Neural Processing Letters (2019) 50:531–548
https://doi.org/10.1007/s11063-019-10032-w


# Weighted Channel-Wise Decomposed Convolutional Neural Networks

**Yao Lu[1] · Guangming Lu[1] · Yuanrong Xu[1]**

Published online: 4 April 2019


**Abstract**
Currently, block term decomposition is widely utilized to factorize regular convolutional kernels with several groups to decrease parameters. However, networks designed based on this method lack adequate information interactions from every group. Therefore, the Weighted Channel-wise Decomposed Convolutions (WCDC) are proposed in this paper, and the relevant networks can be called WCDC-Nets. The WCDC convolutional kernel employ the channel-wise decomposition to reduce the parameters and computational complexity to the bone. Furthermore, a tiny learnable weighted module is also utilized to dig up connections of the outputs from channel-wise convolutions in the WCDC kernel. The WCDC filter can be easily applied in many popular networks and can be trained end to end, resulting in a significant improvement of model's flexibility. Experimental results on the benchmark datasets showed that WCDC-Nets can achieve better performances with much fewer parameters and flop pointing computations.

**Keywords** Block term decomposition · Group convolutions · Channel-wise convolutions · Weighted channel-wise decomposed convolutions

## 1 Introduction

Convolutional Neural Networks (CNNs) have made a great breakthrough in various computer vision areas. However, these architectures usually have large numbers of parameters and floating point operations. In order to improve the efficiency of networks, researchers have proposed many methods to reduce the parameters. One of the most popular techniques is employing "$1 \times 1$" (or "point-wise") convolutions to project the input information into a low dimension space, then adopting regular convolutions with larger filter size to process informa-

✉ Guangming Lu
luguangm@hit.edu.cn

Yao Lu
yaolu_1992@126.com

Yuanrong Xu
xuyuanrong1988@126.com

[1] Harbin Institute of Technology (Shenzhen), Shenzhen, China

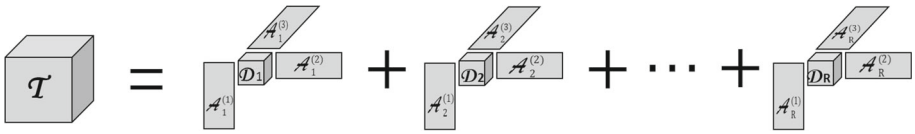


**Fig. 1** Block term decomposition of a 3-rd order tensor

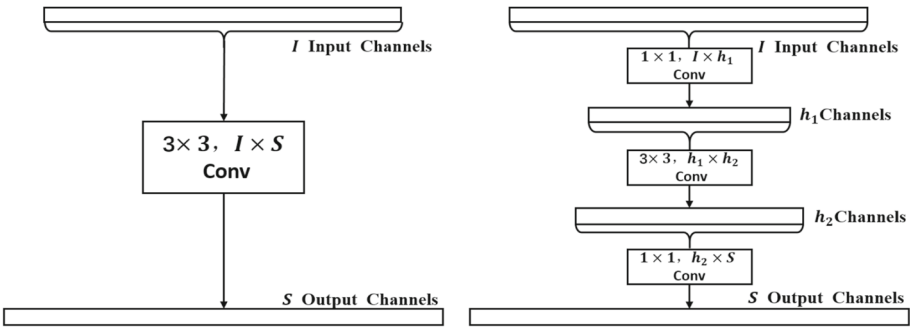
tion in this low dimension space, which can reduce parameters effectively and also be called "bottleneck" structure. This kind of structures take advantage of "point-wise" convolutions to map the input information into any dimensional space and organize the information across channels. Consequently, it can improve the representational ability of networks. Bottleneck is widely employed in ResNet [5], GoogLeNet [10,16,17] and DenseNet-BC [8].

Recently, many researches are focusing on reducing the redundancy of the convolutional kernels, because the kernels are usually very sparse, which is a hot study issue in the area of decreasing parameters and saving flop pointing computations. Since a regular convolutional kernel includes the spatial extent and channel extent, and the former is always much smaller than the latter, for instance, a convolutional kernel with size "$3 \times 3 \times 32 \times 32$". Therefore, "group convolutions" have been proposed to remove the redundancy from channel extent, which equally divides the input channels into a specific number of groups and separately performs the corresponding convolutions in each group. Specially, when the number of groups is set to 1, it will be equivalent to the regular convolutions, and when groups are equal to the input channels, it will become "channel-wise" (or "depth-wise") convolutions.

Following this study issue, the combination of bottleneck and group convolutions has been utilized in the network such as ResNeXt [18] and CRU Network [19]. Besides that, Chen [19] first gives the theoretical explanation based on the block term decomposition [3], where convolutional kernel can be seen as a 4-th order tensor and can be approximated by a sum of low rank tucker. Figure 1 illustrates the decomposition of a 3-rd order tensor. However, in practice, when the networks are designed, some hyper parameters such as the number of groups and the low rank are all fixed based on the experiences, which results in approximating the original convolutions kernels not well. To compensate for these shortcomings, Interleaved Group Convolutions (IGC) [21] has been introduced by Zhang, which initially splits the input channels to $L$ parts. Suppose every part has $M$ channels, then every part is fed to the corresponding primary group convolutions ($L$ groups). Next, all the output feature maps are concatenated together and permuted in order to be divided into $M$ partitions. At last, "$1 \times 1$" group convolutions with $M$ groups perform separately on those $M$ partitions and concatenate all their output feature maps.

Although IGC block can remedy no connections of different groups along the channel extent, it also needs a lot of expertise knowledge to design the models. In addition, although they all use the group convolutions to decrease parameters, the small divided convolutional filer still performs on several input feature maps and needs to produce a certain number of output feature maps in every group. Accordingly, all the small divided convolutional kernels have 4 dimensions, which can't reduce the filters' redundancy to the bone.

To address this problem, Weighted Channel-wise Decomposition Convolutions (WCDC) is introduced in this paper, which is a combination of bottleneck structure, channel-wise decomposed convolutions and a tiny learnable weighted module. In the WCDC structure, input information is initially projected into a low dimension space (also called "bottleneck", and set $bottleneck = R$), then these $R$ "$3 \times 3$" spatial convolutions with 2 dimensions respectively perform on $R$ feature map channels. At the following step, a learnable weighted




**(a)** Regular convolutions **(b)** Bottleneck convolutions

**Fig. 2** Comparison of regular and bottleneck structure convolutions

module is adopted to predict the weights of $R$ different spatial convolutional filters' outputs. The weighted module consists of one global average pooling layer, one nonlinear activation layer and two fully connected layers. Finally, the feature maps are scaled by the learned weights and fed to "$1 \times 1$" convolutions to be mapped into a new dimension space.

The superiorities of the WCDC-Nets are summarized as follows:

1. All the convolutions except "point-wise" convolutions in the process of implementation are just 2 dimensional plane with size "$3 \times 3$". Accordingly, the parameters are decreased by this method as low as possible.
2. A learnable weighted module is introduced to explicitly predict weights of all the groups. This small module can be easily inserted into any backbones of the networks and trained end to end simply. Consequently, it can bridge the gaps among different groups and significantly improve the model's flexibility.
3. We replace the regular convolutions of ResNet and DenseNet with the WCDC kernels and implement various versions based on them. At last, we test these architectures on CIFAR10 and CIFAR100 datasets from the aspects of parameters' efficiency and computational complexity. Experimental results demonstrated that WCDC-Nets can obtain competitive results with less parameters and flop pointing computations.

## 2 Related Work

In order to decrease the parameters and save the storage memory, many practical methods have been introduced and utilized in many applications. In this section, we will briefly illustrate some works related to ours and the novelty of the proposed method.

### 2.1 Bottleneck Structure

Bottleneck structure is widely employed in most of the popular networks, for example, GoogLeNet, Resnet and DenseNet. Comparison of the regular convolution and bottleneck structure is shown in Fig. 2. In the bottleneck structure, the information is embedded into a low dimension space through a "$1 \times 1$" convolutional layer (also called "point-wise" convolution). Then, a regular convolution layer with kernel size "$s \times s$" (e.g., $s = 3$ or $s = 5$)

will be attached to produce the features in the low dimension space. Finally, another "point-wise" convolutional layer can be utilized to increase the dimensions of output features. This process is shown in Fig. 2b. According to the Network in Network [14] (NiN), the "point-wise" convolution can organize the information along the channel extent and improve the representational ability of networks, which contributes to reducing the output dimensions effectively. Therefore, the number of the parameters and the computational complexities (Flops) can be significantly decreased by the bottleneck structure.

### 2.2 Group Convolutions

Group convolutions are first employed in AlexNet to distribute the model on two GPUs to keep the memory available, and they have been implemented by many libraries such as Caffe [11], Tensorflow [1] and Torch [2]. Group convolutions can effectively reduce parameters and computations, because the input feature map channels and convolutional filters are all divided into a specific number of groups and only conducting the convolutional operations in every group. An "extreme" version of group convolutions is "channel-wise" convolutions (also called "depth-wise convolutions" in Tensorflow [1]), where the convolutional filters are channel-wise decomposed and every obtained small filter is respectively performed on every input channel. In other words, the number of groups is equal to the input channels.

### 2.3 Mathematical View on Combination of Bottleneck and Group Convolution

It is generally known that a regular convolutional kernel can be seen as a tensor mathematically. Also, Some famous methods are proposed to decompose a tensor, such as CANDECOMP/PARAFAC (CP) Decomposition, Tucker Decomposition and their combination method Block Term Decomposition [3]. The block term decomposition can approximate a high order tensor in a sum of several low-rank Tuckers. Specially, if the rank of every Tucker equals to one, it will degrade to CP Decomposition and if there is only one Tucker, it will degrade to Tucker Decomposition. Chen [19] has proposed a more generalized version of block term decomposition, which first bridges the widening gap between theory and practical convolutional architecture (combination of bottleneck structure and group convolutions). Figure 3 shows the computational process of convolutional architectures based on the generalized block term decomposition. Specifically, the input feature maps are initially projected by a "$1 \times 1$" convolutional layer and then fed to group convolutions, where the input tensors are divided equally into $R$ groups along the channel axis and separately performed convolution operation within every group. Finally, the results are concatenated together and projected by another "$1 \times 1$" convolutional layer.

## 3 Weighted Channel-Wise Decomposed Convolutional Neural Networks

This section firstly demonstrates the Weighted channel-wise decomposed convolutions (WCDC) in the Sect. 3.1. Then, the WCDC kernel's parameters and computational complexity are analyzed in the Sect. 3.2. Finally, the detailed architecture of WCDC-Nets is illustrated in the Sect. 3.3.

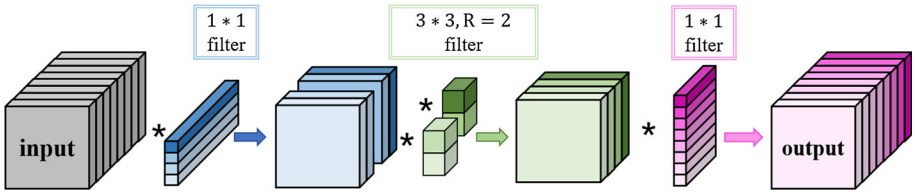



**Fig. 3** Generalized block term decomposition: the input feature maps are first projected by "point-wise" convolutions into a new dimension space. Then these embedding feature maps are fed to the group convolutions (2 groups), where the input are divided equally into 2 groups along the channel axis and separately performed convolutional operation within every group. Finally, the results are concatenated together and projected by "point-wise" convolutions again

### 3.1 Weighted Channel-Wise Decomposed Convolutions

Initially, the weighted block term decomposition and its definition are introduced in this section. Then, the mathematical view of the regular networks will be demonstrated by employing weighted block term decomposed convolutions. Finally, this section will illustrate how to utilize the channel-wise convolutions in the weighted block term decomposed convolutions.

#### 3.1.1 Weighted Block Term Decomposition

As illustrated by Chen [19], a regular tensor can be approximated by the summation of several simple convolutions, which can be formulated by generalized block term decomposition as Eq. (1), we denote a regular tensor with $N$-th order or mode (the number of dimensions of a tensor) by $\mathcal{T}$, ($\mathcal{T} \in \mathbb{R}^{s_1 \times s_2 \times \cdots \times s_N}$), where $s_n$ indicates the dimensions of the $n$th mode.

$$\begin{aligned} \mathcal{T} &\approx \sum_{r=1}^{R} \mathcal{D}_r \bullet_1^{\delta} \mathcal{A}_r^{(1)} \bullet_2^{\delta} \mathcal{A}_r^{(2)} \bullet_3^{\delta} \cdots \bullet_N^{\delta} \mathcal{A}_r^{(N)}, \\ \text{where } &\begin{cases} \mathcal{D}_r \in \mathbb{R}^{s_1^* \times s_2^* \times \cdots \times s_N^*} \\ \mathcal{A}_r^{(n)} \in \mathbb{R}^{s_n \times s_n^*}, \quad n \in \{1, 2, \ldots, N\} \end{cases} \end{aligned} \tag{1}$$

$R$ is the number of decomposed block terms, and $\mathcal{D}_r$ refers to the $r$-th core tensor with rank $(s_1^*, s_2^*, \ldots, s_N^*)$. In addition, the matrix obtained of the $n$th mode from the $r$th group is denoted by $\mathcal{A}_r^{(n)}$. Finally, $\bullet_n^{\delta}$ is the generalized $mode-n$ product and $\delta$ is the *elementwise operations*, such as nonlinear activation.

In the current popular networks, the decomposed convolutional kernels are all designed according to the experiences and fixed at the first stage of building the models. Thus, these architectures are too rigid, and they cannot approximate the regular convolutional kernels and produce the features well. However, from the Eq. (1), it is critical to find appropriate core tensors to approximate the regular tensor well. Unfortunately, traditional algorithms are all limited to being inserted into the current popular convolutional neural networks. In addition, they are inefficient and difficult to be trained end-to-end. From this point, the weighed module is introduced to explicitly learn the extent of every basic block term's importance. The formulation is shown as the following equation:

$$\mathcal{T} = \sum_{r=1}^{R} \lambda_r \mathcal{D}_r \bullet_1^{\delta} \mathcal{A}_r^{(1)} \bullet_2^{\delta} \mathcal{A}_r^{(2)} \bullet_3^{\delta} \cdots \bullet_N^{\delta} \mathcal{A}_r^{(N)} \tag{2}$$

where $\lambda_r$ denotes the weight of the $r$-th term block.

### 3.1.2 Weighted Block Term Decomposed Convolutions in Regular Networks

Following [19], suppose a 4-th order convolutional kernel $\mathcal{T} \in \mathbb{R}^{s_1 \times s_2 \times s_3 \times s_4}$, where the first two orders indicate the spatial kernel size with $s_1 \times s_2$. $s_3$ and $s_4$ refer to the number of input and output channels, respectively. In practice, the "$3 \times 3$" kernel are utilized by the proposed convolutional neural networks in this paper. Accordingly, the value of $s_1$ and $s_2$ is very small and it isn't necessary to decompose the tensor from the first two orders. Finally, Eq. (2) can be simplified as below:

$$\begin{aligned} \mathcal{T} &= \sum_{r=1}^{R} \lambda_r \mathcal{D}_r \bullet_3^{\delta} \mathcal{A}_r^{(3)} \bullet_4^{\delta} \mathcal{A}_r^{(4)}, \\ \text{where } &\begin{cases} \mathcal{D}_r \in \mathbb{R}^{s_1 \times s_2 \times s_3^* \times s_4^*} \\ \mathcal{A}_r^{(3)} \in \mathbb{R}^{s_3 \times s_3^*} \\ \mathcal{A}_r^{(4)} \in \mathbb{R}^{s_4 \times s_4^*} \end{cases} \end{aligned} \tag{3}$$

Therefore, given an input tensor $\mathcal{I} \in \mathbb{R}^{w \times h \times s_3}$, which is conducted by the novel convolutional kernel $\mathcal{T}$ based on Eq. (3), the output tensor $\mathcal{O} \in \mathbb{R}^{w \times h \times s_4}$ can be obtained as the following equation:

$$\begin{aligned} \mathcal{O}(x, y, c) = \sum_{r=1}^{R} \lambda_r \sum_{q=1}^{s_4^*} \delta \Bigg\{ \sum_{p=1}^{s_3^*} \sum_{i=x-\delta_1}^{x+\delta_1} \sum_{j=y-\delta_2}^{y+\delta_2} \\ \mathcal{D}_r(i - x + \delta_1, j - y + \delta_2, p, q) \\ \delta \left( \sum_{m=1}^{s_3} \mathcal{I}(i, j, m) \mathcal{A}_r^{(3)}(m, q) \right) \Bigg\} \mathcal{A}_r^{(4)}(c, q) \end{aligned} \tag{4}$$

where $\delta_1 = \lfloor s_1/2 \rfloor$ and $\delta_2 = \lfloor s_2/2 \rfloor$ . We employ $\boldsymbol{\mathcal{X}}_r^{(1)} \in \mathbb{R}^{s_1 \times s_2 \times s_3^*}$ and $\boldsymbol{\mathcal{X}}_r^{(2)} \in \mathbb{R}^{s_1 \times s_2 \times s_4^*}$ to represent the intermediate result. Thus, the Eq. (4) can be denoted as below:

$$\mathcal{X}_r^{(1)}(i, j, q) \triangleq \delta \left( \sum_{m=1}^{s_3} \mathcal{I}(i, j, m) \mathcal{A}_r^{(3)}(m, q) \right), \tag{5}$$

$$\begin{aligned} \mathcal{X}_r^{(2)}(x, y, q) \triangleq \delta \Bigg\{ \sum_{p=1}^{s_3^*} \sum_{i=x-\delta_1}^{x+\delta_1} \sum_{j=y-\delta_2}^{y+\delta_2} \mathcal{X}_r^{(1)}(i, j, p) \\ \mathcal{D}_r(i - x + \delta_1, j - y + \delta_2, p, q) \Bigg\}, \end{aligned} \tag{6}$$

$$\mathcal{O}(x, y, c) \triangleq \sum_{r=1}^{R} \lambda_r \sum_{q=1}^{s_4^*} \mathcal{X}_r^{(2)}(x, y, q) \mathcal{A}_r^{(4)}(c, q) \tag{7}$$

### 3.1.3 Channel-Wise Decomposition

In order to improve the efficiency of parameters, from Eq. (3), we always set $s_3^*$ and $s_4^*$ very small to decrease the number of parameters. In the proposed networks, $s_3^*$ and $= s_4^*$ are set to 1 to decompose the regular convolutional kernels along every input and output channels.

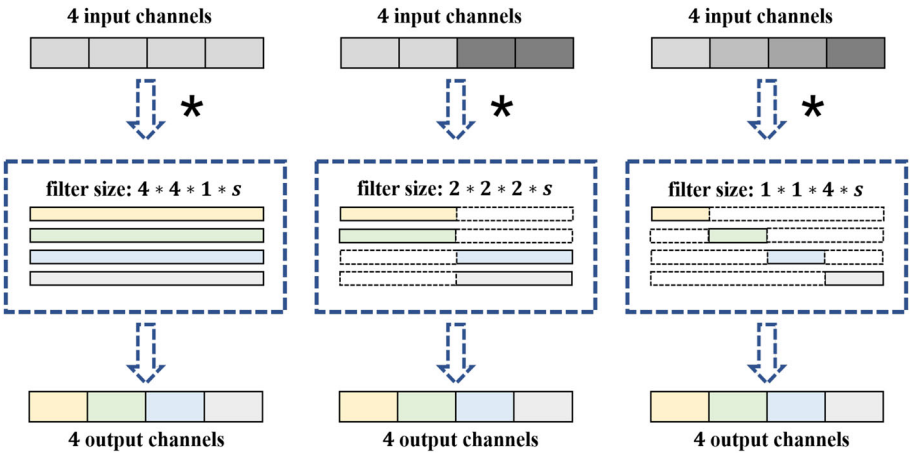



**Fig. 4** Comparisons of regular, group and channel-wise decomposed convolutions: the number of input and output channels is 4. **a** In regular convolution, filter size is $4*4*1*s$, where $s$ indicates the spatial size of filter (e.g., $3*3=9$) and 1 denotes the group of regular convolutional filters, which implies the filter conduct on all of input channels and the filter itself is not divided into different groups. **b** When the number of groups is 2, the input channels and filter are all divided into 2 groups equally. **c** This is a special case of group convolutions, when the number of groups is equal to input and output channels, filter degrades to 4 simple 2-nd order tensor and every small tensor conducts on just one input channel, which can largely decrease the parameters

This is the channel-wise decomposition and illustrated in Fig. 4c. Consequently, the number of groups $R$ is equal to *bottleneck* and every basic core tensor is 2-nd order, which leads to largely decreasing the number of parameters. Then, the weighted channel-wise decomposed convolutions can be formulated by the flowing equations:

$$\mathcal{X}_r^{(1)}(i,j) \triangleq \delta\left(\sum_{m=1}^{s_3} \mathcal{I}(i,j,m)\mathcal{A}_r^{(3)}(m)\right), \tag{8}$$

$$\mathcal{X}_r^{(2)}(x,y) \triangleq \delta\Bigg\{\sum_{i=x-\delta_1}^{x+\delta_1}\sum_{j=y-\delta_2}^{y+\delta_2} \mathcal{X}_r^{(1)}(i,j) \mathcal{D}_r(i-x+\delta_1, j-y+\delta_2)\Bigg\}, \tag{9}$$

$$\mathcal{O}(x,y,c) \triangleq \sum_{r=1}^{R} \lambda_r \mathcal{X}_r^{(2)}(x,y)\mathcal{A}_r^{(4)}(c) \tag{10}$$

where the basic core kernels are all degraded to matrices and denoted by $\boldsymbol{\mathcal{D}}_r \in \mathbb{R}^{s_1 \times s_2}$. Additionally, $\mathcal{A}_r^{(3)} \in \mathbb{R}^{s_3}$ and $\mathcal{A}_r^{(4)} \in \mathbb{R}^{s_4}$ are also degraded to vectors.

It is notable that the channel-wise decomposition is a special case of the group convolutions. When performing the group convolutions, the input channels are divided equally into $R$ groups (based on Eq. (2), $s_3^* = bottleneck/R$), then conducting regular convolutions in every group. At last, all the output channels from each group are concatenated. The differences of group and channel-wise decomposed convolutions are shown in Fig. 4.

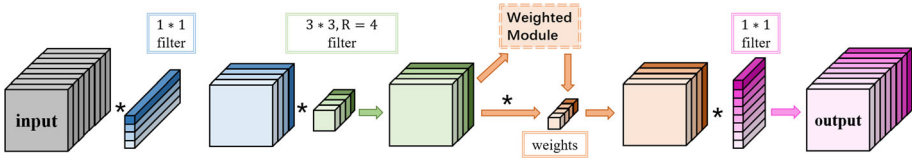



**Fig. 5** WCDC architecture: the input feature maps (8 channels) are first projected by "point-wise" convolutions ("$1 \times 1 \times 8 \times 4$") into a new dimension space, then the output feature maps are fed to the "channel-wise" convolutions (4 "$3 \times 3 \times 1 \times 1$" kernels) respectively. At the next stage, a learnable module will predict each group's weight according to the output feature maps of preceding "channel-wise" convolutions. After the output feature maps are scaled based on learned weights, they will be projected by "point-wise" convolutions ("$1 \times 1 \times 4 \times 8$") again

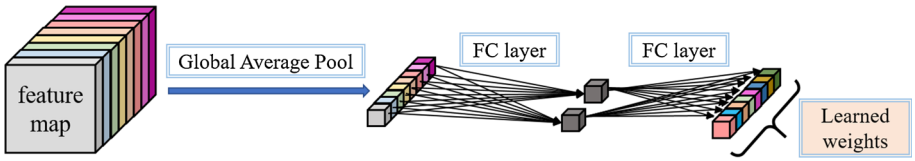



**Fig. 6** Weighted module architecture: suppose the number of input feature maps is 8, a average global pool layer is adopted to pool the feature maps' size into "$1 \times 1 \times 8$". In order to make sure the number of predicted weights is equivalent to the channels of input feature maps, we employ fully connected layers with 2 hidden neural nodes and 8 output neural nodes to predict the weights, and we denote them as $fc$, [2, 8], additionally, between them, there exists a ReLU activation layer. Finally, a sigmoid layer is attached at the end of last fully connected layer. In this figure, the nonlinearity layers haven't been drawn

From the Eqs. (8), (9) and (10), the WCDC convolutional kernels can be implemented based on the sequence of "$1 \times 1$" convolutions, "$s_1 \times s_2$" group convolutions, a learnable module and "$1 \times 1$" convolutions. Figure 5 demonstrates the process of computation. In practice, the group convolutional kernel's spatial size is always set to "$3 \times 3$", and the number of input and output feature maps of group convolutions are always the same and called *bottleneck*. It is noteworthy that the obtained weight of each group is learned in accordance with the output features of group convolutions. And the weighted module is designed based on the SE method introduced in [7] (also be called SE module). It includes one global average pool layer, one nonlinear ReLU activation layer, two fully connected layers and one nonlinear sigmoid activation layer. The weighted module is shown in Fig. 6. Then we multiply output features by the corresponding weights along the channel axis. Finally, the product is fed to the last point-wise convolutions. From the aspect of the structure, SE-Nets employ the SE module at the last of the layer. Different from the SE-Nets, in the WCDC-Nets, the SE module is utilized at the middle channel-wise convolutional layer. From another aspect of theoretical analysis, in Eq. (2), SE module is employed to predict the weights of the outputs from every groups in WCDC-Nets. However, in SE-Nets, it doesn't have any theoretical illustrations. Therefore, the functions of the SE module in the SE-Nets and WCDC-Nets are completely different.

### 3.2 Analysis of WCDC Kernel's Parameters and Computational Complexity

For given a regular convolutional kernel with size "$s \times D \times D$", in which, $s$ indicates the spatial kernel size (e.g.,$3 \times 3 = 9$) and $D$ is the number of input and output channels. Then the parameters $Params_{(r)}$ can be computed as following:

$$Params_{(r)} = D^2 s \tag{11}$$

Suppose the *bottleneck* is $R$, which implies the input features isn't embedded into $R$ dimensional space. If the regular filter is replaced with the WCDC kernel, the WCDC kernel's parameters $Params_{(w)}$ can be obtained as below (since the parameters of weighted module is very small, it is ignored):

$$\begin{aligned} Params_{(w)} &= D \times R + R \times s + R \times D \\ &= 2DR + sR = (2D + s)R \end{aligned} \tag{12}$$

From the Eqs. (11) and (12), we can get :

$$\begin{aligned} &\text{when } Params_{(w)} = Params_{(r)}, \\ &R = \frac{D^2 s}{2D + s} \text{ ( in practice, } s \ll D) \\ &\approx \frac{D^2 s}{2D} = \frac{s}{2} D \end{aligned} \tag{13}$$

According to Eq. (13), if $s = 9$, when $Params_{(w)} = Params_{(r)}$, $R \approx \frac{9}{2}D$, it indicates WCDC-Nets will be much wider than the regular CNNs with same parameters. In other words, if the *bottleneck* $R$ is smaller than $D$, the WCDC-Nets can largely decrease the number of the parameters. In addition, since the number of the "multi-add" operations is "$w \times h$" (indicating the width and height of the feature map) times of the parameters, the WCDC-Nets also significantly reduce the computational complexity.

### 3.3 Implementation Details

According to Eqs. (8), (9) and (10), WCDC kernels consist of three convolutional layers and a learnable weighted module, which can be implemented by regular operations. In addition, the *bottleneck* (number of groups) can be set to arbitrary value. Therefore, the introduced WCDC kernels are easily equipped and flexibly ported to most of the popular architectures. Since the ResNets and DenseNets are both the classical models and have achieved state-of-the-art performances, the WCDC-Nets are designed based on them by replacing their regular convolutions with WCDC kernels. The following paragraph will illustrate the detailed WCDC-Nets implemented on the ResNets [6] and DenseNets [8] performed on CIFAR [12] datasets.

Similar to ResNets [6], the WCDC-ResNets include 3 big blocks (or stages) and every big block contains $B$ basic modules. The output size in each stage is $32 \times 32$, $16 \times 16$ and $8 \times 8$, respectively. Therefore, the basic module of ResNets $\begin{pmatrix} 1 \times 1, K \\ 3 \times 3, K, R = 1 \\ 1 \times 1, 4K \end{pmatrix}$ is replaced with $\begin{pmatrix} 1 \times 1, K \\ 3 \times 3, K, R = K \\ \text{global average pool} \\ fc, [K/8, K] \\ 1 \times 1, 4K \end{pmatrix}$ in WCDC-ResNets, where $K$ represents the number of output channels of *bottleneck*, and $K$ will be double every time when it steps into the next stage. $R$ denotes the number of groups of "$3 \times 3$" convolutions. In addition, "$fc, [h_1, h_2]$" denotes the fully connected layer with $h_1$ hidden nodes and $h_2$ output nodes, respectively.

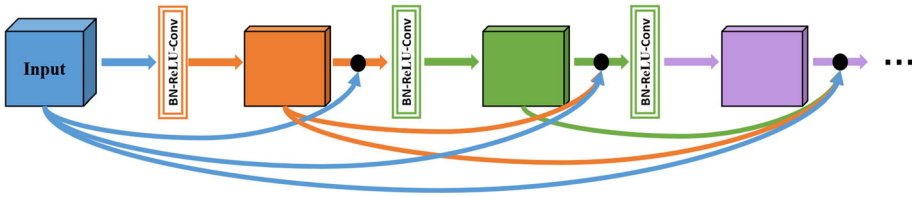



**Fig. 7** Dense block: before performing every regular operations such as "BN-ReLU-Conv", the input of every basic convolutional module is composed of all its preceding modules' output feature maps, and every module produces the same number feature map channels. Black filled circle represents concatenation operation

DenseNets [8] consists of three big dense connected blocks and a transition module between two successive dense blocks. The output size of each big block is same with the ResNets. Different from ResNet's module, whose input is element-wise added to the output of basic convolutional module. In each dense block of DenseNet (Fig. 7), every module has the same number of output feature map channels, and the input feature maps of every module are concatenated to its output feature maps. The WCDC-DenseNet can be constructed by replacing the DenseNets' basic module $\begin{pmatrix} 1\times 1, 4K \\ 3\times 3, K, R=1 \end{pmatrix}$ with $\begin{pmatrix} 1\times 1, 4K \\ 3\times 3, 4K, R=4K \\ \text{global average pool} \\ fc, [0.5K, 4K] \\ 1\times 1, K \end{pmatrix}$, where $K$ indicates the growth rate of each module (see [8]).

## 4 Experimental Result and Analysis

In this section, the performances of WCDC-Net and its variants are evaluated and compared with the existing state-of-the-art architectures, particularly with ResNet and DenseNet, since they are popular and have achieved high performances. The basic modules of ResNet and DenseNet are replaced with WCDC module. If the WCDC-Nets can produce better performances than the relevant ResNet and DenseNet, it can prove the proposed WCDC kernel is more efficient and practical than the traditional convolutional kernel.

### 4.1 Datasets

*CIFAR-10* CIFAR-10 (C10) datasets [12] has 60,000 32 × 32-pixel colored nature scene images in total, which includes 50,000 images for training and 10,000 images for test in 10 classes. Same with the common practice, we also make a data augmentation: zero-padding the images first with 4 pixels on each side, then cropping from those $40\times 40$ images randomly to produce 32 × 32 ones again. Finally, the images will be horizontally flipped. This new augmentation datasets will be denoted by C10+.

*CIFAR-100* Similar to CIFAR-10, CIFAR-100 (C100) datasets [12] also contains 60,000 32 × 32-pixel colored nature scene images, 50,000 images for training and 10,000 images for test, but it has 100 classes. CIFAR-100 has much more abundant kinds of objects than CIFAR-10, so that it is more appropriate to validate the capability of every model. We also make a data augmentation of C100 in the same way as C10, which is marked as C100+.

*ImageNet datasets* The regular ILSVRC 2012 classification dataset [4] includes more than 1.2 million training images and 50,000 validation images, from 1000 categories. We adopt

the same data augmentation rule as in [9,18] and employ a single-crop with size $224 \times 224$ at test time. We report classification errors on the validation set.

The same weight initialization and optimization configuration introduced in ResNet and DenseNet are adopted in WCDC-ResNet and WCDC-DenseNet. They all utilize SGD method and Nesterov momentum [15] to optimize. On CIFAR, the mini-batch sizes of WCDC-ResNet and WCDC-DenseNet are 128 and 64 respectively. In addition, the training epoches are set to 160 and 300. The optimization starts from the initial learning rate with 0.1, which is divided by 10 at 50% and 75% of the total number of training epochs. The momentum is 0.9 and weight decay is $2e{-}4$ and $1e{-}4$ respectively. On ImageNet, the mini-batch size is 256, and the initial learning rate is 0.6, which is divided by 10 every 30 epoches. Finally, the number of training epoches is 100. The momentum is 0.9 and weight decay is $4e{-}5$.

### 4.2 Comparisons with State-of-the-Art Models

Special emphasis will be placed on verifying that WCDC-Nets have a better performance in spirit of using fewer or same parameters than all the compared networks. Various versions of WCDC-ResNets and WCDC-DenseNets have been trained by setting different layers and growth rates, and the finally results are shown in Table 1. It is notable that, on the CIFAR datasets, almost all the WCDC-ResNets and WCDC-DenseNets achieve better accuracies compared with relevant ResNets and DenseNets with same structure settings or parameters. Additionally, they also largely outperform other classical architectures, such as Network in Network [14] and FractalNet [13] with much fewer parameters. Especially, the WCDC-DenseNet ($K = 48$, $B = 27$) obtains the best performance on CIFAR datasets (shown in bolditalic) with the same parameters of DenseNet ($K = 40$, $B = 31$). Its error rates drop to 3.39% and 16.98% on C10+ and C100+, respectively. It is paid attention to that, the number of the output channels in the first block of WCDC-DenseNet is much smaller than that of the following blocks. Therefore, in order to obtain high-quality detailed features in the first stage, in the WCDC-DenseNets ($K = 48$, $B = 27$), the original DenseNet's modules in the first stage are not replaced with the WCDC modules.

The WCDC-ResNets are also compared with the traditional models on the ImageNet datasets. From the Table 2, it's apparent that, under the same structure settings, the WCDC-ResNet-50 ($K = 64$) can significantly decrease the number of the parameters and the Flops with achieving a competitive performance. Then, under approximately the same parameters, the WCDC-ResNet-50 ($K = 80$) can produce better accuracy than the ResNets and the SE-ResNets. Moreover, the number of the WCDC-ResNet's Flops is also reduced from $3.3G$ to $2.4G$ compared with the traditional models.

### 4.3 Model's Capacity

In order to verify the WCDC-Net's capacity, the modules $B$ and growth rate $K$ are increased gradually in the implementations of WCDC-ResNets and WCDC-DenseNets. As shown in Table 1, there is a significant trend that the novel models obtain general better accuracy in the C10+ and C100+ columns as the increasing of parameters. The error rates of WCDC-ResNets gradually drop from 7.89% to 7.37% on C10, 4.72% to 3.96% C10+, 28.28% to 26.95% on C100 and 22.63% to 19.64% on C100+ datasets. Additionally, the error rates of WCDC-DenseNets gradually drop from 4.52% to 3.39% on C10+ and 21.65% to 16.98% on C100+. Accordingly, it implies that WCDC-Nets can sufficiently employ the additional

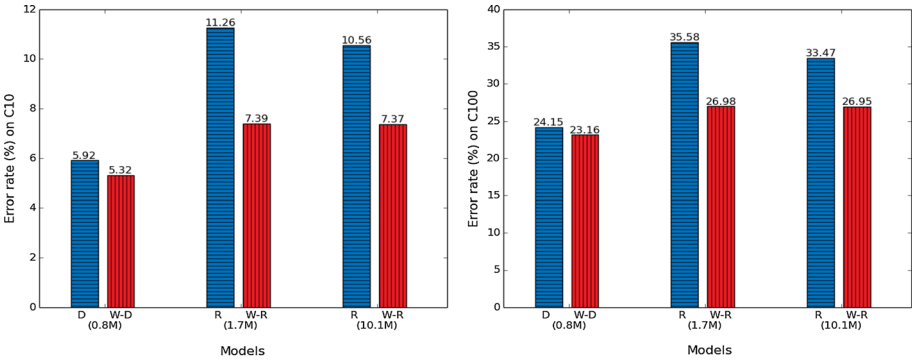
**Table 1** Test error rate (%) on CIFAR datasets: results that are better than relevant competing methods are marked bold and the overall best results are marked bolditalic

| Method | Version and structure settings | Depth | Params | C10 | C10+ | C100 | C100+ |
|---|---|---|---|---|---|---|---|
| NiN | Reported in [14] | – | – | 10.41 | 8.81 | 35.68 | – |
| FractalNet | Reported in [13] | 21 | 38.6$M$ | 10.18 | 5.22 | 35.34 | 23.30 |
| | With dropout/drop path [13] | 21 | 38.6$M$ | 7.33 | 4.60 | 28.20 | 23.73 |
| ResNet | Reported in [5] | 110 | 1.7$M$ | – | 6.61 | – | – |
| | Reported in [9] | 110 | 1.7$M$ | 13.63 | 6.41 | 44.74 | 27.22 |
| | With stochastic depth [9] | 110 | 1.7$M$ | 11.66 | 5.23 | 37.80 | 24.58 |
| | | 1202 | 10.2$M$ | – | 4.91 | – | – |
| | Wide ResNet [20] | 16 | 11.0$M$ | – | 4.81 | – | 22.07 |
| | Pre-activation (reported in [6]) | 164 | 1.7$M$ | 11.26 | 5.46 | 35.58 | 24.33 |
| | | 1001 | 10.2$M$ | 10.56 | 4.62 | 33.47 | 22.71 |
| WCDC-ResNet | $K = 16, B = 18$ | 164 | 0.9$M$ | 7.89 | 4.72 | 28.28 | **22.63** |
| | $K = 32, B = 9$ | 83 | 1.7$M$ | 7.39 | **4.40** | **26.98** | **20.77** |
| | $K = 16, B = 111$ | 1001 | 5.1$M$ | 8.26 | **4.38** | 27.57 | **20.91** |
| | $K = 32, B = 55$ | 497 | 10.1$M$ | 7.37 | **3.96** | **26.95** | **19.64** |
| DenseNet | $K = 12, B = 16$ (reported in [8]) | 100 | 0.8$M$ | 5.92 | 4.51 | 24.15 | 22.27 |
| | $K = 40, B = 31$ (reported in [8]) | 190 | 25.6$M$ | – | 3.46 | – | 17.18 |
| WCBC-DenseNet | $K = 12, B = 16$ | 148 | 0.6$M$ | **5.38** | 4.52 | **23.62** | **21.65** |
| | $K = 20, B = 11$ | 103 | 0.8$M$ | ***5.32*** | 4.30 | ***23.16*** | **20.61** |
| | $K = 48, B = 27$ $(*)$ | 247 | 25.6$M$ | – | ***3.39*** | – | ***16.98*** |

"+" indicates the data augmentation (translation and/or mirroring) of datasets. The "$*$" in the second column of WCDC-DenseNet indicates WCDC-DenseNet @ $\times$ 32 $\times$ 16, and the other WCDC-DenseNets are all WCDC-DenseNet @ $\times$ 32 $\times$ 16 $\times$ 8. Where "@*stage*" represents at which stages the model utilizes WCDC kernels. All the DenseNets and WCDC-DenseNets have bottleneck structures

**Table 2** Test error rate (%) on ImageNet datasets

| Method | Params | Flops | Top-1 | Top-5 |
|---|---|---|---|---|
| ResNet-50 ($K = 64$) [5] | $25.6M$ | $3.3G$ | 24.7 | 7.8 |
| SE-ResNet-50 ($K = 64$) [5] | $28.1M$ | $3.3G$ | 23.3 | 6.6 |
| WCDC-ResNet-50 ($K = 64$) | $15.0M$ | $1.6G$ | 25.9 | 8.7 |
| WCDC-ResNet-50 ($K = 80$) | $22.1M$ | $2.4G$ | 22.8 | 6.3 |




**(a)** Comparisons on C10 **(b)** Comparisons on C100

**Fig. 8** Comparisons on C10 and C100 datasets: "D" and "R" indicate the DenseNet and ResNet, and "W-D" and "W-R" denote WCDC-DenseNet and WCDC-ResNet. The number of parameters is marked at the bottom of each pair of contrastive models

representation of bigger size model. These better results also suggest that WCDC-Nets can alleviate the trouble of overfitting well.

### 4.4 Overfitting

With the increasing of models' parameters, the WCDC-Nets are intended to address the overfitting problem. Since C10 and C100 are the datasets without any data augmentations, it can be used to prove this point effectively. As shown in Fig. 8, the C10 and C100 error rates of WCDC-ResNet and WCDC-DenseNet compared with relevant ResNet and DenseNet with same model size have been plotted. It is apparent that all the WCDC-Nets achieve better performances compared with those relevant regular architectures with same parameters. In particular, the error rates of WCDC-ResNet ($1.7M$) are respectively declined by 3.87% and 8.6% on C10 and C100 datasets. Also, for the WCDC-ResNet ($10.1M$), the error rates are decreased by 3.19% and 6.52% on C10 and C100, respectively. This is because the WCDC convolutions just employ 2 dimensional plane convolutional kernels in every group. Therefore, the final convolutional kernel is much sparser and improves the representational ability of every 2 dimensional plane convolutional kernel. Although the parameters of WCDC-Nets is equivalent to the traditional models, the finally used WCDC convolutional kernels in WCDC-Nets are much bigger and sparser. Accordingly, WCDC-Nets have a better power of learning information and they are difficult to get into the overfitting trouble when the models have more parameters without adequate training data.

**Table 3** Test error rate (%) on CIFAR datasets with same structure settings

| Method | Structure settings | Depth | Params | C10 | C10+ | C100 | C100+ |
|---|---|---|---|---|---|---|---|
| ResNet | $K = 16, B = 18$ | 164 | $1.7M$ | 11.26 | 5.46 | 35.58 | 24.33 |
| WCDC-ResNet | | | $\mathbf{0.9M}$ | 7.89 | 4.72 | 28.28 | 22.63 |
| ResNet | $K = 16, B = 111$ | 1001 | $10.2M$ | 10.56 | 4.62 | 33.47 | 22.71 |
| WCDC-ResNet | | | $\mathbf{5.1M}$ | 8.26 | **4.38** | 27.57 | **20.91** |
| DenseNet | $K = 12, B = 16$ | 100 | $0.8M$ | 5.92 | 4.51 | 24.15 | 22.27 |
| WCDC-DenseNet | | 148 | $\mathbf{0.6M}$ | **5.38** | 4.52 | **23.62** | 21.65 |

The smaller model sizes and error rates are marked bold

**Table 4** Test error rate (%) on CIFAR datasets with same parameters

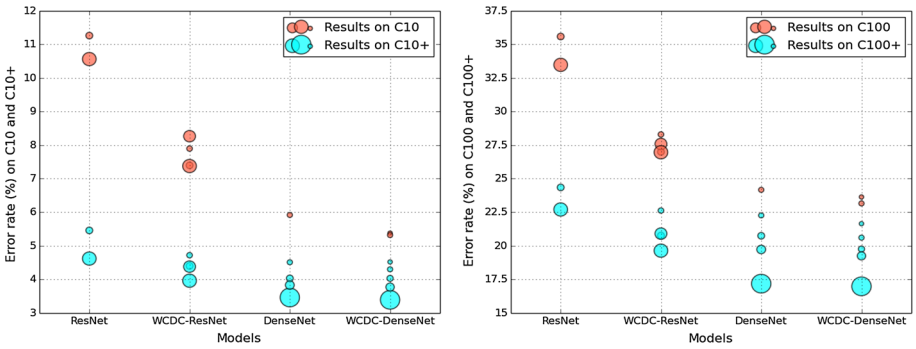
| Method | Structure settings | Depth | Params | C10 | C10+ | C100 | C100+ |
|---|---|---|---|---|---|---|---|
| ResNet | $K = 16, B = 18$ | 164 | $1.7M$ | 11.26 | 5.46 | 35.58 | 24.33 |
| WCDC-ResNet | $K = 32, B = 9$ | 83 | | 7.39 | 4.40 | 26.98 | 20.77 |
| ResNet | $K = 16, B = 111$ | 1001 | $10.2M$ | 10.56 | 4.62 | 33.47 | 22.71 |
| WCDC-ResNet | $K = 32, B = 55$ | 497 | | 7.37 | 3.96 | 26.95 | 19.64 |
| DenseNet | $K = 12, B = 16$ | 100 | $0.8M$ | 5.92 | 4.51 | 24.15 | 22.27 |
| WCDC-DenseNet | $K = 20, B = 11$ | 103 | | **5.32** | 4.30 | **23.16** | 20.61 |
| DenseNet | $K = 15, B = 20$ | 124 | $1.7M$ | – | 4.03 | – | 20.75 |
| WCDC-DenseNet | $K = 24, B = 12$ | 112 | $1.5M$ | – | 4.03 | – | 19.76 |
| DenseNet | $K = 18, B = 20$ | 124 | $2.5M$ | – | 3.83 | – | 19.73 |
| WCDC-DenseNet | $K = 28, B = 13$ | 121 | $2.4M$ | – | 3.77 | – | 19.25 |
| DenseNet | $K = 40, B = 31$ | 190 | $25.6M$ | – | 3.46 | – | 17.18 |
| WCDC-DenseNet | $K = 48, B = 27(*)$ | 247 | | – | **3.39** | – | **16.98** |

The smaller model sizes and error rates are marked bold

### 4.5 Parameter Efficiency

Since the WCDC convolutions are utilized in every module, fewer parameters are used to learn the object features. Therefore, the parameters' ability of retrieving information is largely improved. In order to test the efficiency of parameters, the WCDC-Nets are evaluated from two aspects. First, the WCDC-Nets are designed based on ResNet and DenseNet with same growth rate $K$ and modules $B$ (or depth) to test their performances with fewer parameters. The final results are shown in Table 3. From the table, it is evident that the merits of WCDC-Nets are notable, especially, the two versions of WCDC-ResNet just utilize half of the relevant ResNets' parameters and achieve much better performances. What is the most apparent is that, compared with the ResNets ($1.7M$), the error rates of WCDC-ResNet ($0.9M$) are respectively declined by 3.37%, 0.74%, 7.3% and 1.7% on C10, C10+, C100 and C100+. Additionally, WCDC-DenseNet with the least parameters ($0.6M$) also produces lower error rates on C10, C100 and C100+ than that of the DenseNet ($0.8M$). As for C10+, it also achieves a competitive result (4.52% vs. 4.51%).

From the another aspect, WCDC-Nets are designed with almost the same parameters compared with the relevant ResNet and DenseNet. The experimental results are also shown in Table 4. All the WCDC-Nets obtain better performances than the traditional architectures.




**(a)** Comparisons on C10 and C10+ **(b)** Comparisons on C100 and C100+

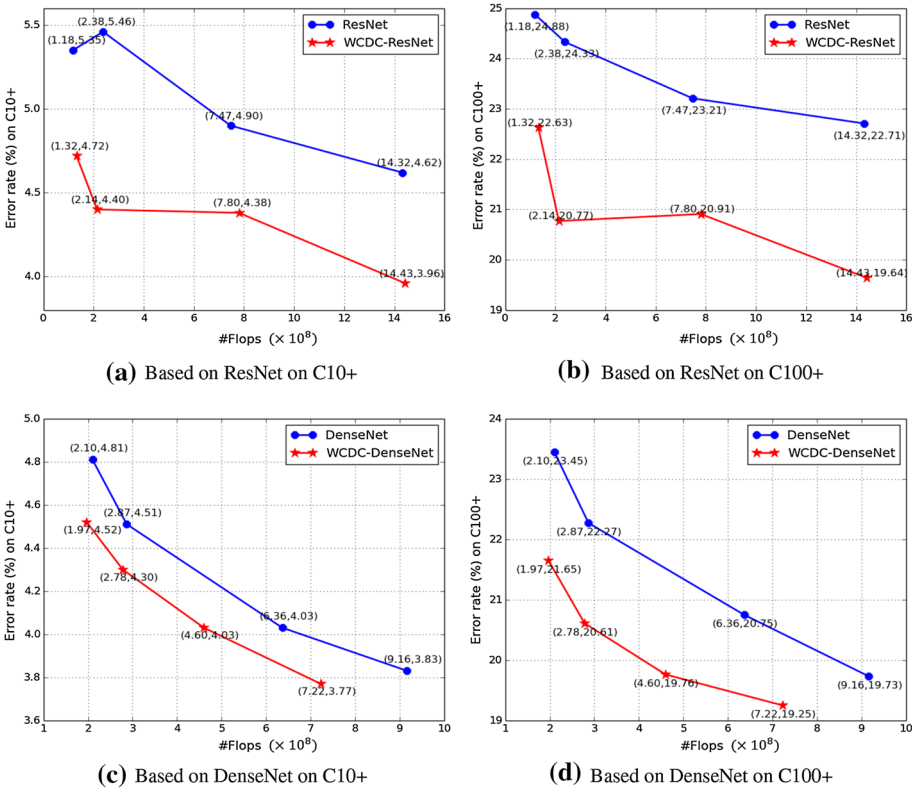
**Fig. 9** Comparisons of parameters efficiency: one circle represents one network, and the area of the circle indicates the size of the model

Finally, all the results obtained by various models with different model sizes are shown in Fig. 9. One circle represents one network, and the bigger the circle is, the larger the relevant architecture is. From Fig. 9, WCDC-Nets achieve better results with much smaller model size and obtain best performances with same model size compared with the relevant networks. It is notable that the WCDC-Nets have significant improvements on parameters' efficiency.

Since the WCDC filters employ the channel-wise decomposition, when the WCDC-Nets have the same structure settings, the final kernel is sparse and remove the redundancy from the traditional filters. And for WCDC-Nets with same parameters, they will have much larger filters to produce features, which implies the WCDC-Nets are much wider. In addition, the learnable weighted module will fuse the information from every group and further strengthen the learning ability of convolutional kernels. All these factors will lead to improving the representational power of parameters. Consequently, from the comparisons of these two aspects, WCDC-Nets have significantly and effectively improve the parameter's efficiency.

### 4.6 Computational Complexity

As shown in Fig. 10, the error rates of C10+ and C100+ are respectively compared based on the implementations of ResNet and DenseNet. As the increasing of complexity, the error rates of all the networks have a general decreasing trend. Additionally, the final results also reveal that WCDC-ResNets and WCDC-DenseNets both obtain better accuracy than traditional models with fewer times of flop operations ("multiply-add" steps). Especially, in Fig. 10b, the WCDC-ResNet with fewest Flops obtains the lower error rate than ResNet with the most Flops on C100+ [$(1.32 \times 10^8,\ 22.63\%)$ vs $(14.82 \times 10^8,\ 22.71\%)$]. The best performance of ResNet can be obtained by WCDC-ResNet through utilizing just less $1/10$ computational complexity of ResNet. The less computational complexity is attributed to the WCDC kernels in the basic block. Due to this channel-wise decomposition, the input feature maps and convolutional filters are all divided into a specific number (equal to the *bottleneck*) of groups. In other words, every small filter is 2 dimensional and just performs on one input channel, which leads to much fewer flop operations.



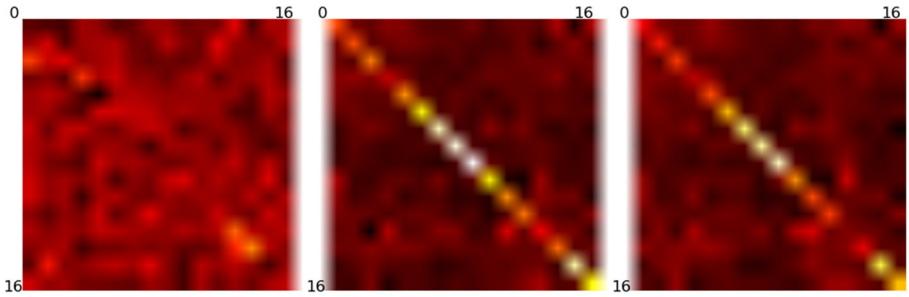

**(a)** Based on ResNet on C10+ **(b)** Based on ResNet on C100+

**(c)** Based on DenseNet on C10+ **(d)** Based on DenseNet on C100+

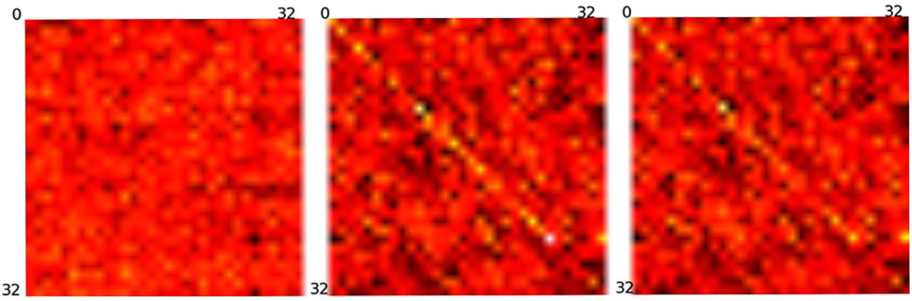
**Fig. 10** Comparisons of computational complexity on C10+ and C100+ datasets

### 4.7 Further Investigation of WCDC Kernels

In order to further explore the effects of channel-wise decomposition and the learnable weighted module, the inter-layer correlation between the adjacent filter layers are computed. They are evaluated on C100 from ResNet ($K = 16$, $B = 18$) and WCDC-ResNet ($K = 16$, $B = 18$). The results are shown in Fig. 11. Figure 11a, b display the "$3 \times 3$" convolutional layers from the 10th and 28th modules with 16 and 32 filters, respectively. The left images in Fig. 11a, b are from ResNet, it is clear to find that the regular kernels are rambling. But in the middle images from WCDC-ResNet ($K = 16$, $B = 18$), which shows the channel-wise decomposed filters correlation before the weighted module. It is visible that the channel-wise decomposition enforces the diagonalization of the entire filter. Especially, in Fig. 11a, there are 16 lighter points distributing on the diagonal line. In addition, compared with ResNet ($K = 16$, $B = 18$), the diagonal line is much lighter while the left of image is much darker, which validates the channel-wise decomposed convolutional kernel is much sparser than the regular kernel. The right images are the results after the feature maps scaled by the weighted module along the channel extent. It is interesting that the diagonal line becomes blurred and darker slightly, but the other part becomes a little lighter. It suggests that the WCDC-kernels are not only sparse but also a better approximation of regular convolutions from another point of view.




**(a)** Comparisons of 16 filters




**(b)** Comparisons of 32 filters

**Fig. 11** Inter-layer filter correlation. The left images of **a** and **b** are from ResNet ($K = 16, B = 18$), the middle and the right are respectively from channel-wise decomposed kernels before and after weighted module of WCDC-ResNet ($K = 16, B = 18$). The diagonal sparsity learned by channel-wise decomposed kernels is apparent in the correlation of filters. The larger the correlation, the lighter the corresponding pixel

## 5 Conclusion

A novel WCDC kernel was introduced in this paper. It can largely decrease the parameters and the computational Flops. Additionally, the WCDC kernels can approximate regular convolutions much better, because the tiny weighted module was employed to predict weights and explicitly establish connections of all the different groups. Finally, experimental results showed that the WCDC models can significantly improve the parameters' efficiency and decrease the computational complexity with better performances. In the future work, we will explore more efficient convolutional filter based on the WCDC kernel. Furthermore, in practice, the mobile networks will be designed to address the problem of deploying the networks on the mobile devices effectively.

**Acknowledgements** This work is supported by NSFC fund (61332011), Shenzhen Fundamental Research fund (JCYJ20170811155442454, JCYJ20180306172023949), and Medical Biometrics Perception and Analysis Engineering Laboratory, Shenzhen, China.

## References

1. Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, Citro C, Corrado GS, Davis A, Dean J, Devin M et al (2016) Tensorflow: large-scale machine learning on heterogeneous distributed systems. arXiv preprint arXiv:1603.04467

2. Collobert R, Bengio S, Marithoz J (2002) Torch: a modular machine learning software library. Technical report IDIAP-RR 02-46, IDIAP
3. De Lathauwer L (2008) Decompositions of a higher-order tensor in block terms—part II: definitions and uniqueness. SIAM J Matrix Anal Appl 30(3):1033–1066
4. Deng J, Dong W, Socher R, Li LJ, Li K, Fei-Fei L (2009) Imagenet: a large-scale hierarchical image database. In: IEEE conference on computer vision and pattern recognition, 2009. CVPR 2009, pp 248–255. IEEE
5. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: The IEEE conference on computer vision and pattern recognition (CVPR), pp 770–778
6. He K, Zhang X, Ren S, Sun J (2016) Identity mappings in deep residual networks. In: European conference on computer vision. Springer, Berlin, pp 630–645
7. Hu J, Shen L, Sun G (2017) Squeeze-and-excitation networks. arXiv preprint arXiv:1709.01507
8. Huang G, Liu Z, Weinberger KQ, van der Maaten L (2017) Densely connected convolutional networks. arXiv preprint arXiv:1608.06993v4
9. Huang G, Sun Y, Liu Z, Sedra D, Weinberger K (2016) Deep networks with stochastic depth. In: European conference on computer vision. Springer, Berlin, pp 646–661
10. Ioffe S, Szegedy C (2015) Batch normalization: accelerating deep network training by reducing internal covariate shift. In: International conference on machine learning, pp 448–456
11. Yangqing J, Evan S, Jeff D, Sergey K, Jonathan L (2014) Caffe: convolutional architecture for fast feature embedding. Eprint Arxiv, pp 675–678
12. Krizhevsky A, Hinton G (2009) Learning multiple layers of features from tiny images
13. Larsson G, Maire M, Shakhnarovich G (2016) Fractalnet: ultra-deep neural networks without residuals. arXiv preprint arXiv:1605.07648
14. Lin M, Chen Q, Yan S (2013) Network in network. arXiv preprint arXiv:1312.4400
15. Sutskever I, Martens J, Dahl G, Hinton G (2013) On the importance of initialization and momentum in deep learning. In: International conference on machine learning, pp 1139–1147
16. Szegedy C, Liu W, Jia Y, Sermanet P (2014) Going deeper with convolutions. In: Computer vision and pattern recognition, pp 1–9
17. Szegedy C, Vanhoucke V, Ioffe S, Shlens J, Wojna Z (2016) Rethinking the inception architecture for computer vision. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 2818–2826
18. Xie S, Girshick R, Dollár P, Tu Z, He K (2016) Aggregated residual transformations for deep neural networks. arXiv preprint arXiv:1611.05431
19. Yunpeng C, Xiaojie J, Bingyi K, Jiashi F, Shuicheng Y (2017) Sharing residual units through collective tensor factorization in deep neural networks. arXiv preprint arXiv:1703.02180
20. Zagoruyko S, Komodakis N (2016) Wide residual networks. arXiv preprint arXiv:1605.07146
21. Zhang T, Qi GJ, Xiao B, Wang J (2017) Interleaved group convolutions for deep neural networks. ArXiv e-prints