



Review of Lightweight Deep Convolutional Neural Networks

Fanghui Chen¹ · Shouliang Li¹ · Jiale Han¹ · Fengyuan Ren¹ · Zhen Yang¹

Received: 25 August 2023 / Accepted: 30 October 2023 / Published online: 28 November 2023
© The Author(s) under exclusive licence to International Center for Numerical Methods in Engineering (CIMNE) 2023

Abstract

Lightweight deep convolutional neural networks (LDCNNs) are vital components of mobile intelligence, particularly in mobile vision. Although various heavy networks with increasingly deeper and wider have continuously broken accuracy records since 2012, with the spring of terminals and mobile devices, neural networks that can match them have become a core role in practical applications. In this review, we focus on several representative lightweight Deep Convolutional Neural Networks (DCNN) technologies that hold significant potential for advancing the field. More than 190 references screened out in terms of architecture design and model compression, in which over 50 representative ones are emphasized from the perspectives of methods, performance, advantages, and drawbacks, as well as underlying framework support and benchmark datasets. With a comprehensive analysis, we put forward some existing problems and offer prospects of lightweight DCNN for future development.

1 Introduction

DCNNs are preferred in mobile intelligence more than ever. Since 2012, deep learning techniques have prompted CNNs to flourish as the mainstream status of the computer vision field [1]. The powerful local modeling ability of deep convolution neural networks endows it dazzling in computer vision tasks such as image classification, object detection, segmentation, recognition, etc. As we witnessed, to pursue better performance, the shape of the deep networks becomes increasingly deeper and wider. Thus, the data-driven DNN has massive parameters to ensure performance on highly parallel hardware devices, which requires an awful amount of hardware resources to train the parameter deluge. Inevitably, most deep CNN models are large-scale and computation-intensive. Specifically, AlexNet [1] consumes more than 200MB of memory, VGGNet [2] takes up more than 500MB, and ResNet50 [3] is about 95MB. Due to the high resource demands, most models with excellent performance

suffer from many limitations in real-world scenarios, especially for edge intelligence (EI) with widespread application demands.

At present, deep learning primarily adopts the cloud-end paradigm in practical applications. This approach entails exchanging information between cloud computing servers and mobile devices, employing deep learning algorithms to address real-world issues. In this process, the edge terminal devices send requests to the cloud computing center through the network, and the computing center then returns the processed results to each corresponding terminal device. However, the paradigm heavily relies on network coverage and stability, making it time-consuming, tedious, error-prone, and pose potential security risks. Edge terminal scenarios, such as smartphones, autopilot systems, and drones, have a superior demand in real-time and security performance for visual applications. Traditional cloud-based models may not meet the performance demands of these edge terminal scenarios. In fact, there are many redundancy connections in different layers of a DNN. Lightweight CNNs are potential candidates for such edge scenarios to solve vision tasks, but pruning a large-scale network to fit a resource-constrained terminal device is full of challenges. The lightweight technology aims to explore and eliminate those idle neurons without significantly decreasing the performance. It generally refers to lightweight model design or model compression. Indeed, some optimization is selective in specific mobile application scenarios as essential, such as machine

✉ Fengyuan Ren
rfy@lzu.edu.cn

✉ Zhen Yang
zhenyang@lzu.edu.cn

¹ School of Information Science and Engineering, Lanzhou University, Lanzhou 730000, China

learning libraries selection, hardware platforms deployment, etc. In recent years, more and more industries and computer vision communities have ventured into this area. This wave of interest has spurred a flood of impactful studies and breakthroughs.

Table 1 lists the surveys [4–19] that refer to light-weight strategy within the past 5 years. The works [4–11] concentrate on light-weight methods from the viewpoint of structural models and compression technologies. Others [12–19] deem lightweight technology an essential part of EI. [4] reviews those lightweight networks maturely applied in object detection, a branch of computer vision tasks. [5, 7, 11] only present some classical lightweight models while lacking a comprehensive overview of lightweight techniques as well as recent improvements. [6] just pecked at light-weight technologies in terms of artificial design, model compression, and architecture search. [8] mainly reviewed those convolution variants with high computational efficiency, and did not cover other promising lightweight within the scope of CNNs. [9, 10] emphasized model compression while neglecting its peer technologies like model design. And in [12–19] focuses on the techniques of compacting and accelerating DNN models. However, it does not cover the underlying support framework and lacks the latest technology due

to obsolescence. [13, 14, 16] mainly emphasized EI and the relationship between edge computing and intelligent applications. [15, 17–19] start from the perspective of deep learning, which encompasses a broad range of technologies and applications to provide specific guidance.

Although there are many works that try to elaborate the lightweight paradigm, they either don't cover comprehensive key technologies or fully consider the characteristics within lightweight CNN architectures. Additionally, recent improvements and trends for future directions are vague in those works. Therefore, this paper aims to bridge the gaps by comprehensively analyzing the state-of-the-art techniques adopted in lightweight DCNNs, incorporating underlying supports. To establish a more complete and up-to-date resource regarding this pivotal topic for researchers and practitioners alike, we carefully retrieve articles that are technically representative for summarizing. We will elaborate on the evolution of lightweight DCNNs. It has been driven by the deployment to edge terminals. The concept of 'lightweight' is evaluated from aspects such as the number of parameters, computation complexity, memory consumption, etc. All involved literatures are categorized into two classes respectively on algorithms and libraries—one focusing on algorithms designed specifically for DCNNs

Table 1 Relative surveys and their focus

References	Published year	Focus
[4]	2018	Focus on lightweight networks in object detection
[5]	2020	Introduced and discussed six early lightweight CNNs
[6]	2020	A detailed introduction to constructing lightweight neural networks by three methods: artificial design, model compression, and automatic architecture search
[7]	2022	Comparison on mainstream lightweight neural network models
[8]	2022	Focus on the convolution part of CNNs and elaborate on the main design idea by analyzing various compositions of the convolutional part of the model
[9]	2022	Focus on the model compression methods: pruning, quantization, knowledge distillation, and neural architecture search
[10]	2023	Focus on the mechanism of DNN compression technology and divide existing methods into pruning, sparse representation, bits precision, knowledge distillation, and miscellaneous based on this
[11]	2023	Review a few prevalent lightweight CNNs along their architecture, design features, performance metrics, advantages, etc
[12]	2018	Compression and acceleration technology
[13]	2019	Focus on reviewing and discussing the development, motivation, architecture, and theories of EI
[14]	2020	To distinguish EI into artificial intelligence (AI) for edge and AI on edge
[15]	2020	Reviewing the methods of model compression and their principles. Explaining how to use associate methods in accelerator design
[16]	2021	Focus on how to realize EI in a systematic way, including edge caching, edge training, edge inference, edge off-loading
[17]	2022	Focus on DL optimization for mobile applications and devices. Summarize DL-based approaches in different mobile applications
[18]	2022	Enablers of efficient model compression approaches, and the AutoML framework for these methods, relative applications and task-specific accelerations
[19]	2022	Focus on lightweight DL models, different compression techniques, application areas, algorithm-hardware code-sign approaches, available hardware and software tools, and use cases of EI

on resource-constrained edge devices, the other on libraries optimized for hardware components of DCNN inference pipelines in such environments. Thereinto, in the process of pursuing light-weight, the relative software algorithms are further divided into two categories: the design from scratch, and the compression method widely used in large-scale DCNNs. Underlying difficulties, limitations, merits, and disadvantages are discussed in applying these algorithms. Based on the review and analysis, some potential and promising directions associated with lightweight DCNNs are proposed.

Figure 1 describes the overall structure of this survey, which is organized as follows: 2 introduces the motivation of lightweight CNNs, as well as the taxonomic perspective. The representative technical works of lightweight networks are reviewed in 3. Underlying frameworks support and common benchmark datasets are depicted in 4 and 5, respectively. The perspective of lightweight techniques is given in 6. Finally, conclusions are drawn in 7.

2 Motivation and Category

This section first discusses the motivation for utilizing lightweight CNNs and then classifies the corresponding lightweight methods.

2.1 The Motivation of Utilizing Lightweight CNNs

For a long time, artificial neural networks (ANNs) have been seriously hampered by insufficient samples of the data set and low hardware performance. Up until 2009, the ImageNet database was released [20] and there was also sufficient computing power of the hardware, the ImageNet large-scale visual recognition challenge (ILSVRC) [21] was

held spanning the period from 2010 to 2017. The AlexNet [1] in 2012 was a milestone that promoted the great success of deep NNs in the field of image recognition. During the period of ILSVRC, DNNs are capable of identifying objects accurately by executing large-scale computing, the cost is “deep”, i.e., high computational complexity and high memory consumption. However, those large-scale models are not friendly for edge computing which is the mainstream of future general intelligence. Accuracy and real-time property are two main requirements for edge intelligence, which requires both intelligent algorithms and hardware to work in concert. Conventional large-scale intelligent algorithms with high accuracy are hardly deployed on edge devices, while those tiny algorithms are under-powered for the accuracy demands. Moreover, studies have shown that there is a certain degree of redundancies in deep models, whether for edge devices or cloud data centers, the extra costs are unnecessary. And finally, with the failure of Moore’s Law, it is becoming more and more difficult to elevate hardware performance. Neural networks’ lightweight design is a promising solution for these issues. The lightweight network model significantly reduces the number of parameters and computational complexity. Certainly, to achieve fair performance in practical applications requires corresponding support from the underlying libraries and hardware. Currently, the handwritten digit recognition network proposed by LeCun et al. [22] is a paradigm used for exploring technology.

2.2 Category of Lightweight DCNNs Methods

As shown in Fig. 2, the CNNs extract characteristics of the inputs via convolution and pooling operations and put them forward to fully connected (FC) layers to yield outputs. The loss function is then used as an optimization criterion to update the weights of each layer, aiming to minimize the loss

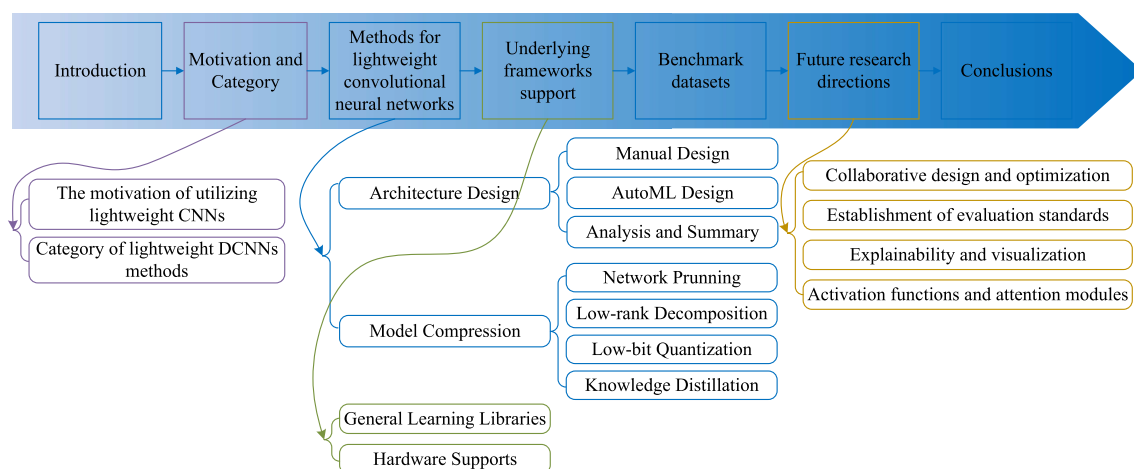


Fig. 1 Overview of this survey

Fig. 2 CNNs propagation pipeline CNNs training (also known as learning) is a process of both forward and backward propagation iteratively. The inference is a forward propagation to calculate the output for unseen data with the learned parameters during training. Here, y_i denotes activation output of every layer and serves as the input to the next layer, which is equivalent to x_i . ω is weight and b is bias. L is loss function, \bar{y} is output

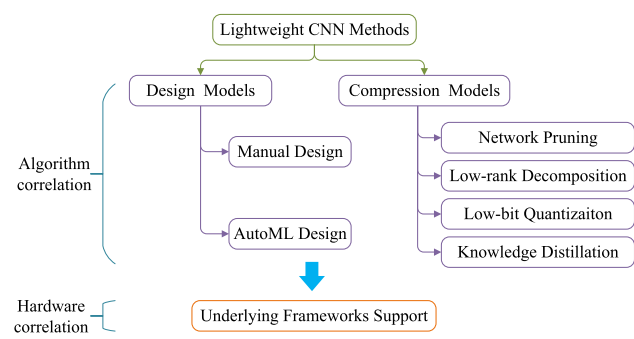
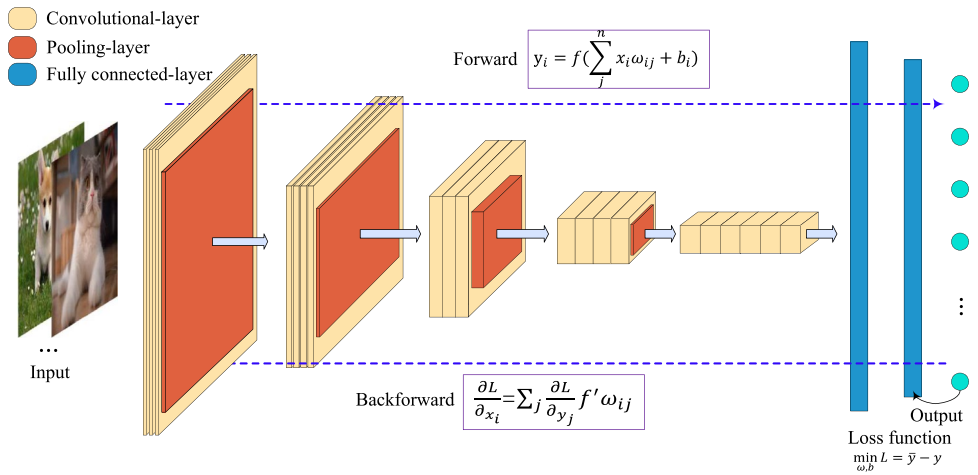


Fig. 3 Taxonomy of lightweight CNNs methods

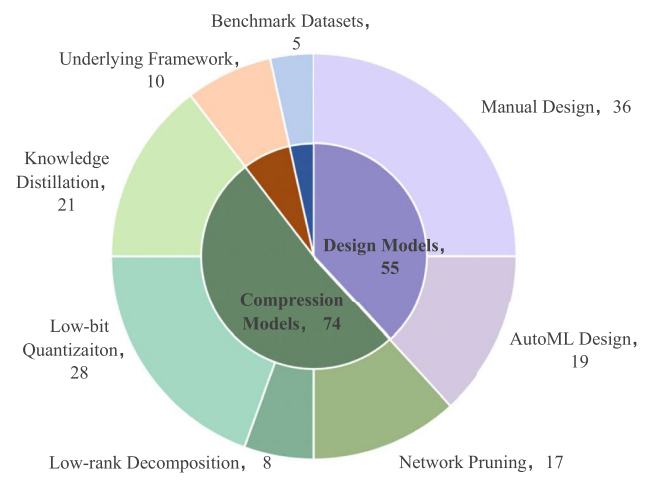


Fig. 4 The item amount statistic within each lightweight strategy and relative supports

and match the expected output. With the learned weights, CNNs are capable of inferring those unencountered tasks.

Large-scale DCNNs may benefit particular visual tasks but certainly require vast hardware resources, and time consuming. In addition, relevant research has shown that there is a lot of redundancy when modeling a large amount of data using deep neural networks [23]. The computational complexity mainly originates from the convolution operation, the number of parameters mainly determined by the full connection layer [24]. To slim the network and reduce its computational complexity, the model itself and the underlying framework need to be optimized to fit particular hardware. Therefore, we classify the lightweight network technology into two categories refer to model correlation and hardware correlation as shown in Fig. 3, where the model correlation involves design and compression. The former mainly includes manual design and automatic model search to obtain an initial lightweight neural network. The latter is tailoring a bulky neural network to a lightweight one, which mainly includes four ways: model pruning, low-rank decomposition, weight quantization, and knowledge distillation. Hardware correlation mainly refers to the accelerated optimization of

convolution operation and the underlying framework-level support when lightweight CNN models are deployed on mobile or embedded devices, such as TensorFlow Lite and TensorRT, as well as related technologies that guide hardware design. The item amount statistics of references for each subcategory are shown in Fig. 4.

3 Methods for Lightweight Convolutional Neural Networks

3.1 Architecture Design

Exploring the sparse hierarchical structure of CNNs without significantly reducing the accuracy of the network is the purpose of designing lightweight CNNs. Two ways towards this destination.

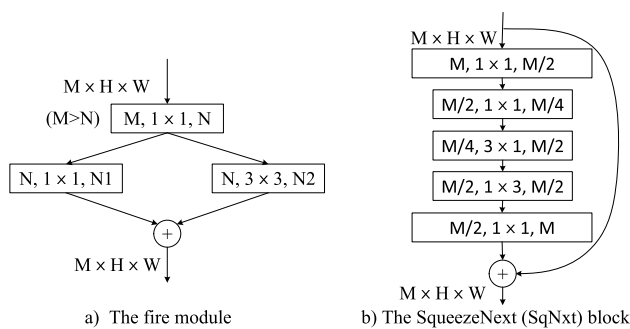


Fig. 5 The fire module and SqueezeNext block. Here, M and N represent the number of channels, while W and H represent the size of the feature map, respectively

3.1.1 Manual Design

Manual design reduces the parameters amount and computational complexity by introducing some specific convolution paradigms, such as group convolution, separable convolution, dilated convolution, etc. The Inception series started from GoogLeNet has made continuous progress in improving accuracy and decreasing the computational complexity of the network [25–28]. Especially, in [28], the decoupling of 3D convolution kernels into a separable 2D paradigm along the direction of the channel, i.e., depthwise convolution (DWConv) or single intra-channel convolution [29, 30], has had a far-reaching impact on the subsequent lightweight schemes.

Via sparse structure designing, a large number of impressive works have emerged. Forrest Iandola et al. [31] proposed the SqueezeNet, in which a sparse convolution module called Fire (Fig. 5a) is represented. The fire model consists of two stages named squeeze and expand, respectively, the former used 1×1 convolution filters [32] to reduce the dimension of characteristic channels, while the latter combined 1×1 and 3×3 convolution filters to support multiple resolutions. SqueezeNet has 50 times fewer parameters than the AlexNet but achieves a competitive level of accuracy on ImageNet as AlexNet. In the improved SqueezeNet [33], a two-stage bottleneck structure is proposed to reduce the number of channels (Fig. 5b), and the separable convolution is utilized to further reduce the parameters. The author also used hardware simulation to determine the best design of the baseline model. The MobileNets series [34, 35] are designed for mobile or embedded devices. MobileNets [34] makes full use of the depthwise separable convolution (DSCConv) which involves depthwise con-volution to filter each input channel, and a 1×1 pointwise convolution (PWConv) to combine the outputs of the depthwise convolution (Fig. 6), and controlling the network size by super parameters. MobileNetV2 [35] co-opted ResNets [3]’s bottleneck module, which combines depth separable convolution with residual connection.

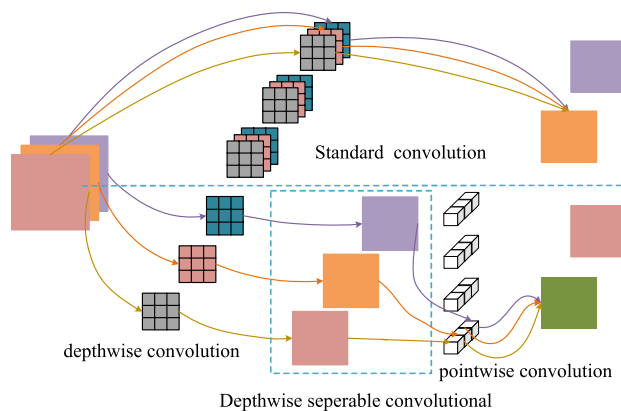


Fig. 6 Standard convolution and depthwise separable convolution

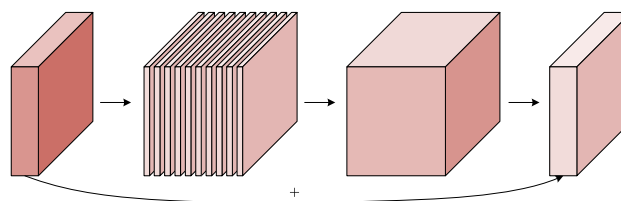


Fig. 7 The inverted residual block. This block expands a compressed input, filters it with a DWConv, and then projects the features back to a lower-dimensional representation using a linear convolution

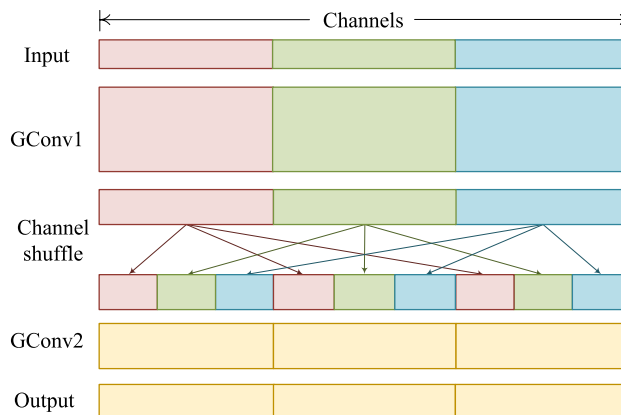


Fig. 8 Channel shuffle with two stacked group convolutions. GConv indicates group convolution. The information between the layers GConv1 and GConv2 is fully communicated through channel shuffle

The author adopted linear transformation in bottleneck to reserve complete information and shortcuts directly between the bottlenecks (Fig. 7) which enables the module to perform inference with higher memory efficiency than standard ones in various neural architectures. ShuffleNet [36], as shown in Fig. 8, is an improvement of MobileNet, and also inherits the merit of group convolutions of AlexNet to compromise representation ability and computational cost.

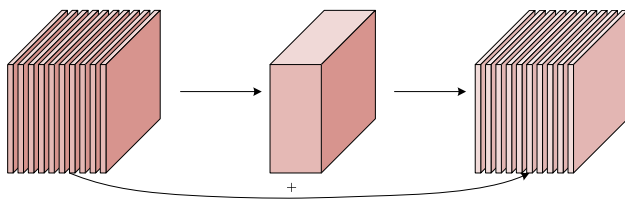


Fig. 9 The sandglass block. This block reverses the inverted residual block between bottlenecks and adds DWConvs (i.e., separated blocks) at both ends of the residual path, both of which are crucial for performance improvement

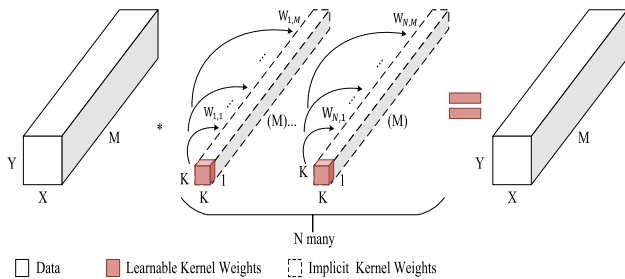


Fig. 10 The blueprint separable convolution. BSCConv exploits correlations between CNN filter kernels along the depth dimension. It represents each filter using a single 2D blueprint kernel distributed across depth via a weight vector

In addition, the shuffle operation is introduced to facilitate information flow exchange for multiple group convolution layers. In ShuffleNetV2 [37], the author claimed that the numerous 1×1 group convolutions and the shuffle operations actually increase the frequency of memory accesses. Therefore, to solve the above problem, channel splitting is employed instead of group-wise convolutions. In the work of [38], the author pointed out that the inverted residual block would induce information loss and gradient fusion. Thus, they add depthwise convolutions at the ends of the residual path (Fig. 9), which can extract richer features. [39] utilized the correlation along the depth direction in DS convolution and proposed the blueprint separable convolution (BSCConv) (Fig. 10). The BSCConv consistently verified improvement based on DSConv models without introducing any further complexity. ChannelNets [40] believe that the fully-connected pattern is the main cause of excessive computational consumption. So three channel-wise convolution operations are proposed, which significantly reduce the number of parameters and computational complexity without accuracy loss. The aforementioned strategy has also been incorporated into 3DCNNs [41] for video applications that typically require higher computational resources.

Other lightweight CNNs concentrate on specific visual tasks [42–46] or dedicated hardware applications [47, 48], such as object detection [42], segmentation [43],

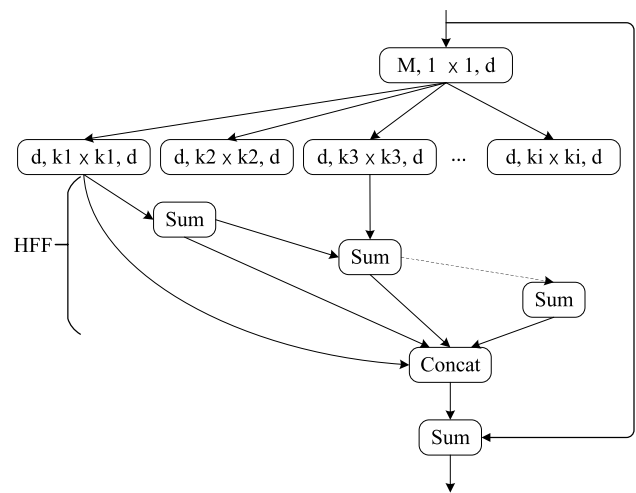


Fig. 11 The block diagram of the efficient spatial pyramid (ESP) module. It consists of a pointwise convolution followed by a spatial pyramid convolution. The former of the module reduces the computation while the latter enlarges the receptive field and removes gridding artifacts through hierarchical feature fusion (HFF)

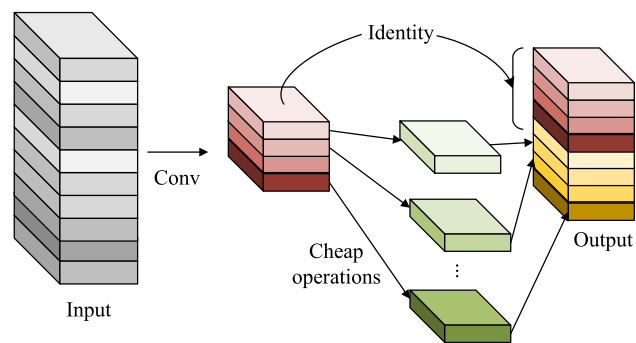


Fig. 12 The Ghost module. It consists of a lightweight “ghost” branch and a heavier “feature” branch in parallel. The former generates feature maps via a series of cheap operations while the latter generates more complex ones. The output is obtained via a concatenation operator, balancing representational power and computational efficiency

and recognition [45, 49]. In [50, 51] (Fig. 11), the dilated convolution is used to enlarge the receptive field without increasing the computational load, memory, and power, which dramatically benefits the semantic segmentation of high-resolution images. What is meaningful are the techniques that leverage visualization to provide design insights. In [52], visualization revealed redundant feature maps as important for effective CNNs. The insight inspired the GhostNet module (Fig. 12) to generate more redundant feature maps through linear transformations for revealing intrinsic information while maintaining compatibility with existing CNNs. The authors also put forward C-GhostNet and G-GhostNet respectively for GPU-like and CPU-like

devices in their subsequent works [47]. In VGNetG [53], the visual analysis enabled utilizing edge operators to substitute for learnable operations in the lower layers, resulting in a parameter-efficient CNN architecture. More recently, works like MobileOne [54] and FalconNet [55] have developed the reparameterization technique [56] into module design. It allows linear branches present during training to be reparameterized as simpler blocks for inference. Concretely, the MobileOne block (Fig. 13) introduces over-parameterized branches to enhance representation capacity during

training, which is then reparameterized into a slimmed-down form for inference, yielding improvements in both accuracy and latency.

In [55], the authors abstracted the Meta Light Block based (Fig. 14) on different lightweight modules. They introduced Reparameterized Spatial Operator (RepSO) and Reparameterized factorized Channel Operator (RefCO) methods to increase the sparsity of the spatial and channel dimensions, respectively. Both strategies leverage structural reparameterization to convert the diverse connections employed during training into equivalent inference units.

We summarize those impressive lightweight works as shown in Table 2. It is clear that besides depthwise convolution and pointwise convolution, the residual connections, rectified linear unit (ReLU) [57] or its variants (i.e. ReLU6 and PReLU) [35, 58], and linear operations are the most commonly involved operations. Furthermore, Squeeze and Excitation (SE) modules [59] (Fig. 15) are often inserted into the blocks as an attention mechanism to elevate the perception abilities of depthwise convolution. In practice, depthwise convolution has lower arithmetic intensity (Fig. 16), making it less efficient than expected [28, 36]. Other works circumvent this problem with different designs to achieve lightweight goals. The ShiftNet [60] proposes a shift operation (Fig. 17) requiring no extra floating point operations (FLOPs) and parameters, readily implemented

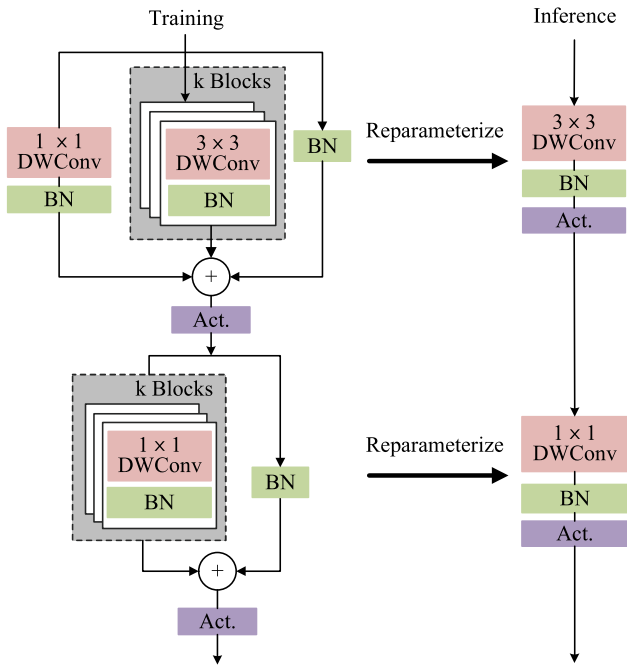


Fig. 13 The MobileOne Block. It has two different structures at train time and test time. Left: Train time MobileOne block with reparameterizable branches. Right: MobileOne block at inference where the branches are reparameterized. Either ReLU or SReLU is used as activation. The trivial over-parameterization factor k is a hyperparameter which is tuned for every variant

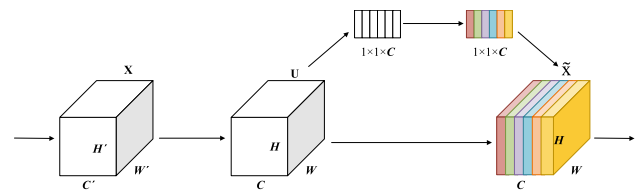
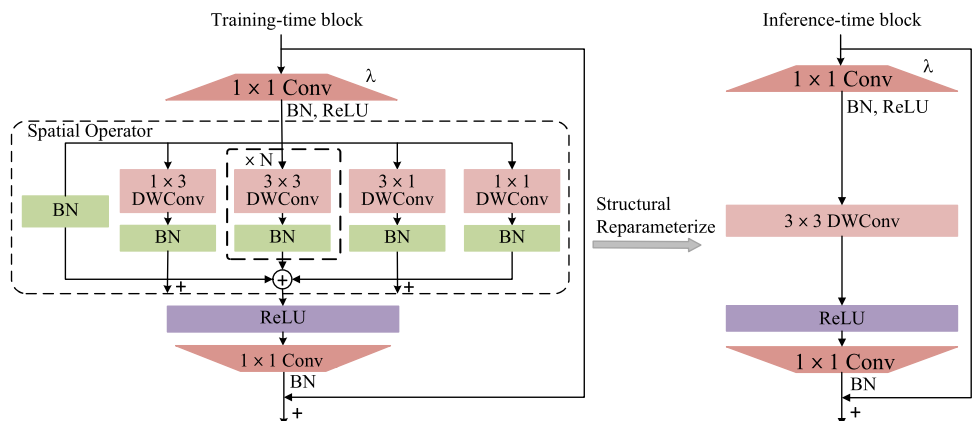


Fig. 15 Illustration of the SE module. It involves a two-step process: squeezing global information by reducing spatial dimensions, and then excitation by learning channel-wise weights to amplify important features

Fig. 14 Meta Light Block with RepSO. The block consists of two 1×1 Conv layers (with an expansion ratio λ) and a single spatial operator layer in between. Left: Train time block with reparameterizable branches. Right: Inference time block where the branches are reparameterized through RepSO



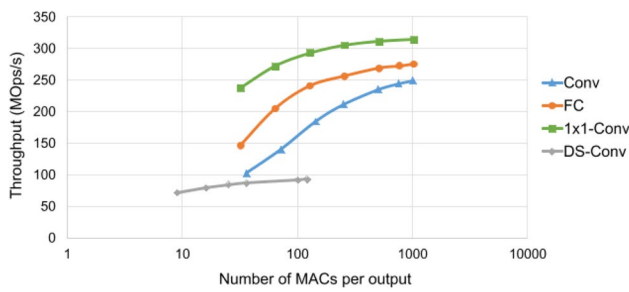


Fig. 16 Throughput variation with number of MACs per output for different types of NN operations [62]

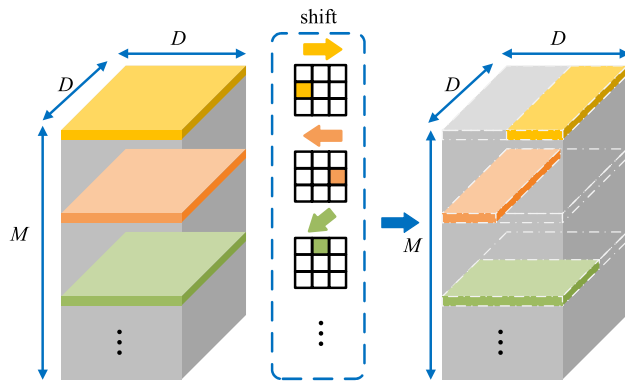


Fig. 17 Shift operation. The operation shifts the pixel values of the input feature map by a specified distance, altering their spatial positions. In the given example with a 3×3 shift matrix, the illuminated cell denotes 1 at that position and the white cells represent 0

under the current environments. The DiCE unit [61] utilizes dimension-wise convolutions and fusion, applying light-weight convolutions across input dimensions and fusing dimension-wise representations compactly (Fig. 18). PeleeNet [44] utilizes conventional convolutions to achieve efficient and real-time object detection.

Overall, the crafted design of blocks aimed to improve the overall performance of the network, and also benefits the

definition of search space for automated machine learning (AutoML) design.

3.1.2 AutoML Design

The manual design has what appears to be a drawback; it heavily relies on the knowledge and experience of specialists to make a compromise between various factors like accuracy, efficiency, computing consumption, and more. As a result, a sub-optimal scheme tends to be obtained instead of an optimal one. Automatic methods typically utilize search algorithms to create an optimal network model that minimizes the need for human labor.

The automatic search approach primarily involves three aspects: search space, search strategy, and evaluation strategy. To achieve a lightweight design for deep networks, the focus on lightweight with limited search space is crucial, and it encompasses three stages: cell-level, stage-level, and layer-level.

1. Cell-level, which explores a one-shot way of the network design. Specifically, it only searches cell-level operations and extends them to the entire network layers. Some early works [65–67] provide certifications of competitive accuracy and low FLOPs of neural network design with this strategy. However, the homogeneity of different layers significantly reduces the accuracy and increases the delay of the neural network. As ShuffleNetV2 [37] pointed out, the cell-level operation is fragmented, resulting in frequent memory access, and may not be hardware-friendly. In addition, this strategy merely uses FLOPs as an approximate metric for measuring the latency, which does not meet the requirements of real-time response.
2. Stage-level aims to obtain a hierarchical search space with different blocks distributed in different stages of a network. MnasNet [68] searched through neural structures composed of multiple stages and connected sequentially, each structure with a variable number of

Fig. 18 DiCE Unit. The unit efficiently encodes the spatial and channel-wise information in the input tensor X using dimension-wise convolutions (DimConv) and dimension-wise fusion (DimFuse) to produce an output tensor Y. In practice, these three-dimensional kernels are executed simultaneously

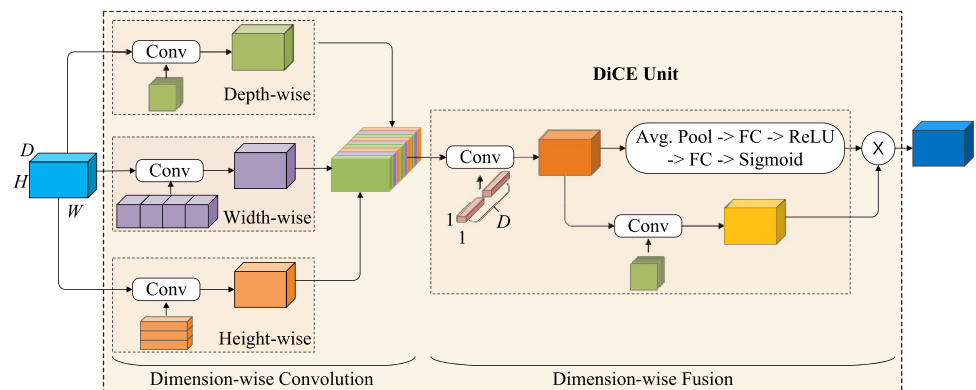


Table 2 The comparison of lightweight modular operations

Building block	Residual connection	Depthwise conv	Pointwise conv	SE module	Activation
Fire module [31]	×	×	✓	×	ReLU
SqueezeNext block [33]	✓	×	✓	×	ReLU
DSConv [28, 34]	×	✓	✓	×	ReLU
MBCConv [35, 63]	✓	✓	✓	×	ReLU6 & Linear
Fused-MBCConv [64]	✓	✓	✓	✓	ReLU6 & Linear
ShuffleNetV1 unit [36]	✓	✓	✓	✓	ReLU
ShuffleNetV2 unit [37]	✓	✓	✓	✓	ReLU & Linear
Sandglass block [38]	✓	✓	✓	✓	ReLU6 & Linear
Ghost block [47, 52]	✓	✓	✓	✓	ReLU & Linear
ESP module [50]	✓	×	✓	×	PReLU & Linear
EESP unit [51]	✓	✓	✓	×	PReLU & Linear
BSConv [39]	✓ [†]	✓	✓	✓	Linear
MobileOne block [54]	✓	✓	✓	✓	ReLU & SEReLU

The ‘activation’ column specifies the activation functions used in each corresponding module. ✓ indicates that the module adopts the operation corresponding to the column header, while × indicates the opposite

[†]The operation commonly employed within BSConv-S refer to [39]

repeated identical layers, and is optimized on the target mobile devices with the external latency considered. Subsequently, MobileNetV3 [69] adopted a similar search algorithm with MnasNet except the search algorithm is a hardware-aware neural architecture search (NAS) complemented by NetAdapt [70]. [71] is also a stage-level search work, and the GPU attribute is considered in its search algorithm. Yet blocks within a stage are identical, there is still potential room for performance elevating.

3. Compared with Stage-Level, Layer-Level takes a further hierarchical step to the search space. [72] applied modular search space to retrieve the predefined supernet middle layer, and the key is to fit the search granularity to the supernet layer level. Nine candidate modules are given in each search space to generate FBNet series networks, and the latency and accuracy are deemed as the contributions of each candidate to the search architecture. In [73], the authors thought that the search space of the aforementioned modular search algorithm is limited, thus, they proposed a memory-efficient algorithm, which can greatly expand the search space in spatial and channel dimensions within the layer.

Furthermore, there are ways to automatically search for lightweight CNN, which take into account finer-grained attributes in search space [71, 74–79]. [74] built ChamNet, considered traits of the hardware platform, and adjusted computing resources in the search algorithm to fit latency and energy constraints. The first mobile GPU-awareness (MOGA) NAS was proposed by Chu et al. [71] for mobile applications. [75, 77] also take into account the traits of

the platform in search space. The convolution kernel size dramatically affects the neural networks performance, thus, MixNets [76] proposed a mixed-depth convolution (MixConv), which mixes multiple kernel sizes into one convolution and integrates them within the AutoML search space to obtain better performance than the previous mobile lightweight models. In addition to the structured search, [78] expands the search space by introducing the previously neglected hyperparametric into it, which makes it more flexible to obtain lightweight networks. [79] believes that the task types should also be considered in the search space.

3.1.3 Analysis and Summary

We summarize the manual design and automatic design of lightweight CNNs, as shown in Table 3. To compare benchmarks, all these works are evaluated on the ImageNet and are measured by the number of parameters and FLOPs as well as mainly concerned with the accuracy of the top one. However, these models are focused on image classification on the ImageNet, and taken for granted that appropriate for other tasks. [50, 79] adopted manual and automatic search, respectively, and take the specific task categories into account simultaneously, high-level tasks such as object detection, semantic segmentation, and fine-grained face recognition, etc., those application properties are crucial for model design [42–46, 79]. Moreover, platform attribution [47, 48, 71, 72, 77] is another critical factor. Many works have reported their inference latency on hardware platforms, as shown in Table 3, column 6, but objective assessment of the performance of these models

Table 3 The comparison of lightweight models on ImageNet

Type	Model	#Parameter (M)	#FLOPs (M)	ImageNets		Inference latency (ms)	Published year	
				Top-1 Acc. (%)	Top-2 Acc. (%)			
Manual	SqueezeNet [31]	4.8	837	57.5	80.30	–	2017	
	SqueezeNext [33] (2.0-SqNxt-23v5)	3.2	708	67.5	88.2	–	2018	
	MobileNetV1 [34]	4.2	575	70.6	89.5	113 (Google Pixel 1)	2017	
	MobileNetV2 [35]	3.4	300	72.0	91.0	75 (Google Pixel 1)	2018	
	ShuffleNetV1 [36]	3.46	140	72.6	–	–	2018	
	ShuffleNetV2 [37]	2.3	146	71.8	–	–	2018	
	PeleeNet [44]	2.8	508	72.6	90.6	26.1 (iPhone 8)	2018	
	ChannelNet-v1 [40]	3.7	407	70.5	–	–	2018	
	GhostNet [47, 52]	5.2	141	73.9	94.1	–	2020	
	MobileNeXt [38]	3.4	300	74.0	–	211 (Google Pixel 4XL)	2020	
	ESPNetV1 [50]	1.46	–	–	–	–	2018	
	ESPNetV2 [51]	3.49	284	72.1	–	–	2019	
	PP-LCNet [48]	3.0	161	71.32	90.03	2.46 (Intel Xeon Gold 6148)	2021	
	VGNetG-1.0MP [80]	0.997	–	67.7	87.9	226 (RTX6000 GPU)	2022	
	DiCENet-E150-B512 [61]	5.1	297	75.7	–	–	2022	
	MobileOne-S0 [54]	2.1	275	71.4	–	0.79	2023	
	Falconnet [55]	2.39	333.1	75.7	–	–	2023	
	Auto	MnasNet-A1 [68]	3.9	312	74	91.8	76 (Google Pixel 1)	2018
		Proxylessnas [81]	4.0	320	74.6	92.2	78 (Google Pixel 1)	2019
		ChamNet-A [74]	–	–	75.4	–	29.8 (Snapdragon 835)	2018
MobileNetV3-Large 1.0 [69]		5.4	219	75.2	–	51 (Google Pixel 1)	2019	
FBNetV1-C [72]		5.5	375	74.9	–	28.1 (Snapdragon 835)	2019	
FBNetV2-L2 [73]		–	423	78.1	–	–	2020	
FBNetV3-A [78]		5.3	357	79.1	94.5	–	2020	
MixNet-S [76]		4.1	256	75.8	92.8	–	2019	
MoGA-A [71]		5.1	304	75.9	92.8	11.8 (Qualcomm 2019)	2019	

is difficult. The most popular measurement indicators of FLOPs and parameters do not entirely reflect the actual model efficiency [62]. In addition, most of the automatic methods rely on structural search with fixed hyperparameters [78], and the whole network still requires manual optimization [69].

3.2 Model Compression

Model compression is to explore the inherent over parameterized and structural redundancy of the network and remove them, thus obtaining a lightweight form. According to different processing views, the model compression is divided into network pruning, low-rank decomposition, low-bit quantization, and knowledge distillation.

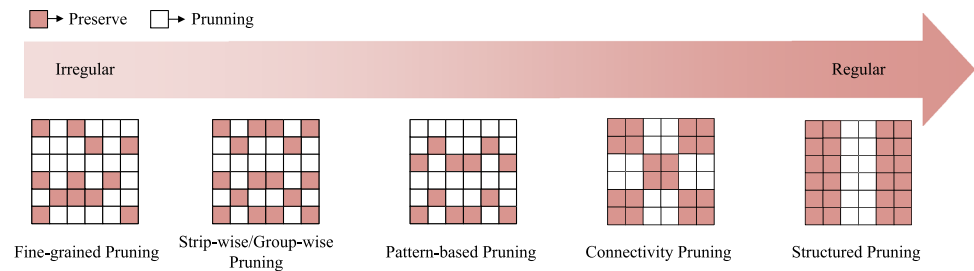
3.2.1 Network Pruning

Network pruning is to eliminate the non-critical redundancy in a pre-trained model without significant performance impairments.

1. Network pruning granularity

According to the redundant paradigm of the pre-trained models, pruning techniques can be classified as coarse-grained pruning (i.e., structured pruning), strip-wise/group-wise pruning, pattern pruning, and fine-grained pruning (i.e., unstructured pruning), as shown in Fig. 19.

- Fine-grained pruning, the pruning granularity of which is a sole neuron or connection. As the sparsity of a model

Fig. 19 Pruning granularity

- usually needs to be determined layer by layer, the practical compression is inconsistent with the theoretical one [82]. In addition, the pruning requires customized hardware to support [83].
- Strip-wise/Group-wise pruning is to prune along some vector of the input/output dimension in the parameter tensor. And the pruned results are irregular, which also requires a customized hardware support system [84, 85].
 - Pattern-based pruning is executed on one layer or the whole of the model according to a set of fixed patterns [86, 87].
 - Connectivity pruning sets the weight values of the pruned filters to 0 thus cutting off the connection between the input and output of certain channels, which enables the model to be accelerated with the existing hardware rather than the customized [88].
 - Structured pruning removes unimportant convolution kernels in a set of channels and directly changes the width of the model. This approach can be accelerated directly using Off-the-Shelf machine learning libraries [89].

2. Pruning methods

Pruning can start from both model structure and the processing of training. In terms of the structure pruning, the weight, the activation function, the gradient in backpropagation, and the batch-normalization all have the possibility to be tailored. From the perspective of model training, pruning approaches include reconstruction error training and regularity training.

Li et al. [89] adopted the L1 norm to prune the weight kernel of the current layer and then removed the features map connected to this weight filter and the corresponding weight channel of the next layer. It is usually conditional to preserve large-norm coefficient features based on the Smaller-Norm-Less Informative hypothesis [90]. He et al. [91] consider the mutual relations between filters and based on the geometric median prune the most redundant filters rather than those relatively less important ones, which still works efficiently even if the norm-based criterion fails. Luo et al. [92] used reconstruction error to minimize the output

deviations of the next layer and pruned the current layer according to the statistics information of its next layer. Dep-Graph [93] explicitly models the dependency between layers by automatically grouping tightly coupled parameters. It enables efficient generalization to diverse neural architectures without tedious individual analysis.

Most of the above works adopt structured pruning techniques and require fine-tuning methods to remedy degeneration in accuracy, the processing of which is tedious, and the generated sub-models are usually sub-optimal. Therefore, [94] adopted a continuous compression proportional control strategy for learning and designed a reinforcement learning reward mechanism according to different scenarios. [95] proposed adaptive batch normalization (BN), whose parameters are updated via several batches rather than fine-tuning. The absence of standardized benchmarks and metrics has confused researchers for a long time [96, 97]. [98] resorted to random search to optimize channel configurations when pruning. It could serve as a baseline to properly evaluate different pruning methods.

3.2.2 Low-Rank Decomposition

The tensor is the fundamental component of CNNs. Low-rank decomposition attempts to reduce the hidden redundancy in tensors, thus decreasing the complexity of convolutional/fully connected layers in CNNs, and speeding up the model inference. One of the most commonly used low-rank decompositions is singular value decomposition (SVD). Let tensor $A \in R^{m \times n}$, and re-write

$$A = USV^T, \quad (1)$$

where $U \in R^{m \times r}$ and $V^T \in R^{r \times n}$ are orthogonal to each other, $S \in R^{r \times r}$ is a diagonal matrix containing singular values of the original matrix A . If there exists $k \ll r$ can replacer, and the decomposition complexity is reduced from $O(mn)$ to $O(k(m + n + 1))$, then the original tensor can be compressed.

In [99, 100], the author decomposed the convolution kernel of $w \times h$ into the convolution kernel form of $w \times 1$ and $1 \times h$. In [101], Tucker is used to decompose the model weight, then the pruned model is deployed on the mobile phone for the experiment. Chen et al. [102] proposed a

decomposition method by using the combination of Tucker and classical prolongation (CP) and improved the efficiency of parameter utilization in the network.

The higher the tensor dimension, the better the decomposition results may be, but the space complexity increases. Therefore, [103] removed the high-order kernel tensor to avoid the aforementioned deficiency, but this method ignores the key part of convolution layer modeling and merely works effectively for the fully connected layer. In [104], the tensor-train method is extended to the convolution layers. Since the convolution structure is already a compact fourth-order tensor compared with the fully connected layer, the final compression level is very limited, and the subsequent extracted features may not be ideal. The prevalent DSConvs in lightweight CNNs make the situation more complex as their more compact tensor maps. [105] proposes an approach that integrates low-rank tensor decomposition with sparse pruning, fully leveraging both coarse and fine structures. This allows for efficient model compression of architectures that utilize DSConvs.

In fact, the low-rank decomposition is similar to designing a lightweight compact structure, the difference is that the latter uses a compact lightweight topology structure to find the basic model, and the former aims to compress a given basic neural network model. At present, the low-rank decomposition is more mature than other methods. And most works require layer-by-layer decomposition and compression. Despite [106] considering the global optimization of all layers during the model compression to avoid trivial layer-wise decomposition issues, it still requires additional retraining processing. In view of the fact that its high cost and limited compression capacity for the convolution layers, this approach does not earn much attention as imagination.

3.2.3 Low-Bit Quantization

Low-bit quantization attempts to analyze the numerical representation of the models, and then map the weights,

activations, and even gradients of the networks to a set of fixed values to compress the numerical representation of the model and improve the inference efficiency. Without significantly impair the accuracy of the network, the original 32-bit (or 16-bit) single-precision floating-point number is projected to a lower-bit representation, such as 8-bit, 4-bit, 2-bit, or even 1-bit. Low-bit representation is one of the most common model compression technologies used in industry, which can not only alleviate the amount of data transmission in hardware, but also reduce multiply-and-accumulate (MAC) operations and energy consumption.

Let x be the input, x^q is the quantization,

$$x^q = clip(R(\frac{x}{s} + Z), \alpha, \beta), \tag{2}$$

Equation 2 denotes that the real number x is projected into the integer range $[\alpha, \beta]$, where s is the quantization factor, and q is the quantized bit width.

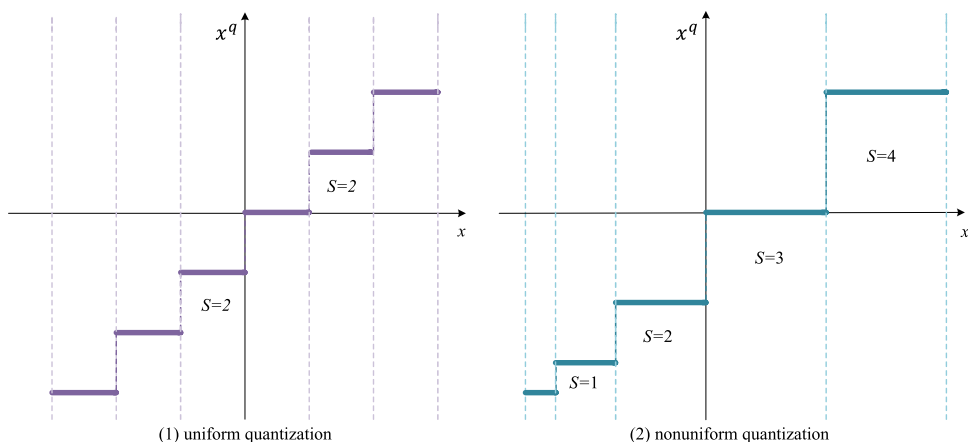
$$s = \frac{\beta - \alpha}{2^b - 1}, \tag{3}$$

Equation 2 depicts an affine quantization. When $Z=0$, this quantization is the so-called scale quantization. Through the inverse operation of Eq. 2, the quantized parameter value is reconstructed as “real value”, but usually the recovered real value is approximate to the original value rather than equal.

1. Quantization regimes

The quantization mechanism represented by R in Eq. 2 mainly includes uniform quantization and nonuniform one, as shown in Fig. 20. Among them, nonuniform quantization distributes input to different step sizes by using variable quantization factor s (Eq. 3) [107, 108]. The quantization result matches the reality better but tends to bring greater overhead costs which is unfriendly to hardware implementation. In practice, binarization quantization is also very commonly utilized. In [109], quantization is taken as an

Fig. 20 Two forms of discrete distribution for quantization. By adjusting s , the distribution of quantization levels is changed to two modes: (1) uniform quantization with the same step sizes, and (2) nonuniform quantization with different step sizes



optimization problem to minimize the difference between the input tensor and the binary expansion part. Uniform quantization is represented by linear quantization which allows the effective implementation of fixed-point operations on hardware. For uniform quantization, it is vital to choose the projecting range $[\alpha, \beta]$, which determines the setting of quantization factor s . There are symmetric quantization and asymmetric quantization, the former is easier to implement, but the imbalance of the actual range may lead to a sub-optimal solution.

2. Quantization granularity

CNN has a hierarchical structure, and different components contribute inconsistently to the model. Different levels of quantization correspond to different granularity. For layer-wise quantization, all elements in this layer share the same quantization parameters [110]. However, different consecutive layers interact with each other, so module-based quantization is proposed to alleviate this problem [111]. For channel-wise quantization, per-channel quantization [112] works on intermediate dimensions of the tensor. Further, one can explore per-row/per-column quantization, but this may lead to the huge burden of inference time, so it is more commonly adopted at the channel level [113].

3. Performance recovery methods

Compressing a huge network to a compact one usually endows degradation of accuracy, thus performance persevering strategy must be involved. The two most utilized methods include post-training quantization (PTQ) and quantization-aware-training (QAT) [114].

The PTQ method is popular among the community and industry since it can restore the performance of the model without requiring the original training pipeline. The method converts the pre-trained network into a fixed-point network and then retrains it offline to restore accuracy [111, 112, 115–119]. However, its commonly used rounding-to-nearest mechanism ignores the contribution of non-diagonal elements in the Hessian matrix, resulting in a sub-optimal solution. Therefore, an improved approximation mechanism for per-layer was proposed [115], and remarkable works have been made by limiting the weights precision of networks such as Resnet18 and Resnet50 to 4-bit, while keeping the accuracy reduction within 1%. [111] quantified the inter-layer correlation which is ignored by AdaRound [115] to restrict the weight bit width of the quantized networks to 2-bit. In [119], the author pointed out that the order of weights and activations is crucial to the network performance, they randomly drop the quantization of activations during PTQ, thus the 2-bit activation for PTQ is realized for the first time.

PTQ has many merits, such as high efficiency and speed in applications. However, it is not as effective in restoring full-precision representation as expected. QAT, on the other hand, performs better in lower bits, such as 4-bit representations, although it takes more training costs and data support. Many works published about this method [120–125]. Since QAT introduced quantization during training, the gradients update may cause accumulating quantization errors, especially in low-precision representation. Therefore, the straight-through estimator (SET) is required [123, 126]. In [121, 122], only the weights are binarized, and the gradients are still presented in full precision without using low bits, which endows much computing cost to backpropagation. In [123], both weights and activations are of low bit width and are quantified definitely, while the gradients are randomly quantized, thus, the CNNs training is accelerated as well as the inference. PACT [127] learns the clipping ranges of activations during training rather than the fixed ranges as in [123]. [128] proposed a so-called quantization interval learning (QIT) which quantizes the network weights and activations to obtain the optimal quantizer. In [129], the quantization step is deemed as a training parameter to better adapt to the quantization distribution, while its improved version [130] can learn to accommodate the negative activations with asymmetric quantization. [131] proposes the CSQ (centered symmetric quantization) quantizer for extreme low-bit quantization (≤ 3 -bit), which is trainable using QAT methods and shows that a simple change of quantization levels can result in significant performance improvement.

Despite the current quantization methods have made great progress in theory, there are still many limitations. On the one hand, most of the unified quantization methods tend to induce accuracy degradation due to the non-uniformed contributions of layers in a hierarchical network. There are mixed-precision quantization methods designed to cope with this problem, but their cost is prohibitive since it requires traversing the mixed quantization space to solve the optimization problem [132, 133]. On the other hand, most of the quantization methods are based on redundant baseline model testing and lack related tests for lightweight design models, let alone on real mobile devices. The quantization is more closely associated with the underlying hardware support, and the customized hardware may be more appropriate for it, but at a higher cost. So does the hardware-aware quantization [134]. Therefore, more corresponding baseline testing and hardware testing may benefit the design of quantization.

3.2.4 Knowledge Distillation

Knowledge distillation (KD) [135, 136] is to acquire and transfer knowledge from the original large teacher model to the lightweight student model, as shown in Fig. 21. It is considered that the “softmax” output of the teacher model

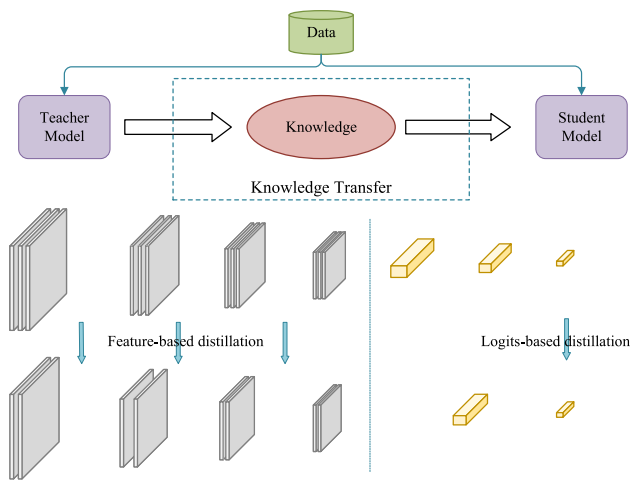


Fig. 21 Knowledge distillation framework

contains more information, which is ignored in the form of one-hot [136], i.e., the highest probability is taken as the correct output while other probabilities as wrong. However, these incorrect outputs actually contain more information which is vital for the learning process. Therefore, a “temperature” strategy is proposed to control the soft probability distribution of the outputs, as depicted in Eq. 4.

$$p_i = \frac{\exp(z_i/T)}{\sum_j \exp(z_j/T)}, \quad (4)$$

where T is the temperature, the z_i calculated for each category is converted as probability p_i by comparing the z_i with other logits. In general, $T = 1$, and when T increases, it produces a soft probability distribution in the class.

The knowledge distillation can be roughly classified as feature-based knowledge and logits-based one according to the distillation location.

- The knowledge transferred by the logits-based method was initially considered as the conditional distribution of the output for a given input sample. From this point of view, the predictions or soft targets from the pre-training teacher model play an important role in the guidance of the student model [137–144]. [139] can be viewed as a supplement to the principle of KD [136]. It decomposes the gradient caused by KD into two items: dark knowledge items and a ground-truth component, and quantifies the contribution of dark knowledge to KD. [138] transfers knowledge through collaborative learning between a group of student models rather than directly transferring with the one-way knowledge between the predefined teacher model and the student model. [141] claimed that the large model is not necessarily a good teacher model for the guidance of the

student model training. They proposed terminating the teacher model training early to alleviate the mismatch between the teacher and student models. [144] pointed out that a good teacher model can train a student model well as long as the teacher and the student always with the same input, radical data enhancement, and sufficient training times. It is found in [142] that when there is a significant capacity difference between the teacher model and the student model, the performance of the student model will be degraded greatly. To alleviate the above problems, multi-level knowledge extraction is introduced as a medium-scale network to bridge the gap between them.

- Feature-based methods may outperform logits-based at the expense of extra computation and memory consumption for distilling deep features during training [145–154]. [145] defined the transferred knowledge according to the information flow between layers, and obtained the extracted knowledge by computing the inner product of features between two layers. [146] added a feature attention mechanism into the network, which helps the student model learn better from the teacher model. [147] proposed to align the distribution i.e., the matching distribution of neuron selectivity patterns between teacher and student models, to improve the performance of the student network. Heo et al. [148] found that the transferring of activation boundaries can greatly improve the transferred efficiency and proposed a knowledge transfer method by transferring the activation boundaries of hidden neurons. [149] put forward a factors-based knowledge transfer method, which can inherit paraphrased information from the teachers’ network. [154] pointed out that students should maintain similarity in pairs in their representation space, rather than imitate the teacher’s representation space. [153] has developed contrastive learning which enables the teacher and student to project the same input to adjacent representations, and different input to apart representations.

It is noted that feature-based distillation requires extra super-parameters adjustment to lever the effects of losses in different layers. Although there is a supervised approach to explore the feature representation of the middle layer in a teacher model, it is unclear which layers and how the layers impact the student model. Compared with feature-based distillation, the logits-based method has a lower cost of training, but its performance compares unfavorably with the former. It is generally agreed that larger models may not necessarily make better teacher models, feature-based distillation is better than soft-label distillation, and the deep-level students model outperforms shallow-level students. Interestingly, soft label distillation in [155] with higher-level semantic features is better than feature-based distillation.

The knowledge distillation field is still in its infancy, requiring further establishment of theories and experiments.

4 Underlying Frameworks Support

The approaches reviewed previously are top-down solutions to develop lightweight network models. However, practical applications require support from the underlying framework to truly obtain improvements in accuracy, speed, and energy efficiency. This underlying framework can be categorized into general learning libraries and hardware-based support.

- General learning libraries [167] can be used for CNNs training and inference, such as in Caffe [163], MXNet [164], Tensorflow [156], TensorRT [166], PyTorch [157], etc., as shown in Table 4. Most of these learning libraries are open source and improved continuously. Among them, PyTorch and Tensorflow are two representative frameworks with strong community and excellent documentation support, and they are also the two most popular frameworks in academic and industrial circles, both have visual tools (Visdom, Tensorboard) support to facilitate the development. They are popular in model training via GPU parallel computing speedup but are not widely available for the inference of edge devices. The libraries' optimization mainly focuses on pipelining, resource management, as well as efficient compiler design [168]. TensorFlow Lite (TF-Lite) and TensorFlow Micro are two extensions of Tensorflow for edge inference.
- To accelerate the processing of deep network models in specific applications, numerous specialized hardware platforms have been developed. As discussed earlier, the reduction in the number of network parameters and multiply-and-accumulate (MAC) may not induce performance improvement or energy consumption reduction

as expected. The core of hardware development is in terms of appropriate throughput and energy efficiency. The MAC operations are easily implemented in parallel and optimized in both spatial and temporal architectures [169]. Temporal architecture is usually implemented in CPUs or GPUs and enhances parallelism via SIMD or SIMT to increase throughput [169, 170]. Representative embedded GPUs involve NVIDIA Jetson TX2 [171], and Intel Edison Kit [172]. In addition, data transmission contributes much to the energy consumption. Spatial architectures are usually designed and implemented based on specific integrated circuits (ASIC) and field programmable gate arrays (FPGA). These designs tend to enhance data reuse and reduce data transmission as much as possible to ensure energy efficiency. Typical works on ASIC include EIE [83] and Eyeriss [173, 174]. EIE is specially designed for pruned lightweight networks. In Eyeriss [173], the authors adopted two methods to improve energy efficiency. One is to reduce transmission through data reuse, and the other is to avoid unnecessary reading and calculation via data statistics [174]. The main difference between them is in structure, while the latter performs better in lightweight applications. More related works refer to FPGAs and ASICs are detailed in [16, 168, 169, 175].

5 Benchmark Datasets

The dataset is the driver for a neural network as well as the requisite of performance verification for it.

CIFAR [176] is a commonly used image classification dataset, with an image pixel of 32×32 . The training and verification set consists of 50 k and 10 k images. CIFAR-10 is composed of 10 mutually exclusive classes, and CIFAR-100 is composed of 100 mutually exclusive classes.

Table 4 Comparison of some learning libraries

Library name	Developers	Languages	Characteristics
TensorFlow [156]	Google's AI	Python	Phe function of data integration
PyTorch [157]	Facebook	Python	Easy-to-use, flexible
Keras [158]	Google	Python	Easy to integrate, efficient, portable
PyTorch Lightning [159]	Facebook	Python	Easy-to-use, reproducibility, readability
Theano [160]	The University of Montreal	Python, C++	Speed and stable optimization
CNTK [161]	Microsoft	C++	Easily realize and combine popular model types
Deeplearning4j [162]	SkyMind	Java, Scala	General-purpose, easy-to-use, flexible
Caffe [163]	The Berkeley Vision, Learning Center (BVLC), community	C++	Lightweight, modular, scalable
MXNet [164, 165]	Apache	C++	Flexible and efficient
TensorRT [166]	NVIDIA	C++, Python	Flexible and efficient, update quickly

ImageNet [20] contains large-scale colored images with pixels of 256×256 , which is composed of 1000 no-overlap categories. There are 1.3 M training images, 100 thousand testing images (100 per class), and 50,000 validation images (50 per class) in ImageNet.

Stanford Dogs dataset [177] contains 20,580 images, which is built for the task of fine-grained image categorization, and composed of 120 breeds of dogs from around the world. The training set contains 14,580 images and the validation contains 6000 images.

There are other public datasets for different types of tasks (e.g., object detection [178] and semantic segmentation [179]). However, current public data sets can not satisfy specific application requirements, more private datasets are available as semi-opened or unopened in academics, companies, or individuals.

6 Future Research Directions

Based on the above analysis, current lightweight technologies have limitations for widespread applications to some extent. Future research should focus on the following promising directions:

- Collaborative design and optimization. As described earlier, lightweight techniques require top-to-bottom cooperation, which requires the involvement of the specific task to explore lightweight ways from all possible perspectives [180]. For example, when designing a lightweight network model, the task complexity and the platform attributes should be considered for tuning. Various compression methods are orthogonal to model design methods [181, 182], i.e., they can cooperate during the design. The progress in lightweight network algorithms also benefits the development of the underlying learning library and hardware. It is worth further focusing on developing the combination [54, 55] of emerging technologies like structural reparameterization with existing lightweight technologies.
- Establishment of evaluation standards. On the one hand, to objectively depict the properties of a lightweight CNN model, the following metrics should be emphasized: the accuracy of the model based on a public dataset, the number of parameters and FLOPs of the model as well as the architecture-related parameters. Moreover, the deployment or development of a lightweight network model should pay attention to the hardware metrics [62], such as latency, power consumption, etc. On the other hand, the scarcity of high-quality datasets is another important hindrance to lightweight network development. In current circumstances, one feasible method is to gradually establish the standard

datasets for specific applications, the other is to adopt alternative strategies to remedy the lack of datasets, such as transfer learning [183], active learning [184], and incremental learning, etc.

- Explainability and visualization. Model visualization technology has emerged recently, which has significant potential to help the researcher understand and improve the models, such as in the design of GhostNet [47, 52] and VGNet [80]. In the future, more fine-grained downstream vision tasks still require fully mining and exploiting the redundancy of the feature maps via visualization technology.
- Activation functions and attention modules. Currently, ReLU is favored by many researchers, and future works on activation function may have space to improve. In addition, lightweight CNNs with the plug-in attention mechanism are full of promising [185–189]. Particularly, lightweight CNNs leveraging multi-head attention mechanisms of Transformers to overcome CNNs' limitations on long-range modeling are gaining momentum [190–194].

7 Conclusions

DCNNs have gained remarkable attention in the field of computer vision, whose performance even have outperformed humans in many applications. It will continue to help drive the artificial intelligence algorithm trend. Although the current technological development has made much progress to some extent, the high computational complexity hinders its portability and performance in various mobile scenarios. Therefore, lightweight network technology naturally appears to match the growing mobile intelligence.

In this paper, the vital technologies of lightweight CNNs in recent years are reviewed, which include but are not limited to manual structured design, automatic architecture search, compression from structure to representation, and abstract knowledge distillation. In addition, we emphasize the significance of dataset and hardware support for lightweight network deployment and prospect several promising future trends corresponding to this field. We try to comb these marvelous works as clearly as possible, thus providing valuable guidance for researchers engaged in this field.

Acknowledgements The authors gratefully acknowledge the anonymous reviewers for their constructive comments. This work is supported in part by the National Natural Science Foundation of China (No. 62132007 and 20210424), the Fundamental Research Funds for the Central Universities of China (No. lzujbky-2022-pd12), and by the Natural Science Foundation of Gansu Province, China (No. 22JR5RA492). All authors have read and agreed to the published version of the manuscript.

Declarations

Conflict of interest The authors intend no competing interest.

References

- Krizhevsky A, Sutskever I, Hinton GE (2012) ImageNet classification with deep convolutional neural networks. In: Bartlett PL, Pereira FCN, Burges CJC, Bottou L, Weinberger KQ (eds) *Advances in neural information processing systems 25: 26th annual conference on neural information processing systems 2012*. Proceedings of a meeting held December 3–6, 2012, Lake Tahoe, NV, USA, pp 1106–1114
- Simonyan K, Zisserman A (2015) Very deep convolutional networks for large-scale image recognition. In: Bengio Y, LeCun Y (eds) *3rd international conference on learning representations, ICLR 2015, San Diego, CA, USA, May 7–9, 2015, conference track proceedings*
- He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: *2016 IEEE conference on computer vision and pattern recognition, CVPR 2016, Las Vegas, NV, USA, June 27–30, 2016*. IEEE Computer Society, pp 770–778
- Li Y, Liu J, Wang L (2018) Lightweight network research based on deep learning: a review. In: *2018 37th Chinese control conference (CCC)*. IEEE, pp 9021–9026
- Zhou Y, Chen S, Wang Y, Huan W (2020) Review of research on lightweight convolutional neural networks. In *2020 IEEE 5th information technology and mechatronics engineering conference (ITOEC)*. IEEE, pp 1713–1720
- Ge D-H, Li H-S, Zhang L, Liu R, Shen P, Miao Q-G (2020) Survey of lightweight neural network. *J. Softw* 31:2627–2653
- Zheng M, Tian Y, Chen H, Yang S, Song F, Gao X (2022) Lightweight network research based on deep learning. In: *International conference on computer graphics, artificial intelligence, and data processing (ICCAID 2021)*, vol 12168. SPIE, pp 333–338
- Ma J, Zhang Y, Ma Z, Mao K (2022) Research progress of lightweight neural network convolution design. *J Front Comput Sci Technol* 16(3):512–528
- Wang CH, Huang KY, Yao Y, Chen JC, Shuai HH, Cheng WH (2022) Lightweight deep learning: an overview. In *IEEE consumer electronics magazine*, pp 1–12
- Mishra R, Gupta H (2023) Transforming large-size to lightweight deep neural networks for IoT applications. *ACM Comput Surv* 55(11):1–35
- Hafiz AM (2023) A survey on light-weight convolutional neural networks: trends, issues and future scope. *J Mob Multimed* 19:1277–1298
- Cheng Y, Wang D, Zhou P, Zhang T (2018) Model compression and acceleration for deep neural networks: the principles, progress, and challenges. *IEEE Signal Process Mag* 35(1):126–136
- Zhou Z, Chen X, Li E, Zeng L, Luo K, Zhang J (2019) Edge intelligence: paving the last mile of artificial intelligence with edge computing. *Proc IEEE* 107(8):1738–1762
- Deng S, Zhao H, Fang W, Yin J, Dustdar S, Zomaya AY (2020) Edge intelligence: the confluence of edge computing and artificial intelligence. *IEEE Internet Things J* 7(8):7457–7469
- Deng L, Li G, Han S, Shi L, Xie Y (2020) Model compression and hardware acceleration for neural networks: a comprehensive survey. *Proc IEEE* 108(4):485–532
- Dianlei X, Li T, Li Y, Xiang S, Tarkoma S, Jiang T, Crowcroft J, Hui P (2021) Edge intelligence: empowering intelligence to the edge of network. *Proc IEEE* 109(11):1778–1837
- Zhao T, Xie Y, Wang Y, Cheng J, Guo X, Bin H, Chen Y (2022) A survey of deep learning on mobile devices: applications, optimizations, challenges, and research opportunities. *Proc IEEE* 110(3):334–354
- Han Cai, Ji Lin, Song Han (2022) Efficient methods for deep learning. In: *Proceedings of computer vision and pattern recognition (CVPR), Advanced Methods and Deep Learning in Computer Vision*, pp 159–190
- Shuvo MH, Islam SK, Cheng J, Morshed BI (2022) Efficient acceleration of deep learning inference on resource-constrained edge devices: a review. *Proc IEEE* 111(1): 42–91
- Deng J, Dong W, Socher R, Li L-J, Li K, Fei-Fei L (2009) ImageNet: a large-scale hierarchical image database. In: *2009 IEEE computer society conference on computer vision and pattern recognition (CVPR 2009)*, 20–25 June 2009, Miami, FL, USA. IEEE Computer Society, pp 248–255
- Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma S, Huang Z, Karpathy A, Khosla A, Bernstein MS, Berg AC, Fei-Fei L (2014) ImageNet large scale visual recognition challenge. *Int J Comput Vis* 115:211–252
- LeCun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. *Proc IEEE* 86(11):2278–2324
- Denil M, Shakibi B, Dinh L, Ranzato MA, de Freitas N (2013) Predicting parameters in deep learning. In: Burges CJC, Bottou L, Ghahramani Z, Weinberger KQ (eds) *Advances in neural information processing systems 26: 27th annual conference on neural information processing systems 2013*. Proceedings of a meeting held December 5–8, 2013, Lake Tahoe, NV, USA, pp 2148–2156
- Denton EL, Zaremba W, Bruna J, LeCun Y, Fergus R (2014) Exploiting linear structure within convolutional networks for efficient evaluation. In: Ghahramani Z, Welling M, Cortes C, Lawrence ND, Weinberger KQ (eds) *Advances in neural information processing systems 27: annual conference on neural information processing systems 2014, December 8–13 2014, Montreal, QC, Canada*, pp 1269–1277
- Szegedy C, Liu W, Jia Y, Sermanet P, Reed SE, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A (2015) Going deeper with convolutions. In: *IEEE conference on computer vision and pattern recognition, CVPR 2015, Boston, MA, USA, June 7–12, 2015*. IEEE Computer Society, pp 1–9
- Szegedy C, Vanhoucke V, Ioffe S, Shlens J, Wojna Z (2016) Rethinking the inception architecture for computer vision. In: *2016 IEEE conference on computer vision and pattern recognition, CVPR 2016, Las Vegas, NV, USA, June 27–30, 2016*. IEEE Computer Society, pp 2818–2826
- Szegedy C, Ioffe S, Vanhoucke V (2016) Inception-V4, inception-ResNet and the impact of residual connections on learning. *CoRR*. <https://arxiv.org/abs/1602.07261>
- Chollet F (2017) Xception: deep learning with depthwise separable convolutions. In: *2017 IEEE conference on computer vision and pattern recognition, CVPR 2017, Honolulu, HI, USA, July 21–26, 2017*. IEEE Computer Society, pp 1800–1807
- Wang M, Liu B, Foroosh H (2016) Design of efficient convolutional layers using single intra-channel convolution, topological subdivision and spatial “bottleneck” structure. *arXiv: Computer Vision and Pattern Recognition*
- Wang M, Liu B, Foroosh H (2017) Factorized convolutional neural networks. In: *2017 IEEE international conference on computer vision workshops, ICCV Workshops 2017, Venice, Italy, October 22–29, 2017*. IEEE Computer Society, pp 545–553
- Iandola FN, Moskewicz MW, Ashraf K, Han S, Dally WJ, Keutzer K (2016) SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <1mb model size. *CoRR*. <https://arxiv.org/abs/1602.07360>
- Lin M, Chen Q, Yan S (2014) Network in network. In: Bengio Y, LeCun Y (eds) *2nd international conference on learning*

- representations, ICLR 2014, Banff, AB, Canada, April 14–16, 2014, conference track proceedings
33. Gholami A, Kwon K, Wu B, Tai Z, Yue X, Jin PH, Zhao S, Keutzer K (2018) SqueezeNext: hardware-aware neural network design. In: 2018 IEEE conference on computer vision and pattern recognition workshops, CVPR Workshops 2018, Salt Lake City, UT, USA, June 18–22, 2018. Computer Vision Foundation/IEEE Computer Society, pp 1638–1647
 34. 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. CoRR. <https://arxiv.org/abs/1704.04861>
 35. Sandler M, Howard AG, Zhu M, Zhmoginov A, Chen L-C (2018) MobileNetV2: inverted residuals and linear bottlenecks. In: 2018 IEEE conference on computer vision and pattern recognition, CVPR 2018, Salt Lake City, UT, USA, June 18–22, 2018. Computer Vision Foundation/IEEE Computer Society, pp 4510–4520
 36. Zhang X, Zhou X, Lin M, Sun J (2018) ShuffleNet: an extremely efficient convolutional neural network for mobile devices. In: 2018 IEEE conference on computer vision and pattern recognition, CVPR 2018, Salt Lake City, UT, USA, June 18–22, 2018. Computer Vision Foundation/IEEE Computer Society, pp 6848–6856
 37. 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
 38. Zhou D, Hou Q, Chen Y, Feng J, Yan S (2020) Rethinking bottleneck structure for efficient mobile network design. In: Vedaldi A, Bischof H, Brox T, Frahm J-M (eds) Computer vision—ECCV 2020—16th European conference, Glasgow, UK, August 23–28, 2020, proceedings, part III, volume 12348 of lecture notes in computer science. Springer, pp 680–697
 39. Haase D, Amthor M (2020) Rethinking depthwise separable convolutions: how intra-kernel correlations lead to improved MobileNets. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 14600–14609
 40. Gao H, Wang Z, Cai L, Ji S (2018) ChannelNets: compact and efficient convolutional neural networks via channel-wise convolutions. IEEE transactions on pattern analysis and machine intelligence, pp 2570–2581
 41. Kopuklu O, Kose N, Gunduz A, Rigoll G (2019) Resource efficient 3D convolutional neural networks. In: Proceedings of the IEEE/CVF international conference on computer vision workshops, pp 1910–1919
 42. Wu B, Iandola F, Jin PH, Keutzer K (2017) SqueezeDet: unified, small, low power fully convolutional neural networks for real-time object detection for autonomous driving. In: Proceedings of the IEEE conference on computer vision and pattern recognition workshops, pp 129–137
 43. Wu B, Wan A, Yue X, Keutzer K (2018) SqueezeSeg: convolutional neural nets with recurrent CRF for real-time road-object segmentation from 3D LiDAR point cloud. In: 2018 IEEE international conference on robotics and automation (ICRA). IEEE, pp 1887–1893
 44. Wang RJ, Li X, Ling CX (2018) Pelee: a real-time object detection system on mobile devices. In Proceedings of the 32nd International Conference on Neural Information Processing Systems (NIPS'18). Curran Associates Inc., Red Hook, NY, USA, pp 1967–1976
 45. Chen S, Liu Y, Gao X, Han Z (2018) MobileFaceNets: efficient CNNs for accurate real-time face verification on mobile devices. In: Biometric recognition: 13th Chinese conference, CCBP 2018, Urumqi, China, August 11–12, 2018, proceedings 13. Springer, pp 428–438
 46. Duong CN, Quach KG, Jalata I, Le N, Luu K (2019) MobiFace: a lightweight deep learning face recognition on mobile devices. In 2019 IEEE 10th international conference on biometrics theory, applications and systems (BTAS). IEEE, pp 1–6
 47. Han K, Wang Y, Chang X, Guo J, Chunjing X, Enhua W, Tian Q (2022) GhostNets on heterogeneous devices via cheap operations. *Int J Comput Vis* 130(4):1050–1069
 48. Cui C, Gao T, Wei S, Du Y, Guo R, Dong S, Lu B, Zhou Y, Lv X, Liu Q et al (2021) PP-LCNet: a lightweight CPU convolutional neural network. arXiv Preprint. <http://arxiv.org/abs/2109.15099>
 49. Duong CN, Quach KG, Jalata I, Le N, Luu K (2019) MobiFace: a lightweight deep learning face recognition on mobile devices. In: 2019 IEEE 10th international conference on biometrics theory, applications and systems (BTAS). IEEE, pp 1–6
 50. Mehta S, Rastegari M, Caspi A, Shapiro L, Hajishirzi H (2018) ESPNet: efficient spatial pyramid of dilated convolutions for semantic segmentation. In: Proceedings of the European conference on computer vision (ECCV), pp 552–568
 51. Mehta S, Rastegari M, Shapiro L, Hajishirzi H (2019) ESP-NetV2: a light-weight, power efficient, and general purpose convolutional neural network. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 9190–9200
 52. Han K, Wang Y, Tian Q, Guo J, Xu C, Xu C (2020) GhostNet: more features from cheap operations. In: 2020 IEEE/CVF conference on computer vision and pattern recognition, CVPR 2020, Seattle, WA, USA, June 13–19, 2020. Computer Vision Foundation/IEEE, pp 1577–1586
 53. Darbani P, Rohbani N, Beitollahi H, Lotfi-Kamran P (2022) RASHT: a partially reconfigurable architecture for efficient implementation of CNNs. *IEEE Trans Very Large Scale Integr Syst* 30(7):860–868
 54. Vasu PKA, Gabriel J, Zhu J, Tuzel O, Ranjan A (2023) MobileOne: an improved one millisecond mobile backbone. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 7907–7917
 55. Cai Z, Shen Q (2023) FalconNet: Factorization for the light-weight ConvNets. arXiv Preprint. <http://arxiv.org/abs/2306.06365>
 56. Ding X, Zhang X, Ma N, Han J, Ding G, Sun J (2021) RepVGG: making VGG-style ConvNets great again. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 13733–13742
 57. Nair V, Hinton GE (2010) Rectified linear units improve restricted Boltzmann machines. In: Proceedings of the 27th international conference on machine learning (ICML-10), pp 807–814
 58. He K, Zhang X, Ren S, Sun J (2015) Delving deep into rectifiers: surpassing human-level performance on ImageNet classification. In: Proceedings of the IEEE international conference on computer vision, pp 1026–1034
 59. Hu J, Shen L, Sun G (2018) Squeeze-and-excitation networks. In: 2018 IEEE conference on computer vision and pattern recognition, CVPR 2018, Salt Lake City, UT, USA, June 18–22, 2018. Computer Vision Foundation/IEEE Computer Society, pp 7132–7141
 60. Wu B, Wan A, Yue X, Jin P, Zhao S, Golmant N, Gholamnejad A, Gonzalez J, Keutzer K (2018) Shift: a zero flop, zero parameter alternative to spatial convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 9127–9135
 61. Mehta S, Hajishirzi H, Rastegari M (2020) DiceNet: dimension-wise convolutions for efficient networks. *IEEE Trans Pattern Anal Mach Intell* 44(5):2416–2425
 62. Lai L, Suda N, Chandra V (2018) Not all ops are created equal! CoRR. <https://arxiv.org/abs/1801.04326>

63. Tan M, Le QV (2019) EfficientNet: rethinking model scaling for convolutional neural networks. In: Chaudhuri K, Salakhutdinov R (eds) Proceedings of the 36th international conference on machine learning, ICML 2019, 9–15 June 2019, Long Beach, CA, USA, volume 97 of proceedings of machine learning research. PMLR, pp 6105–6114
64. Tan M, Le QV (2021) EfficientNetV2: smaller models and faster training. In: Meila M, Zhang T (eds) Proceedings of the 38th international conference on machine learning, ICML 2021, 18–24 July 2021, virtual event, volume 139 of proceedings of machine learning research. PMLR, pp 10096–10106
65. Zoph B, Le QV (2016) Neural architecture search with reinforcement learning. arXiv Preprint. <http://arxiv.org/abs/1611.01578>
66. 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
67. 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
68. Tan M, Chen B, Pang R, Vasudevan V, Sandler M, Howard A, Le QV (2019) MnasNet: platform-aware neural architecture search for mobile. In: IEEE conference on computer vision and pattern recognition, CVPR 2019, Long Beach, CA, USA, June 16–20, 2019. Computer Vision Foundation/IEEE, pp 2820–2828
69. Howard A, Pang R, Adam H, Le QV, Sandler M, Chen B, Wang W, Chen L-C, Tan M, Chu G, Vasudevan VK, Zhu Y (2019) Searching for MobileNetV3. In: International conference on computer vision
70. Yang T-J, Howard A, Chen B, Zhang X, Go A, Sandler M, Sze V, Adam H (2018) NetAdapt: platform-aware neural network adaptation for mobile applications. In: Proceedings of the European conference on computer vision (ECCV), pp 285–300
71. Chu X, Zhang B, Xu R (2019) MoGA: searching beyond MobileNetV3. In: ICASSP 2020—2020 IEEE international conference on acoustics, speech and signal processing (ICASSP), pp 4042–4046
72. Wu B, Dai X, Zhang P, Wang Y, Sun F, Wu Y, Tian Y, Vajda P, Jia Y, Keutzer K (2018) FBNet: hardware-aware efficient ConvNet design via differentiable neural architecture search. CoRR. <https://arxiv.org/abs/1812.03443>
73. Wan A, Dai X, Zhang P, He Z, Tian Y, Xie S, Wu B, Yu M, Xu T, Chen K, Vajda P, Gonzalez JE (2020) FBNetV2: differentiable neural architecture search for spatial and channel dimensions. CoRR. <https://arxiv.org/abs/2004.05565>
74. Dai X, Zhang P, Wu B, Yin H, Sun F, Wang Y, Dukhan M, Hu Y, Wu Y, Jia Y, Vajda P, Uyttendaele M, Jha NK (2018) ChamNet: towards efficient network design through platform-aware model adaptation. CoRR. <https://arxiv.org/abs/1812.08934>
75. Cai H, Zhu L, Han S (2018) ProxylessNAS: direct neural architecture search on target task and hardware. CoRR. <https://arxiv.org/abs/1812.00332>
76. Tan M, Le QV (2019) MixConv: mixed depthwise convolutional kernels. CoRR. <https://arxiv.org/abs/1907.09595>
77. Lin M, Chen H, Sun X, Qian Q, Li H, Jin R (2020) Neural architecture design for GPU-efficient networks. arXiv Preprint. <http://arxiv.org/abs/2006.14090>
78. Dai X, Wan A, Zhang P, Wu B, He Z, Wei Z, Chen K, Tian Y, Yu M, Vajda P, Gonzalez JE (2020) FBNetV3: joint architecture-recipe search using neural acquisition function. CoRR. <https://arxiv.org/abs/2006.02049>
79. Wu B, Li C, Zhang H, Dai X, Zhang P, Yu M, Wang J, Lin Y, Vajda P (2021) FBNetV5: neural architecture search for multiple tasks in one run. CoRR. <https://arxiv.org/abs/2111.10007>
80. Zhang L, Shen H, Luo Y, Cao X, Pan L, Wang T, Feng Q (2022) Efficient CNN architecture design guided by visualization. In: 2022 IEEE international conference on multimedia and expo (ICME). IEEE, pp 1–6
81. Cai H, Zhu L, Han S (2019) ProxylessNAS: direct neural architecture search on target task and hardware. In: 7th international conference on learning representations, ICLR 2019, New Orleans, LA, USA, May 6–9, 2019. OpenReview.net
82. Han S, Pool J, Tran J, Dally WJ (2015) Learning both weights and connections for efficient neural networks. In Proceedings of the 28th international conference on neural information processing systems - volume 1 (NIPS'15). MIT Press, Cambridge, MA, USA, pp 1135–1143
83. Han S, Liu X, Mao H, Pu J, Pedram A, Horowitz MA, Dally WJ (2016) EIE: efficient inference engine on compressed deep neural network. ACM SIGARCH Comput Archit News 44(3):243–254
84. Meng F, Cheng H, Li K, Luo H, Guo X, Lu G, Sun X (2020) Pruning Filter in Filter. In Proceedings of the 34th international conference on neural information processing systems (NIPS'20). Curran Associates Inc., Red Hook, NY, USA, Article 1479, pp 17629–17640
85. Huo Z, Wang C, Chen W, Li Y, Wang J, Wu J (2022) Balanced stripe-wise pruning in the filter. In: International conference on acoustics, speech, and signal processing, pp 4408–4412
86. Ma X, Guo F-M, Niu W, Lin X, Tang J, Ma K, Ren B, Wang Y (2019) PCONV: the missing but desirable sparsity in DNN weight pruning for real-time execution on mobile devices. In: the AAAI conference on artificial intelligence, pp 5117–5124
87. 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 twenty-fifth international conference on architectural support for programming languages and operating systems (ASPLOS '20), pp 907–922
88. Vysogorets A, Kempe J (2021) Connectivity matters: neural network pruning through the lens of effective sparsity. <http://arxiv.org/abs/2107.02306>
89. Li H, Kadav A, Durdanovic I, Samet H, P Graf H (2016) Pruning filters for efficient convnets. arXiv: Computer Vision and Pattern Recognition
90. Ye J, Lu X, Lin Z, Wang JZ (2018) Rethinking the smaller-normless-informative assumption in channel pruning of convolution layers. In: International conference on learning representations
91. He Y, Liu P, Wang Z, Hu Z, Yang Y (2019) Filter pruning via geometric median for deep convolutional neural networks acceleration. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 4335–4344
92. Luo J-H, Wu J, Lin W (2017) ThiNet: a filter level pruning method for deep neural network compression. In: 2017 IEEE international conference on computer vision (ICCV)
93. Fang G, Ma X, Song M, Mi MB, Wang X (2023) DepGraph: towards any structural pruning. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 16091–16101
94. He Y, Lin J, Liu Z, Wang H, Li LJ, Han S (2018) AMC: AutoML for model compression and acceleration on mobile devices. In: European conference on computer vision
95. Li B, Wu B, Su J, Wang G, Lin L (2020) EagleEye: fast sub-net evaluation for efficient neural network pruning. In: European conference on computer vision, pp 639–654
96. Blalock D, Ortiz JGG, Frankle J, Guttig J (2020) What is the state of neural network pruning? Proc Mach Learn Syst 2:129–146
97. Wang H, Qin C, Bai Y, Fu Y (2023) Why is the state of neural network pruning so confusing? On the fairness, comparison setup, and trainability in network pruning. arXiv Preprint. <http://arxiv.org/abs/2301.05219>

98. Li Y, Adamczewski K, Li W, Gu S, Timofte R, Van Gool L (2021) Revisiting random channel pruning for neural network compression. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 191–201
99. Jaderberg M, Vedaldi A, Zisserman A (2014) Speeding up convolutional neural networks with low rank expansions. In: British machine vision conference
100. Zhang X, Zou J, He K, Sun J (2015) Accelerating very deep convolutional networks for classification and detection. *IEEE Trans Pattern Anal Mach Intell.* <https://doi.org/10.1109/TPAMI.2015.2502579>
101. Kim Y, Park E, Yoo S, Choi T-L, Yang L, Shin D (2015) Compression of deep convolutional neural networks for fast and low power mobile applications. *arXiv: Computer Vision and Pattern Recognition*
102. Chen Y, Jin X, Kang B, Feng J, Yan S (2018) Sharing residual units through collective tensor factorization to improve deep neural networks. In: International joint conference on artificial intelligence
103. Su J, Li J, Bhattacharjee B, Huang F (2018) Tensorial neural networks: generalization of neural networks and application to model compression. *arXiv: Machine Learning*
104. Garipov T, Podoprikin D, Novikov A, Vetrov D (2016) Ultimate tensorization: compressing convolutional and fc layers alike. *arXiv: Learning*
105. Hawkins C, Yang H, Li M, Lai L, Chandra V (2021) Low-rank+ sparse tensor compression for neural networks. *arXiv Preprint.* <http://arxiv.org/abs/2111.01697>
106. Chu B-S, Lee C-R (2021) Low-rank tensor decomposition for compression of convolutional neural networks using funnel regularization. *arXiv Preprint.* <http://arxiv.org/abs/2112.03690>
107. Miyashita D, Lee EH, Murmann B (2016) Convolutional neural networks using logarithmic data representation. *CoRR.* <https://arxiv.org/abs/1603.01025>
108. Zhou A, Yao A, Guo Y, Xu L, Chen Y (2017) Incremental network quantization: towards lossless CNNs with low-precision weights. In: 5th international conference on learning representations, ICLR 2017, Toulon, France, April 24–26, 2017, conference track proceedings
109. Guo Y, Yao A, Zhao H, Chen Y (2017) Network sketching: exploiting binary structure in deep CNNs. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 5955–5963
110. Nahshan Y, Chmiel B, Baskin C, Zheltonozhskii E, Banner R, Bronstein AM, Mendelson A (2021) Loss aware post-training quantization. *Mach Learn* 110(11):3245–3262
111. Li Y, Gong R, Tan X, Yang Y, Hu P, Zhang Q, Yu F, Wang W, Gu S (2021) BRECQ: pushing the limit of post-training quantization by block reconstruction. *arXiv Preprint.* <http://arxiv.org/abs/2102.05426>
112. Nagel M, van Baalen M, Blankevoort T, Welling M (2019) Data-free quantization through weight equalization and bias correction. In: 2019IEEE/CVF international conference on computer vision, ICCV 2019, Seoul, Korea (South), October 27–November 2, 2019. *IEEE*, pp 1325–1334
113. Gholami A, Kim S, Dong Z, Yao Z, Mahoney MW, Keutzer K (2021) A survey of quantization methods for efficient neural network inference. *CoRR.* <https://arxiv.org/abs/2103.13630>
114. Nagel M, Fournarakis M, Amjad RA, Bondarenko Y, van Baalen M, Blankevoort T (2021) A white paper on neural network quantization. *CoRR.* <https://arxiv.org/abs/2106.08295>
115. Nagel M, Amjad RA, van Baalen M, Louizos C, Blankevoort T (2020) Up or down? Adaptive rounding for post-training quantization. In: Proceedings of the 37th international conference on machine learning, ICML 2020, 13–18 July 2020, virtual event, volume 119 of proceedings of machine learning research. *PMLR*, pp 7197–7206
116. Banner R, Nahshan Y, Soudry D (2019) Post training 4-bit quantization of convolutional networks for rapid-deployment. In: Wallach HM, Larochelle H, Beygelzimer A, d'Alché-Buc F, Fox EB, Garnett R (eds) *Advances in neural information processing systems 32: annual conference on neural information processing systems 2019, NeurIPS 2019, December 8–14, 2019, Vancouver, BC, Canada*, pp 7948–7956
117. Cai Y, Yao Z, Dong Z, Gholami A, Mahoney MW, Keutzer K (2020) ZeroQ: a novel zero shot quantization framework. In: 2020IEEE/CVF conference on computer vision and pattern recognition, CVPR 2020, Seattle, WA, USA, June 13–19, 2020. *Computer Vision Foundation/IEEE*, pp 13166–13175
118. Hubara I, Nahshan Y, Hanani Y, Banner R, Soudry D (2021) Accurate post training quantization with small calibration sets. In: Meila M, Zhang T (eds) *Proceedings of the 38th international conference on machine learning, ICML 2021, 18–24 July 2021, Virtual Event, volume 139 of proceedings of machine learning research.* *PMLR*, pp 4466–4475
119. Wei X, Gong R, Li Y, Liu X, Yu F (2022) QDrop: randomly dropping quantization for extremely low-bit post-training quantization. In: The tenth international conference on learning representations, ICLR 2022, virtual event, April 25–29, 2022. *OpenReview.net*
120. Courbariaux M, Bengio Y, David J-P (2015) BinaryConnect: training deep neural networks with binary weights during propagations. In: Cortes C, Lawrence ND, Lee DD, Sugiyama M, Garnett R (eds) *Advances in neural information processing systems 28: annual conference on neural information processing systems 2015, December 7–12, 2015, Montreal, QC, Canada*, pp 3123–3131
121. Hubara I, Courbariaux M, Soudry D, El-Yaniv R, Bengio Y (2016) Binarized neural networks. In: Lee DD, Sugiyama M, von Luxburg U, Guyon I, Garnett R (eds) *Advances in neural information processing systems 29: annual conference on neural information processing systems 2016, December 5–10, 2016, Barcelona, Spain*, pp 4107–4115
122. Rastegari M, Ordonez V, Redmon J, Farhadi A (2016) XNOR-Net: ImageNet classification using binary convolutional neural networks. In: Leibe B, Matas J, Sebe N, Welling M (eds) *Computer vision-ECCV 2016-14th European conference, Amsterdam, The Netherlands, October 11–14, 2016, proceedings, part IV, volume 9908 of lecture notes in computer science.* *Springer*, pp 525–542
123. Zhou S, Wu Y, Zekun N, Zhou X, Wen H, Zou Y (2016) DoReFa-Net: training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv: Neural and Evolutionary Computing*
124. Hubara I, Courbariaux M, Soudry D, El-Yaniv R, Bengio Y (2017) Quantized neural networks: training neural networks with low precision weights and activations. *J Mach Learn Res* 18:187:1–187:30
125. Gysel P, Pimentel JJ, Motamedi M, Ghiasi S (2018) Ristretto: a framework for empirical study of resource-efficient inference in convolutional neural networks. *IEEE Trans Neural Netw Learn Syst* 29(11):5784–5789
126. Bengio Y, Léonard N, Courville AC (2013) Estimating or propagating gradients through stochastic neurons for conditional computation. *CoRR.* <https://arxiv.org/abs/1308.3432>
127. Choi J, Wang Z, Venkataramani S, Chuang PI-J, Srinivasan V, Gopalakrishnan K (2018) PACT: parameterized clipping activation for quantized neural networks. *CoRR.* <https://arxiv.org/abs/1805.06085>
128. Jung S, Son C, Lee S, Son JW, Han J-J, Kwak Y, Hwang SJ, Choi C (2019) Learning to quantize deep networks by optimizing

- quantization intervals with task loss. In: IEEE conference on computer vision and pattern recognition, CVPR 2019, Long Beach, CA, USA, June 16–20, 2019. Computer Vision Foundation/IEEE, pp 4350–4359
129. Esser SK, McKinstry JL, Bablani D, Appuswamy R, Modha DS (2020) Learned step size quantization. In: 8th international conference on learning representations, ICLR 2020, Addis Ababa, Ethiopia, April 26–30, 2020. OpenReview.net
 130. Bhalgat Y, Lee J, Nagel M, Blankevoort T, Kwak N (2020) LSQ+: improving low-bit quantization through learnable offsets and better initialization. In: 2020IEEE/CVF conference on computer vision and pattern recognition, CVPR workshops 2020, Seattle, WA, USA, June 14–19, 2020. Computer Vision Foundation/IEEE, pp 2978–2985
 131. Asim F, Park J, Azamat A, Lee J (2022) Centered symmetric quantization for hardware-efficient low-bit neural networks. British Machine Vision Association (BMVA)
 132. Dong Z, Yao Z, Arfeen D, Gholami A, Mahoney MW, Keutzer K (2020) HAWQ-V2: Hessian aware trace-weighted quantization of neural networks. *Adv Neural Inf Process Syst* 33:18518–18529
 133. Yao Z, Dong Z, Zheng Z, Gholami A, Yu J, Tan E, Wang L, Huang Q, Wang Y, Mahoney M et al (2021) HAWQ-V3: dyadic neural network quantization. In: International conference on machine learning. PMLR, pp 11875–11886
 134. He Y, Lin J, Liu Z, Wang H, Li L-J, Han S (2018) AMc: AutoML for model compression and acceleration on mobile devices. In: Proceedings of the European conference on computer vision (ECCV), pp 784–800
 135. Bucila C, Caruana R, Niculescu-Mizil A (2006) Model compression. In: Eliassi-Rad T, Ungar LH, Craven M, Gunopulos D (eds) Proceedings of the twelfth ACM SIGKDD international conference on knowledge discovery and data mining, Philadelphia, PA, USA, August 20–23, 2006. ACM, pp 535–541
 136. Hinton GE, Vinyals O, Dean J (2015) Distilling the knowledge in a neural network. CoRR. <https://arxiv.org/abs/1503.02531>
 137. Romero A, Ballas N, Kahou SE, Chassang A, Gatta C, Bengio Y (eds) (2015) FitNets: hints for thin deep nets. In: Bengio Y, LeCun Y (eds) 3rd International conference on learning representations, ICLR 2015, San Diego, CA, USA, May 7–9, 2015, conference track proceedings
 138. Zhang Y, Xiang T, Hospedales TM, Lu H (2018) Deep mutual learning. In: 2018 IEEE conference on computer vision and pattern recognition, CVPR 2018, Salt Lake City, UT, USA, June 18–22, 2018. Computer Vision Foundation/IEEE Computer Society, pp 4320–4328
 139. Furlanello T, Lipton ZC, Tschannen M, Itti L, Anandkumar A (2018) Born-again neural networks. In: Dy JG, Krause A (eds) Proceedings of the 35th international conference on machine learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10–15, 2018, volume 80 of proceedings of machine learning research. PMLR, pp 1602–1611
 140. Yang C, Xie L, Su C, Yuille AL (2019) Snapshot distillation: teacher–student optimization in one generation. In: IEEE conference on computer vision and pattern recognition, CVPR 2019, Long Beach, CA, USA, June 16–20, 2019. Computer Vision Foundation/IEEE, pp 2859–2868
 141. Cho JH, Hariharan B (2019) On the efficacy of knowledge distillation. In: 2019 IEEE/CVF international conference on computer vision, ICCV 2019, Seoul, Korea (South), October 27–November 2, 2019. IEEE, pp 4793–4801
 142. Mirzadeh SI, Farajtabar M, Li A, Levine N, Matsukawa A, Ghasemzadeh H (2020) Improved knowledge distillation via teacher assistant. In: Proceedings of the AAAI conference on artificial intelligence, vol 34, pp 5191–5198
 143. Chen D, Mei J-P, Zhang H, Wang C, Feng Y, Chen C (2022) Knowledge distillation with the reused teacher classifier. In: IEEE/CVF conference on computer vision and pattern recognition, CVPR 2022, New Orleans, LA, USA, June 18–24, 2022. IEEE, pp 11923–11932
 144. Beyer L, Zhai X, Royer A, Markeeva L, Anil R, Kolesnikov A (2022) Knowledge distillation: a good teacher is patient and consistent. In: IEEE/CVF conference on computer vision and pattern recognition, CVPR 2022, New Orleans, LA, USA, June 18–24, 2022. IEEE, pp 10915–10924
 145. Yim J, Joo D, Bae J-H, Kim J (2017) A gift from knowledge distillation: fast optimization, network minimization and transfer learning. In: 2017IEEE conference on computer vision and pattern recognition, CVPR 2017, Honolulu, HI, USA, July 21–26, 2017. IEEE Computer Society, pp 7130–7138
 146. Zagoruyko S, Komodakis N (2017) Paying more attention to attention: improving the performance of convolutional neural networks via attention transfer. In: 5th international conference on learning representations, ICLR 2017, Toulon, France, April 24–26, 2017, conference track proceedings. OpenReview.net
 147. Huang Z, Wang N (2017) Like what you like: knowledge distill via neuron selectivity transfer. CoRR. <https://arxiv.org/abs/1707.01219>
 148. Heo B, Lee M, Yun S, Choi JY (2018) Knowledge transfer via distillation of activation boundaries formed by hidden neurons. CoRR. <https://arxiv.org/abs/1811.03233>
 149. Kim J, Park S, Kwak N (2018) Paraphrasing complex network: network compression via factor transfer. In: Bengio S, Wallach HM, Larochelle H, Grauman K, Cesa-Bianchi N, Garnett R (eds) Advances in neural information processing systems 31: annual conference on neural information processing systems 2018, NeurIPS 2018, December 3–8, 2018, Montréal, Canada, pp 2765–2774
 150. Heo B, Kim J, Yun S, Park H, Kwak N, Choi JY (2019) A comprehensive overhaul of feature distillation. In: 2019 IEEE/CVF international conference on computer vision, ICCV 2019, Seoul, Korea (South), October 27–November 2, 2019. IEEE, pp 1921–1930
 151. Park W, Kim D, Lu Y, Cho M (2019) Relational knowledge distillation. In: IEEE conference on computer vision and pattern recognition, CVPR 2019, Long Beach, CA, USA, June 16–20, 2019. Computer Vision Foundation/IEEE, pp 3967–3976
 152. Peng B, Jin X, Li D, Zhou S, Wu Y, Liu J, Zhang Z, Liu Y (2019) Correlation congruence for knowledge distillation. In: 2019IEEE/CVF international conference on computer vision, ICCV 2019, Seoul, Korea (South), October 27–November 2, 2019. IEEE, pp 5006–5015
 153. Tian Y, Krishnan D, Isola P (2019) Contrastive representation distillation. CoRR. <https://arxiv.org/abs/1910.10699>
 154. Tung F, Mori G (2019) Similarity-preserving knowledge distillation. In: 2019 IEEE/CVF international conference on computer vision, ICCV 2019, Seoul, Korea (South), October 27–November 2, 2019. IEEE, pp 1365–1374
 155. Zhao B, Cui Q, Song R, Qiu Y, Liang J (2022) Decoupled knowledge distillation. In: IEEE/CVF conference on computer vision and pattern recognition, CVPR 2022, New Orleans, LA, USA, June 18–24, 2022. IEEE, pp 11943–11952
 156. Abadi M, Barham P, Chen J, Chen Z, Davis A, Dean J, Devin M, Ghemawat S, Irving G, Isard M, Kudlur M, Levenberg J, Monga R, Moore S, Murray DG, Steiner B, Tucker PA, Vasudevan V, Warden P, Wicke M, Yu Y, Zheng X (2016) TensorFlow: a system for large-scale machine learning. In: Keeton K, Roscoe T (eds) 12th USENIX symposium on operating systems design and implementation, OSDI 2016, Savannah, GA, USA, November 2–4, 2016. USENIX Association, pp 265–283
 157. Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, Killeen T, Lin Z, Gimelshein N, Antiga L, Desmaison A, Köpf A, Yang EZ, DeVito Z, Raison M, Tejani A, Chilamkurthy

- S, Steiner B, Fang L, Bai J, Chintala S (2019) PyTorch: an imperative style, high-performance deep learning library. In: Wallach HM, Larochelle H, Beygelzimer A, d'Alché-Buc F, Fox EB, Garnett R (eds) *Advances in neural information processing systems 32: annual conference on neural information processing systems 2019, NeurIPS 2019*, December 8–14, 2019, Vancouver, BC, Canada, pp 8024–8035
158. Keras. <https://keras.io/>. Accessed 16 Nov 2022
 159. PyTorch lightning. <https://www.pytorchlightning.ai/>. Accessed 20 Dec 2022
 160. Theano. <https://github.com/Theano/Theano>. Accessed 16 Nov 2022
 161. The microsoft cognitive toolkit. <https://learn.microsoft.com/en-us/cognitive-toolkit/>. Accessed 16 Dec 2022
 162. Deeplearning4j suite overview. <https://deeplearning4j.kondu.it.ai/>. Accessed 16 Nov 2022
 163. Jia Y, Shelhamer E, Donahue J, Karayev S, Long J, Girshick RB, Guadarrama S, Darrell T (2014) Caffe: convolutional architecture for fast feature embedding. In: Hua KA, Rui Y, Steinmetz R, Hanjalic A, Natsev A, Zhu W (eds) *Proceedings of the ACM international conference on multimedia, MM'14*, Orlando, FL, USA, November 03–07, 2014. ACM, pp 675–678
 164. Chen T, Li M, Li Y, Lin M, Wang N, Wang M, Xiao T, Xu B, Zhang C, Zhang Z (2015) MxNet: a flexible and efficient machine learning library for heterogeneous distributed systems. CoRR. <https://arxiv.org/abs/1512.01274>
 165. Apache MXNet. <https://mxnet.apache.org/versions/1.9.1/>. Accessed 23 Dec 2022
 166. NVIDIA TensorRT. <https://developer.nvidia.com/zh-cn/tensorrt>. Accessed 20 Dec 2022
 167. Kechit Goyal. Title, deep learning frameworks in 2023 you can't ignore. <https://www.upgrad.com/blog/top-deep-learning-frameworks/>. Accessed 09 Jan 2023
 168. Shuvo MMH, Islam SK, Cheng J, Morshed BI (2023) Efficient acceleration of deep learning inference on resource-constrained edge devices: a review. *Proc IEEE* 111(1):42–91
 169. Sze V, Chen Y-H, Yang T-J, Emer JS (2017) Efficient processing of deep neural networks: a tutorial and survey. *Proc IEEE* 105(12):2295–2329
 170. Xu X, Ding Y, Hu SX, Niemier M, Cong J, Hu Y, Shi Y (2018) Scaling for edge inference of deep neural networks. *Nat Electron* 1(4):216–222
 171. Jetson TX2 Module. <https://developer.nvidia.com/embedded/jetson-tx2>. Accessed 09 Dec 2022
 172. Intel Edison development platform. https://www.intel.com/content/dam/support/us/en/documents/edison/sb/edison_pb_331179002.pdf. Accessed 29 Dec 2022
 173. Chen Y-H, Krishna T, Emer JS, Sze V (2017) Eyeriss: an energy-efficient reconfigurable accelerator for deep convolutional neural networks. *IEEE J Solid State Circuits* 52(1):127–138
 174. Chen Y-H, Yang T-J, Emer JS, Sze V (2019) Eyeriss V2: a flexible accelerator for emerging deep neural networks on mobile devices. *IEEE J Emerg Sel Top Circuits Syst* 9(2):292–308
 175. Baischer L, Wess M, TaheriNejad N (2021) Learning on hardware: a tutorial on neural network accelerators and co-processors. arXiv Preprint. <http://arxiv.org/abs/2104.09252>
 176. Krizhevsky A, Hinton G (2009) CIFAR-100 (Canadian institute for advanced research). Technical report, CIFAR
 177. Khosla A, Jayadevaprakash N, Yao B, Fei-Fei L (2011) Novel dataset for fine-grained image categorization. In: *First workshop on fine-grained visual categorization*, IEEE conference on computer vision and pattern recognition, Colorado Springs, CO, June 2011
 178. Lin T-Y, Maire M, Belongie SJ, Hays J, Perona P, Ramanan D, Dollár P, Zitnick CL (2014) Microsoft COCO: common objects in context. In: Fleet DJ, Pajdla T, Schiele B, Tuytelaars T (eds) *Computer vision—ECCV 2014—13th European conference*, Zurich, Switzerland, September 6–12, 2014, proceedings, part V, volume 8693 of lecture notes in computer science. Springer, pp 740–755
 179. Cordts M, Sebastian Ramos MO, Rehfeld T, Enzweiler M, Benenson R, Franke U, Roth S, Schiele B (2016) The cityscapes dataset for semantic urban scene understanding. In: 2016 IEEE conference on computer vision and pattern recognition, CVPR 2016, Las Vegas, NV, USA, June 27–30, 2016. IEEE Computer Society, pp 3213–3223
 180. Lai L, Suda N (2018) Rethinking machine learning development and deployment for edge devices. CoRR. <https://arxiv.org/abs/1806.07846>
 181. Polino A, Pascanu R, Alistarh D (2018) Model compression via distillation and quantization. arXiv Preprint. <http://arxiv.org/abs/1802.05668>
 182. Liu Z, Mu H, Zhang X, Guo Z, Yang X, Cheng K-T, Sun J (2019) MetaPruning: meta learning for automatic neural network channel pruning. In: *Proceedings of the IEEE/CVF international conference on computer vision*, pp 3296–3305
 183. Liu Y, Yang G, Qiao S, Liu M, Qu L, Han N, Wu T, Yuan G, Wu T, Peng Y (2023) Imbalanced data classification: using transfer learning and active sampling. *Eng Appl Artif Intell* 117(Part):105621
 184. Ren P, Xiao Y, Chang X, Huang P-Y, Li Z, Gupta BB, Chen X, Wang X (2022) A survey of deep active learning. *ACM Comput Surv* 54(9):180:1–180:40
 185. Wang Q, Wu B, Zhu P, Li P, Zuo W, Hu Q (2020) ECA-Net: efficient channel attention for deep convolutional neural networks. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp 11534–11542
 186. Zhang Q-L, Yang Y-B (2021) SA-Net: shuffle attention for deep convolutional neural networks. In: *ICASSP 2021—2021 IEEE international conference on acoustics, speech and signal processing (ICASSP)*. IEEE, pp 2235–2239
 187. Woo S, Park J, Lee J-Y, Kweon IS (2018) CBAM: convolutional block attention module. In: *Proceedings of the European conference on computer vision (ECCV)*, pp 3–19
 188. Cao, Y., Xu, J., Lin, S., Wei, F., & Hu, H (2019) GCNet: non-local networks meet squeeze-excitation networks and beyond. In: *Proceedings of the IEEE/CVF international conference on computer vision*, pp 1971–1980
 189. Li X, Hu X, Yang J (2019) Spatial group-wise enhance: improving semantic feature learning in convolutional networks. arXiv Preprint. <http://arxiv.org/abs/1905.09646>
 190. Mehta S, Rastegari M (2021) MobileViT: light-weight, general-purpose, and mobile-friendly vision transformer. arXiv Preprint. <http://arxiv.org/abs/2110.02178>
 191. Wu H, Xiao B, Codella N, Liu M, Dai X, Yuan L, Zhang L (2021) CvT: introducing convolutions to vision transformers. In: *Proceedings of the IEEE/CVF international conference on computer vision*, pp 22–31
 192. Maaz M, Shaker A, Cholakkal H, Khan S, Zamir SW, Anwer RM, Khan FS (2022) EdgeNext: efficiently amalgamated CNN-transformer architecture for mobile vision applications. CoRR. <https://arxiv.org/abs/2206.10589>
 193. Chen Y, Dai X, Chen D, Liu M, Dong X, Yuan L, Liu Z (2022) Mobile-former: bridging MobileNet and transformer. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp 5270–5279
 194. Zhang J, Li X, Li J, Liu L, Xue Z, Zhang B, Jiang Z, Huang T, Wang Y, Wang C (2023) Rethinking mobile block for efficient neural models. arXiv Preprint. <http://arxiv.org/abs/2301.01146>

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the

author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.