers before SPP layer in SPPNet to update the network parameters, Fast R-CNN also exhibits obviously faster training and testing than the SPPNet.

Given consumption of substantial computing resources by Fast R-CNN caused by reliance on candidate region selection, Ren et al. (2015) developed a region proposal network (RPN) for real-time object detection in 2017 to replace the selective search method. As a fully convolutional network (FCN), RPN can obtain a series of objectness scores from images of arbitrary size. Ren et al. combined RPN with Fast R-CNN to constitute Faster R-CNN, which is a network sharing the features of convolution layers. Figure 17 illustrates the object detection process with the Faster R-CNN. Its network comprises two parts: RPN that generates region proposals and a detector that adopts uses the proposed regions. Faster R-CNN achieved the best detection results on the PASCAL VOC2007, 2012 and MS COCO datasets, with the computational process almost being real time.

In 2016, Redmon et al. proposed a new detection strategy for the object detection problem. Unlike previous practices, most detection networks locate objects based on classification, and You Only Look Once (YOLO) is considered as a new detection approach (Redmon et al. 2016). It transforms the object detection problem into a probabilistic regression problem of multiple bounding boxes and related categories. Based on GoogLeNet, YOLO uses a single neural network and a single evaluation to directly predict the bounding boxes and categories from the entire input image. Figure 18 illustrates the object detection idea of YOLO. Initially, the network segments the input image into SS grids, with each grid being responsible for detecting target objects whose center point falls within the grid, and for predicting B bounding boxes and marking the corresponding scores. A high confidence score indicates that the possibility of the object being contained in the bounding box is high. The IoU values of predicted and actual bounding boxes are regarded as the confidence scores, which are dependent jointly on the probability $P_r(object)$ of bounding box containing object and the geometric accuracy IoU_{pred}^{truth} of bounding box. Relevant computational formula is:

$$P_r\left(Class_i|Object\right) * P_r(Object) * IoU_{pred}^{truth} = P_r\left(Class_i\right) * IoU_{pred}^{truth} \tag{9}$$

where $P_r(object)$ denotes the probability of object, IoU_{pred}^{truth} is the geometric accuracy, and $P_r\left(Class_i|Object\right)$ represents the conditional category probability. Every bounding box of YOLO contains five predicted values: x, y, w, h, and *confidence*, of which (x, y) denotes

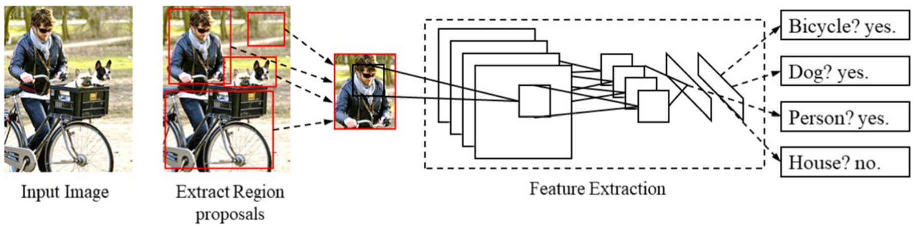


Fig. 14 R-CNN-based object detection process

the coordinate position of bounding box center relative to the grid boundary with *w* and *h* denoting the predicted width and height relative to the entire image. All the four values are proportionality coefficients between 0 and 1. Besides, confidence denotes the confidence score.

Single shot detector (SSD) abandons the current practice of using bounding box hypothesis to resample pixels or features, which achieves acceleration goal by eliminating the bounding box proposal and the subsequent pixel or feature recombination stages (Liu et al. 2016). Additionally, SSD uses small convolution kernels to predict the category and offset of objects at the bounding box locations, and employs independent predictors to detect different aspect ratios. With the above-mentioned filters, it also performs multi-scale detection in multiple feature maps at the later network stages. On the VOC 2007 test set, the detection speed of SSD is 59 FPS, while that of Faster R-CNN is only 7 FPS, and of YOLOv1 is 45 FPS.

Despite faster recognition than the Faster R-CNN, YOLOv1 has weaker object localization ability than the Faster R-CNN due to the selection of only a few bounding boxes for prediction, as well as the low generalization rate regarding object aspect ratio. YOLOv2, as an improved method, uses a variety of strategies to enhance the localization accuracy (Redmon and Farhadi 2017), such as (1) BN: YOLOv2 adds BN layer after each convolution layer, and deletes the dropout, thereby enhancing its mAP by 2.4%. (2) Anchor Boxes: YOLOv2 draws on the anchor free strategy in Faster R-CNN to facilitate the network convergence. (3) YOLOv2 utilizes a feature network called Darknet-19, which contributes to lowering the computational burden by approximately 33%.

YOLOv3, by introducing multi-scale fusion of residual networks and features, further improves the network recognition rate (Redmon and Farhadi 2018). It deepens the network depth through introducing of residual network construction to the Darknet. Since the improved network has 53 convolution layers, it is also named Darknet-53. On ImageNet, Darknet-53 has similar recognition rate to that of ResNet-101 and ResNet-152 even though its recognition speed is much faster.

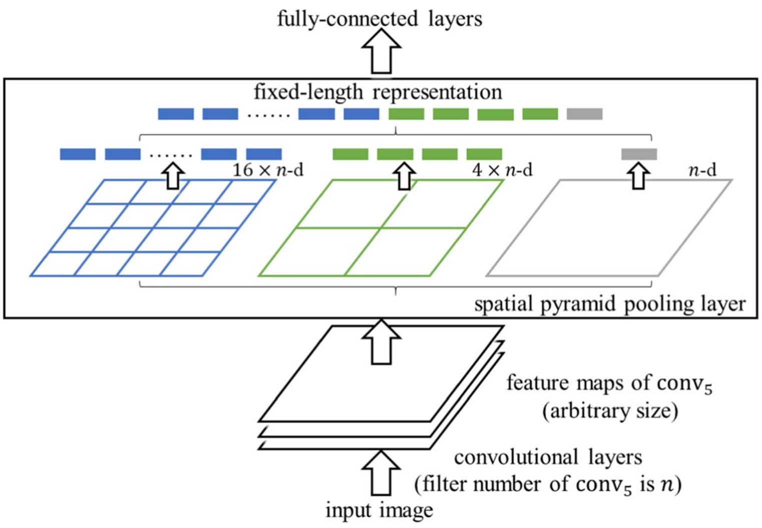


Fig. 15 Spatial pyramid pooling architecture

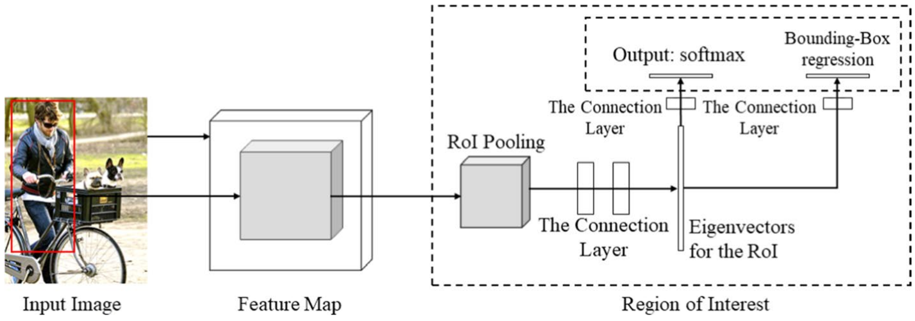
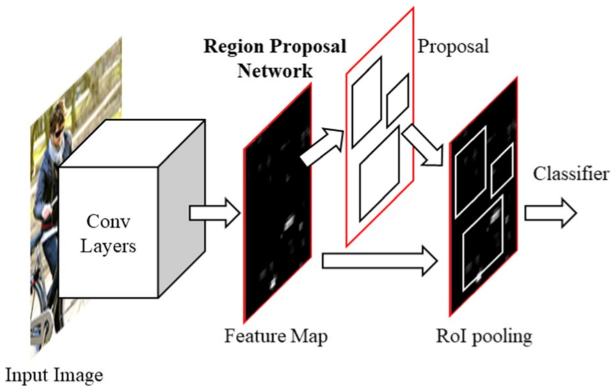


Fig. 16 Fast R-CNN-based object detection process

Proposed in 2020, YOLOv4 also introduces vrarious CNNs optimization strategies developed in recent years into the network architecture (Bochkovskiy et al. 2020). YOLOv4 applies the more capable CSPNet as a backbone. Through Mosaic data augmentation, YOLOv4 combines four training images into one for training, thereby increasing the model robustness. Besides, YOLOv4 introduces self-adversarial training to enhance the network resistance to adversarial attacks. On the MS COCO dataset, YOLOv4 utilizes Tesla V100 GPU to achieve a 43.5% AP network accuracy with an inference speed of nearly 65 FPS. In the same year, Glenn uploaded YOLOv5 to the Github (Jocher 2020). Compared with the previous series, YOLOv5 adds the function of adaptive anchor calculation, and the aspect ratio of the anchor is also employed as a parameter for backpropagation and iteration to adapt to groundtruth. Apart from that, YOLOv5 is used to modify the CSPnet structure and then applied to the neck of the detector in comparison with YOLOv4. Combined with these optimization strategies, YOLOv5 achieves more efficient training results.

In 2021, a network architecture for YOLOX was put forward by Ge et al., who considered that coupled detector heads may impair the network performance and took the decoupled head approach to improving the AP by 4.2% compared to the end-to-end approach (Ge et al. 2021). In addition, Ge et al. presented a sample matching scheme of SimOTA (Simplified Optimal Transport Assignment) to calculate the matching relationship between the anchor box and groundtruth based on the network's own prediction instead of using the anchor as the prior frame. YOLOX obtains faster and higher accuracy on the coco dataset

Fig. 17 Faster R-CNN-based object detection process



compared to existing models. Table 7 details the comparisons of algorithmic differences among R-CNN, SPPNet, Fast R-CNN, Faster R-CNN, YOLOv1–v5, YOLOX and SSD at each step.

In particular, many ingenious network structures have been generated to cope with the problem of multi-scale targets in target detection networks. An architecture known as the Feature Pyramid Network (FPN) that generates a series of bottom-up feature maps through the up-sampling layer and concatenates them with the previous feature maps of the same size was proposed by Lin et al. (Lin et al. 2017b). The PANet structure given by Wang et al. takes the feature map after FPN up-sampling and then passes it through a series of top-down down-sampling/citewang2019panet. In addition, the PANet also adds a shortcut across multiple layers to the top-down and bottom-up modules for concatenating the feature maps. Other than that, the ThunderNet designed by Qin et al. uses the Global Fusion Model (GFM) to fuse the features of each size and perform a secondary down-sampling process (Qin et al. 2019). Cross-scale connectivity was further investigated in works such as Kong et al. (2018), Kim et al. (2018), and Zhao et al. (2019). The simple and efficient weighted bidirectional feature pyramid network (BiFPN) further realizes the cross-scale connectivity optimization based on the PANet. In other words, an additional connection to the additional downsampling layer is provided in the original downsampling layer to fuse more features without significantly increasing computational cost. This design allows the ThunderNet to perform better in lightweight networks. Furthermore, the Exact Fusion Model (EFM) proposed by Wang et al. selects fusion features based on the size of the anchor (Wang et al. 2020a). Figure 19 visualizes the difference between FPN, PANet, BiFPN, GFM and EFM architectures.

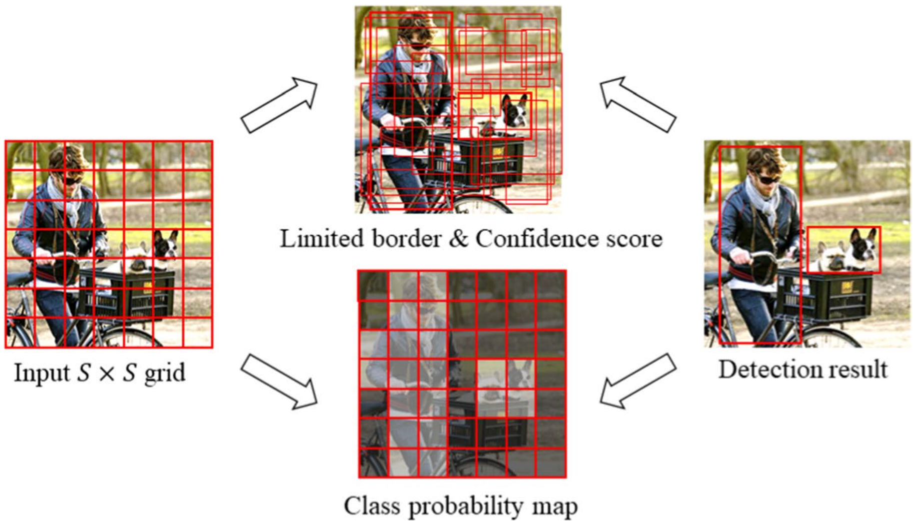


Fig. 18 YOLO-based object detection process

3.6 Transformer encoder

The attention mechanism raised by Vaswani et al. (2017) was originally commonly used as a backbone in the field of Natural Language Processing (NLP) (Radford et al. 2018, 2019; Devlin et al. 2018). In 2020, Dosovitskiy et al. applied the idea of the attention mechanism to the field of computer vision and also proposed the Vision Transformer (ViT) module (Dosovitskiy et al. 2020). With the support of large-scale datasets, the ViT model can achieve accuracy comparable to CNNs models. Figure 20 shows the architecture of the transformer encoder in the ViT module. The input to the transformer encoder is a 1D token embedding, and the 3D feature map should be flattened into a 2D block before the input. The Transformer encoder consists of multi-layer and multi-head self-attention (MHSA) and MLP blocks. Besides, the Layer Norm layer is applied before each block, and residual concatenation is adopted after each block. Beyond that, the MLP block contains GELU and two Layer Norm layers. Performance comparison and analysis of convolution with transformer are summarized in Table 8.

However, some studies have demonstrated that ViT model are lacking in optimizability compared with CNN, which is due to the lack of spatial inductive biases in ViT (Graham et al. 2021; Dai et al. 2021; Liu et al. 2021b; Wang et al. 2021; Yuan et al. 2021; Chen et al. 2021c). Therefore, using a convolution strategy in the ViT model to weaken such biases can improve its stability and performance. In 2021, Graham et al. combined Transformer and CNNs and proposed a network called LeViT (Graham et al. 2021). Graham et al. introduced the idea of attention bias for combining location information in ViT. The Pyramid Vision Transformer (PVT) proposed by Wang et al. incorporates the Transformer into a CNNs in order to overcome the shortcomings of the Transformer encoder for dense prediction tasks (Wang et al. 2021). The PVT can be trained on dense partitions of images to achieve high output resolution. Yuan et al. proposed a network architecture called Tokens-To-Token Vision Transformer (T2T-ViT) (Yuan et al. 2021). By recursively aggregating adjacent Tokens into one Token, the image is eventually structured into Tokens step by step. Besides, experiments show that the T2T-ViT architecture provides an efficient backbone with a deeper and narrower backbone under the CNNs architecture. Mehta et al. concatenated the Vit into MobileNet v2 and proposed MobileViT, a lightweight network for mobile devices (Mehta et al. 2021). The results revealed that MobileViT obviously outperformed other lightweight networks for different tasks and data sets.

Zhang et al. and Zhu et al. combined the ViT module and yolo for identifying images captured by Unmanned Aerial Vehicles (UAVs) (Zhang et al. 2021; Zhu et al. 2021a). It should be noticed that these datasets have many problems, such as extremely different target scales, complex backgrounds, high densities, and a large number of object occlusions. The experimental results show that the attention mechanism can make up for the deficiency of global information by convolution to some extent, and the results obtained by their models are excellent.

Instead of using the feature map directly as the input of the Transformer encoder in the ViT, the Visual Transformers (VTs) proposed by Wu et al. convert the feature map into a series of visual tokens by Tokenizer, and then the processed visual tokens are projected onto the original map and the original image by projector (Wu et al. 2020b). Then, experiments demonstrate that VTs improve the ResNet accuracy of ImageNet top-1 by 4.6 to 7 points while using fewer FLOPs and parameters.

A hybrid network structure called Conformer (Peng et al. 2021) was produced by Peng et al. It is the first network that combines the CNNs and Transformer modules in

Table 7 Comparisons among R-CNN, SPPNet, Fast R-CNN, Faster R-CNN, YOLOv1–v5, YOLOX and SSD

process	Region proposal extraction	Feature extraction	Feature classification
R-CNN	Selective search	–	SVM & Bounding Box Regression
SPPNet	Selective search	SPP pooling	SVM & Bounding Box Regression
Fast R-CNN	Selective search	RoI pooling	softmax & Bounding Box Regression
Faster R-CNN	RPN network	RoI pooling	softmax & Box Regression
YOLOv1	–	Goog-LeNet	softmax & Bounding Box Regression
SSD	–	VGG	Log-softmax & Bounding Box Regression
YOLOv2	–	Darknet	softmax & Bounding Box Regression
YOLOv3	–	Darknet & FPN	GIoU & Bounding Box Regression
YOLOv4	–	CSPDarknet & PANet	CIOU & Bounding Box Regression
YOLOv5	–	CSPDarknet & PANet	GIOU & Bounding Box Regression
YOLO X	–	Darknet	IOU & SimOTA

parallel to interact with both local and global features at each stage through the Feature Coupling Unit (FCU), thus possessing the advantages of both the CNNs and the Transformer. In addition, on the ImageNet, Conformer outperforms the VT (DeiT-B) by 2.3%, while on MSCOCO, it performs better than ResNet-101 by 3.7% and 3.6% in mAPs for object detection and instance segmentation, respectively. Srinivas et al. proposed an architecture named BoTNet, which significantly improved the baselines by replacing spatial convolution with global self-attention in the last three bottleneck blocks of ResNet (Srinivas et al. 2021). Dai et al. proposed a network named CoAtNets network structure (Dai et al. 2021), and suggested that the deep structure of the CNNs and the attention mechanism can be associated by simple relative attention. Besides, they found that stacking convolutional layers and the Transformer encoder in principle can

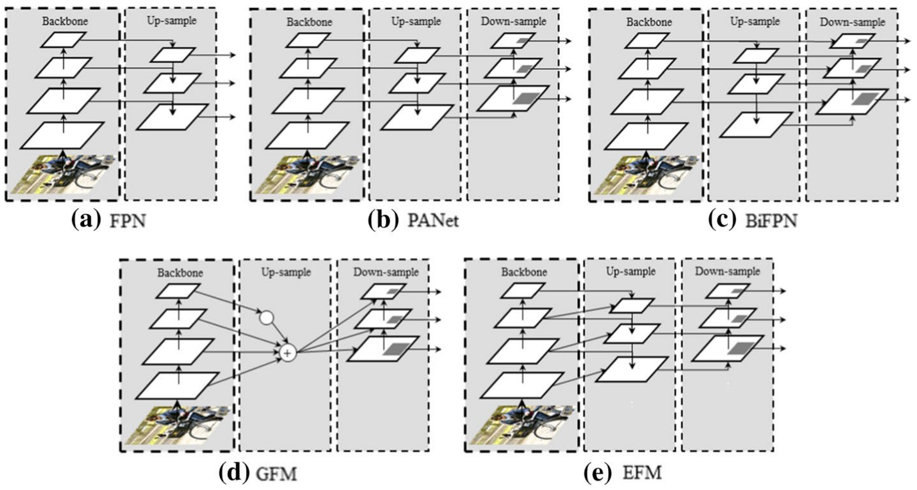


Fig. 19 PANet architecture

yield good results. CoAtNets achieved 88.56% accuracy on ImageNet-21K and 90.88% with the extension of JFT-3B. Liu et al. introduced a hierarchical Transformer called Swin Transformer (Liu et al. 2021b) that achieved 87.3% accuracy on Imagenet-1K by restricting the self-attention computation to non-overlapping local windows through shifted windows, while allowing cross-window connectivity.

4 Network architecture compression

Since the 1990s, the innovation of CNNs architectures has been continuously explored, and the complexity of CNNs and the resources required for their training have also been increasing. Meanwhile, on the premise of ensuring the network accuracy, compression and acceleration of model training and recognition become necessary in the field of CNNs architecture optimization. CNNs have a large number of redundant parameters from the convolution layers to the FC layers (Hu et al. 2016). The output values of most activated neurons are close to 0. The model features can be expressed even if these neurons are eliminated. This phenomenon is called over-parameterization. By decreasing redundancy in the network architectures, the training and recognition can be accelerated, and the memory can be reduced.

Figure 21 illustrates different types of classification methods for CNNs architecture optimization. Among them, the network pruning techniques include weight connection pruning and neuron pruning. The tensor decomposition techniques contain singular value decomposition, Tucker decomposition, canonical polyadic (CP) decomposition, block term decomposition and tensor-train decomposition. The network quantization techniques consist of binary quantization, ternary quantization and Hash quantization. We elaborate on the optimization ideas of CNNs architectures from the following four aspects: network pruning, tensor decomposition, network quantization and knowledge transfer.

4.1 Network pruning

Previously, network pruning has been demonstrated as an effective method for reducing network complexity and overfitting (Han et al. 2015a). Currently, pruning is the most common method of compressing network models. It is of note that the concept of pruning is not limited to CNNs. Dropout and DropConnect represent two quite typical techniques of model pruning. The idea of Dropout is to randomly set the output of several neurons to 0, while the idea of DropConnect is to randomly set the connections between some neurons to 0. Consistent with these two ideas, there are two methods called neuron pruning and weight connection pruning. Figure 22 presented below depicts the pruning of synapses and neurons. Generally, with the weight connection pruning, the weight matrices are stored using sparse matrices, and the unimportant weight connections are set to 0. As for the neuron pruning, it directly removes neurons with low importance.

Actually, the method for pruning CNNs models is not limited to the above. The currently common pruning practice can generally be described as training the network, pruning, retraining the weights, pruning again, and retraining until the steps satisfying the set conditions are formed. Figure 23 displays the network pruning methods classified according to pruning granularities. From fine to coarse, they are Fine-grained pruning, Vector-level pruning, Kernel-level pruning, Group-level pruning and Filter-level pruning. They

Table 8 Performance comparison and analysis of convolution with transformer(all inference are performed on ImageNet.1K of input size 224×224)

Name	Year	Accuracy (% *Top*1)	Params	FLOPs	Contribution
LeViT-128	2021	9.2	84.7	406M	Attention bias for combining location information in ViT
LeViT-192		10.9	85.7	658 M	
LeViT-256		18.9	86.8	1120 M	
LeViT-384		39.1	87.6	2353 M	
PVT-Tiny	2021	75.1	13.2M	1900M	Overcomed the shortcomings of the transformer for dense prediction tasks
PVT-Small		79.8	24.5 M	3800 M	
PVT-Medium		81.2	44.2 M	6700 M	
PVT-Large		81.7	61.4 M	9800 M	
T2T-ViTt-14	2021	80.7	21.5M	5200M	Structuring images as tokens
T2T-ViTt-19		81.4	39.0 M	8400 M	
T2T-ViTt-24		82.2	64.1 M	13,200 M	
MobileViT-XS	2021	74.8	2.3 M	–	Combined inverse residual and ViT
MobileViT-S		78.4	5.6 M		
VT-R34	2021	79.9	19.2 M	3236 M	Input the image into transformer by mapping
VT-R50		80.6	21.4 M	3412 M	
VT-R101		82.4	41.5 M	7129 M	
Conformer-Ti	2021	81.3	23.5 M	–	Combined CNNs and Transformer modules in parallel
Conformer-S		83.4	37.7 M		
Conformer-B		84.1	83.3 M		
BoT-S1-50	2021	20.8 M	79.1	4270 M	Replacing spatial convolution with global self-attention
CoAtNet-0	2021	81.6	25M	4.2B	Stacking convolutional layers and transformer encoder
CoAtNet-1		83.3	42 M	8.4B	
CoAtNet-2		84.1	75 M	15.7B	
CoAtNet-3		84.5	168 M	34.7B	
Swin-T	2021	81.3	29 M	755.2 M	Restricted self-attention to non-overlapping local windows and connect them

Table 8 (continued)

Name	Year	Accuracy (% *Top*1)	Params	FLOPs	Contribution
Swin-B		83.5	88 M	278.1 M	

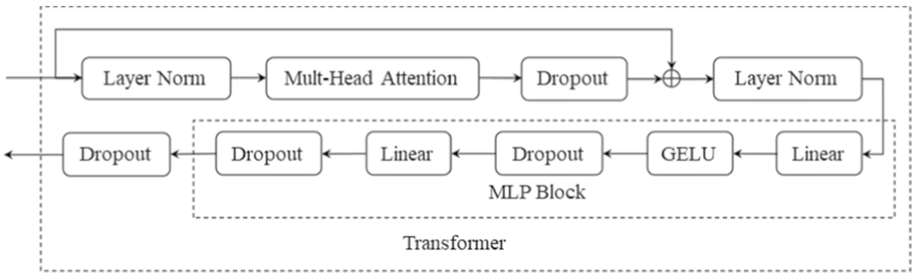


Fig. 20 Transformer encoder architecture

alter the intrinsic network topology, and require design of a specific algorithm to support the pruning operation, which belong to unstructured pruning. In Sect. 4.1, various pruning methods are described in detail, and performance comparison and analysis of pruning methods are summarized in Table 9.

4.1.1 Fine-grained pruning

Fine-grained pruning refers to pruning the connections between neurons or the neurons themselves in a network. Any unimportant parameters in the convolution kernels can be pruned. This concept has been applied prior to the proposal of deep neural networks (DNNs). LeCun et al. put forward the optimal brain damage (OBD) in 1989, reducing the parameters in actual NNs by four times (LeCun et al. 1990). With OBD, the networks automatically delete parameters, and thus the network training speed is significantly improved and the recognition accuracy is slightly enhanced. Hanson et al. introduced a weight decay term in the error function to make the networks sparse (Hanson and Pratt 1988). In other words, they reduced the network complexity by lowering the number of hidden nodes.

As early as in 1993, Hassibi et al. proposed the optimal brain surgeon (OBS), and then compared its performance with that of OBD (Hassibi et al. 1994). Their results revealed that (1) The pruning of networks based on the Hessian matrix of loss function was more accurate than the weight decay-based pruning. (2) In OBS, the constraints on the Hessian matrix were discarded, and thus its generalization ability was more powerful than OBD. Nevertheless, the second derivatives of the loss functions for OBS and OBD require considerable computational burdens in the case of large network architectures. In 2015, Srinivas et al. performed pruning operations on the neurons in FC layers instead of network connections (Srinivas and Babu 2015). Their pruning technique for the FC layers highly resembled that of OBS. Through substantial derivative approximations, their method removed the dependence on training data. In the same year, Han et al. proposed a deep compression framework, with which they compressed DNNs through three steps, respectively, pruning, quantization and Huffman coding (Han et al. 2015b). The deep compression proposed by Han et al. in the same year further optimized the network architectures, which comprehensively applied techniques including pruning, quantization and coding (Han et al. 2015a).

Guo et al. pointed out in 2016 that the importance of parameters would continuously change with the initiation of network training (Guo et al. 2016), so restoring important pruned connections would play a crucial role in improving the network performance. The dynamic network surgery proposed by them adds a restore operation in the pruning

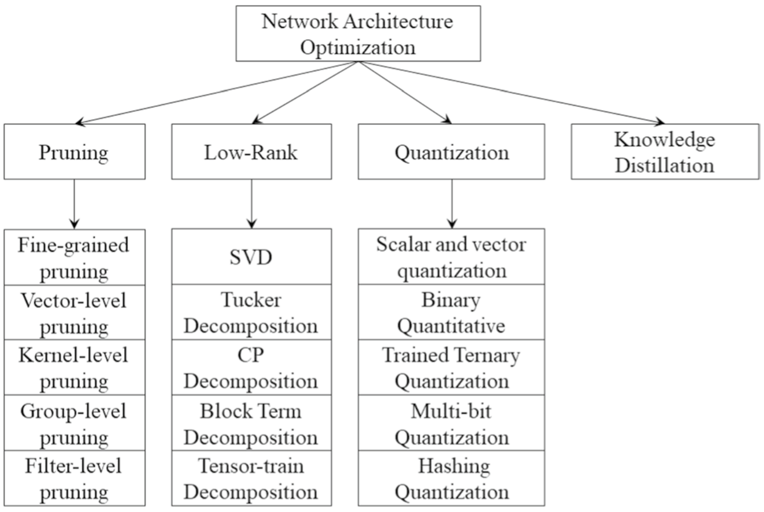


Fig. 21 Four types of classification methods for CNNs architecture optimization

process. Therefore, the pruned network connections can be reactivated when they become important. These two operations are performed alternately following each training, thus tremendously improving the online learning efficiency. In the same year, Zhou et al. (2016) processed loss functions with sparse constraints by adopting forward-backward splitting method, which significantly reduced the number of neurons in networks. To address the failure to directly combine and apply the Winograd's minimal filtering algorithm and the network pruning technique, in 2017, Liu et al. (2018b) proposed to shift the ReLU activation function to the Winograd domain before pruning the Winograd transformed weights. This approach reduces the number of multiplication operands by 10.4, 6.8 and 10.8 times, respectively, on the CIFAR-10, CIFAR-100 and ImageNet datasets. In 2019, Chen et al. proposed the fine-grained dynamic method called channel gating to prune CNNs inference (Hua et al. 2019). Channel gateway identifies areas of the feature map at every CNNs layer which make fewer contributions to classification outcomes, and also closes a channel subset to calculate the activation in such unimportant areas. In 2021, Zhang et al. proposed ClickTrain to provide higher model accuracy and compression through fine-grained architecture-preserving pruning (Zhang et al. 2021a). In the same year, Sun et al. proposed a fine-grained pruning method called DominoSearch (Sun et al. 2021). They argued that existing $N : M$ fine-grained sparse neural network algorithms were only applicable to the situation that each layer features the same $N : M$ sparsity and the accuracy drops

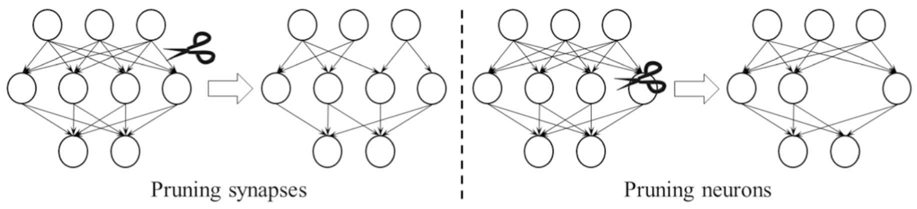


Fig. 22 Pruning of synapses and neurons

dramatically at sparsity >80%. DominoSearch can find mixed $N : M$ sparsity schemes from pre-trained networks to achieve higher accuracy than traditional schemes under the same complexity constraint.

4.1.2 Vector-level and kernel-level pruning

Compared with fine-grained pruning, the granularities processed by vector- and kernel-level pruning are coarser. The vector-level pruning trims the vectors inside the convolution kernels, while the kernel-level pruning removes the 2D convolution kernels in the filters. In 2017, Mao et al. explored different levels of pruning granularities and found that the vector-level pruning occupies less memory space than the fine-grained pruning, since it requires fewer indices to indicate the pruning parameters (Mao et al. 2017). Generally, structured pruning is more friendly to memory access than the unstructured pruning. Thus, practically, the vector, convolution kernel and filter pruning techniques are more effective for hardware implementation.

Anwar et al. were the first to propose the idea of kernel-level pruning in 2017, where the concept of intra kernel strided sparsity was introduced (Anwar et al. 2017). This method performs pruning operations using N particle filters corresponding to N convolution layers. By transforming the fine-grained pruning into structured pruning, the sub-vectors are pruned with a fixed step size. In 2020, Ma et al. and Niu et al. introduced fine-grained pruning patterns into the coarse-grained structures for kernel-level pruning (Ma et al. 2020; Niu et al. 2020). PCONV contains two kinds of sparsity. One is the sparse convolution pattern (SCP) generated by the convolution kernel pruning, which improves accuracy because of its distinctive vision characteristics. The other refers to the connectivity sparsity generated by the convolution kernel pruning, which maintains balanced workload on filter computation. In 2021, inspired by neurobiology, a kernel-level lossless pruning method called ResRep (Ding et al. 2021) was presented by Ding et al. Unlike traditional pruning methods, it divides the CNNs into a memory part for maintaining performance and an oblivious part

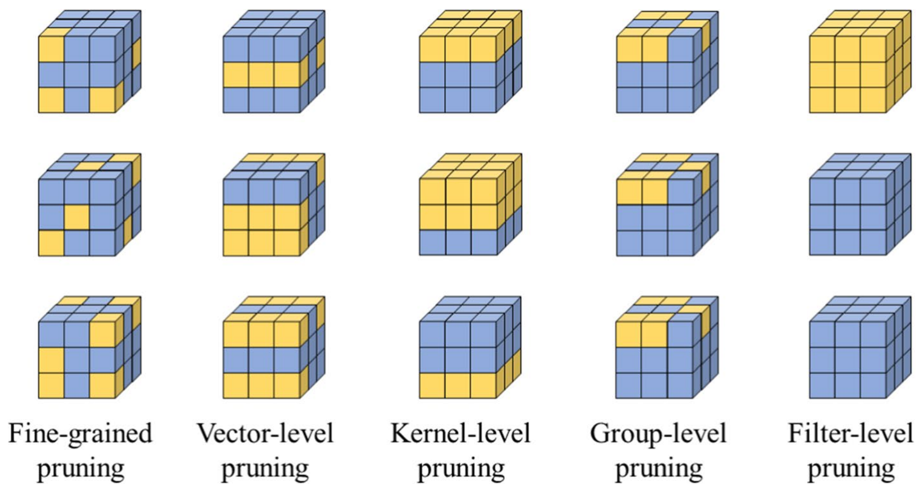


Fig. 23 Pruning methods classified according to pruning granularities

Table 9 Comparison and analysis of pruning methods

Pruning	Methods	Target	Performance
Fine-grained pruning	Srinivas and Babu (2015)	FC layers	Provides 87.45% compression with <1 % accuracy loss on LeNet
	Han et al. (2015a)	Conv & FC layers	Provides 88.89% compression on AlexNet and 89.89% compression on VGGNet
	Han et al. (2015b)	Conv & FC layers	Provides 98.97% CNNs layer compression and 89.80% FN layer compression on VGGNet
	Guo et al. (2016)	Conv & FC layers	Compress parameters in LeNet-5 and AlexNet by a factor of 108× and 17.7×
	Zhou et al. (2016)	fully connected layers	Provides 70% compression with <1 % accuracy loss on first FC layers
	Liu et al. (2018b)	Conv layers	Compress the CNNs models by a factor of 10.4×, 6.8× and 10.8×
	Hua et al. (2019)	Conv layers	Provides 2.3× acceleration for ImageNet when the theoretical FLOP reduction is 2.8×
	Zhang et al. (2021a)	Conv layers	Reduces the end-to-end time cost of the state-of-the-art method by up to 2.3×
	Sun et al. (2021)	Conv layers	For sparsity of 87.5% of model, improves top-1 accuracy by 2.1%. For 227M complexity, improves top-1 accuracy by 1.3%
Vector-level and kernel-level pruning	Mao et al. (2017)	Conv & FC layers	Saves about 2× the memory references compared to fine-grained sparsity
	Anwar et al. (2017)	Conv & FC layers	Provides 70% compression with <1 % accuracy loss on the CIFAR-10 network
	Ma et al. (2020)	Conv layers	Compared to TensorFlow-Lite, TVM and Alibaba mobile neural network speedups of 39.2×, 11.4× and 6.3× respectively
	Niu et al. (2020)	Conv layers	Compared to TensorFlow-Lite, TVM and Alibaba mobile neural network speedups of 44.5×, 11.4× and 7.1× respectively
	Ding et al. (2021)	Conv layers	Reduced FLOPs by 45% on ResNet-50 with no decrease in accuracy
Group-level pruning	Lebedev and Lempitsky (2016)	Conv layers	Provides a 3.5× increase in speed on AlexNet with an accuracy loss of less than 1%
	Wen et al. (2016)	Conv layers	Reduces the weight matrix sizes by 50%, 70.7%, and 36.1% on the CIFAR-10 dataset; Enables 5.1× and 3.1× acceleration for AlexNet on the CPU and GPU, respectively
	Lee et al. (2020)	Conv layers	the error rate of 2D unaligned group-level pruning is 3.12% lower at ResNet-20 network on CIFAR-10 than former 2D aligned group-level pruning in condition of 95% sparsity
	Jo et al. (2020)	Conv layers	Reduces the accuracy loss by 3.77% compared to the globally pruned threshold pruned model at 95% sparsity
	Liu et al. (2021)	Conv layers	Provides a 3× increase in speed on Faster R-CNN with an mAP drop 0.8%

Table 9 (continued)

Pruning	Methods	Target	Performance
Filter-level pruning	Li et al. (2016)	Conv & FC layers	Provides 66% compression on VGG-16 and 62% compression on ResNet-110
	Luo et al. (2017)	Conv & FC layers	Reduces 3.31× FLOPs and 16.63× compressions on VGG-16 at the cost of lowering the top-5 accuracy by a mere 0.52%; Reduces of FLOPs by 2.26× and compression by 2.06× while lowering the top-5 accuracy only by 1%
	He et al. (2017)	Conv layers	Provides a 2× increase in speed on ResNet, Xception with an accuracy loss of 1.4% and 1.0%
	Liu et al. (2017)	Conv layers	Provides 20× compression in model size and 5× reduction in computing operations for VGG-Net
	Wang et al. (2019b)	Conv & FC layers	Provides 92.8% parametric-reduction ratio and 73.5% FLOPs-reduction ratio with less than 0.3% accuracy loss on VGG-16 for CIFAR-10
	Xu (2020)	Conv & FC layers	Provides 3.21× parametric-reduction ratio and 16.92× FLOPs-reduction ratio with 0.55% top-5 accuracy drop
	Tang et al. (2021b)	Conv layers	Reduce 55.3% FLOPs with 0.57% top-1 accuracy drop on ResNet-34

for learning pruning. The memory part is trained using SGD, and the latter is pruned with the rule of penalized gradient.

4.1.3 Group-level pruning

Traditional group-level pruning refers to trimming parameters according to the same sparsity patterns on filters, requiring the filters to have identical sparsity patterns. As shown in Fig. 24, when an identical sparsity pattern exists in every convolution kernel, the convolution filters can be expressed as a thinner dense matrix.

In 2016, Lebedev and Lempitsky (2016) adopted group-wise brain damage for pruning convolution kernel input to obtain the optimal receptive field in a data-driven manner. In the same year, Wen et al. (2016) proposed a structured sparsity learning (SSL) method for standardizing the DNN architectures, which enabled effective acceleration on DNNs with higher unstructured sparsities (at low sparsities, negative acceleration may occur).

Conventional methods prune weight groups at group-size-aligned locations. In 2020, Lee et al. proposed unaligned group-level pruning, which could enhance compressed model accuracy. Their method is able to seek the optimal solution for unaligned group selection issue through dynamic programming (Lee et al. 2020). Jo et al. put forward Grouped Dynamic Sparse Training (DST) (Jo et al. 2020). DST can promote model sparsity degree through repeating pruning and recovering at each step during training (Liu et al. 2020b). The method applies DST in group-level pruning, thus learning the criteria of every filter and judging the importance of weight groups. In 2021, Liu et al. proposed a layer grouping algorithm to prune complex network architecture such as residual connections and group/depth-wise convolution (Liu et al. 2021c). Apart from that, they adopted Fisher information-based unified metric to evaluate the importance of single channels and coupled channels, and coupled channels for complex architecture were automatically found. Experiments prove that this pruning algorithm can be effectively deployed in a variety of complex structures, such as ResNeXt, MobileNetV2, and the RegNet.

4.1.4 Filter-level pruning

Filter-level pruning is used for convolution filters or channels, which is more effective than other methods since after trimming one layer, the number of channels in the next layer decreases correspondingly. In 2016, Li et al. put forward holistic global pruning (Li et al. 2016). By filter pruning, this idea compensates for the inadequacy of magnitude-based methods in pruning the FC layers. Besides, they directly removed the convolution kernels exerting less impact on output accuracy, as well as the corresponding feature maps through non-sparse connections. On CIFAR10, the inference cost can be reduced by up to 34% for VGG-16 and by up to 38% for ResNet-110 while regaining close to the original accuracy by network retraining.

In 2017, Luo et al. proposed a filter-level pruning technique called ThiNet (Luo et al. 2017), where the probability information of the next convolution layer were used instead of the current one for guiding the filter-level pruning in the current layer. In the same year, He et al. (2017) removed redundant convolution kernels and their corresponding feature maps by channel pruning that was based on LASSO regularization and linear least squares, followed by the reconstruction of the remaining network. According to the test on VGG-16, their method exhibited lower accuracy loss than most existing optimization algorithms under the same acceleration. Liu et al. recommended applying the scaling factor of BN layers to evaluate the filter importance (Liu et al. 2017). By trimming channels with close to zero scaling factors, the filters can be trimmed without introducing overhead into the networks.

In 2019, Wang et al. argued that network importance should be evaluated independently. Therefore, redundancy between filters is not considered, and the reduction of parameters does not necessarily indicate the reduction of calculation costs (Wang et al. 2019b). To solve the existing problem, they further proposed a cop-based pruning algorithm. The algorithm is able to detect redundant filters according to the users' own preferences by adding the parameter number and computational cost to the importance. Li et al. divided an undirected FC graph representing a pre-trained CNNs model into several subgraphs using spectral clustering algorithm, who finally kept one filter for one group and retrained the pruned model (Li et al. 2019). In 2020, Xu et al. developed the filter level pruning method, namely, PNFM, which could determine the importance of pruned filter according to the change rate in feature image output at the latter layer (Xu 2020). Besides, a cluster algorithm-based method capable of making tiny datasets for pruning was used to improve pruning efficacy.

In 2021, Tang et al. dynamically removed redundant filters by embedding the manifold information of all instances into the pruned network space (Tang et al. 2021b). The manifold information between instances and pruned sub-networks is aligned by the recognition complexity and feature similarity of the images in the training set, thus maximizing the exploitation of the redundancy in a given network structure.

4.2 Tensor decomposition

Obviously, is that the pruning process generally requires substantial pretraining operations. Despite the tremendously improved operating performance of the constructed networks, they often consume more time during the training process. Hence, the idea of tensor decomposition is applied to compress the network size and improve the network speed, which is usually accomplished by singular value decomposition, Tucker decomposition, CP decomposition, and block term decomposition.

Tensor decomposition can be described as a process of decomposing the original high-rank tensor into several low-rank tensors. For computers, this means reducing the computational burdens of convolution, which can effectively remove the redundant information in the networks.

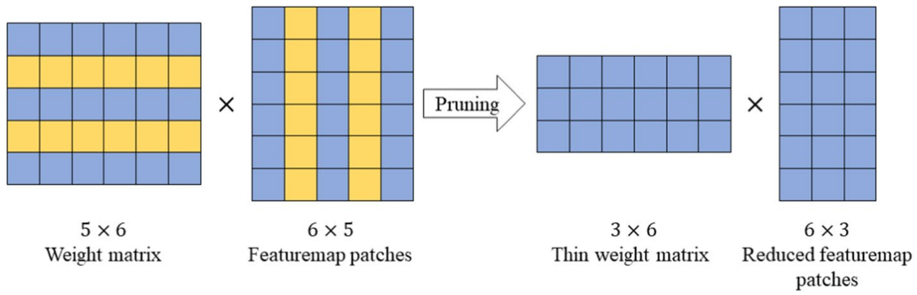


Fig. 24 Group-level pruning

4.2.1 Singular value decomposition

As a classical algorithm in the ML field, singular value decomposition (SVD) is often adopted for the eigendecomposition of low-rank matrices. Through SVD, the weight tensor (as the convolution kernel) is divided into two parts. That is, the original convolution layer is replaced with two consecutive layers. The mathematical expression of SVD is:

$$W \approx USV^T \tag{10}$$

where $W \in \mathbb{R}^{m \times n}$ denotes the decomposed tensor, $U \in \mathbb{R}^{m \times m}$ and $V \in \mathbb{R}^{n \times n}$ represent the orthogonal matrices, and $S \in \mathbb{R}^{m \times n}$ is a diagonal matrix.

Figure 25 illustrates the full convolution and SVD convolution processes, where the convolution kernel K in CNNs is a four-dimensional (4D) tensor, $K \in \mathbb{R}^{d \times d \times S \times T}$ is a 4D tensor with S, d and T respectively representing the input channel, convolution kernel size and output channel. Tensors $P \in \mathbb{R}^{T \times K}$ and $W' \in \mathbb{R}^{K \times d \times d \times S}$ are decomposed from the tensor K, that is:

$$K \approx PW' \tag{11}$$

When the complexity of original tensor is $O\left(d^2 ST\right)$, the complexity of decomposed tensors P and W' is $O(TK) + O\left(d^2 KS\right)$. Moreover, smaller K value indicates stronger compression effect.

In 2013, Denil et al. (2013) exploited the idea of tensor decomposition for improving the CNNs performance. Denil stated that compared to OBD, their work enables predictions before rather than after network training, limiting the number of parameters. On the MNIST dataset, this method can predict over 95% of network weights without compromising the network accuracy. By borrowing the tensor decomposition idea used by Jaderberg et al. (2014) and Denton et al. (2014) accelerated CNNs by the tensor low-rank expansion technique. Jaderberg et al. proposed a 4-layer CNNs trained on the scene character classification dataset, which decomposes $k \times k$ filters into asymmetric $k \times 1$ and $1 \times k$ filters. It achieves 2.5-times acceleration without accuracy loss, and 4.5-times acceleration with less than 1% accuracy drop. Denton et al. (2014) discovered an appropriate low-rank parameter approximation by exploiting the linear architecture of NNs. It achieves 2× acceleration of CPU and GPU while ensuring that the accuracies are within 1% from the original accuracy. In 2015, Zhang et al. borrowed the asymmetric tensor decomposition used by Jaderberg to accelerate the overall network operation, which achieves a 3.8× speedup on the VGG-16 model (Zhang et al. 2015). On the basis of SVD, Liu et al. added sparsity (representing the filter weights) to reduce the redundancy of network parameters through sparse decomposition of convolution kernels (Liu et al. 2015). This approach achieves sparsity of over 90% convolution layers in AlexNet, and less than 1% accuracy loss on the ILSVRC2012 dataset.

In 2020, Li et al. addressed the problem that pruning operations could not be completed in network structures with shortcut connections (e.g., ResNet) by complementing filter pruning with SVD (Li et al. 2020). They suggested combining pruning and decomposition using sparsity inducing matrices and proposed a unified representation. Additionally, they introduce various algorithms such as binary search, gradient based learning rate adjustment layer balancing and annealing methods.

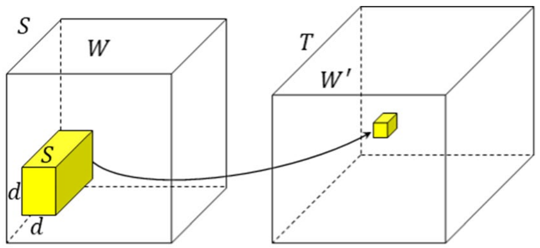
4.2.2 Tucker decomposition

Figure 26 depicts the Tucker decomposition process, which aims to decompose the convolution kernel $K \in \mathbb{R}^{d\times d\times S\times T}$ into a kernel tensor and several factor matrices. Its mathematical expression is:

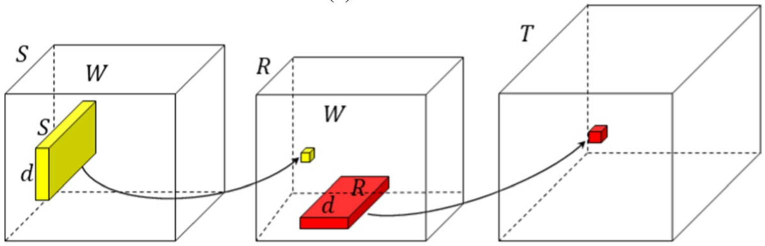
$$K = \sum_{r_3=1}^{R_3} \sum_{r_4=1}^{R_4} C_{r_3,r_4} U^{(3)}_{s,r_3} U^{(4)}_{t,r_4} \tag{12}$$

where $C \in \mathbb{R}^{d\times d\times R_3\times R_4}$, $U^{(3)} \in \mathbb{R}^{S\times R_4}$, and $U^{(4)} \in \mathbb{R}^{T\times R_4}$.

In 2015, Kim et al. (2015) proposed a holistic compression method based on the variational-Bayesian low-rank selection and Tucker tensor decomposition. Owing to the substantially reduced model size, runtime and energy consumption, the networks compressed using this method can be transplanted to run on mobile devices. In 2020, Lin et al. presented a generalized Higher Order Tucker Articulated Kernels (HOT-CAKE) method (Lin et al. 2020). This method performs input channel decomposition which guides Tucker rank selection, high-order Tucker decomposition as well as fine tuning at every CONV layer. Experiments show that HOTCAKE compresses pre-compressed models and produces latest lightweight networks. Kozyrskiy et al. proposed a new method combining tucker decomposition and quantization of weights and activations (Kozyrskiy and Phan 2020). This method includes the greedy one-step and multi-step algorithms for the task of multilinear rank selection, and quality recovery with the application of Tucker decomposition and quantization.



(a)



(b)

Fig. 25 Full and SVD convolution processes: **a** full convolution, **b** SVD

4.2.3 CP decomposition

Figure 27 displays the process of CP decomposition, which can be regarded as a special case of Tucker decomposition. Obviously, in Tucker decomposition, the algorithm is decomposed according to the rank of tensor itself, while in CP decomposition, presetting the rank value is necessary for iteration, and the rank selection of decomposition matrix is an NP hard problem. After selecting the appropriate rank, its algorithm can be mathematically expressed as:

$$K = \sum K^x \times K^y \times K^s \times K^t \tag{13}$$

where $K^x \in \mathbb{R}^{d\times R}$, $K^y \in \mathbb{R}^{d\times R}$, $K^s \in \mathbb{R}^{S\times R}$, and $K^t \in \mathbb{R}^{S\times R}$.

Lebedev et al. put forward a kernel tensor decomposition method based on CP decomposition, which decomposes the convolution kernel into four first-order kernel tensors by nonlinear least squares technique (Lebedev et al. 2014). For the 36-class ILSVRC classification experiment, this method can attain an 8.5-times speedup on the CPU. The experimental results also suggest that the tensor decomposition exerts a regularization effect. Additionally, Lebedev et al. also pin-pointed in their study that a good SGD learning rate is hardly attainable, implying implies that optimizing the single layer decomposition in the ImageNet model is not an easy task. In 2018, Astrid and Lee (2017) proposed a network compression method based on optimizing CP decomposition of all convolution layers. After each single-layer network is decomposed, the entire network is fine-tuned to overcome the decrease in network accuracy resulting from the unstable CP decomposition. In 2019, Zhou et al. proposed two methods for estimating the rank of the CP tensor from noise measurements (Zhou et al. 2019). One directly employs CNNs in CP rank estimation. The other introduces pre-decomposition into feature acquisition, which can input rank-one components to CNN.

4.2.4 Block term decomposition

Wang and Cheng (2016) proposed a CNNs acceleration technique called block term decomposition (BTD) in 2016. Figure 28 illustrates the tensor BTD process. The convolution kernel is assumed to be a 3D tensor $\mathcal{T} \in \mathbb{R}^{S\times T\times P}$, where P represents the spatial dimension. The mathematical expression of the algorithm is:

$$\mathcal{T} = \sum_{r=1}^{R} \mathcal{G}_r \times A_r \times B_r \times C_r \tag{14}$$

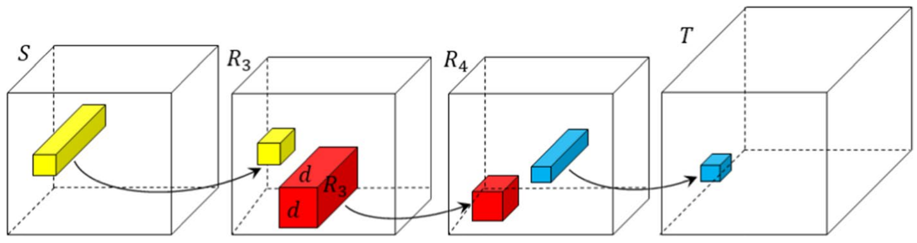


Fig. 26 Tucker decomposition process

where $\mathcal{G}_r \in \mathbb{R}^{s \times t \times p}$ denotes the core tensor of the rth decomposition with $A_r \in \mathbb{R}^{S \times s}$, $B_r \in \mathbb{R}^{T \times t}$, $C_r \in \mathbb{R}^{P \times p}$ representing the factor matrices along each dimension. Lowercase letters s, t and p denote the ranks of each dimension.

According to comprehensive experiments on ILSVRC-12, the algorithm significantly reduces computational complexity, but with negligible loss in accuracy. For the widely applied VGG-16 model, this method attains a 6.6× acceleration on the entire PC network, and a 5.91× acceleration on the mobile device maintaining the increase in top-5 error less than 1%.

In 2020, Ye et al. explored the relevance of the weight matrix and employed low-rank block-term tensors to approximate the weight matrix (Ye et al. 2020). The layer structure using this low-rank weight matrix is named block-term tensor layers (BT-layers), which are well adapted to CNNs. From experimental results, it can be seen that BT-layers bring great compression to the network, and the inputs and outputs in BT-layers are reshaped into low-dimensional higher-order tensors. In this case, the CNNs obtains better representational capabilities.

4.2.5 Tensor-train decomposition

In 2015, Novikov et al. used tensor-train decomposition (TTD) (Oseledets 2011) to transform the dense weight matrix of the FC layer into tensor-train (TT) format (Novikov et al. 2015). This method compresses the dense weight matrix of the FC layer of VGG by a factor of up to 200000, resulting in a compression factor of up to 7 times for the entire network. Figure 29 illustrates the processing of TTD. The filter is assumed to be a 4D tensor $\mathcal{K} = l_1 \times 1_2 \times C \times S$, where $l_1 \& 1_2$ represents the size of kernel, $C = c_1 ... c_d$ and $S = s_1 ... s_d$ represent the number of input and output channels respectively. The mathematical expression is:

$$\mathcal{K} = \mathcal{G}_0 \circ \mathcal{G}_1 \circ \mathcal{G}_2 \circ ... \circ \mathcal{G}_{d-1} \circ \mathcal{G}_d \tag{15}$$

where $\mathcal{G}_d \in \mathbb{R}^{r_i \times c_i s_i \times r_{i+1}}$ denotes the tensor-train format of the ith decomposition for $0 < i < d$.

In 2021, Sharma et al. applied TTD to Convolutional auto-encoders (CAEs) (Sharma et al. 2021). By adjusting the tensor levels, they managed to use the model to adjust the number of learning parameters without changing the network structure. In the same year, Lee et al. combined TTD and data quantization approaches to build a model called QTTNet (Lee et al. 2021). Their method is able to simultaneously lower the bit-width of the data with fewer trainable parameters. Wang et al. considered that tensor decomposition introduces a significant loss of accuracy, and they proposed a new novel nonlinear tensor

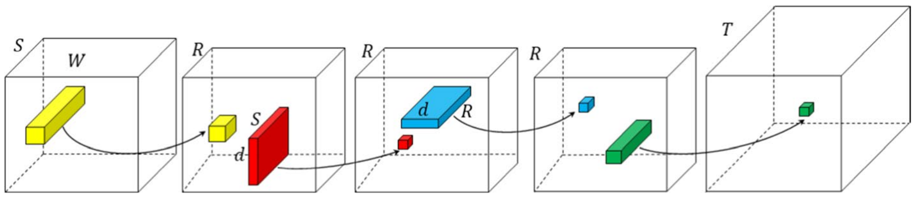


Fig. 27 CP decomposition process

train (NTT) format (Wang et al. 2021c). NTT contains the additional nonlinear activation function to compensate for the accuracy loss that the normal TT fails to provide.

4.3 Network quantitative

Compared to pruning, which reduces the model size by removing weights and tensor decomposition, and the complexity by decomposing the convolution kernels, the quantization aims to lower the number of bits required for storing weights. Memory can be significantly reduced through quantization, i.e. by compressing the original network by decreasing the number of bits required for each parameter. Depending on the quantified results, network quantization methods can be divided into scalar and vector quantization, binary quantization, ternary quantization, multi-bit quantization and hashing quantization. Where binary quantization, ternary quantization, multi-bit quantization and hashing quantization belong to fixed-point quantization.

4.3.1 Scalar and vector quantization

The idea of quantizing CNNs with scalars or vectors is borrowed from the early lossy compression technology, which adopted codebook and quantization code for restoring the original data. Vector quantization defines a quantizer, i.e. a mapping function that converts a dimensional vector into a vector in the codebook.

To optimize the quantizer, the Lloyd optimality conditions must be satisfied. Hence, K-means clustering algorithm can be applied in solving the optimal quantizer. In 2011, Jegou et al. developed a product quantization (PQ) algorithm for FC layers (Jegou et al. 2010). The PQ quantizer decomposes the original vector space into Cartesian products of multiple low-dimensional vector spaces, and performs K-means on the decomposed low-dimensional vector spaces to calculate the codebook. The quantizer conversion process can be mathematically expressed as:

$$\begin{gathered} x \xrightarrow{q} q(x) \\ x \in R^D,\ q(x) \in C = c_i \\ i \in I, I = 0, \ldots, k-1 \end{gathered} \tag{16}$$

where $q(.)$ stands for mapping function, c_i denotes the centrioles, I represents the predefined index set, and k refers to the size of codebook.

After constructing the PQ quantizer, the authors put forward two methods, namely, the symmetric distance computation (SDC) and the asymmetric distance computation

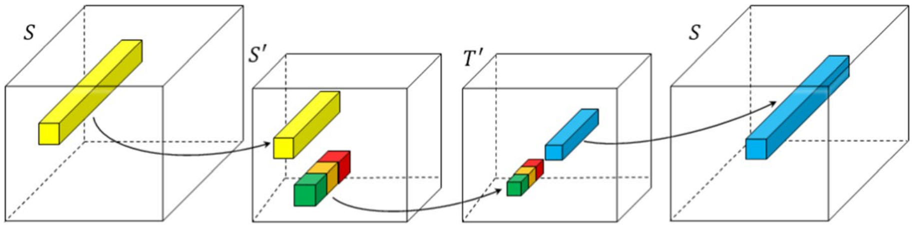


Fig. 28 BTD process

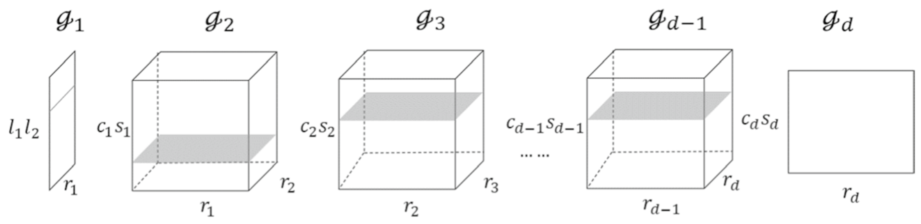


Fig. 29 TTD process

(ADC), to address the similarity search problem. The corresponding mathematical formulas of the two computation methods are:

$$\hat{d}(x,y) = d(q(x),y) = \sqrt{\sum_j d\left(q_j(x), q_j(y)\right)} \tag{17}$$

$$\hat{d}(x,y) = d(x,q(y)) = \sqrt{\sum_j d\left(u_i(x), q_j\left(u_j(x)\right)\right)} \tag{18}$$

where x denotes the query vector, y denotes a certain vector in the dataset with $d(.)$ and $\hat{d}(.)$ respectively representing the actual and approximate distances between vectors.

It can be observed intuitively that the primary difference between the two algorithms lies in whether the query vector is quantized. Additionally, the authors proposed the IVFADC algorithm, which modifies the ADC algorithm to two-layer quantization. The first layer is a coarse quantizer, where $k' = \sqrt{n}$ clusters are obtained based on K-means in the original vector space. The second layer performs PQ quantization on the residual datasets of each data and its corresponding quantization center, further reducing the computational burden.

In 2014, Gong et al. quantized weights by the K-means method (Gong et al. 2014). After clustering, weights of the same clusters were represented using the same index and mean center corresponding to the index. Meanwhile, the cluster index number and codebook were stored. Experimental results showed that for the 1000 ImageNet classification task, this method achieves 16–24× compression ratios with less than 1% accuracy drop. Wu et al. pinpointed that although many existing algorithms can attain good compression effects, they mostly target the FC layers rather than the entire network (Wu et al. 2016). Using the PQ algorithm, the authors accelerated and compressed the CNNs concurrently. This method can achieve 4–6× acceleration and 15–20× compression ratio almost without loss of precision and with less than 1% accuracy loss. In 2017, Cheng et al. performed the test using a similar method, which yielded the same acceleration and compression ratio. In 2019, Lee et al. proposed a DNN accelerator based on model compression technology for vector quantization (Lee et al. 2019). This algorithm well compresses NN models at FC layers and convolution layers. Experiments have demonstrated that it can achieve a compression rate of 10-20 times, 4.2× reduction for memory access and 2.05× throughput in every cycle for batch-one inference compared with the most advanced technology.

4.3.2 Binary quantization

Binary quantization refers to substituting binary weights for floating-point weights in the forward and reverse training of DNN. Courbariaux et al. introduced Binary Connect in 2015, which is a method for training DNNs using binary weights in the forward and backward propagation processes (Courbariaux et al. 2015). It retains the accuracy of the stored weights that are gradient accumulations. According to the experiments, nearly state-of-the-art results are obtained using Binary Connect on the MNIST, CIFAR-10, and SVHN with the same arrangements. Another work by Courbariaux et al. shows that quantifying both weights and activations can lead to greater compression of network storage (Courbariaux et al. 2016). A binarized neural network (BNN) proposed by Courbariaux et al. can achieve an accuracy equivalent to full-precision baseline on CIFAR-10 dataset (Courbariaux et al. 2016). To extend the BNN required by ImageNet classification task, Tang et al. optimized the training strategies for BNN (Tang et al. 2017), and reinforced strategy accuracy. In 2016, Rastegari et al. proposed XNOR-Net, which quantified both the input of the filter and the convolution layer to binary (Rastegari et al. 2016). Compared with BinaryConnect and BNN, the method exhibits an advantage on ImageNet, with a leading accuracy of 16%.

In 2019, Ding et al. suggested applying distribution loss in explicit regularization of the activation flow, and put forward a framework for systematic formulation of loss (Ding et al. 2019b). Wang et al. presented the binary convolutional neural network learning method (CI-BCNN) based on channel-wise interaction. Their work has generated lower computing and storage costs on both CIFAR10 and MNIST (Wang et al. 2019c; Ding et al. 2019b). In the same year, Nguyen et al. considered that frequent access to off-chip memory resulted in slow network processing speed. They proposed a Tera-OPS stream hardware accelerator to binary YOLO networks (Nguyen et al. 2019). The binary weight stores the entire network model in block RAMs with a field-programmable gate array (FPGA), which considerably reduces off-chip access, thus, significantly enhancing performance. In 2020, Lin et al. argued that binary networks such as Xnor-net had serious accuracy degradation problems. Some of the best binary networks (Wang et al. 2019c; Ding et al. 2019b) still experience severe precision degradation with large datasets such as ImageNet (Liu et al. 2020a). They proposed a binary network called ReActNet. ReActNet introduced react-Sign and React-Prelu, thus enabling explicit learning for distribution reshape and shift with almost zero cost. In 2021, J.Redfern et al. proposed a binary CNNs (BCNN) with all matrix operations quantized to 1-bit precision (Redfern et al. 2021). Substantial experiments of image recognition and object detection have proven that their algorithm is more effective when compared with the latest solutions.

4.3.3 Ternary quantization

Binarization quantifies floating-point numbers to 1bit directly, while ternary quantifies weights to 2bit. Although 2bit can express four numbers, it usually takes $\{-1, 0, 1\}$. In 2016, Li et al. proposed the ternary weight network (TWN), which was similar to BWN but constrained all weights to be ternary values (Li et al. 2016). Zhu et al. proposed trained ternary quantization (TTQ), which can lower the weight precision of neural networks to ternary values (Zhu et al. 2016). This method allows precision improvement of the ResNet-32, 44 and 56 models on CIFAR-10, as well as the AlexNet model on ImageNet. In 2017, Wen et al. put forward a distributed deep learning method called TernGrad, proving its convergence (Wen et al. 2017). The method uses ternary gradients to accelerate data parallelism. Experiments showed that the application of TernGrad on AlexNet causes no precision loss, whereas the mean accuracy loss on GoogLeNet is less than 2%. In the same year, Wang et al. presented a novel fixed-point factorized network (FFN) for pretrained models, in order to reduce the computational complexity and storage requirements (Wang et al. 2017). The weights significantly eliminates the multiply-accumulate (MAC) operations that consume the most resources. Substantial experiments on large ImageNet classification tasks demonstrated that the proposed FFN only requires one-thousandth of multiply operation while attaining comparable precision. In 2018, Wang et al. developed a two-step quantization (TSQ) framework, which decomposes the network quantization problem into the following two steps: the first step is to keep the weights at full precision, and sparsely quantize the activation values. The second step is to quantify the elements in various layers through layer-by-layer iteration (Wang et al. 2018). The layerwise quantization is capable of correcting quantization errors.

In 2020, Li et al. proposed Ternary Residual Quantization (TRQ), which introduced a stemresidual framework to ternary quantization (Li et al. 2021b). For ResNet-18, TRQ achieves only 0.3% performance drop on CIFAR-100. For VGG-Small, TRQ consistently surpasses the baseline by a margin of 2.1% and 3.5% on CIFAR-10 and CIFAR-100, respectively. Razani et al. proposed mixed quantized models, namely Smart Quantization (SQ) (Razani et al. 2021). SQ can realize adaptive combination of binary and ternary quantization. Quantization depth can be revised directly using the regularization function, and thus that the model needs to be merely trained once. Experiment shows that this method can adapt to quantization depth successfully and maintain a high model accuracy on MNIST and CIFAR10 benchmarks. Liu et al. developed pruning ternary quantization (PTQ), unifying pruning and quantization to provide a scope of size-accuracy trade-off (Liu et al. 2021a). PTQ converts regular weights to ternary orthonormal based on simple pruning and L2 projection. PTQ can provides no more than 46x compression ratio on ResNet-18, and corresponding accuracy reaches 65.36%. Additionally, PTQ compresses the ResNet-18 model by 48× and ResNet-50 model by 30×, while the top-1 accuracy on ImageNet decreases slightly by 4.4% and 1.68%, respectively.

4.3.4 Multi-bit quantization

The floating-point weights are not necessarily required to quantizate to 1-bit or 2-bit. Low-order quantization operations also generally bring good performance to the network without large loss of accuracy. In 2016, Lin et al. proposed a bit-width allocation algorithm for fixed-point quantizers of deep convolutional networks (DCNs) (Lin et al. 2016). The method uses the signal-to-quantization noise ratio (SQNR) as an index of performance classification, and derives the quantizer step size from the optimal quantizer design theory. Zhou et al. developed DoReFa-Net for training CNNs with low bitwidth weights and it realized activation with low bitwidth parameter gradients (Zhou et al. 2016). Particularly, in backward pass, parameter gradients will also be reduced to low bitwidth numbers through stochastic quantization prior to propagation to convolutional layers.

To solve the problem of accuracy loss resulting from binary network, Lin et al. adopted the linear combination of multiple binary weight bases for approximating the full precision weight in 2017, and also utilized multiple binary activation to decrease information loss (Lin et al. 2017a). In 2018, Leng et al. employed the idea of alternate direction multiplier method (ADMM) to decouple the discrete constraints of network from the continuous parameters, thereby transforming the original problem into several sub-problems (Leng et al. 2018). In 2020, Wu et al. investigated the low-bit quantization issue in face recognition (FR), and raised a new rotation consistent margin (RCM) loss function required by high-efficient low-bit FR quantization (Wu et al. 2020a). The results after compression to 3 bits and 4 bits reveal the superiority of the proposed method. Yang et al. believed that the traditional network quantization method added difficulties for the network to learn the low-order weight (Yang et al. 2020b). Thus, the differential method is used to search for discrete weights in neural network in order to avoid the non-differentiable problem of quantization functions.

In 2021, Yamamoto et al. proposed a learnable companding quantization (LCQ) where the network will flexibly and non-uniformly control weights and activation quantization to 2, 3, and 4 bits (Yamamoto 2021). In addition, a weight normalization technique called limited weight normalization (LWN) for increasing the stability of quantization training was suggested by them. Wang et al. argued that traditional quantization methods need to ensure the consistency of the dataset to perform bit-wise search, which makes the search of large-scale datasets costly in practical applications (Wang et al. 2021). Considering that, they proposed a Generalizable Mixed-Precision Quantization (GMPQ) method by capacity-aware attribution imitation to maintain the attribution rank consistency between the quantization model and the full-precision model, hence allowing the model to be generalized to large-scale datasets with only a small amount of data.

4.3.5 Hashing quantization

In 2015, Chen et al. proposed a novel network architecture called HashNet (Chen et al. 2015b). The network adopts a random weight sharing strategy to reduce the memory usage, and achieves a substantial reduction in the model size by reducing the inherent network redundancy. Figure 30 illustrates a Hashnet with random weight sharing when the compression factor is 1/4. The network comprises a hidden layer, four input units and two output units, and the connections in the network are randomized into three groups, of which V_1 and V_1 represent two virtual weight matrices. As shown in Fig. 30, the same weight values are indicated by the same color. Chen et al. encoded the weights with hash codes, and applied a low-cost hash function to randomly group the connection weights into hash buckets. All connections in the same hash buckets shared a parameter value. Experimental results demonstrated that the HashNet greatly reduces the storage requirements of NNs while fundamentally maintaining the generalization performance.

Hu et al. pinpointed in 2018 that the weights optimized by Chen et al.'s method were still floating-point numbers (Hu et al. 2018). To address the existing problem, they proposed a method called BWNH to train the binary weight networks through hashing. Based on their view, training a binary weight network can essentially be regarded as a hash problem. They proposed an alternate optimization method to learn hash codes. On the CIFAR10, CIFAR100 and ImageNet, the performance of BWNH is far superior to the current technological level. In 2020, Yuan et al. put forward a novel central similarity measure, i.e., Central Similarity Quantization (CSQ) (Yuan et al. 2020). CSQ can simulate the relations for similar and dissimilar pairs in order to improve the discriminability in produced hash codes.

4.4 Knowledge transfer

The concept of knowledge transfer can be described as the utilization of information acquired from large, complex, integrated neural network to form a compact neural network. Figure 31 describes the method of knowledge transfer, where the information flow shifts from a complex teacher network to a simpler student network. That is to say, the former network is marked by training the data of the latter network.

Training a compact model with the synthetic data generated by the complex network can well approximate the function and lower the possibility of overfitting. More importantly, it yields a more streamlined network than the original model, thus providing a new perspective for model compression and complex network acceleration. Bucilu et al. (2006) were the first to apply the idea of secondary learning to deep networks in 2006, who proposed a knowledge transfer-based model compression technique. This method fits the function mapping of teacher model by making the labels of student model as close as possible to those of teacher model. Three experimental methods are used to generate pseudo data, namely random, naive Bayesian estimation and MUNGE. The experimental results showed that the size of student network is reduced by 1000 times while the running speed increases by 1000 times (Bucilu et al. 2006).

Different from Bucilu et al.'s work, Ba et al. proposed to use the Logit inputs of Softmax layer as the supervisory information instead of labels. Their model allows the Logits output by the student model to fit the Logit values of the teacher model (Ba and Caruana 2013). Additionally, believing that a deeper network architecture is unnecessary, they designed a student model that is shallower than the teacher model (Ba and Caruana 2013). Every layer of the student model is wider than that of the teacher model, aiming to ensure that the network parameters of the two models are identical. Ba and Caruana (2013) used the logit labels of dataset obtained from the teacher network as knowledge to guide the training of student network, which could achieve recognition precision equivalent to that of deep

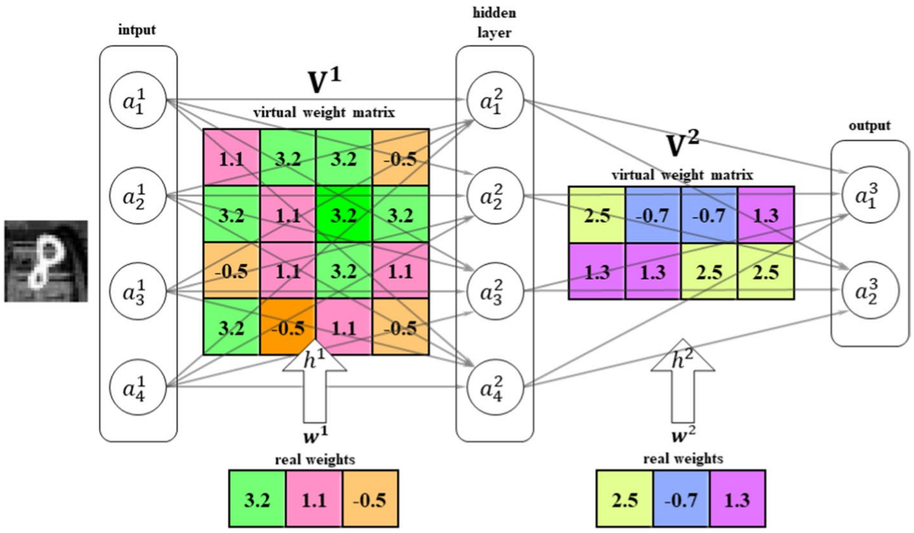


Fig. 30 Illustration of a Hashnet with random weight sharing at a compression factor of 1/4 (Chen et al. 2015b)

networks on both the TIMIT and CIFAR-10 databases. Experimental results suggested the possibility of training a shallower but wider student network to imitate the teacher network, whose effect is almost as good as that of the teacher network.

In 2014, Hinton et al. made further improvements to the above work. The knowledge distilling (KD) proposed by them uses an appropriate T value to generate a soft probability label, revealing the similarity between data structures (Hinton et al. 2015). In this way, the student network can achieve ultra-high precision on the MNIST dataset with less inference time. The framework can be summarized as follows: Suppose that T is a teacher network that outputs softmax $P_T = softmax\left(a_T\right)$, where a_T denotes the vector of teacher pre-softmax activation. 1) If the teacher model is a single network, a_T represents the weighted sum of output layer. 2) If the teacher model is an integrated result, both P_T and a_T are obtained by output averages (arithmetic average and geometric average, respectively) of different networks. Suppose that S is a student network, the parameter is W_S, and the output probability is $P_S = softmax\left(a_S\right)$, where a_S denotes the pre-softmax output of student. The student network is trained to make its output P_S similar to the teacher output P_T, as well as the true label y_{true}. Since P_T may be considerably close to a hot code representation of sample true label, the relaxation $\tau > 1$ is introduced to soften the signal generated by the teacher network output, providing more information during the training process. The output $\left(P_S^\tau\right)$ of student network has the same relaxation effect as the softened output $\left(P_T^\tau\right)$ of teacher. Their respective mathematical formulas are:

$$P_T^\tau = softmax\left(\frac{a_T}{\tau}\right),\ P_S^\tau = softmax\left(\frac{a_S}{\tau}\right) \tag{19}$$

The optimizer then trains the student network according to the following loss function:

$$\mathcal{L}_{KD}(W_s) = \mathcal{H}(y_{ture}, P_s) + \lambda\mathcal{H}(P_T^\tau, P_S^\tau) \tag{20}$$

where $\mathcal{H}$ refers to cross entropy, and λ denotes an adjustable parameter for balancing two cross entropies. Experiments have demonstrated that students trained in this way perform better than those trained directly using training data.

Romero et al. stated that the KD framework can attain encouraging results even when the architecture of student network is slightly deeper (Romero et al. 2014). However, with

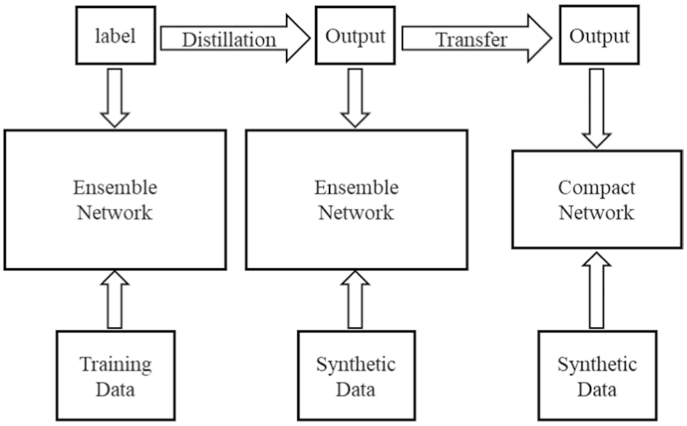


Fig. 31 Illustration of knowledge transfer

the increasing in depth of student network, KD training still faces the difficulty of optimizing deep networks. In existing problem, Romero et al. proposed a network architecture called FitNet (Romero et al. 2014), which uses Hints to train the first half of network, and then applies KD to finish the network training. The process of Hints-based student model training can be described as follows: As shown in Fig. 32a, the network training begins with a trained teacher network, as well as randomly initialized FitNet. As shown in Fig. 32b, a regressor parameterized by W_r is added onto the top of FitNet guided layer by adopting Hints. Meanwhile, the FitNet parameter W is introduced to the guided layer for minimizing the W_{Guided} value. As presented Fig. 32c, the KD framework is used to train W_S of the entire FitNet, in order to minimize the W_S^* value. Unlike Ba et al.'s conclusion, Romero et al. believed that using a finer and deeper network (i.e. a student network) to simulate a wider and shallower teacher network can bring higher classification accuracy (Romero et al. 2014). The depth of student network guarantees its performance, while its thinness reduces the computational complexity. To address the dimensional inconsistency between the outputs of student and teacher model middle layers, a regressor is added to FitNet. Initially, the first half of FitNet is trained using Hints. Then, the entire FitNet is trained by conventional network distillation. Experimental results demonstrated that FitNet increases the network depth and reduces the network parameters while enhancing the accuracy relative to the teacher model.

In 2015, Chen et al. obtained the network architecture of student model by proposing a function-preserving transformation-based network growth method, and named it Net-2Net (Chen et al. 2015a). Figure 33 compares the Net2Net workflow with the conventional method. It consists of Net2WiderNet and Net2DeeperNet, two strategies for network width and depth growth, respectively. Afterwards, it uses distillation to train the student model, which allows fast transfer of useful information from the teacher network to a deeper (or wider) student network. Obviously, the core idea of Net2Net aims to reuse the trained network parameters by copying, and then generate network on this basis.

Unlike the previous practice of representing knowledge with soft label probability, Luo et al. used the output of high-level neurons of teacher network to represent the domain knowledge needing to be transferred in 2016 (Luo et al. 2016). They selected the neurons most relevant to face recognition based on the learned essential characteristics of human face representations. The proposed approach does not cause any loss of information. The trained student network obtains a higher compression rate and a higher accuracy rate on the LFW than the teacher network. In 2017, Zagoruyko et al. supervised the learning of student network by using the attention feature maps in teacher network that can provide vision-related position information, and carried out attention transfer at the low, medium and high levels, greatly improving the performance of deep CNNs such as residual networks (Zagoruyko and Komodakis 2016a). In 2018, Lucas et al. developed an optimization method that combines Fisher pruning and knowledge transfer (Theis et al. 2018). Initially, a pretrained high-performance network is used to generate substantial saliency maps as domain knowledge. Subsequently, network is trained with the saliency maps, and redundant feature maps are removed by Fisher pruning, thereby accelerating the network operation by up to 10 times during image saliency prediction.

In 2019, Wang et al. developed the online ensemble distillation (OED) method which supported automatic pruning of blocks/layers on a target network through transferring the knowledge from a teacher network following an end-to-end mode (Wang et al. 2019a). OED method can adopt the fast iterative shrinkage-thresholding algorithm for quickly and strictly removing excess blocks, and corresponding soft masks equate zero. Experiments demonstrate that OED method compresses and accelerates multiple CNNs architectures while enhancing the robustness for pruned networks. In the same year, Ding et al. combined teacher–student learning with Tucker decomposition for compressing and accelerating convolutional layers based on CNN-DBLSTM (bidirectional long short-term memory) in optical character recognition (OCR) tasks, which greatly lowered footprint and computational cost without or with a slight decline in recognition accuracy (Ding et al. 2019a). In 2020, Touvron et al. introduced a teacher–student strategy specific to transformers depending on distillation, which was motivated by inducing convolutional bias into transformers (Touvron et al. 2021). By applying data-augmentation, different optimizers, and stochastic depth, the reference vision transformer achieves top-1 accuracy of 83.1% on ImageNet with no external data. In 2021, Aghli et al. considered that pruning methods exhibited excellent performance on simple networks like VGG or AlexNet while they could not be applied on complex network structures such as ResNets because pruning methods might cause the network structure to be broken (Aghli and Ribeiro 2021). They combined the pruning and distillation methods, based on the value pruning method only on a specific number of layers within the ResNet architecture, aiming to prevent damage to the network structure. Secondly, knowledge distillation structure and loss function are introduced for compressing the untrimmed layer. This compression method greatly reduces the size of the model while maintaining accuracy very close to the baseline model.

In 2021, Yao et al. presented a KD strategy for homogeneous and heterogeneous teacher–student pairs (Yao et al. 2021), which utilizes a detection KD pipeline to distill homogeneous and heterogeneous detector pairs. Apart from that, they considered that the traditional feature imitation paradigm loses semantic information in FPN by focusing on information-rich foreground regions through imitation masks. Therefore, contrastive distillation was introduced for the purpose of capturing the different feature regions between encoded associated information, and automatically performing soft matching between feature pairs in the FPN so as to provide useful information for the student network. Zhou et al. held that existing KD methods overlook the intrinsic correlation between individual knowledge and relational knowledge (Zhou et al. 2021). Thus, they decided to distill holistic knowledge by means of an attribution graph constructed between instances. Holistic knowledge is represented as a unified graph-based embedding learnt by aggregating the individual knowledge of the associated neighborhood samples, while the student network is learned by refining holistic knowledge in a contrastive manner. Zhu et al. proposed the Complementary Relation Contrastive Distillation (CRCD) method to distill the information

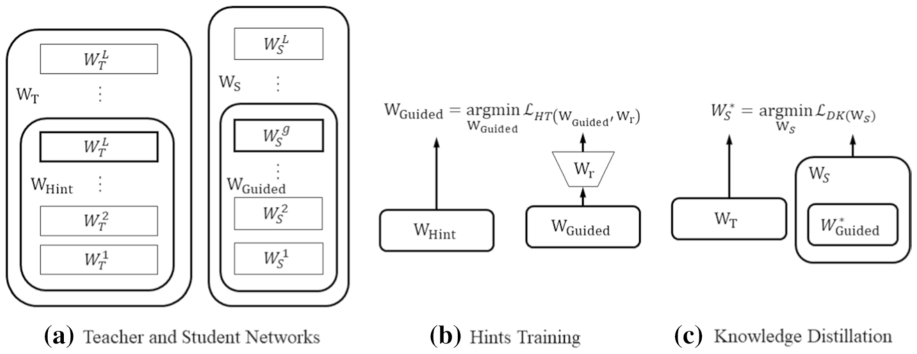

W_T^L
W_T
W_T^L
W_{Hint}
W_T^2
W_T^1
W_S^L
W_S
W_S^g
W_{Guided}
W_S^2
W_S^1
$W_{Guided} = \arg\min_{W_{Guided}} \mathcal{L}_{HT}(W_{Guided}, W_r)$
W_r
W_{Hint}
W_{Guided}
$W_S^* = \arg\min_{W_S} \mathcal{L}_{DK}(W_S)$
W_T
W_S
W^*_{Guided}
(a) Teacher and Student Networks
(b) Hints Training
(c) Knowledge Distillation

Fig. 32 Hints-based training of student model: **a** Teacher-student network, **b** Hints-based training, **c** KD framework (Romero et al. 2014)

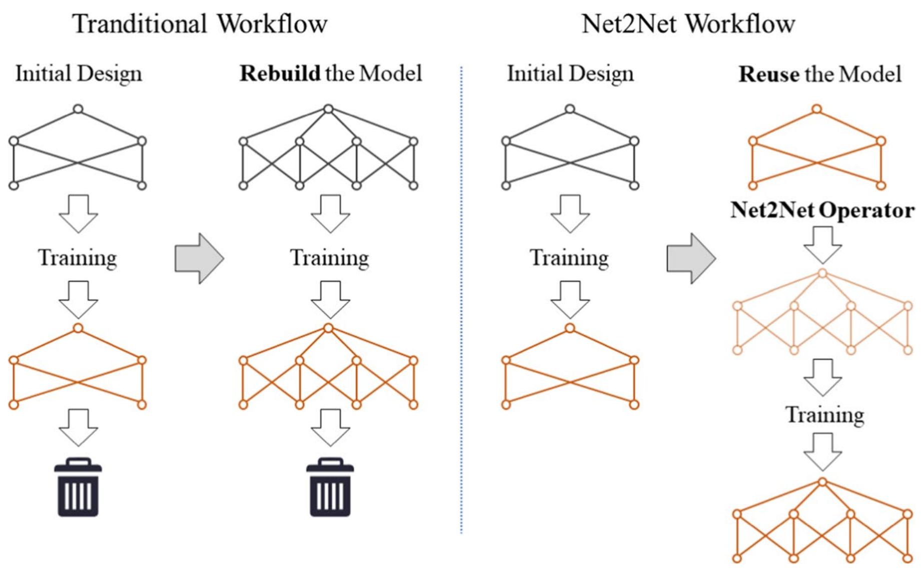


Fig. 33 Comparison of Net2Net workflow with conventional method (Chen et al. 2015a)

coming between samples (Zhu et al. 2021b). The method estimates the mutual relation modeled by two complementary elements in an anchor-based manner, and the anchor–student relationship is distilled under the supervision of its corresponding anchor–teacher relationship. Guo et al. considered that the feature information obtained in regions excluding objects is usually ignored in the target detection task for student networks (Guo et al. 2021). In addition, they considered that we should attach importance to information on feature regions, and presented algorithms containing decoupled features of the neck and decoupled proposals of the classification head suggested by decoupled features (DeFeat) to refine the network information. To reduce the high computational cost required to train large teacher models, Ji et al. proposed to train student networks to refine knowledge without pre-training teacher networks via Feature Refinement via Self-Knowledge Distillation (FRSKD) (Ji et al. 2021), which utilizes an auxiliary self-teacher network to transmit refined knowledge for the classifier network and performs self-knowledge distillation using soft labeling and feature map distillation.

5 Neural architecture search

The performance of CNNs in the target task is determined by the architecture. Among them, the better the network architecture, the better the efficiency and performance. The design of CNNs architecture usually requires specialized knowledge, which seriously hinders the development and application of deep learning. Neural Architecture Search (NAS) aims to reduce the need for designer experience by finding the optimal network structure. In 2016, Zoph et al. proposed a reinforcement learning-based NAS algorithm and achieved performance that surpassed artificial network architectures (Zoph and Le 2016).

Existing NAS algorithms mainly include search space, search strategy, and performance evaluation strategy (Elsken et al. 2019). Among them, there are three common forms of search space, namely overall form, modular form and level form. In more detail, the overall form includes direct connection (Zoph and Le 2016) and multiple branch (Lu et al. 2019; Maziarz et al. 2019). For the search space, the overall form can obtain a network architecture that is more suitable for the corresponding task, but requires more computational cost. Multiple branch has more complex network architecture and stronger network expression ability. In contrast, the general scope and unit structure of the modular form (Zhang et al. 2020; Zhong et al. 2020; Tan et al. 2019; Xu et al. 2021) are smaller, thus enhancing the search efficiency and transferability of the neural network structure. In the level form (Liu et al. 2019; Fang et al. 2020), the connections of the network as a whole and the way of operation within the connection layer are allowed to search, which can significantly reduce human intervention. Furthermore, an advanced search strategy will largely determine the performance of the searched CNNs architecture. In mainstream algorithms, commonly used search strategies can be mainly divided into evolutionary algorithm (Xie and Yuille 2017; Michel et al. 2019; Lu et al. 2019; Chu et al. 2020b; Yang et al. 2020a), reinforcement learning (Zoph and Le 2016; Tan et al. 2019; Zhong et al. 2020; Cheng et al. 2020), gradient descent-based method (Chen and Hsieh 2020; Chu et al. 2020a; Xu et al. 2019) and optimization algorithm based on sequential model (Liu et al. 2018a; Perez-Rua et al. 2018; Dong et al. 2018).

The acquisition of a successful NAS method usually means training and evaluating thousands of models, which inevitably requires up to thousands of GPU days (Zoph et al. 2018; Real et al. 2019). The huge search budget makes it difficult for the widespread application of NAS. Guo et al. adopted one-shot methods to reduce the computational cost based on weight sharing techniques (Guo et al. 2020). Through three stages of supernet training, subnet searching and subnet retraining, the search cost is reduced from thousands of GPU days to dozens of GPU days. NAS with Batch Normalization (BN-NAS) Chen et al. (2021a) proposed by Chen et al. accelerates the evaluation and training process. At the initial stage of training, the performance of the subnet is predicted. In supernet training, only BN parameters are trained, which further improves the training efficiency. An architecture named Dynamic Slimmable Network (DS-Net) proposed by Li et al. can adjust the number of filters of the network for different input dynamics (Li et al. 2021a). DS-Net contains a bifurcated two-stage training scheme inspired by NAS. Through In-place Ensemble Bootstrapping and Sandwich Gate Sparsification, the supernet training efficiency and Gate assisted training are improved respectively. The hourglass-inspired approach (HourNAS) proposed by Yang et al. can implement high-speed NAS algorithms. In their view, the vital blocks in the guaranteed path in the CNNs architecture that resemble the neck of an hourglass restrict the flow of information, while other blocks that resemble an hourglass sphere determine the complexity of the entire network (Yang et al. 2021). Since HourNAS preferentially searches for more important blocks, 77.0% of the architectures with Top-1 accuracy can be obtained in 0.1 GPU days. The relativistic architecture performance predictor proposed by Xu et al. can be used to rank different architectures (Xu et al. 2021). In more detail, a cell-based search space provides a unified supernet for all architectures and encodes them into tensors. By pairwise ranking loss functions, better architectures are obtained.

In addition, the architecture design of CNNs with transformer is also worth considering. For CNNs architectures adding ViT modules, ViT embedding depth, embedding dimension and number of heads usually need to be considered. Existing CNNs with transformer architectures are usually based on human design, which is highly dependent on the designers engineering experience in the application of attention mechanisms. Chen et al. first proposed a novel ViT model architecture search algorithm named AutoFormer (Chen et al. 2021b) which can be used for efficient training of the transformer supernet. The trained supernet is able to generate more high-quality transformers by directly inheriting the weights without additional fine-tuning or retraining. The AttentiveNAS strategy proposed by Wang et al. improves on the existing two-stage NAS (Wang et al. 2021a). In addition to the comparison of sampling strategies such as BestUp and WorstUp, sampling is effectively guided to the best or worst Pareto front during training.

6 CNNs applications

With strong capabilities and wide application fields, CNNs has good performance in various tasks. In the fields of video object segmentation, medical imaging, and face recognition, convolution-based derivative network architectures have achieved accuracy and efficiency that approach or even surpass humans. This section will briefly explore the application of CNNs in three fields.

6.1 Video object segmentation

With the large-scale development of short video platforms, the task of video object segmentation becomes more and more important. Intelligent analysis of these videos has become an extremely challenging task today. As a basic technique to solve such problems, video object segmentation algorithms provide good performance (Chen et al. 2020; Huang et al. 2020; Wang et al. 2021b). The Full-duplex Strategy Network (FSNet) proposed by Ji et al. provides robust video object segmentation Ji et al. (2021). The function of the relational cross-attention module (RCAM), a bidirectional interaction module included in FSNet, is to extract discriminative features of appearance and motion branches and ensure mutual constraints. Additionally, a bidirectional purification module (BPM) is introduced for updating inconsistent features.

In video object segmentation tasks, specific information can be dramatically distorted due to rapid movement, which increases the challenge. Existing works mainly focus on how to make full use of limited information for accurate segmentation (Lai et al. 2020; Seong et al. 2020; Wang et al. 2020b). Through the novel two-branch architecture, Mao et al. incorporated the advantages of offline learning and online learning in inductive reasoning (Mao et al. 2021). Furthermore, the semi-supervised video object segmentation (VOS) task incorporates the transformer architecture for the first time.

6.2 Medical imaging

The application of CNNs in medical image analysis has achieved great success. CNNs is commonly used in clinical applications such as disease diagnosis, disease localization, treatment planning, and treatment delivery. With the development of CNNs networks, computer program-based medical image analysis devices have achieved near or even surpassing human accuracy in many tasks.

Good performance in medical image analysis is built on a large human-annotated training set, which requires expertise and clinical experience (Roy et al. 2020; Tang et al. 2021a). The context relation encoder (CRE) proposed by Tang et al. enhances the feature relationship around the object boundary by the correlation between foreground and background, which achieves good results on a small number of medical image datasets. Based on the generative confrontation method, Dey et al. built a deformable template, in which the specificity of the generated template for attributes such as age and disease was significantly improved (Dey et al. 2021). Ye et al. proposed an unsupervised CNNs framework for motion tracking in Cardiac tagging magnetic resonance imaging (t-MRI) images (Ye et al. 2021). The estimation of the motion field is performed via bi-directional generative diffeomorphic registration neural network and differentiable composition layer. Zhao et al. proposed a segmentation-mesh-classification network (SMCN), which combines geometric and positional information for information diagnosis (Zhao et al. 2021b). Furthermore, SMCN builds an authoritative anatomical correspondence-aware organ mesh model by completing the task of learning the segmentation of pancreas and mass.

6.3 Face recognition

Face recognition is a biometric technology for identification based on facial feature information, which has now become a ubiquitous biometric authentication method. Despite more than 30 years of research and development, traditional face recognition technology still has some insurmountable problems. For example, face recognition models are vulnerable to spoofing attacks in some security protection tasks (Erdogmus and Marcel 2014; Costa-Pazo et al. 2016). Furthermore, most defense algorithms are ineffective against unprevented attacks (de Freitas Pereira et al. 2013). George et al. proposed a frame-level RGB-D face presentation attack detection (PAD) method (George and Marcel 2021). Among them, a loss function named cross-modal focal loss (CMFL) can be used to supervise a single channel in a multi-stream structure. Zhao et al. proposed a forgery detection network combined with an attention mechanism, which uses attention machine to pay attention to different local information in the network (Zhao et al. 2021). Furthermore, low-level texture features and high-level semantic features are aggregated under the guidance of attentional maps.

Since facial features change with age, most algorithms must adopt methods of reducing feature correlation or multi-age feature fusion to ensure the accuracy of network recognition. In this regard, Huang et al. proposed a framework called MTLFace for learning age-invariant identity-related representation (Huang et al. 2021). Based on the attention mechanism, facial features are divided into age-related features and identity-related features. To enhance the face recognition performance, a large-scale public dataset of cross-age faces annotated with age and gender is built.

7 Summarizations and prospects

Given the rapid evolution of hardware performance, deep CNNs will become the mainstream algorithm for computer vision-related tasks. Stronger computing power will support the computation of deeper networks and the processing of more sample data, thereby enhancing the nonlinear fitting and generalization abilities of models. Undoubtedly, CNNs architecture design is a promising research field, which may become one of the most extensively applied AI technologies in the future.

Concerning the CNNs migration problem on the mobile platforms, the pursuit of lighter network architectures is a major development direction. Strategies such as network architecture design, pruning, sparsification and tensor decomposition can all achieve network compression to some extents.

In this paper, the research achievements in recent years are summarized. The challenges and future prospects of CNNs development are listed as follows:

(1) Numerous neural network architectures, including CNNs, lack interpretability. The DeconvNet proposed by Zeiler et al. in 2013 replaces the convolution and pooling layers to monitor the learning scheme during model training, which has been verified on AlexNet. According to experimental results, most neurons in AlexNet are in an inactive state, and this phenomenon can be optimized by choosing smaller filters and convolution step size.

(2) Hyperparameter selection still relies on manual comparison and experience, although even subtle changes in hyperparameter values exert a tremendous influence on the network performance. The strategy for hyperparameter selection requires a standardized design concept, as well as a reasonable optimization method for adjusting the hyperparameter values. Neural architecture search (NAS) algorithms are also necessary to design networks that search for the right network architecture automatically.

(3) Effective training of CNNs requires powerful hardware resources, such as GPUs. However, it remains necessary to explore how to utilize CNNs in embedded and mobile devices effectively. Designing hardware-friendly deep models facilitates the engineering implementation of deep learning, which is also a key research direction of network architecture optimization.

(4) The bottom-up and top-down feature map stitching in the pyramid architecture provided richer semantic information for multi-size feature extraction in the target detection task. However, shortcut connections lack more theoretical justification. The introduction of MHSA idea can provide more global information for CNN, but the large number of FC structures in the MHSA module greatly increase the computational load of the network. A more advanced attention extraction algorithm is necessary.

(5) Regarding network pruning algorithms, most of the existing methods remove redundant connections or neurons in networks. Such low-level pruning has non-structural risks. During computer operation, irregular memory access modes hinder further network acceleration instead. Additionally, the evaluation systems for convolution kernels and their weight importance remain rather simple. Hence, it is of vital necessity to propose a more effective way to measure the influence of pruning objects on the models.

(6) Tensor decomposition can tremendously accelerate the model operation process, and the mathematical principles applied during the decomposition process are more conducive to explaining the optimization mechanism of network architectures intuitively. However, tensor decomposition has a poor acceleration effect on network models with small convolution kernels, and is generally incapable of compressing the size of network models.

(7) Network quantization can significantly reduce the model size. However, it increases the operational complexity. During quantization, some special processing is required. Otherwise, the precision loss will be severer. Besides, the quantization itself usually loses some precision. An appropriate quantization strategy will be able to reduce the complexity of the model while minimizing the loss of accuracy. In addition, a hybrid

accuracy quantization strategy can be used to reduce the magnitude of the parameters to a reasonable degree based on the contribution.

(8) Knowledge transfer can guide the training of student network via the teacher network, which has a strong application value in small sample contexts. However, the training difficulty varies among different student network architectures. Besides, the final effect also varies. Hence, to construct student network architectures based on the teacher network architectures, the designers are required to have richer theoretical foundation and engineering experience. Moreover, the commissioning cycle of knowledge transfer is longer than other methods.

(9) Most model acceleration methods achieve optimization for image recognition tasks, while few are dedicated to accelerating tasks in other fields of computer vision, such as object detection. Additionally, the evaluation systems for network compression algorithms are rather weak, which generally focus on the comparison of network parameters and running time. As future research directions, the size and speed of networks can be balanced, and a network performance evaluation system under multiple scenarios can be provided.

Acknowledgements This work was supported by the National Natural Science Foundation of China under Grants No. 61573330 and 61720106009.

References

Aghli N, Ribeiro E (2021) Combining weight pruning and knowledge distillation for CN compression. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 3191–3198

Alzubaidi L, Al-Shamma O, Fadhel MA, Farhan L, Zhang J, Duan Y (2020) Optimizing the performance of breast cancer classification by employing the same domain transfer learning from hybrid deep convolutional neural network model. Electronics 9(3):445

Alzubaidi L, Zhang J, Humaidi AJ, Al-Dujaili A, Duan Y, Al-Shamma O, Santamaría J, Fadhel MA, Al-Amidie M, Farhan L (2021) Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. J Big Data 8(1):1–74

Alzubaidi L, Al-Shamma O, Fadhel MA, Farhan L, Zhang J (2018) Classification of red blood cells in sickle cell anemia using deep convolutional neural network. In: International conference on intelligent systems design and applications. Springer, Cham, pp 550–559

Anwar S, Hwang K, Sung W (2017) Structured pruning of deep convolutional neural networks. ACM J Emerg Technol Comput Syst JETC) 13(3):1–18

Astrid M, Lee S-I (2017) Cp-decomposition with tensor power method for convolutional neural networks compression. In: 2017 IEEE international conference on Big Data and Smart Computing (BigComp). IEEE, pp 115–118

Ba LJ, Caruana R (2013) Do deep nets really need to be deep? Adv Neural Inf Process Syst 3:2654–2662

Bengio Y (2013) Deep learning of representations: Looking forward. In: International conference on statistical language and speech processing. Springer, Berlin, pp 1–37

Bochkovskiy A, Wang C-Y, Liao H-YM (2020) Yolov4: optimal speed and accuracy of object detection. arXiv preprint. arXiv:2004.10934

Bucilu C, Caruana R, Niculescu-Mizil A (2006) Model compression. In: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining, pp 535–541

Chen X, Hsieh C-J (2020) Stabilizing differentiable architecture search via perturbation-based regularization. In: International conference on machine learning (PMLR), pp 1554–1565

Chen T, Goodfellow I, Shlens J (2015a) Net2net: Accelerating learning via knowledge transfer. Computer Science

Chen W, Wilson J, Tyree S, Weinberger K, Chen Y (2015b) Compressing neural networks with the hashing trick. In: International conference on machine learning (PMLR), pp 2285–2294

Chen X, Li Z, Yuan Y, Yu G, Shen J, Qi D (2020) State-aware tracker for real-time video object segmentation. In: Proceedings of the IEEE/CVF international conference on computer vision and pattern recognition, pp 9384–9393

Chen B, Li P, Li B, Lin C, Li C, Sun M, Yan J, Ouyang W (2021a) BN-NAS: neural architecture search with batch normalization. In: Proceedings of the IEEE/CVF international conference on computer vision, pp 307–316

Chen M, Peng H, Fu J, Ling H (2021b) Autoformer: searching transformers for visual recognition. In: Proceedings of the IEEE/CVF international conference on computer vision, pp 12270–12280

Chen Y, Dai X, Chen D, Liu M, Dong X, Yuan L, Liu Z (2021c) Mobile-former: bridging mobilenet and transformer. arXiv preprint. arXiv:2108.05895

Cheng A-C, Lin CH, Juan D-C, Wei W, Sun M (2020) InstaNAS: instance-aware neural architecture search. In: Proceedings of the AAAI conference on artificial intelligence, vol 34, pp 3577–3584

Chollet F (2017) Xception: deep learning with depthwise separable convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 1251–1258

Choudhary T, Mishra V, Goswami A, Sarangapani J (2020) A comprehensive survey on model compression and acceleration. Artif Intell Rev 53(7):5113–5155

Chu X, Zhou T, Zhang B, Li J (2020a) Fair DARTS: eliminating unfair advantages in differentiable architecture search. In: European conference on computer vision. Springer, Munich, pp 465–480

Chu X, Zhang B, Xu R (2020b) Multi-objective reinforced evolution in mobile neural architecture search. In: European European conference on computer vision. Springer, Munich, pp 99–113

Clevert D-A, Unterthiner T, Hochreiter S (2015) Fast and accurate deep network learning by exponential linear units (ELUS). arXiv preprint. arXiv:1511.07289

Costa-Pazo A, Bhattacharjee S, Vazquez-Fernandez E, Marcel S (2016) The replay-mobile face presentation-attack database. In: International conference of the Biometrics Special Interest Group (BIOSIG). IEEE, pp 1–7

Courbariaux M, Bengio Y, David J-P (2015) BinaryConnect: training deep neural networks with binary weights during propagations. In: Advances in neural information processing systems, pp 3123–3131

Courbariaux M, Hubara I, Soudry D, El-Yaniv R, Bengio Y (2016) Binarized neural networks: training deep neural networks with weights and activations constrained to +1 or −1. arXiv preprint. arXiv:1602.02830

Csáji BC et al (2001) Approximation with artificial neural networks, vol 24(48). MSc thesis, Faculty of Sciences, Eötvös Loránd University, p 7

Dai Z, Liu H, Le QV, Tan M (2021) Coatnet: marrying convolution and attention for all data sizes. arXiv preprint. arXiv:2106.04803

Dalal N, Triggs B (2005) Histograms of oriented gradients for human detection. In: 2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05), vol 1. IEEE, pp 886–893

de Freitas Pereira T, Anjos A, De Martino JM, Marcel S (2013) Can face anti-spoofing countermeasures work in a real world scenario? In: Proceedings of the 2013 international conference on biometrics (ICB). IEEE, pp 1–8

Denil M, Shakibi B, Dinh L, Ranzato M, Freitas ND (2013) Predicting parameters in deep learning. University of British Columbia, Vancouver

Denton EL, Zaremba W, Bruna J, LeCun Y, Fergus R (2014) Exploiting linear structure within convolutional networks for efficient evaluation. In: Advances in neural information processing systems, pp 1269–1277

Devlin J, Chang M-W, Lee K, Toutanova K (2018) Bert: pre-training of deep bidirectional transformers for language understanding. arXiv preprint. arXiv:1810.04805

Dey N, Ren M, Dalca AV, Gerig G (2021) Generative adversarial registration for improved conditional deformable templates. In: Proceedings of the IEEE/CVF international conference on computer vision, pp 3929–3941

Ding H, Chen K, Huo Q (2019a) Compressing CNN–DBLSTM models for ocr with teacher–student learning and Tucker decomposition. Pattern Recogn 96:106957

Ding R, Chin T-W, Liu Z, Marculescu D (2019b) Regularizing activation distribution for training binarized deep networks. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 11408–11417

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, pp 4510–4520

Dong J-D, Cheng A-C, Juan D-C, Wei W, Sun M (2018) DPP-Net: deviceevice-aware progressive search for pareto-optimal neural architectures. In: Proceedings of the European conference on computer vision (ECCV), pp 517–531

Dosovitskiy A, Beyer L, Kolesnikov A, Weissenborn D, Zhai X, Unterthiner T, Dehghani M, Minderer M, Heigold G, Gelly S, et al (2020) An image is worth 16×16 words: transformers for image recognition at scale. arXiv preprint. arXiv:2010.11929

Elsken T, Metzen JH, Hutter F (2019) Neural architecture search: a survey. J Mach Learn Res 20(1):1997–2017

Erdogmus N, Marcel S (2014) Spoofing face recognition with 3d masks. IEEE Trans Inf Forensics Security 9(7):1084–1097

Fang J, Sun Y, Zhang Q, Li Y, Liu W, Wang X (2020) Densely connected search space for more flexible neural architecture search. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 10628–10637

Fukushima K, Miyake S (1982) Neocognitron: a self-organizing neural network model for a mechanism of visual pattern recognition. In: Competition and cooperation in neural nets. Springer, Heidelberg, pp 267–285

Ge Z, Liu S, Wang F, Li Z, Sun J (2021) YOLOX: exceeding YOLO Series in 2021. arXiv preprint. arXiv:2107.08430

George A, Marcel S (2021) Cross modal focal loss for rgbd face anti-spoofing. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 7882–7891

Fukushima K (1989) Neocognitron: a hierarchical neural network capable of visual pattern recognition. Neural Netw 1:119–130

Girshick R (2015) Fast R-CNN. In: Proceedings of the IEEE international conference on computer vision, pp 1440–1448

Girshick R, Donahue J, Darrell T, Malik J (2014) Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 580–587

Gong Y, Liu L, Ming Y, Bourdev L (2014) Compressing deep convolutional networks using vector quantization. Comput Sci+++

Graham B, El-Nouby A, Touvron H, Stock P, Joulin A, Jégou H, Douze M (2021) Levit: a vision transformer in convnet's clothing for faster inference. arXiv preprint. arXiv:2104.01136

Gu J, Wang Z, Kuen J, Ma L, Shahroudy A, Shuai B, Liu T, Wang X, Wang G, Cai J et al (2018) Recent advances in convolutional neural networks. Pattern Recogn 77:354–377

Gulcehre C, Cho K, Pascanu R, Bengio Y (2014) Learned-norm pooling for deep feedforward and recurrent neural networks. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, pp. 530–546

Guo Y, Yao A, Chen Y (2016) Dynamic network surgery for efficient DNNs. arXiv preprint. arXiv:1608.04493

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. Springer, Munich, pp 544–560

Guo J, Han K, Wang Y, Wu H, Chen X, Xu C, Xu C (2021) Distilling object detectors via decoupled features. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 2154–2164

Han S, Mao H, Dally WJ (2015a) Deep compression: compressing deep neural networks with pruning, trained quantization and Huffman coding. Fiber 56(4):3–7

Han S, Pool J, Tran J, Dally WJ (2015b) Learning both weights and connections for efficient neural networks. MIT, Cambridge

Han D, Kim J, Kim J (2017) Deep pyramidal residual networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 5927–5935

Hanson S, Pratt L (1988) Comparing biases for minimal network construction with back-propagation. Adv Neural Inf Process Syst 1:177–185

Hassibi B, Stork DG, Wolff G, Watanabe T (1994) Optimal brain surgeon: extensions and performance comparison. In: Cowan JD, Tesauro G, Alspector J (eds) Advances in neural information processing systems, vol 6. Morgan Kaufmann, San Mateo, pp 263–270

Håstad J, Goldmann M (1991) On the power of small-depth threshold circuits. Comput Complex 1(2):113–129

He K, Sun J (2015) Convolutional neural networks at constrained time cost. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 5353–5360

He K, Zhang X, Ren S, Sun J (2015a) Spatial pyramid pooling in deep convolutional networks for visual recognition. IEEE Trans Pattern Anal Mach Intell 37(9):1904–1916

He K, Zhang X, Ren S, Sun J (2015b) Delving deep into rectifiers: surpassing human-level performance on ImageNet classification. In: Proceedings of the IEEE international conference on computer vision, pp 1026–1034

He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 770–778

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, pp 1389–1397

Hinton GE, Salakhutdinov RR (2006) Reducing the dimensionality of data with neural networks. Science 313(5786):504–507

Hinton G, Vinyals O, Dean J (2015) Distilling the knowledge in a neural network. Comput Sci 14(7):38–39

Howard AG, Zhu M, Chen B, Kalenichenko D, Wang W, Weyand T, Andreetto M, Adam H (2017) Mobilenets: efficient convolutional neural networks for mobile vision applications. arXiv preprint. arXiv:1704.04861

Howard A, Sandler M, Chu G, Chen L-C, Chen B, Tan M, Wang W, Zhu Y, Pang R, Vasudevan V et al (2019) Searching for mobilenetv3. In: Proceedings of the IEEE/CVF international conference on computer vision, pp 1314–1324

Hua W, Zhou Y, De Sa C, Zhang Z, Suh GE (2019) Boosting the performance of cnn accelerators with dynamic fine-grained channel gating. In: Proceedings of the 52nd Annual IEEE/ACM international symposium on microarchitecture, pp 139–150

Huang G, Liu Z, Van Der Maaten L, Weinberger KQ (2017) Densely connected convolutional networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 4700–4708

Huang G, Liu S, Van der Maaten L, Weinberger KQ (2018) Condensenet: an efficient densenet using learned group convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 2752–2761

Huang X, Xu J, Tai Y-W, Tang C-K (2020) Fast video object segmentation with temporal aggregation network and dynamic template matching. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 8879–8889

Huang H, Zhang J, Shan H (2021) When age-invariant face recognition meets face age synthesis: a multi-task learning framework. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 7282–7291

Hubel DH, Wiesel TN (1962) Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. J Physiol 160(1):106–154

Hubel DH, Wiesel TN (2009) Republication of The Journal of Physiology (1959) 148, 574-591: Receptive fields of single neurones in the cat's striate cortex. 1959. J Physiol 587(Pt 12):2721–2732

Hu H, Peng R, Tai YW, Tang CK (2016) Network trimming: a data-driven neuron pruning approach towards efficient deep architectures. arXiv preprint. arXiv:1607.03250

Hu Q, Wang P, Cheng J (2018) From hashing to CNNs: training binary weight networks via hashing. In: 32nd AAAI conference on artificial intelligence, vol 32

Ioffe S, Szegedy C (2015) Batch normalization: accelerating deep network training by reducing internal covariate shift. In: International conference on machine learning (PMLR), pp 448–456

Jaderberg M, Vedaldi A, Zisserman A (2014) Speeding up convolutional neural networks with low rank expansions. Comput Sci 4(4):XIII

Jegou H, Douze M, Schmid C (2010) Product quantization for nearest neighbor search. IEEE Trans Pattern Anal Mach Intell 33(1):117–128

Ji G-P, Fu K, Wu Z, Fan D-P, Shen J, Shao L (2021) Full-duplex strategy for video object segmentation. In Proceedings of the IEEE/CVF international conference on computer vision, pp 4922–4933

Ji M, Shin S, Hwang S, Park G, Moon I-C (2021) Refine myself by teaching myself: Feature refinement via self-knowledge distillation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 10664–10673

Jia Y et al (2013) An open source convolutional architecture for fast feature embedding. In: Proceedings of the 22nd ACM international conference on multimedia, Orlando, FL, pp 675–678

Jo G, Lee G, Shin D (2020) Exploring group sparsity using dynamic sparse training. In: 2020 IEEE international conference on consumer electronics-Asia (ICCE-Asia). IEEE, pp 1–2

Jocher G (2020) Yolo v5. https://github.com/ultralytics/yolov5 Accessed July 2020

Khan A, Sohail A, Zahoora U, Qureshi AS (2020) A survey of the recent architectures of deep convolutional neural networks. Artif Intell Rev 53(8):5455–5516

Kim YD, Park E, Yoo S, Choi T, Yang L, Shin D (2015) Compression of deep convolutional neural networks for fast and low power mobile applications. Comput Sci 71(2):576–584

Kim S-W, Kook H-K, Sun J-Y, Kang M-C, Ko S-J (2018) Parallel feature pyramid network for object detection. In: Proceedings of the European conference on computer vision (ECCV), pp 234–250

Kong T, Sun F, Tan C, Liu H, Huang W (2018) Deep feature pyramid reconfiguration for object detection. In: Proceedings of the European conference on computer vision (ECCV), pp 169–185

Kozyrskiy N, Phan A-H (2020) Cnn acceleration by low-rank approximation with quantized factors. arXiv preprint. arXiv:2006.08878

Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. Adv Neural Inf Process Syst 25:1097–1105

Lai Z, Lu E, Xie W (2020) Mast: a memory-augmented self-supervised tracker. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 6479–6488

Lebedev V, Lempitsky V (2016) Fast convnets using group-wise brain damage. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 2554–2564

Lebedev V, Ganin Y, Rakhuba M, Oseledets I, Lempitsky V (2014) Speeding-up convolutional neural networks using fine-tuned cp-decomposition. In: International conference on learning representations (ICLR Poster)

LeCun Y, Denker JS, Solla SA (1990) Optimal brain damage. In: Touretzky DS (ed) Advances in neural information processing systems. Morgan Kaufmann, San Francisco, pp 598–605

LeCun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. Proc IEEE 86(11):2278–2324

LeCun Y, Kavukcuoglu K, Farabet C (2010) Convolutional networks and applications in vision. In: Proceedings of 2010 IEEE international symposium on circuits and systems. IEEE, pp 253–256

Lee KH, Verma N (2013) A low-power processor with configurable embedded machine-learning accelerators for high-order and adaptive analysis of medical-sensor signals. IEEE J Solid-State Circuits 48(7):1625–1637

Lee H, Wu Y-H, Lin Y-S, Chien S-Y (2019) Convolutional neural network accelerator with vector quantization. In: 2019 IEEE international symposium on circuits and systems (ISCAS). IEEE, pp 1–5

Lee K, Kim H, Lee H, Shin D (2020) Flexible group-level pruning of deep neural networks for on-device machine learning. In: 2020 Design, automation & test in Europe cnference & exhibition (DATE). IEEE, pp 79–84

Lee D, Wang D, Yang Y, Deng L, Zhao G, Li G (2021) QTTNet: quantized tensor train neural networks for 3D object and video recognition. Neural Netw 141:420–432

Leng C, Dou Z, Li H, Zhu S, Jin R (2018) Extremely low bit neural network: Squeeze the last bit out with ADMM. In: Thirty-Second AAAI Conference on Artificial Intelligence

Li F, Zhang B, Liu B (2016) Ternary weight networks. arXiv preprint. arXiv:1605.04711

Li H, Kadav A, Durdanovic I, Samet H, Graf HP (2016) Pruning filters for efficient convnets. arXiv preprint. arXiv:1608.08710

Li L, Zhu J, Sun M-T (2019) A spectral clustering based filter-level pruning method for convolutional neural networks. IEICE Trans Inf Syst 102(12):2624–2627

Li C, Wang G, Wang B, Liang X, Li Z, Chang X (2021a) Dynamic slimmable network. In: Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition, pp 8607–8617

Li Y, Ding W, Liu C, Zhang B, Guo G (2021b) TRQ: Ternary neural networks with residual quantization. In: Proceedings of the AAAI conference on artificial intelligence

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, pp 8018–8027

Lin M, Chen Q, Yan S (2013) Network in network. arXiv preprint. arXiv:1312.4400

Lin D, Talathi S, Annapureddy S (2016) Fixed point quantization of deep convolutional networks. In: International conference on machine learning (PMLR), pp 2849–2858

Lin X, Zhao C, Pan W (2017a) Towards accurate binary convolutional neural network. In: Advances in neural information processing systems

Lin T-Y, Dollár P, Girshick R, He K, Hariharan B, Belongie S (2017b) Feature pyramid networks for object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 2117–2125

Lin R, Ko C-Y, He Z, Chen C, Cheng Y, Yu H, Chesi G, Wong N (2020) Hotcake: higher order tucker articulated kernels for deeper CNN compression. In: 2020 IEEE 15th international conference on solid-state & integrated circuit technology (ICSICT). IEEE, pp 1–4

Liu B, Wang M, Foroosh H, Tappen M, Pensky M (2015) Sparse convolutional neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 806–814

Liu W, Anguelov D, Erhan D, Szegedy C, Reed S, Fu C-Y, Berg AC (2016) SSD: single shot multibox detector. In: European conference on computer vision. Springer, Cham, pp 21–37
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, pp 2736–2744
Liu C, Zoph B, Neumann M, Shlens J, Hua W, Li L-J, Fei-Fei L, Yuille A, Huang J, Murphy K (2018a) Progressive neural architecture search. In: Proceedings of the European conference on computer vision (ECCV), pp 19–34
Liu X, Pool J, Han S, Dally WJ (2018b) Efficient sparse-Winograd convolutional neural networks. arXiv preprint. arXiv:1802.06367
Liu C, Chen L-C, Schroff F, Adam H, Hua W, Yuille AL, Fei-Fei L (2019) Auto-deeplab: hierarchical neural architecture search for semantic image segmentation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 82–92
Liu Z, Shen Z, Savvides M, Cheng K-T (2020a) Reactnet: towards precise binary neural network with generalized activation functions. In: European conference on computer vision. Springer, Berlin, pp 143–159
Liu J, Xu Z, Shi R, Cheung RC, So HK (2020b) Dynamic sparse training: Find efficient sparse network from scratch with trainable masked layers. arXiv preprint. arXiv:2005.06870
Liu D, Chen X, Fu J, Liu X (2021a) Pruning ternary quantization. arXiv preprint. arXiv:2107.10998
Liu Z, Lin Y, Cao Y, Hu H, Wei Y, Zhang Z, Lin S, Guo B (2021b) Swin transformer: hierarchical vision transformer using shifted windows. arXiv preprint. arXiv:2103.14030
Liu L, Zhang S, Kuang Z, Zhou A, Xue J-H, Wang X, Chen Y, Yang W, Liao Q, Zhang W (2021c) Group fisher pruning for practical network compression. In: International conference on machine learning (PMLR), pp 7021–7032
Lowe DG (2004) Distinctive image features from scale-invariant keypoints. Int J Comput Vis 60(2):91–110
Lu Z, Whalen I, Boddeti V, Dhebar Y, Deb K, Goodman E, Banzhaf W (2019) NSGA-Net: neural architecture search using multi-objective genetic algorithm. In: Proceedings of the genetic and evolutionary computation conference, pp 419–427
Luo P, Zhu Z, Liu Z, Wang X, Tang X (2016) Face model compression by distilling knowledge from neurons. In: Proceedings of the 30th AAAI conference on artificial intelligence. AAAI, Phoenix
Luo J-H, 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, pp 5058–5066
Ma N, Zhang X, Zheng H-T, Sun J (2018) Shufflenet v2: Practical guidelines for efficient cnn architecture design. In: Proceedings of the European conference on computer vision (ECCV), pp 116–131
Ma X, Guo F-M, Niu W, Lin X, Tang J, Ma K, Ren B, 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, vol 34, pp 5117–5124
Maas AL, Hannun AY, Ng AY, et al (2013) Rectifier nonlinearities improve neural network acoustic models. Proc ICML 30:3. Citeseer
Mao H, Han S, Pool J, Li W, Liu X, Wang Y, Dally WJ (2017) Exploring the regularity of sparse structure in convolutional neural networks. arXiv preprint. arXiv:1705.08922
Mao Y, Wang N, Zhou W, Li H (2021) Joint inductive and transductive learning for video object segmentation. In: Proceedings of the IEEE/CVF international conference on computer vision, pp 9670–9679
Maziarz K, Tan M, Khorlin A, Chang K-YS, Gesmundo A (2019) Evo-nas: Evolutionary-neural hybrid agent for architecture search
Mehta S, Rastegari M (2021) MobileViT: Light-weight, general-purpose, and mobile-friendly vision transformer. arXiv preprint. arXiv:2110.02178
Michel G, Alaoui MA, Lebois A, Feriani A, Felhi M (2019) Dvolver: Efficient pareto-optimal neural network architecture search. arXiv preprint. arXiv:1902.01654
Mikolov T, Karafiát M, Burget L, Černocky J, Khudanpur S (2010) Eleventh annual conference of the international speech communication association
Montremerlo M, Beeker J, Bhat S, Dahlkamp H (2008) The stanford entry in the urban challenge. J Field Robot 7(9):468–492
Nair V, Hinton GE (2010) Rectified linear units improve restricted Boltzmann machines. In: International conference on international conference on machine learning (ICML)
Nguyen DT, Nguyen TN, Kim H, Lee H-J (2019) A high-throughput and power-efficient FPGA implementation of YOLO CNN for object detection. IEEE Trans Very Large Scale Integr (VLSI) Syst 27(8):1861–1873
Niu W, Ma X, Lin S, Wang S, Qian X, Lin X, Wang Y, Ren B (2020) PatDNN: achieving real-time DNN execution on mobile devices with pattern-based weight pruning. In: Proceedings of the 25th

international conference on architectural support for programming languages and operating systems, pp 907–922

Novikov A, Podoprikhin D, Osokin A, Vetrov D (2015) Tensorizing neural networks. arXiv preprint. arXiv:1509.06569

Oseledets IV (2011) Tensor-train decomposition. SIAM J Sci Comput 33(5):2295–2317

Peng Z, Huang W, Gu S, Xie L, Wang Y, Jiao J, Ye Q (2021) Conformer: local features coupling global representations for visual recognition. arXiv preprint. arXiv:2105.03889

Perez-Rua J-M, Baccouche M, Pateux S (2018) Efficient progressive neural architecture search. arXiv preprint. arXiv:1808.00391

Punyani P, Gupta R, Kumar A (2020) Neural networks for facial age estimation: a survey on recent advances. Artif Intell Rev 53(5):3299–3347

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, pp 6718–6727

Radford A, Narasimhan K, Salimans T, Sutskever I (2018) Improving language understanding by generative pre-training

Radford A, Wu J, Child R, Luan D, Amodei D, Sutskever I et al (2019) Language models are unsupervised multitask learners. OpenAI Blog 1(8):9

Rastegari M, Ordonez V, Redmon J, Farhadi A (2016) XNOR-Net: Imagenet classification using binary convolutional neural networks. In: European conference on computer vision. Springer, Cham, pp 525–542

Razani R, Morin G, Sari E, Nia VP (2021) Adaptive binary-ternary quantization. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 4613–4618

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, vol 33, pp 4780–4789

Redfern AJ, Zhu L, Newquist MK (2021) BCNN: a binary CNN with all matrix OPS quantized to 1 bit precision. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 4604–4612

Redmon J, Farhadi A (2017) Yolo9000: better, faster, stronger. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 7263–7271

Redmon J, Farhadi A (2018) Yolov3: an incremental improvement. arXiv preprint. arXiv:1804.02767

Redmon J, Divvala S, Girshick R, Farhadi A (2016) You only look once: unified, real-time object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 779–788

Ren S, He K, Girshick R, Sun J (2015) Faster R-CNN: towards real-time object detection with region proposal networks. Adv Neural Inf Process Syst 28:91–99

Romero A, Ballas N, Kahou SE, Chassang A, Gatta C, Bengio Y (2014) Fitnets: Hints for thin deep nets. rXiv preprint. arXiv:1412.6550

Roy AG, Siddiqui S, Pölsterl S, Navab N, Wachinger C (2020) squeeze & exciteguided few-shot segmentation of volumetric images. Med Image Anal 59:101587

Sandler M, Howard A, Zhu M, Zhmoginov A, Chen L-C (2018) Mobilenetv2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE conference on computer vision and pattern recognition, pp 4510–4520

Seong H, Hyun J, Kim E (2020) Kernelized memory network for video object segmentation. In: European conference on computer vision. Springer, Cham, pp 629–645

Sharma M, Markopoulos PP, Saber E, Asif MS, Prater-Bennette A (2021) Convolutional auto-encoder with tensor-train factorization. In: Proceedings of the IEEE/CVF international conference on computer vision, pp 198–206

Simard PY, Steinkraus D, Platt JC et al (2003) Best practices for convolutional neural networks applied to visual document analysis. In: ICDAR, vol 3

Simonyan K, Zisserman A (2014) Very deep convolutional networks for large-scale image recognition. arxiv preprint. arXiv:1409.1556

Srinivas S, Babu RV (2015) Data-free parameter pruning for deep neural networks. Comput Sci. https://doi.org/10.5244/C.29.31

Srinivas S, Sarvadevabhatla RK, Mopuri KR, Prabhu N, Kruthiventi SS, Babu RV (2016) A taxonomy of deep convolutional neural nets for computer vision. Front Robotics AI 2:36

Srinivas A, Lin T-Y, Parmar N, Shlens J, Abbeel P, Vaswani A (2021) Bottleneck transformers for visual recognition. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 16519–16529

Srivastava RK, Greff K, Schmidhuber J (2015) Training very deep networks. In: Advances in neural information processing systems

Sun W, Zhou A, Stuijk S, Wijnhoven R, Nelson AO, Corporaal H et al (2021) DominoSearch: find layer-wise fine-grained N:M sparse schemes from dense neural networks. In: Advances in neural information processing systems 34 (NeurIPS 2021)

Sundermeyer M, Schlüter R, Ney H (2012) Lstm neural networks for language modeling. In: 13th annual conference of the international speech communication association

Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A (2015) Going deeper with convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 1–9

Szegedy C, Ioffe S, Vanhoucke V, Alemi AA (2017) Inception-v4, inception-resnet and the impact of residual connections on learning. In: 31st AAAI conference on artificial intelligence

Takahashi N, Mitsufuji Y (2021) Densely connected multi-dilated convolutional networks for dense prediction tasks. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 993–1002

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, pp 2820–2828

Tang W, Hua G, Wang L (2017) How to train a compact binary neural network with high accuracy? In: 31st AAAI conference on artificial intelligence

Tang H, Liu X, Sun S, Yan X, Xie X (2021a) Recurrent mask refinement for few-shot medical image segmentation. In: Proceedings of the IEEE/CVF international conference on computer vision, pp 3918–3928

Tang Y, Wang Y, Xu Y, Deng Y, Xu C, Tao D, Xu C (2021a) Manifold regularized dynamic network pruning. In: Proceedings of the IEEE/CVF international conference on computer vision and pattern recognition, pp 5018–5028

Theis L, Korshunova I, Tejani A, Huszár F (2018) Faster gaze prediction with dense networks and fisher pruning. arXiv preprint. arXiv:1801.05787

Touvron H, Cord M, Douze M, Massa F, Sablayrolles A, Jégou H (2021) Training data-efficient image transformers and distillation through attention. In: International conference on machine learning (PMLR), pp 10347–10357

Uijlings JR, Van De Sande KE, Gevers T, Smeulders AW (2013) Selective search for object recognition. Int J Comput Vis 104(2):154–171

Vanhoucke V, Senior A, Mao MZ (2011) Improving the speed of neural networks on CPUS. In: Deep learning and unsupervised feature learning workshop

Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł, Polosukhin I (2017) Attention is all you need. In: Advances in neural information processing systems, pp 5998–6008

Wang P, Cheng J (2016) Accelerating convolutional neural networks for mobile applications. In: Proceedings of the 24th ACM international conference on Multimedia, pp 541–545

Wang P, Cheng J (2017) Fixed-point factorized networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 4012–4020

Wang P, Hu Q, Zhang Y, Zhang C, Liu Y, Cheng J (2018) Two-step quantization for low-bit neural networks. In: Proceedings of the IEEE Conference on computer vision and pattern recognition, pp 4376–4384

Wang Z, Lin S, Xie J, Lin Y (2019a) Pruning blocks for cnn compression and acceleration via online ensemble distillation. IEEE Access 7:175703–175716

Wang W, Fu C, Guo J, Cai D, He X (2019b) COP: customized deep model compression via regularized correlation-based filter-level pruning. Neurocomputing 464:533–545

Wang Z, Lu J, Tao C, Zhou J, Tian Q (2019c) Learning channel-wise interactions for binary convolutional neural networks. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 568–577

Wang C-Y, Liao H-YM, Wu Y-H, Chen P-Y, Hsieh J-W, Yeh I-H (2020a) Cspnet: a new backbone that can enhance learning capability of CNN. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops, pp 390–391

Wang N, Zhou W, Li H (2020b) Contrastive transformation for self-supervised correspondence learning. arXiv preprint. arXiv:2012.05057

Wang D, Li M, Gong C, Chandra V (2021a) Attentivenas: improving neural architecture search via attentive sampling. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 6418–6427

Wang Y, Xu Z, Wang X, Shen C, Cheng B, Shen H, Xia H (2021b) End-to-end video instance segmentation with transformers. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 8741–8750

Wang D, Zhao G, Chen H, Liu Z, Deng L, Li G (2021c) Nonlinear tensor train format for deep neural network compression. Neural Netw 144:320–333

Wang C, Liu B, Liu L, Zhu Y, Hou J, Liu P, Li X (2021d) A review of deep learning used in the hyperspectral image analysis for agriculture. Artif Intell Rev 54:5205–5253

Wang Z, Xiao H, Lu J, Zhou J (2021e) Generalizable mixed-precision quantization via attribution rank preservation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp 5291–5300

Wang W, Xie E, Li X, Fan D-P, Song K, Liang D, Lu T, Luo P, Shao L (2021f) Pyramid vision transformer: a versatile backbone for dense prediction without convolutions. arXiv preprint. arXiv:2102.12122

Wen W, Wu C, Wang Y, Chen Y, Li H (2016) Learning structured sparsity in deep neural networks. Adv Neural Inf Process Syst 29:2074–2082

Wen W, Xu C, Yan F, Wu C, Wang Y, Chen Y, Li H (2017) Terngrad: Ternary gradients to reduce communication in distributed deep learning. arXiv preprint. arXiv:1705.07878

Wu J, Leng C, Wang Y, Hu Q, Cheng J (2016) Quantized convolutional neural networks for mobile devices. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 4820–4828

Wu Y, Wu Y, Gong R, Lv Y, Chen K, Liang D, Hu X, Liu X, Yan J (2020a) Rotation consistent margin loss for efficient low-bit face recognition. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 6866–6876

Wu B, Xu C, Dai X, Wan A, Zhang P, Yan Z, Tomizuka M, Gonzalez J, Keutzer K, Vajda P (2020b) Visual transformers: token-based image representation and processing for computer vision. arXiv preprint. arXiv:2006.03677

Xie L, Yuille A (2017) Genetic CNN. In: Proceedings of the IEEE international conference on computer vision, pp 1379–1388

Xie S, Girshick R, Dollár P, Tu Z, He K (2017) Aggregated residual transformations for deep neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 1492–1500

Xu H (2020) Pnfm: a filter level pruning method for cnn compression. In: Proceedings of the 3rd international conference on information technologies and electrical engineering, pp 49–54

Xu B, Wang N, Chen T, Li M (2015) Empirical evaluation of rectified activations in convolutional network. arXiv preprint. arXiv:1505.00853

Xu Y, Xie L, Zhang X, Chen X, Qi G-J, Tian Q, Xiong H (2019) PC-DARTS: partial channel connections for memory-efficient architecture search. arXiv preprint. arXiv:1907.05737

Xu Y, Wang Y, Han K, Tang Y, Jui S, Xu C, Xu C (2021) Renas: relativistic evaluation of neural architecture search. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 4411–4420

Yamamoto K (2021) Learnable companding quantization for accurate low-bit neural networks. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 5029–5038

Yang Z, Wang Y, Chen X, Shi B, Xu C, Xu C, Tian Q, Xu C (2020a) CARS: continuous evolution for efficient neural architecture search. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 1829–1838

Yang Z, Wang Y, Han K, Xu C, Xu C, Tao D, Xu C (2020b) Searching for low-bit weights in quantized neural networks. arXiv preprint. arXiv:2009.08695

Yang Z, Wang Y, Chen X, Guo J, Zhang W, Xu C, Xu C, Tao D, Xu C (2021) Hournas: extremely fast neural architecture search through an hourglass lens. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 10896–10906

Yao L, Pi R, Xu H, Zhang W, Li Z, Zhang T (2021) G-DetKD: towards general distillation framework for object detectors via contrastive and semantic-guided feature imitation. In: Proceedings of the IEEE/CVF international conference on computer vision, pp 3591–3600

Ye J, Li G, Chen D, Yang H, Zhe S, Xu Z (2020) Block-term tensor neural networks. Neural Netw 130:11–21

Ye M, Kanski M, Yang D, Chang Q, Yan Z, Huang Q, Axel L, Metaxas D (2021) DeepTag: an unsupervised deep learning method for motion tracking on cardiac tagging magnetic resonance images. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 7261–7271

Yuan L, Wang T, Zhang X, Tay FE, Jie Z, Liu W, Feng J (2020) Central similarity quantization for efficient image and video retrieval. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 3083–3092

Yuan L, Chen Y, Wang T, Yu W, Shi Y, Jiang Z, Tay FE, Feng J, Yan S (2021) Tokens-to-token vit: training vision transformers from scratch on imagenet. arXiv preprint. arXiv:2101.11986

Zagoruyko S, Komodakis N (2016a) Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. arXiv preprint. arXiv:1612.03928

Zagoruyko S, Komodakis N (2016b) Wide residual networks, British Machine Vision Conference

Zeiler MD, Fergus R (2014) Visualizing and understanding convolutional networks. In: European conference on computer vision. Springer, Cham, pp 818–833

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

Zhang Q, Zhang M, Chen T, Sun Z, Ma Y, Yu B (2019) Recent advances in convolutional neural network acceleration. Neurocomputing 323:37–51

Zhang T, Cheng H-P, Li Z, Yan F, Huang C, Li H, Chen Y (2020) Autoshrink: a topology-aware NAS for discovering efficient neural architecture. In: Proceedings of the AAAI conference on artificial intelligence, vol 34, pp 6829–6836

Zhang Z, Lu X, Cao G, Yang Y, Jiao L, Liu F (2021a) Vit-yolo: Transformer-based yolo for object detection. In: Proceedings of the IEEE/CVF international conference on computer vision, pp 2799–2808

Zhang C, Yuan G, Niu W, Tian J, Jin S, Zhuang D, Jiang Z, Wang Y, Ren B, Song SL et al (2021b) Clicktrain: efficient and accurate end-to-end deep learning training via fine-grained architecture-preserving pruning. In: Proceedings of the ACM international conference on supercomputing, pp 266–278

Zhao Q, Sheng T, Wang Y, Tang Z, Chen Y, Cai L, Ling H (2019) M2det: a single-shot object detector based on multi-level feature pyramid network. In: Proceedings of the AAAI conference on artificial intelligence, vol 33, pp 9259–9266

Zhao T, Cao K, Yao J, Nogues I, Lu L, Huang L, Xiao J, Yin Z, Zhang L (2021a) 3D graph anatomy geometry-integrated network for pancreatic mass segmentation, diagnosis, and quantitative patient management. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 13743–13752

Zhao H, Zhou W, Chen D, Wei T, Zhang W, Yu N (2021b) Multi-attentional deepfake detection. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 2185–2194

Zhong Z, Yang Z, Deng B, Yan J, Wu W, Shao J, Liu C-L (2020) Blockqnn: efficient block-wise neural network architecture generation. IEEE Trans Pattern Anal Mach Intell 43(7):2314–2328

Zhou H, Alvarez JM, Porikli F (2016) Less is more: towards compact CNNs. In: European conference on computer vision. Springer, Berlin, pp 662–677

Zhou M, Liu Y, Long Z, Chen L, Zhu C (2019) Tensor rank learning in CP decomposition via convolutional neural network. Signal Process Image Commun 73:12–21

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, pp 10387–10396

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:2004.10934

Zhu C, Han S, Mao H, Dally WJ (2016) Trained ternary quantization. arXiv preprint. arXiv:1612.01064

Zhu X, Lyu S, Wang X, Zhao Q (2021a) TPH-YOLOv5: improved yolov5 based on transformer prediction head for object detection on drone-captured scenarios. In: Proceedings of the IEEE/CVF international conference on computer vision, pp 2778–2788

Zhu J, Tang S, Chen D, Yu S, Liu Y, Rong M, Yang A, Wang X (2021b) Complementary relation contrastive distillation. In: Proceedings of the IEEE/CVF international conference on computer visionand pattern recognition, pp 9260–9269

Zoph B, Le QV (2016) Neural architecture search with reinforcement learning. arXiv preprint. arXiv:1611.01578

Zoph B, Vasudevan V, Shlens J, Le QV (2018) Learning transferable architectures for scalable image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 8697–8710