Xu *et al. Journal of Cloud Computing* (2022) 11:54
https://doi.org/10.1186/s13677-022-00330-5

Journal of Cloud Computing:
Advances, Systems and Applications

RESEARCH

Open Access

RDPNet: a single-path lightweight CNN with re-parameterization for CPU-type edge devices

Jiarui Xu, Yufeng Zhao* and Fei Xu

Abstract

Deep convolutional neural networks have produced excellent results when utilized for image classification tasks, and they are being applied in a growing number of contexts. Model inference on edge devices is challenging due to the unending complicated structures needed to improve performance, which adds a significant computing burden. According to recent research, the often utilized residual structure in models does not support model inference. The idea of structural reparameterization is put out to address this shortcoming. The RepVGG produced with this method is a high-performance, quick-inference single-path network. Even after reparameterization, the model still needs GPUs and other specialized computing libraries to accelerate inference, however this still has a limit on how quickly the model can infer at the edge. We construct RDPNet using depthwise separable convolution and structural reparameterization to further reduce model size and accelerate inference. When utilizing an Intel CPU, this is a straightforward network that may be utilized for inference. For re-parameterization, we specifically adopt Depthwise separable convolution as the basic convolution form. Create a multi-branch model for training on the training side, and then simplify it into a single-branch model that the edge devices can easily infer. Research demonstrates that compared to alternative lightweight networks that can attain SOTA performance, RDPNet offers a superior trade-off between accuracy and latency.

Keywords: Structural re-parameterization, Depthwise separable convolution, Lightweight networks, Edge devices

Introduction

In recent years, the development of CNN research is obvious to all, and very significant results have been achieved in the fields of image classification. Provides a solid foundation for autonomous driving, object recognition and object tracking scenarios. With the development of cloud-edge computing, these tasks are becoming more and more perfect in real-world applications. In this context, deploying CNN models to the edge to meet the needs of various cloud-edge computing scenarios has become a research issue that must be considered. Among these, the creation of the model architecture is the cornerstone and essential stage for advancing CNNs. Early designs that consist of stacks of single convolutional layers, like [1] and [2], perform well on classification tasks, sometimes even outperforming human recognition levels. Complex structures like ResNet [3], the Inception [4–7], and [8] were proposed in order to improve the model's performance, and these further improved outcomes in classification tasks.

On the other side, as a model performs better, computational complexity and network model parameters likewise rise. The conventional method involves sending massive amounts of edge-generated data to cloud computing centers for processing, and then sending the findings back to the edge in order to make complicated networks usable in a larger range of application situations. This method cannot meet latency requirements

---

*Correspondence: zyfzy99@163.com

Computer Science and Engineering, Xi'an Technological University, No. 2 Xuefu Middle Road, 710021 Xi'an, China

and consumes a lot of bandwidth and energy. The majority of edge technology leverages the CPU to do complex tasks and remain efficient, taking into account cost, latency, and the type of work that needs to be done. Model inference becomes difficult as a result of the direct deployment of computationally intensive models to the edge.

This issue has been addressed recently by lowering the model memory occupation and speeding up inference. such as knowledge distillation [9, 10], network pruning [11, 12], and quantization [13, 14]. These techniques to compress data are based on current models. The performance of the model is limited to the pre-trained Baseline, and the compressed model performs worse than the starting network. With fewer parameters and computations, compact network designs [15–19] achieve acceptable accuracy. By using depthwise separable convolution (DSC) in place of the conventional convolutional form, the model can be made lighter. The multi-branch ConvNet layered by Bottleneck structure is the key to achieving good performance with the proposed residual structure. The connection of residual structures, however, is beneficial for model inference but not for model training. These issues also exist in the lightweight neural network used in e.g. [18] and its extension using the residual structure. To decrease the amount of memory needed for inference, [20, 21] suggested converting the multi-branch model into a straight-pipe structure. The model nevertheless needs specialized software libraries and hardware for acceleration.

Our work in this study primarily focuses on designing the underlying convolutional blocks by hand in order to construct a lightweight CNN. We contend, in the spirit of RepVGG, that the form of DSC that results from reparameterization enables the model to preserve accurate inference while maintaining efficiency. Howard et al. [17] and Chollet [22] suggested DW for feature extraction of spatially correlated information and PW for channel information integration in relation to DSC. According to MobileNetV2, the latter PW is more important because it incorporates channel correlation during feature extraction, which is crucial for block feature extraction. The similar point of view is presented with several model modifications in [23, 24]. We also aware of how important PW is for feature extraction while developing a multi-branch Module. So, we made the decision to exclude the Identity mapping branch from PW. Additionally, group convolutions are used in place of the convolutional forms of other branches in the DW multi-branch architecture to match the spatially correlated extraction properties of DSC.

These characteristics guide the construction of a new DSC-based multi-branch module, which is then re-parameterized into a single-path model for inference. By stacking modules with the proper depth and width requirements, we create the RDPNet. According to experimental findings, RDPNet excels at building lightweight neural networks and makes excellent trade-offs while performing image classification and object recognition tasks.

The main contributions of this paper are as follows:

(1) We build a multi-branch structure based on the depthwise separable convolution in the form of convolution, and through the process of re-parameterization, we make it into a single-branch structure that makes inference by edge devices easier.
(2) We suggest RPDNet, a thin neural network appropriate for portable embedded devices. In comparison to other networks that are capable of achieving SOTA performance, experiments reveal that RDPNet offers a better trade-off between accuracy and delay, and the model implementation is both easier to use and more efficient.

The remainder of this essay is organized as follows: Section 2 reviews relevant work in brief. In the Section 3, we first analyze the impact of different model architectures on inference. Then, the construction and transformation of RDPNet-Module and the overall design of the model are introduced. Presentation of experimental findings and discussion (Section 4). The entire text is summarized in Section 5, which also offers a look ahead to upcoming work.

Related work

Compact network and multi-branch model

On the ImageNet dataset, the single-path network [2] obtains an accuracy of 70%, however the performance of the model is constrained by gradient vanishing and accuracy loss brought on by the stacking of deep convolutional layers. A substantial amount of computation is also added by the stacked single-path convolutional layer network. Some network lightweight strategies have been presented in order to expedite training and save computation. To minimize the dimension, [25] suggests adding 1×1 convolution before 3×3 convolution. To put a limit on model parameters, [17] and [22] suggest DSC.

The network structure was then significantly altered by the ResNet [3] proposal, and the residual structure was adopted as the core model for feature extraction networks. The network employing the residual structure is referred to in this paper as a multi-branch network since there is a view [26] that the residual structure is an integration of many shallow networks. According to a different perspective [27], the residual structure facilitates information transfer between layers, allows for the reuse of features during forward propagation, and solves the gradient disappearance issue during back propagation.

In a nutshell, the addition of Identity Mapping increases the independence of each convolutional block, better alleviates gradient disappearance during training, and prevents model degradation during stacking. Sandler et al. [18] proposes the inverse residual structure by combining deep separable convolution and residual structure. Zhang et al. [15]and Ma et al. [16] uses channel shuffling and group convolution to optimize on this basis. Subsequent studies have proposed more detailed methods for inverting residual structures. Han et al. [24] uses feature multiplexing to improve the inverted residual structure so that the model has fewer parameters. Yang et al. [23] observes that the functions of the two PW parts are different, and adjusting the convolution dimension of the two PWs obtains better results. Additionally, [28] reduces connectivity between convolutional layers to simplify operations and adds more intricate nonlinear functions to make up for the loss of network depth. Zhou et al. [29] makes inferences about the depth axis redundancy of DSC based on a quantitative analysis of convolution kernels. It swaps the original weight for a predetermined weight to simplify calculations.

Despite the fact that these recently presented models have few FLOPs, memory footprint and inference performance are increased by several computation-saving techniques and residual structures. At the same time, the model is challenging to customize and execute due to its complex multi-branch structure, and pruning and quantization do not work well with it.

Some study areas have changed from manually creating neural networks to structural systems that adaptively execute systematic search for particular tasks as a result of the emergence of GPUs. A number of networks [30–33] have been presented in a search area comparable to ResNet/MobileNetV2. However, [32] and manually configuring the search space demand a significant amount of computational power, and the network model discovered by searching requires a lot of hardware. The best solution can be found by searching the body of existing information, and even better is to determine the best model architecture for the model that already exists.

Structural re-parameterization

In order to speed up the training process, DiracNet [34] suggested a Block that had the re-presentation $y = \sigma(x + f(x))$ and changed the network into a single-branch structure based on the linear relationship between the convolutional and BN layers. However, due to nonlinearity's limitations, the performance of the DiracNet model developed using ResNet structure is constrained.

In contrast to [20, 21], which indirectly generates a three-branch structure for training, this method enables the model to perform better. Ding et al. [21] suggests that the model be trained to have a complicated multi-branch structure to accommodate different network requirements. This structure can then be re-parameterized into a 3×3 convolution-based network.Operations similar to converting multiple convolution blocks into a single block through linear transformation can be understood as reparameterization operations [35–37]. The purpose of linear transformation is to reduce the amount of parameters and increase its inference efficiency. Structural re-parameterization has the benefit of allowing the model to be optimally trained using complicated network types for improved model performance. The altered canonical network allows for improved quantization, pruning, and the computation of the library's accelerated operators. Accordingly, the flaw is that the 3×3 convolution model after conversion still requires a significant amount of computation for edge devices, necessitating additional compression processing. We suggest RDPNet, which was inspired by the re-parameterization approach. Create a DSC structure with a single path for inference and a complex multi-branch structure for training.

RDPNet

In this section, We compare the inference effects of common network structures under different hardware, then we examine the benefits of adopting the straight-tube topology to deploy inference on edge devices. The network structure is then described using a combination of structural re-parameterization and depthwise separable convolution. A lightweight model RDPNet is built using the existing overall network architecture as a reference.

Problem description

Multi-path branching

An example of a single-path ensemble is ResNet. Identity Mapping, which is more suited to the training and depth of the model, is used by the network to implement the deep network during the training phase. Despite being jointly trained, there isn't much of a correlation between the blocks in the network. The key is that the model's capacity to better match data is made possible by the efficient multi-branch structure. The residual structure, which enables the model to delay the disappearance of gradients during training, is usually thought to be the cause of ResNet's exceptional performance.

Figure 1 depicts various multi-branch network architectures in their extended form. When the number of input and output channels of Block is the same, the

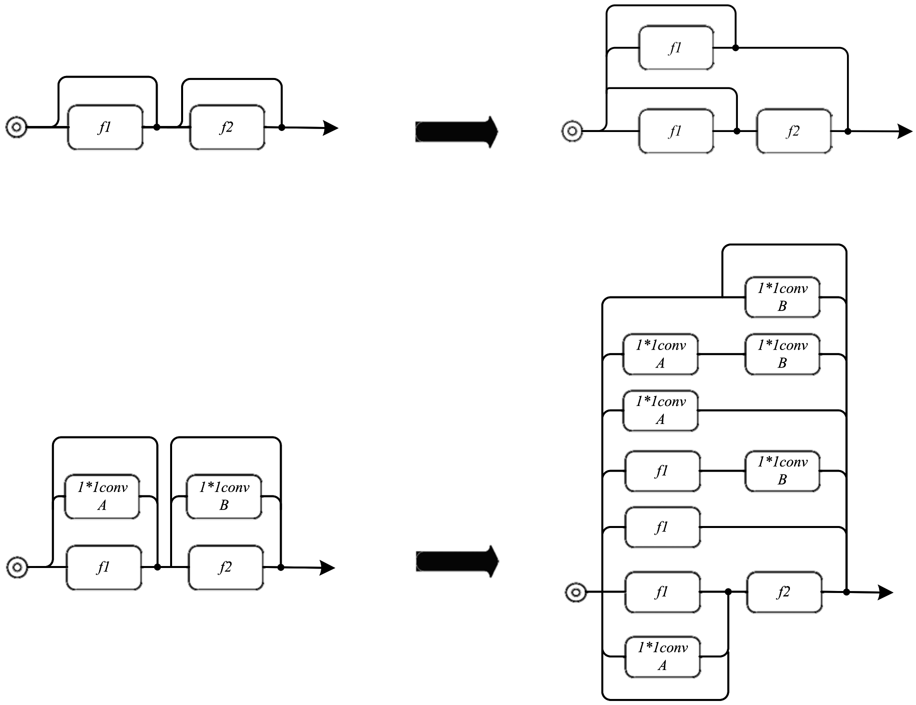

Fig. 1 Multi-branch structure expansion diagram, multi branch network of two Residuals Blocks(top), multi branch of two Repvgg Blocks(bottom)

difference from ResNet is that the training model of RepVGG adds a 1×1 convolutional branch to expand the path. In the experimental section of [20], ablation experiments are performed on multi-way branches, which demonstrate that the logical addition of branches to the model is one of the factors contributing to its good performance. Reusing features and adding numerous routes to the model is a solid solution to improve model performance and reduce the number of parameters [8]. Model delay increases when more complicated models are used. By using linear transformations, structural reparameterization has the benefit of reducing computation. The model's study from the aforementioned viewpoint also demonstrates that a suitably complex structure can actually improve the model's performance. Even though the model's solution space remained the same after reparameterization, the solution path was improved. The side shows that an effective optimization method is one of the keys to improving the model

Hardware computing mode

The CPU's computing technique is distinguished by a variety of complex computing models and a limited number of single calculations. The benefit is that it can handle many computing tasks of varying complexity and computing kinds simultaneously, making it appropriate for situations in which edge hardware is present. GPU computing is characterized by simple computation types, few repetitions, and a significant number of single calculations. When processing massive volumes of data, multi-memory parallel computing is a computing technique that is necessary.

The GPU of the cloud computing facility accelerates the VGG-like model created by stacking three conventional convolutions three times. The key to enabling the model to carry out a high number of tensor-type operations on the training side is the large number of arithmetic logic units (ALUs) that exist on the GPU. Large-scale tensor-type calculations cannot be supported by the edge CPU due to a lack of ALUs and cache settings. As a result, the deep learning model's inference is constrained at the edge and it is unable to satisfy the latency criteria necessary in some cases (Table 1).

The structure of the model has a direct impact on the inference speed at the CPU's edge. The experimental gear consists of an Nvidia RTX3060ti GPU, an Intel 9400f CPU, a batch size of 1, a feature map size of 224× 224, and networks with black markers that use DSC. We discovered that models employing DSC ran on CPU substantially more quickly. A number of factors, including transporting hardware, computing libraries, and underlying computing frameworks, must be taken into account in real-world settings. These elements will affect model inference more significantly.

Table 1 The inference latency of different types of network structures on GPU and CPU

Model	Param(M)	FLOPs(G)	CUDA(ms)	CPU(ms)
VGG16	138	15.5	467.0	122.78
Resnet18	11.7	1.8	22.29	65.58
Resnet50	25.6	4.1	85.55	107.67
MobilenetV3-small	3.6	0.1	20.31	30.43
Mobilenetv2-1x	3.5	0.3	38.29	45.62
Shufflunetv2-1.5x	3.5	0.3	28.31	36.05
RepVGG-A0	7.0	1.4	4.83	52.34
RDPNet-1.0x	1.3	0.3	12.45	23.14

Consequences of structural changes

When the model is inferred, there is a distinct distinction between the multi-branch structure and the straight-tube form. To achieve the goal of accelerating the computation on the CPU for the analysis in the preceding section, we built the model as a computing mode with low memory footprint for a single calculation.

ShffleNetv2 [16] makes the argument that even while some multi-branch models have minimal FLOPs, this statistic is insufficient to assess how well the model performs when making inferences at the edge. The effect of each operator in the network model on the latency during inference is the unique performance that needs to be taken into account. For instance, although though the residual structure's shortcut uses a little amount of FLOPs, it has a high MAC. The model's inference speed increases by 20 % after the shortcut is eliminated. As a result, we came to the conclusion that the multi-branch model does not enable model inference. On an Intel-CPU, we tested the speed of the straight-tube construction and the multi-branch structure. According to experimental findings, the straight-tube structure's inference speed is noticeably quicker than the residual structure with numerous convolutions.

Figure 2 displays the specific outcomes. We stack the model from 4 to 25 layers, with conv_0 denoting straight structure, conv_2 denoting residual structure containing two convolutional layers, and conv_3 denoting three convolutional layers.

Summary of the problem

From the theoretical analysis of the multi-branch structure and the speed test results of the presence or absence of depthwise separable convolution, we summarize some points. (1) The convolution form based on DSC greatly reduces the amount of model parameters and is more conducive to reasoning on CPU devices. (2) Although the multi-branch model can improve the model performance, it is not conducive to model inference. (3) It is not difficult to see from the model speed measurement that the computing library and hardware conditions have a great influence on the model inference. Based on the observations in this section, we design a lightweight

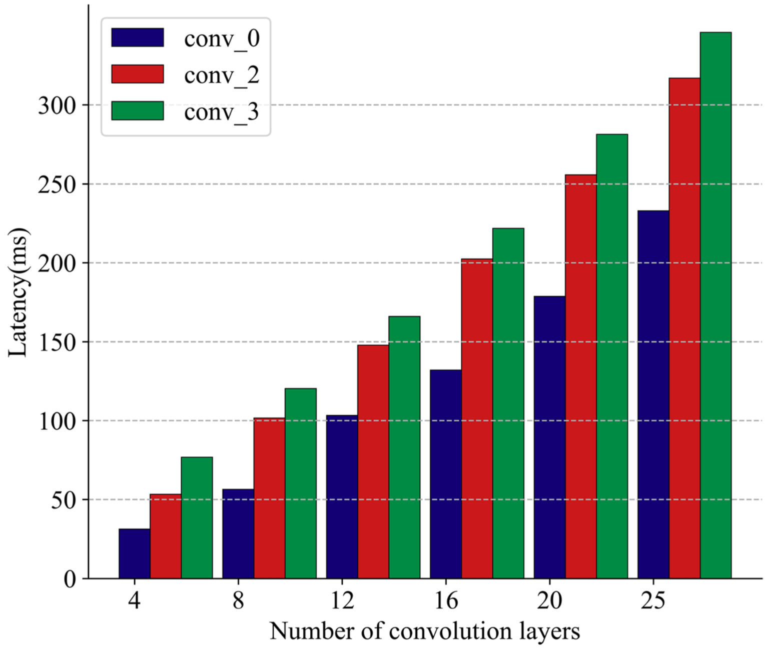

Fig. 2 Comparison of inference delay between residual structure and straight structure

network for the usage characteristics of the model on CPU devices, and deploy it on embedded devices without the assistance of specific computing libraries for accelerate.

RDPNet-Module

Depthwise separable convolution

A number of lightweight networks have been derived from DSC, which is a crucial link in lightweight networks and whose features are commonly used in lightweight networks. This technique is used in this paper, and unlike conventional convolution, DSC splits conventional convolution into two pieces. Pointwise convolution(PW) is the process of combining the features to create a new dimension feature vector whereas depthwise convolution(DW) is the process of filtering the input feature vector. The model's computation and parameters are drastically decreased after such change. Pointwise convolution is the process of combining the features to create a new dimension feature vector whereas depthwise convolution is the process of filtering the input feature vector. The model's computation and parameters count are drastically decreased after such change.

DSC works well since the model requires less work to compute because to its computational form. More nonlinearity is added after the convolution is divided into two to make up for the accuracy loss brought on by the point-by-point convolution's reduction of feature correlation. The HS activation function took the place of the RELU activation function in MobileNetv3 [30], which enhanced the model's performance. Additionally, more research is being done on the correlation between DW channels, for example, employing channel shuffling to boost the correlation between channels [15, 16]. We discovered that the network employing DSC has a large speedup on CPU in contrast to the speed test performance on GPU. We think that this is because the depthwise convolution, which increases the computational parallelism of the model, groups and convolves the feature maps. The number of parameters and overall processing needed for a single convolution operation are decreased by breaking the regular convolution into two sections. This way, it is compatible with the CPU's low-computation and multi-batch operating modes, hitting additional processing RAM to meet the goal of accelerating inference.

Conventional transformation

The inference capabilities of the multi-branch model and the straight-tube model at the edge were examined in the preceding section. The multi-branch structure has an effect on the model inference even though it can help the model reach higher accuracy. We build and alter the model using the structural re-parameterization method to keep the model's high performance and high-speed inference capability.

Lossless model transformation using linear principles is possible through structural re-parameterization in convolutional and network architectures. The network can be trained in a multi-branch structure on the training side thanks to structural reparameterization. The multi-path convolution layer and BN layer have a linear transformation relationship that causes the complex multi-path structure to be losslessly converted into a single-path structure. Although the solution space of the model has remained the same, the solution path has been extended and optimized in light of the function fitting data. This makes it simpler for the model to improve its accuracy while being trained.

Citing the formula described in [20]. x represents the branch path number. When $C_{\text{in}} = C_{\text{out}}$, $\left\{\alpha^x, \beta^x, \gamma^x, \delta^x\right\}$ represents the mean value, standard deviation and learning scaling factor obtained by the BN layer in the 3×3 convolutional layer, 1×1 branch, and Identity branch, respectively and bias term. In inference mode, the convolution-BN layers are directly combined into one convolutional layer. i is represented as the current number of channels, $\forall 1 \leq i \leq C_{\text{out}}$, for BN in different branches:

$$BN\left(M, \alpha^x, \beta^x, \gamma^x, \delta^x\right)_i = (M_i - \alpha_i)\frac{\gamma_i}{\beta_i} + \delta_i \tag{1}$$

If W^{Δ} is represented as the weight value of reparameterization, and b^{Δ} represents the bias term after reparameterization, the output of the above BN layer can be converted into:

$$BN(M)_i = \frac{\gamma_i}{\beta_i} W_i - \frac{\alpha_i \gamma_i}{\beta_i} \delta_i = W_i^{\Delta} + b_i^{\Delta} \tag{2}$$

Add the bias terms in each channel to obtain the final bias term, and denote the modified weights and bias terms as W and b, respectively. The final convolution kernel is created by filling and superimposing the 3×3 convolution kernel with the two 1×1 convolutions, which reduces the multi-branch path to a single branch.

Block building

Reparameterization is to use a set of parameters to represent multiple sets of parameters according to the linear relationship between the convolution and the BN layer to simplify the operation. We build a special DSC module based on multi-branch paths. The training-end model, which is primarily made up of Depth-Block and Point-Block, is displayed in Fig. 3a.

We transform the complex multiplex structure into a DSC form that is optimized for CPU, drawing inspiration from the model transformations in RepVGG and DBB.

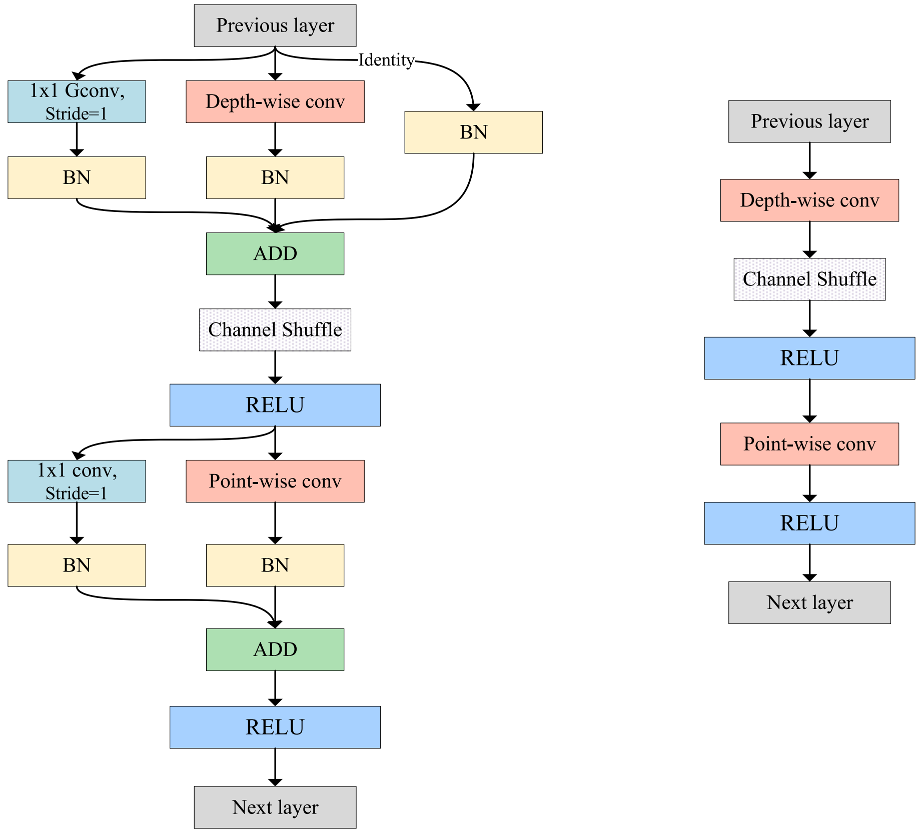

Fig. 3 RDPNet multi-branch training model and single-path inference model, multi branch structure(left), single branch structure(right)

We employ the same multiplex construction and reparameterization in the Depth-block as in the RepVGG-Block. Convolution is used to group the input feature map at Depth-Block. In Point-Block, an 1×1 convolutional two-path branch is built.It should be noticed that the Point-Block module component is where the RDPNet feature map downsampling is completed. Our proposed RDPNet module is shown in Fig. 3a represents a multi-branch training model, (b) represents a single-path inference model.The information flow of the Depth-Block module in dimension matching is $y = x + f_1(x) + g_1(x)$, where $f_1(x)$ is the depthwise convolution, $g_1(x)$ is the 1×1 branch group convolution, and x is the Identity Mapping. The representation of the Point-Block module is $y = f_2(x) + g_2(x)$, where $f_2(x)$ is pointwise convolution and $g_2(x)$ is 1×1 branch convolution.

When adding Identity Mapping to the Point-Block, the model's performance will suffer. Additionally, according to [22] and [18], we think that DW and PW separately finish various stages of the convolution and are in charge of various feature extractions. The absence of channel information was caused by the addition of Identity Mapping. In the residual structure, it is more precisely represented as $y = x + f(x)$ in Identity Mapping. The issues of disappearing gradients and compressing low-dimensional data are lessened when $f(x)$ is minimal since the lower layer network roughly obtains the output of the higher layer x. In the case of creating multi-way branches and matching dimensions, the function of point-wise is to join features to create feature vectors of additional dimensions. The model will lose the channel information retrieved in PW if x is significantly greater than $f(x)$.

After extracting the depth features from the Depth-Block in accordance with the specifications of DSC, we introduced the Channel shuffle module to increase the information linkage between channels.The outcomes of the ablation trials demonstrate how much Channel shuffle enhances the model's performance. We can perform a lossless transformation of the training end model thanks to the linear relationship between the training end and the deployment end. Figure 4 displays the Module deployment model (b).

Depth-block transform

We re-parameterize Depth-Block based on standard transforms in this part. Depth convolution is a

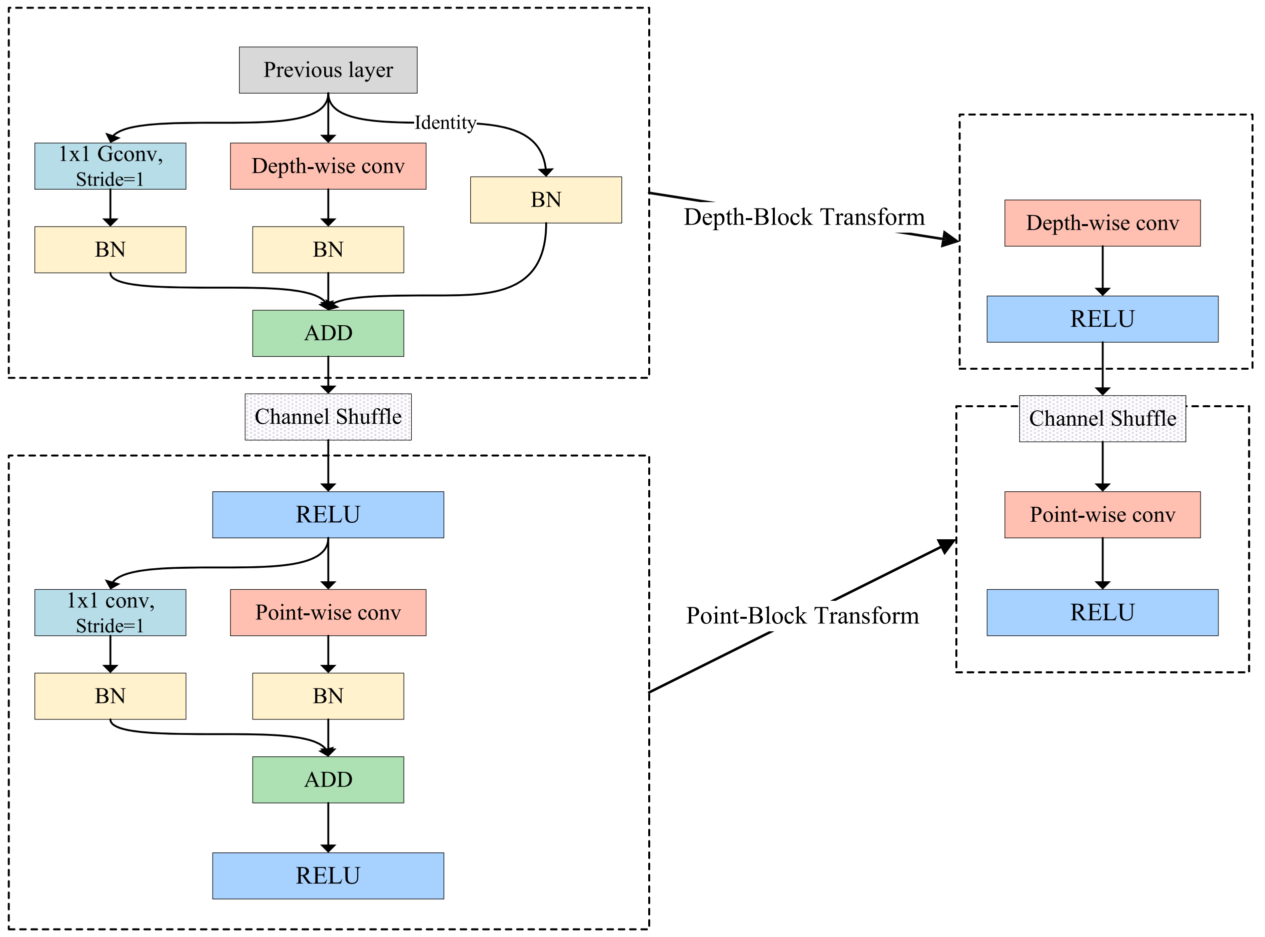

Fig. 4 RDPNet multi-branch training model and single-path inference model

convolution technique that produces a single feature map from a single channel of data using a single convolution kernel. The feature map produced in this manner matches the quantity of input channels. According to Eq.(2) we can re-parameterize the weights within each channel. where x is the input, W^* is the reparameterized convolution kernel, b^* is the reparameterized bias term, and the number of convolution channels is z. $\forall 1 \le i \le z$, the output form M of convolution BN can be written as:

$$M_i = \left(X \otimes W_i^* + b_i^*\right) \tag{3}$$

Combine each channel as:

$$M = \text{Concat}(M_1, \ldots, M_i) \tag{4}$$

Because depthwise convolutions are performed as grouped convolutions, branched 1×1 convolutions are also constructed in the same way. The convolution process in each channel is reparameterized and concatenated using standard transformation rules.

Point-block transform

This Transform component is just a straightforward linear addition. The acquired convolution kernel weights and bias terms are, of course, $\left\{W^1, W^2, b^1, b^2\right\}$ through standard transformation. W^* is the reparameterized convolution kernel, b^* is the reparameterized bias term. Our reparameterization of two 1×1 convolutions is handled as follows:

$$W^* = W^1 \oplus W^2 \tag{5}$$

$$b^* = b^1 + b^2 \tag{6}$$

The Block built by the Module we built and [23, 38] differ from one another. The latter, which is helpful for computing libraries to accelerate and enhance accuracy, linearly transforms irregular convolution blocks into regular convolution blocks. A multi-path complex model is transformed into a thin, single-path lightweight model using our module, which is useful for CPU inference. Our

Module transforms a complex multi-path model into a simple single-path lightweight model that is suitable for CPU inference.

When building the model, we followed the following guidelines. (1) Because the purpose of building the model is to reduce weight and satisfy the calculation method of CPU. The construction of the deployment-side model is related to the computing method of edge hardware. (2) The model construction on the training side is related to the performance of the model. The model on the training side is more similar to the optimization of the model, which refers to the multi-channel construction of the classical network. Therefore, we improve the model based on the characteristics of DSC. (3) The structural re-parameter is the transformation relationship between linear operations such as the corresponding convolution and BN layers. It is necessary to pay attention to the model correspondence between train-time and deploy-time.The schematic diagram of the reparameterization transformation is shown in Fig. 4.

Model architecture

RDPNet's overall architecture is depicted in Table 2 along with changes for various widths. RDPNet differs from other topologies in that the model, after reparameterization, comprises of DSC and nonlinear modules. This makes it possible to use the model for edge-based rapid inference more effectively. To downsample the feature map, we divided the model into 5 Stages, each with a Stride of 2. In order to output the classification results when dealing with image classification jobs, a global average pooling layer and a fully connected layer are added after Stage 4.

The general architecture that we created, with its [1, 1, 4, 14, 1] layer count, is comparable to that created by RepVGG and ResNet. Focus the main feature extraction work on the stage where the feature map size is 14×14. We initially utilize a one-layer convolutional RepVGG-Block for feature extraction, since the feature maps that need to be processed at the beginning of the model are larger. At the same time, an appropriate number of channels are employed to ensure model performance in fully linked layers to preserve more features.

Table 2 RDPNet series architecture

Stage	Operator	Stride	Output size	Channel width
Stage-0	RepVGGBlock	2	112 × 112	$1 \times min(64, 64\alpha)$
Stage-1	RDP-Module	2	56 × 56	$1 \times 64\alpha$
Stage-2	RDP-Module	2	28 × 28	$4 \times 128\alpha$
Stage-3	RDP-Module	2	14 × 14	$14 \times 256\alpha$
Stage-4	RDP-Module	2	7 × 7	$1 \times 512\alpha$

For the width of the model, the paradigm of ResNet and VGG is followed, set to [64, 128, 256, 512], and the width is scaled by α. The initial value of α is [1, 1, 1, 2.5], and the scales are [0.25, 0.5, 0.75, 1, 1.25, 1.5] respectively, and 6 sizes of networks are constructed. Used as a benchmark against thin networks and scalability schemes like the MobileNet and ShuffleNet series.

We use the basic architectural form to build the model, which is designed to meet the inference requirements of edge devices. From the perspective of model architecture, the main content of this paper is not to focus on the effect that the model can be stacked with more convolutional layers, but to focus on the performance and latency of the model. This enables efficient inference of models on edge devices.

Experiment

We test the model's effectiveness on image classification tasks on CIFAR10/100 [39] and ImageNet [40] using RDPNet. As the Backbone, we used the PASCAL2007 [38] dataset to validate the model's performance on downstream tasks. When compared to other models, the deploy-time model's inference speed and performance are evaluated. The image classification experiments in this paper are based on Nvidia RTX3060Ti, Intel-CPU 9400f, memory 16GB, and the number of threads is 8. The deep learning framework used is pytorch-1.7, and the API used for model delay speed measurement is torch-profiler.

Image classification

Cifar10/100 dataset

Cifar10/100 consist of 50k training set data and 10k test set data respectively. The size of the picture is 32×32, and it is divided into 10 categories and 100 categories. We set the number of iterations to 200 epochs during training, used an SGD optimizer with a momentum of 0.9, a weight decay of 0.0001, and an initial learning rate of 0.1. The learning rate decay strategy is Cosine Annealing Method. Only random horizontal flipping of images is used for image enhancement.

We swap out the Stride in the first two Stages for 1 to ensure that the final output feature map size is 4×4. This is done to adapt to the feature map size. With this change, we are able to obtain the RDPNet series' results for the CIFAR10/100 dataset. With a 32×32 input image size, we ran speed tests and performed inference of the network model on the CPU. The latency value represents the amount of time needed to analyze an image. We averaged the outcomes of 500 time-lapse tests in order to remove natural errors. The outcomes are displayed in

Table 3 Table caption

Model	Param(M)	FLOPs(M)	CIFAR10 Acc.(%)	CIFAR100 Acc.(%)	Latency(ms)
RDPNet-0.25x	0.2	6.7	89.7	68.4	11.79
RPDNet-0.5x	0.46	21.9	92.0	70.5	13.0
RPDNet-0.75x	0.8	45.9	92.4	72.3	13.75
RDPNet-1.0x	1.3	78.5	93.7	73.8	14.67
RDPNet-1.25x	1.9	117.7	94.5	75.1	16.76
RDPNet-1.5x	2.6	164.3	94.7	75.6	17.36

Table 4 Comparison of RDPNet and MobileNet series on CIFAR10/100

Model	Param(M)	FLOPs(M)	CIFAR10 Acc.(%)	CIFAR100 Acc.(%)	Latency(ms)
MobileNetV1-0.25x	0.24	13.01	**90.4**	68.23	14.18
MobileNetV2-0.25x	0.25	8.97	89.9	67.5	25.60
MobileNetV3-small-0.35	0.31	3.45	88.8	66.5	21.48
RDPNet-0.25x	0.20	6.70	89.7	**68.4**	**11.79**
MobileNetV1-0.5x	0.82	47.21	91.8	70.8	17.59
MobileNetV2-0.5x	0.70	27.32	92.0	72.5	27.21
MobileNetV3-small-0.75x	0.72	11.40	92.0	70.4	23.68
RDPNet-0.75x	0.80	45.9	**92.4**	**72.3**	**13.75**
MobileNetV1-0.75x	1.82	102.66	92.7	72.2	18.71
MobileNetV2-0.75x	1.63	55.13	93.1	73.2	37.13
MobileNetV3-large-0.75x	1.73	40.67	93.7	73.9	33.54
RDPNet-1.25x	1.86	117.7	**94.5**	**75.1**	**16.76**
MobileNetV1-1.0x	3.22	179.34	93.4	73.4	23.36
MobileNetV2-1.0x	2.24	92.40	93.6	74.9	42.23
MobileNetV3-large-1.0x	2.98	68.45	93.7	75.2	35.90
RDPNet-1.5x	2.60	164.3	**94.7**	**75.6**	**17.36**

Table 3 after we scale the network width to create models with 6 specifications and validate them.

The findings indicate that in terms of accuracy and CPU inference speed, the RDPNet series of models perform significantly better than the MobileNet series of networks. Between them, RDPNet-1.25x's accuracy on the CIFAR dataset is 94.5 and 75.1 percent, respectively. The inference speed surpasses the performance of the current network by a wide margin. The trade-off between accuracy and inference speed is very good. Table 4 contains information.

ImageNet dataset

We validate some models on the ImageNet dataset in order to assess the classification performance of the model on a sizable dataset. For testing, this dataset includes 1.3 million and 50,000 images from 1000 different object categories. RDPNet-1x and RDPNet-1.25x were our choices. The model architecture we employed matches that in Table 2 in every way. With a momentum of 0.9, the SGD optimizer was employed. We use a weight decay of 0.0001 to reduce overfitting. The learning rate decay rate depends on the batch size and epoch size, and the learning rate adjustment strategy uses the cosine annealing algorithm. Only

Table 5 The performance of some RDPNet models on the ImageNet dataset

Model	Top-1 Acc.(%)	Top-5 Acc.(%)	Latency(ms)
MobileNetV2-0.5x	65.03	85.72	38.52
MobileNetV3-large-0.35x	64.78	83.56	23.83
ShuffleNetV2-0.5x	61.32	83.73	25.58
RDPNet-0.75x	**67.23**	**88.93**	**18.58**
MobileNetV1-1x	70.99	89.68	40.76
MobileNetV2-1x	72.15	90.65	46.62
MobileNetV3-small-1.25x	70.67	89.51	37.08
ShuffleNetV2-1.5x	71.63	90.15	36.76
RDPNet-1.25x	**73.65**	**91.82**	**23.60**

Table 6 On the PASCAL2007 dataset, RDPNet's performance as the backbone is contrasted with that of other models

Method	Backbone	mAP(%)	Latency(ms)
Faster-RCNN	MobileNetV3-large-0.35x	32.17	57.62
	RDPNet-0.75x	**31.67**	**43.62**
	MobileNetV3-small-1.25x	21.05	65.34
	RDPNet-1.25x	21.36	**55.39**

upscaling and horizontal flipping are employed for image enhancement. For comparison, we chose FLOPs and Param models that are similar to the RDPNet family. The CPU device is the foundation for the inference latency test. Wait test The average value of ten delay tests was chosen to prevent human error. The outcomes are displayed in Table 5.

We make use of a list of networks with SOTA performance. Accuracy, computational complexity, and delayed inference are all thoroughly assessed. The performance on the ImageNet dataset is top-1 accuracy. The latency test uses an Intel-CPU i5-9400f for an average of 500 inferences.

Object detection on PASCAL

On edge devices based on CPU type, we test the model's performance on downstream tasks. On the Faster-RCNN [41] object detection framework, we use RDPNet-1.25x as a Backbone. For validation, we made use of the PASCAL2007 [38] dataset. we applied pretrained weights from ImageNet dataset. We only alter Backbone's latency for CPU-based object detection tasks when it comes to the object detection framework. Table 6 shows its comparative effect as a backbone and the same type of network

Semantic segmentation

We use RDPNet as the backbone to conduct semantic segmentation experiments on the PASCAL2012 dataset [38]. Unlike image classification tasks, semantic segmentation models are generally divided into two parts: one part is used for classification, and the other part is used to segment regions. Our framework of choice is the DeeplabV3plus [42, 43] model, with model training on an Nvidia Tesla K80. We imported a pretrained model on the ImageNet dataset for comparison with MobileNetV3 as the backbone model. The SGD optimizer is used when training the model, the momentum is 0.9, the initial learning rate is 0.01, and the latency test is based on Intel CPU-9400f.The results are shown in Table 7. The semantic segmentation framework with RDPNet as the backbone performs better in performance and latency.

Table 7 Semantic segmentation experiments on the PASCAL2012 dataset

Method	Backbone	mIoU(%)	Latency(s)
DeeplabV3+	MobileNetv2-0.5x	0.708	0.317
	MobileNetV3-large-0.35x	0.693	0.292
	RDPNet-0.75x	**0.721**	**0.237**
	MobileNetV2-1.0x	0.711	0.379
	MobileNetV3-small-1.25x	0.768	0.350
	RDPNet-1.25x	**0.782**	**0.276**

Table 8 Ablation Study for Identity and Channel shuffle

No Identity for Point-wise	Channel shuffle	Accuracy(%)	Latency(ms)
		70.6	12.43
√		71.3	12.40
	√	72.5	13.54
√	√	73.8	14.67

Ablation study

We discovered that the channel features extracted by PW have a significant impact on the network's overall performance during the construction of the multi-branch model. Additionally, I used Channel shuffle to interactively process the data between channels, which enhanced the model's general performance. Based on this, we ran ablation experiments on these two components to confirm their role in the enhancement of network performance as a whole. For validation on CIFAR100, we used RDPNet-1.0x with a batch-size selection of 128 and an input image size of 32×32. According to experiments, alterations to these two components have almost no impact on the CPU's network inference delay (Table 8).

Limitations

With less emphasis on model depth and convergence efficiency, the aim of this paper is to establish a network model for effective inference at the edge for CPU computing. At the edge, there are various types of hardware, and each model has a different set of requirements. Simply changing the model's width and depth has a significant impact on it and might not produce the desired results. More precise pruning and quantization are required, as well as cost-effective model size adjustment. The framework found by NAS technology may be referenced in the network architecture construction, which can further enhance the model performance.

Conclusion

We propose the RDPNet, a DSC and RELU-based network with a straight-tube structure that is suitable for CPU operation. The structural re-parameterization method is used, which takes into account the characteristics of CPU calculation. Transform highly accurate complex models into simple one-way models for CPU inference. In terms of accuracy and latency, RDPNet achieves a better trade-off when compared to other networks that achieve SOTA performance. Model implementation is simpler and more efficient.

Abbreviations

DSC: Depthwise separable convolution; PW: Point-wise; DW: Depth-wise.

Acknowledgements

We would like to thank the authors of the literature cited in this article for providing helpful ideas for this study. The authors would like to thank anonymous reviewers for their helpful insights and suggestions that have greatly improved the content and presentation of this paper.

Authors' contributions

Jiarui Xu, Yufeng Zhao, Fei Xu conceived and designed the study. Jiarui Xu performed the simulations. Jiarui Xu wrote the paper. All authors reviewed and edited the manuscript. All authors read and approved the final manuscript.

Funding

[1] Shaanxi Provincial Department of Science and Technology Regional Innovation Capability Guidance Plan (2022QFY01-14): Research and exploration of intelligent construction and operation of sports venues based on digital twins; [2] Xi'an Beilin District Science and Technology Project (GX2137): Research on key technologies of edge computing framework for cloud robots; [3] Aerospace High Trust Embedded Software Engineering Technology Laboratory Fund: Research on Trusted Artificial Intelligence Model Verification and Code Generation Based on TensorFlow Computational Graph Model.

Availability of data and materials

The datasets used in this paper are public datasets, which are cited in the references.

Declarations

Competing interests

The authors declare that they have no competing interests.

Received: 23 July 2022 Accepted: 9 September 2022

References

1. Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. Commun ACM 60(6):84-90
2. Simonyan K, Zisserman A (2015) Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2014
3. He K, Zhang X, Ren S, et al. (2016) Deep residual learning for image recognition. arXiv Comput Vis Patt Recognit 2016:770-778
4. Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A (2015) Going deeper with convolutions. arXiv Comput Vis Patt Recognit 2015:1-9
5. Szegedy C, Vanhoucke V, Ioffe S, Shlens J, Wojna Z (2016a) Rethinking the inception architecture for computer vision. Comput Vis Patt Recognit 2016:2818-2826
6. Szegedy C, Ioffe S, Vanhoucke V, Alemi AA (2016b) Inception-v4, inception-resnet and the impact of residual connections on learning. Thirty-first AAAI conference on artificial intelligence 2017
7. Ioffe S, Szegedy C (2015) Batch normalization: Accelerating deep network training by reducing internal covariate shift. International conference on machine learning PMLR 2015:448-456
8. Huang G, Liu Z, van der Maaten L, Weinberger KQ (2016) Densely connected convolutional networks. Proceedings of the IEEE conference on computer vision and pattern recognition 2017:4700-4708
9. Crowley EJ, Gray G, Storkey A (2018) Moonshine: Distilling with cheap convolutions. Advances in Neural Information Processing Systems 2018:31
10. Polino A, Pascanu R, Alistarh D (2018) Model compression via distillation and quantization. arXiv preprint arXiv:1802.05668, 2018
11. Han S, Mao H, Dally WJ (2015a) Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. arXiv preprint arXiv:1510.00149, 2015
12. Han S, Pool J, Tran J, Dally WJ (2015b) Learning both weights and connections for efficient neural networks. Advances in neural information processing systems 2015:28
13. Courbariaux M, Bengio Y, David JP (2015) Binaryconnect: Training deep neural networks with binary weights during propagations. Advances in neural information processing systems 2015:28
14. Rastegari M, Ordonez V, Redmon J, Farhadi A (2016) Xnor-net: Imagenet classification using binary convolutional neural networks. European conference on computer vision. Springer, Cham, 2016: 525-542
15. Zhang X, Zhou X, Lin M, Sun J (2018) Shufflenet: An extremely efficient convolutional neural network for mobile devices. Proceedings of the IEEE conference on computer vision and pattern recognition 2018:6848-6856
16. Ma N, Zhang X, Zheng HT, Sun J (2018) Shufflenet v2: Practical guidelines for efficient cnn architecture design. Proceedings of the European conference on computer vision (ECCV) 2018:116-131
17. Howard A, Zhu M, Chen B, Kalenichenko D, Wang W, Weyand T, Andreetto M, Adam H (2017) Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861, 2017
18. Sandler M, Howard A, Zhu M, Zhmoginov A, Chen LC (2018) Mobilenetv2: Inverted residuals and linear bottlenecks. Proceedings of the IEEE conference on computer vision and pattern recognition 2018:4510-4520
19. Tan M, Chen B, Pang R, Vasudevan VK, Sandler M, Howard A, Le QV (2018) Mnasnet: Platform-aware neural architecture search for mobile. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition 2019: 2820-2828
20. Ding X, Zhang X, Ma N, Han J, Ding G, Sun J (2021a) Repvgg: Making vgg-style convnets great again. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition 2021:13733-13742
21. Ding X, Zhang X, Han J, Ding G (2021b) Diverse branch block: Building a convolution as an inception-like unit. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition 2021:10886-10895
22. Chollet F (2017) Xception: Deep learning with depthwise separable convolutions. Proceedings of the IEEE conference on computer vision and pattern recognition 2017:1251-1258
23. Yang H, Shen Z, Zhao Y (2021) Asymmnet: Towards ultralight convolution neural networks using asymmetrical bottlenecks. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition 2021:2339-2348
24. Han K, Wang Y, Tian Q, Guo J, Xu C, Xu C (2019) Ghostnet: More features from cheap operations. Proceedings of the IEEE/CVF conference on computer vision and pattern recognition 2020:1580-1589
25. Iandola F, Han S, Moskewicz MW, Ashraf K, Dally WJ, Keutzer K (2016) Squeezenet: Alexnet-level accuracy with 50x fewer parameters and<0.5mb model size. arXiv preprint arXiv:1602.07360 2016
26. Veit A, Wilber MJ, Belongie S (2016) Residual networks behave like ensembles of relatively shallow networks. Advances in neural information processing systems 2016:29
27. He K, Zhang X, Ren S, Sun J (2016) Identity mappings in deep residual networks. European conference on computer vision. Springer, Cham 2016:630-645
28. Li Y, Chen Y, Dai X, Chen D, Liu M, Yuan L, Liu Z, Zhang L, Vasconcelos N (2021) Micronet: Improving image recognition with extremely low flops. Proceedings of the IEEE/CVF International Conference on Computer Vision 2021:468-477
29. Zhou D, Hou Q, Chen Y, Feng J, Yan S (2020) Rethinking bottleneck structure for efficient mobile network design. European Conference on Computer Vision. Springer, Cham 2020:680-697
30. Howard A, Pang R, Adam H, Le QV, Sandler M, Chen B, Wang W, Chen LC, Tan M, Chu G, Vasudevan VK, Zhu Y (2019) Searching for mobilenetv3.

Proceedings of the IEEE/CVF international conference on computer vision 2019:1314-1324

31. 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. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition 2019:10734-10742
32. Real E, Aggarwal A, Huang Y, Le QV (2019) Regularized evolution for image classifier architecture search. Proceedings of the aaai conference on artificial intelligence 33(01):4780-4789
33. Tan M, Le QV (2019) Efficientnet: Rethinking model scaling for convolutional neural networks. International conference on machine learning. PMLR 2019:6105-6114
34. Zagoruyko S, Komodakis N (2017) Diracnets: Training very deep neural networks without skip-connections. arXiv preprint arXiv:1706.00388, 2017
35. Guo S, Alvarez JM, Salzmann M (2018) Expandnets: Linear over-parameterization to train compact convolutional networks. Advances in Neural Information Processing Systems 33:1298-1310
36. Cao J, Li Y, Sun M, Chen Y, Lischinski D, Cohen-Or D, Chen B, Tu C (2020) Do-conv: Depthwise over-parameterized convolutional layer. arXiv preprint arXiv:2006.12030
37. Ding X, Guo Y, Ding G, Han J (2019) Acnet: Strengthening the kernel skeletons for powerful cnn via asymmetric convolution blocks. Proceedings of the IEEE/CVF international conference on computer vision. 2019:1911-1920
38. Everingham M, Gool LV, Williams C, Winn J, Zisserman A (2010) The pascal visual object classes (voc) challenge. International journal of computer vision 88(2):303-338
39. Krizhevsky A (2009) Learning multiple layers of features from tiny images. Technical report 2009:7
40. Deng J, Dong W, Socher R, Li LJ, Li K, Fei-Fei L (2009) Imagenet: A large-scale hierarchical image database. 2009 IEEE conference on computer vision and pattern recognition Ieee 2009:248-255
41. Ren S, He K, Girshick R, Sun J (2015) Faster r-cnn: Towards real-time object detection with region proposal networks. Advances in neural information processing systems 2015:28
42. Chen LC, Papandreou G, Schroff F, Adam H (2017) Rethinking atrous convolution for semantic image segmentation. arXiv preprint arXiv:1706.05587, 2017
43. Chen LC, Zhu Y, Papandreou G, Schroff F, Adam H (2018) Encoder-decoder with atrous separable convolution for semantic image segmentation. Proceedings of the European conference on computer vision (ECCV) 2018:801-818

Publisher's Note