The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

Decoupled Convolutions for CNNs

Guotian Xie,[1,2*] **Ting Zhang,**[4] **Kuiyuan Yang,**[3] **Jianhuang Lai,**[1,2] **Jingdong Wang**[4]
[1]School of Data and Computer Science, Sun Yat-Sen University
[2]Guangdong Province Key Laboratory of Information Security
[3]DeepMotion, [4]Microsoft Research
xieguotian1990@gmail.com,{Ting.Zhang, jingdw}@microsoft.com
kuiyuanyang@deepmotion.ai, stsljh@mail.sysu.edu.cn

Abstract

In this paper, we are interested in designing small CNNs by decoupling the convolution along the spatial and channel domains. Most existing decoupling techniques focus on approximating the filter matrix through decomposition. In contrast, we provide a two-step interpretation of the standard convolution from the filter at a single location to all locations, which is exactly equivalent to the standard convolution. Motivated by the observations in our decoupling view, we propose an effective approach to relax the sparsity of the filter in spatial aggregation by learning a spatial configuration, and reduce the redundancy by reducing the number of intermediate channels. Our approach achieves comparable classification performance with the standard uncoupled convolution, but with a smaller model size over CIFAR-100, CIFAR-10 and ImageNet.

Introduction

Since AlexNet (Krizhevsky, Sutskever, and Hinton 2012) successfully applied Convolutional Neural Network (CNN) to ImageNet and won the challenge by a large margin in 2012, CNNs become the most widely used model for image classification (He et al. 2016), object detection (Ren et al. 2015; Redmon and Farhadi 2016) and image segmentation (Long, Shelhamer, and Darrell 2015; Kolesnikov and Lampert 2016) and so on. CNNs have become deeper and deeper (Simonyan and Zisserman 2014; Szegedy et al. 2015; He et al. 2015; 2016; Huang et al. 2016), ranging from tens of layers to thousands of layers to pursue better performance, and have become wider and wider as well, such as Wide Residual Networks (Zagoruyko and Komodakis 2016).

Another research direction is designing more effective filters. There have been many works on filter design, and most of them can be categorized into two types. One is to decompose the filter matrix into several low rank matrices (Ioannou et al. 2015; Denton et al. 2014; Zhang et al. 2015; Kim et al. 2015; Tai et al. 2015; Jaderberg, Vedaldi, and Zisserman 2014; Mamalet and Garcia 2012), the other is to view the filter as a sparse matrix, where some works sparsify the channel extent, e.g., group convolution (Ioannou et al. 2016; Zhang et al. 2017), channel-wise convolution or separable filters (Chollet 2016) and other works sparsify the spatial extent with smaller filters, e.g., 3×3, 1×3 and 3×1 (Szegedy et al. 2016). In this paper, in contrast to design the filters, we are interested in decoupling the convolution along the spatial and channel domains and propose an effective approach based on the decoupled interpretation.

We start from analyzing the process of convolution on the input, and decompose this process into two steps. First each location in the input is projected across the channel domain. In this way, the projection along channel domain is not related to the spatial information of the input. Second, we accumulate the projections of the locations across spatial domain, and this process is only related to the spatial relationship. We reformulate the decoupled two steps in a convolution form, first conducting 1×1 across channel-domain convolution, and then conducting across spatial-domain convolution with a spatial configuration. This process is denoted as decoupling spatial convolution.

From this decoupling view, we found that the decoupled structure of standard spatial convolution is unbalance, in which the 1×1 across channel-domain convolution is in a high dimensional space that might lead to redundancy, whereas the across spatial-domain convolution is a structured sparse group convolution. To solve this problem, we propose a balance decoupling spatial convolution (BDSC) to relax the sparsity of across spatial-domain convolution by learning a spatial configuration, and to reduce the redundancy of across channel-domain convolution by reducing the intermediate output channels. In this way, we found in our experiments that, the performance of the models using our decoupling convolution drops slightly comparing with the standard spatial convolution, yet the model size is smaller than models of standard spatial convolution.

Our contributions in this paper are:

1. We decouple the standard spatial convolution of CNN into two parts, an across channel-domain convolution and an across spatial-domain convolution.
2. We propose the balance decoupling spatial convolution to relax the sparsity of the filter in spatial aggregation by learning a spatial configuration, and to reduce the redundancy of 1×1 across channel-domain convolution by re-

[*]This work was done when Guotian Xie was an intern at Microsoft Research, Beijing, P.R. China.

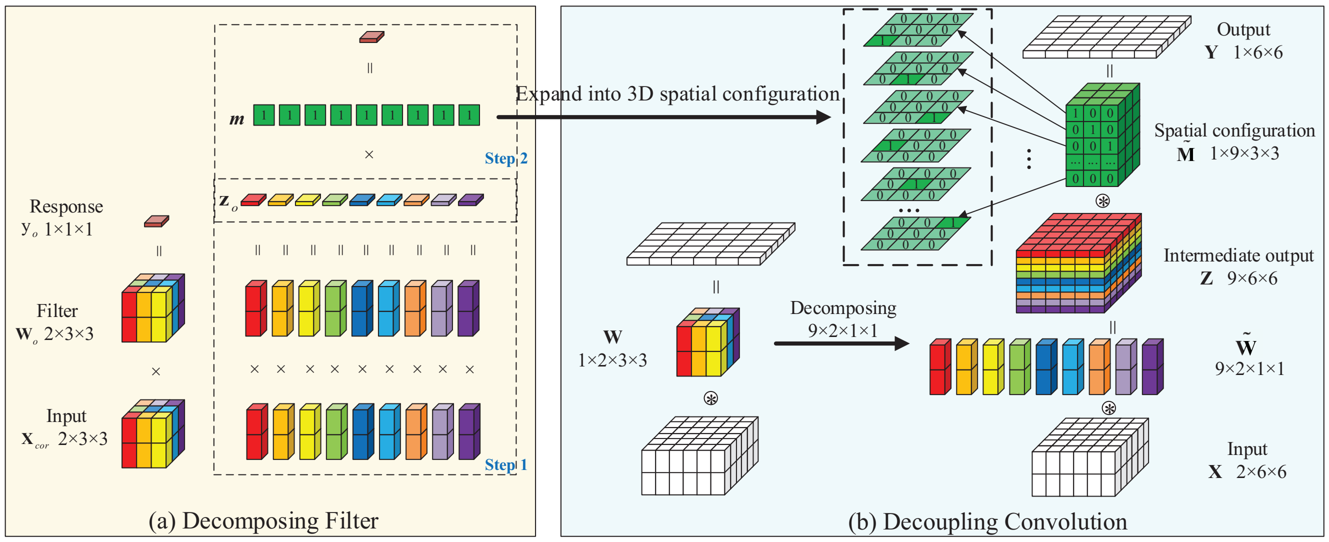

Figure 1: Illustrating the decoupled convolution by (a) decomposing filter and (b) decoupling convolution. In (a), an entry is obtained by first the projection across channel-domain and then the aggregation across spatial-domain. In (b), the convolution is decoupled into an across channel-domain convolution and an across spatial-domain convolution.

ducing the number of intermediate output channels.

3. Our experiments on CIFAR-100, CIFAR-10 and ImageNet demonstrate that models using the proposed balance decoupling spatial convolution get slightly drop in performance comparing with models using the standard spatial convolution, but with a smaller model size.

Decoupling Spatial Convolution

A convolutional layer maps a three-dimensional tensor, denoted as input $\mathbf{X} \in \mathcal{R}^{C_{in} \times H \times W}$, to a three-dimensional tensor, denoted as output $\mathbf{Y} \in \mathcal{R}^{C_{out} \times H \times W}$, where $H \times W$ is the spatial size of feature map in the tensor (the spatial size of the input feature map and the output feature map are assumed to be the same), C_{in} is the number of channels in the input and C_{out} is the number of channels in the output. The filters in a convolutional layer are parameterized by a four dimensional tensor $\mathbf{W} \in \mathcal{R}^{C_{out} \times C_{in} \times K_h \times K_w}$, where $K_h \times K_w$ is the spatial size of the filter and $\mathbf{W}(o, \cdot, \cdot, \cdot)$ is the oth filter, denoted as $\mathbf{W}_o \in \mathcal{R}^{C_{in} \times K_h \times K_w}$, $o = 1, \cdots, C_{out}$. We will first show the process of filter decomposition, and then reformulate this process into convolution decoupling. All vectors in this paper are column vectors.

Decomposing Filter

Let $\mathbf{Y}_o \in \mathcal{R}^{H \times W}$ be the oth feature map of the output, we have $\mathbf{Y}_o = \mathbf{W}_o \circledast \mathbf{X}$, where $\circledast$ denotes the convolution operation. An entry y in $\mathbf{Y}_o$ is obtained through first the projection across channel-domain and then the aggregation across spatial-domain. Let the corresponding input denoted as $\mathbf{X}_{cor} \in \mathcal{R}^{C_{in} \times K_h \times K_w}$,

1. **Across channel-domain projection.** Decomposing $\mathbf{W}_o$ along the spatial-domain and then we obtain $\{\mathbf{w}^o_{u,v}\}_{u=1,\cdots,K_h, v=1,\cdots,K_w}$, where $\mathbf{w}^o_{u,v} = \mathbf{W}_o(\cdot, u, v) \in \mathcal{R}^{C_{in}}$ is the column of $\mathbf{W}_o$ at the position (u, v). Accordingly, decomposing $\mathbf{X}_{cor}$ as $\{\mathbf{x}^{cor}_{u,v}\}$ corresponding to $\{\mathbf{w}^o_{u,v}\}_{u=1,\cdots,K_h, v=1,\cdots,K_w}$. Then the output of across channel-domain projection is obtained as,

$$\mathbf{z}_o = \begin{bmatrix} {\mathbf{w}^o_{1,1}}^T \mathbf{x}^{cor}_{1,1} \\ {\mathbf{w}^o_{1,2}}^T \mathbf{x}^{cor}_{1,2} \\ \cdots \\ {\mathbf{w}^o_{K_h,K_w}}^T \mathbf{x}^{cor}_{K_h,K_w} \end{bmatrix}. \tag{1}$$

Here, $S^* = K_h \times K_w$ and $\mathbf{z}_o \in \mathcal{R}^{S^*}$ is the intermediate output. This process is illustrated in Figure 1 (a) step 1.

2. **Across spatial-domain aggregation.** The spatial-domain aggregation is performed on the intermediate outputs using a spatial mask $\mathbf{M} \in 1^{K_h \times K_w}$, and we denote $\mathbf{m} = vec(\mathbf{M})$,where $vec(\cdot)$ is a function to vectorize a tensor. The output of the aggregation across spatial-domain is,

$$y_o = \mathbf{m}^T \mathbf{z}_o, \tag{2}$$

which is equivalent to the response of the o_{th} filter on the input $\mathbf{X}_{cor}$. This process is shown in Figure 1 (a) step 2.

Figure 1 (a) shows an example of this decomposition process, in which $C_{in} = 2$, and the spatial size of the filter $\mathbf{W}_o$ is 3×3. The first step decomposes $\mathbf{W}_o$ along the spatial-domain into 9 single columns, and each is multiplied with the corresponding column of $\mathbf{X}_{cor}$ to get the intermediate feature, which has 9 responses, corresponding to $9 = 3 \times 3$ different spatial locations. Then across spatial-domain aggregation is conducted on the 9 responses, using the mask $\mathbf{m} = [1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1]^T$.

The above analysis is for one filter at a single location. When there are C_{out} filters, the intermediate output of the first step becomes $\mathbf{z} = [\mathbf{z}_1^T, \mathbf{z}_2^T, \cdots, \mathbf{z}_{C_{out}}^T]^T$, and the final output of the second step is,

$$\mathbf{y} = \hat{\mathbf{M}}\mathbf{z} \tag{3}$$

$$= \begin{bmatrix} \mathbf{m}^T & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{m}^T & \cdots & \mathbf{0} \\ \cdots & \cdots & \cdots & \cdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{m}^T \end{bmatrix} \begin{bmatrix} \mathbf{z}_1 \\ \mathbf{z}_2 \\ \cdots \\ \mathbf{z}_{C_{out}} \end{bmatrix}, \tag{4}$$

where $\mathbf{z} \in \mathcal{R}^S$, $\hat{\mathbf{M}} \in \{0,1\}^{C_{out} \times S}$, $S = C_{out} \times K_h \times K_w$, and $\mathbf{y} \in \mathcal{R}^{C_{out}}$.

Collecting all locations of output maps together, we show that each step actually can be formed as a convolution. As a result, the convolution is decoupled into an across channel-domain convolution and an across spatial-domain convolution.

Decoupling Convolution

In this section, we will give mathematical formulation to show that each step can be regarded as a convolution.

Across channel-domain convolution. It is easy to see that the projection across channel-domain is equivalent to a 1×1 convolution with filters $\mathbf{W}$ being reshaped to $\tilde{\mathbf{W}} \in \mathcal{R}^{S \times C_{in} \times 1 \times 1}$, and

$$\tilde{\mathbf{W}} = \begin{bmatrix} {\mathbf{w}_{1,1}^1}^T \\ \cdots \\ {\mathbf{w}_{K_h,K_w}^1}^T \\ \cdots \\ {\mathbf{w}_{K_h,K_w}^2}^T \\ \cdots \\ {\mathbf{w}_{K_h,K_w}^{C_{out}}}^T \end{bmatrix}. \tag{5}$$

Therefore the across channel-domain convolution is given as,

$$\mathbf{Z} = \tilde{\mathbf{W}} \circledast \mathbf{X}, \tag{6}$$

where $\mathbf{Z}$ is a three-dimensional tensor with size being $S \times H \times W$.

Across spatial-domain convolution. We first expand the spatial mask $\mathbf{M}$ into a mask tensor similar to the one shown in Figure 1(b). The resulting mask tensor denoted as $\mathbf{M}_e \in \{0,1\}^{(K_h \times K_w) \times K_h \times K_w}$ satisfies that there is only one entry valued as 1 in $\mathbf{M}_e(k, \cdot, \cdot)$ $(k = 1, \cdots, (K_h \times K_w))$ and all the entries valued as 1 are at different locations of spatial map of size $K_h \times K_w$. As a result, we can see that across spatial-domain aggregation is equivalent to a $K_h \times K_w$ convolution with filters $\tilde{\mathbf{M}} \in \{0,1\}^{C_{out} \times S \times K_h \times K_w}$,

$$\mathbf{Y} = \tilde{\mathbf{M}} \circledast \mathbf{Z}, \tag{7}$$

$$\text{where} \quad \tilde{\mathbf{M}} = \begin{bmatrix} \tilde{\mathbf{m}}^T & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \tilde{\mathbf{m}}^T & \cdots & \mathbf{0} \\ \cdots & \cdots & \cdots & \cdots \\ \mathbf{0} & \mathbf{0} & \cdots & \tilde{\mathbf{m}}^T \end{bmatrix}_{C_{out} \times C_{out}}, \tag{8}$$

$$\text{and} \quad \tilde{\mathbf{m}} = vec(\mathbf{M}_e). \tag{9}$$

In summary, the convolution can be decoupled as,

$$\mathbf{Y} = \tilde{\mathbf{M}} \circledast (\tilde{\mathbf{W}} \circledast \mathbf{X}), \tag{10}$$

where the input is first fed into across channel-domain operation which is a 1×1 convolution that maps the input to a very high dimensional space, and then fed into across spatial-domain operation which is a $K_h \times K_w$ convolution that handles the spatial information and meanwhile performs the dimension reduction.

Figure 1 (b) shows the decoupling convolution process and the comparison with the standard convolution. We can see that for each convolution $\mathbf{Y} = \mathbf{W} \circledast \mathbf{X}$, there exists a 1×1 convolution tensor $\tilde{\mathbf{W}}$ and a $K_h \times K_w$ convolution tensor $\tilde{\mathbf{M}}$, such that $\mathbf{Y} = \mathbf{W} \circledast \mathbf{X} = \tilde{\mathbf{M}} \circledast (\tilde{\mathbf{W}} \circledast \mathbf{X})$. That is, the two-step interpretation is exactly equivalent to the standard convolution. After the decoupling, $\tilde{\mathbf{W}}$ is not related to spatial-domain any more. It maps the input in current feature space to another feature space. On the other hand, $\tilde{\mathbf{M}}$ encodes the spatial relationship that if the entry of $\tilde{\mathbf{M}}$ is 1, meaning that the corresponding feature is related, otherwise is 0. So we denote $\tilde{\mathbf{M}}$ as the spatial configuration.

Balance Decoupling Spatial Convolution

From the decoupled spatial convolution shown in Equation (10), we observe that: (i) the spatial configuration $\tilde{\mathbf{M}}$ is corresponding to a structured sparse group convolution; and (ii) the 1×1 convolution of filters $\tilde{\mathbf{W}}$ maps the features from a low dimensional space into a high dimensional space (from input with C_{in} dimensions to output with $S = C_{out} \times K_h \times K_w$ dimensions). This is an unbalance structure, and we think that the intermediate representation contains too many channels and the spatial configuration $\tilde{\mathbf{M}}$ are too sparse. Motivated by these observations, we propose the balance decoupling spatial convolution (BDSC), with a learned spatial configuration and an unaggressive 1×1 convolution by setting $S = C_{in}$.

Relax the Sparsity of Spatial Configuration

The across spatial-domain convolution is a 3×3 fixed sparse group convolution. In fact, we can learn a spatial configuration $\tilde{\mathbf{M}}_l$ to relax the fixed sparse constraint. In the training of standard convolution neural network, it is not easy to learn the spatial configuration directly. Instead, we add a floating-point precision tensor $\mathbf{Q}$ corresponding to the spatial configuration $\tilde{\mathbf{M}}_l$, and update this floating-point precision tensor $\mathbf{Q}$. When performing forward propagation, we constrain that $\tilde{\mathbf{M}}_l = sign(\mathbf{Q})$. The approximated gradients, however, are not so smooth. So we adopt the techniques of XNOR (Rastegari et al. 2016) to learn $\tilde{\mathbf{M}}_l$ by introducing a vector α,

$$\tilde{\mathbf{Q}}(o, \cdot, \cdot, \cdot) = \alpha(o)\tilde{\mathbf{M}}_l(o, \cdot, \cdot, \cdot). \tag{11}$$

According to XNOR net, the best $\tilde{\mathbf{M}}_l$ and α to approximate $\mathbf{Q}$ by $\tilde{\mathbf{Q}}$ are,

$$\begin{aligned} \tilde{\mathbf{M}}_l &= sign(\mathbf{Q}), \\ \alpha(o) &= \frac{1}{n}\|\mathbf{Q}(o, \cdot, \cdot, \cdot)\|_1, \end{aligned} \tag{12}$$

where $n = S \times K_h \times K_w$. More details about the derivation can be found in the paper (Rastegari et al. 2016). Then we approximate the gradient w.r.t. $\mathbf{Q}$ $g_{\mathbf{Q}}$ as $g_{\tilde{\mathbf{Q}}}$, i.e., we use $g_{\tilde{\mathbf{Q}}}$ to update $\mathbf{Q}$. The training process of the spatial configuration $\tilde{\mathbf{M}}_l$ is shown in Algorithm 1.

Note that the spatial configuration $\tilde{\mathbf{M}}_l$ learned using Algorithm 1 are valued as -1 or 1, which can be easily transfered to be valued as 0 or 1 by $\tilde{\mathbf{M}} = \frac{1}{2}(\tilde{\mathbf{M}}_l + \mathbf{1})$.

```
Data: Input $\mathbf{X}$, float-type $\mathbf{Q}$ corresponding to the
      spatial configuration, and gradients from
      backward $\frac{\partial L}{\partial \mathbf{Y}}$
Result: Feature maps $\mathbf{Y}$, $\mathbf{Q}$ after updating, spatial
        configuration $\tilde{\mathbf{M}}_l$, $\alpha$ for approximation
clamp $\mathbf{Q}$ to range [-1,1]
$\tilde{\mathbf{M}}_l = sign(\mathbf{Q})$
for $o_{th}$ filter in this layer do
    $\alpha(o) = \frac{1}{n}\|\mathbf{Q}(o,\cdot,\cdot,\cdot)\|_1$
    $\tilde{\mathbf{Q}}(o,\cdot,\cdot,\cdot) = \alpha(o)\tilde{\mathbf{M}}_l(o,\cdot,\cdot,\cdot)$
end
$\mathbf{Y}$=ConvolutionForward($\mathbf{X}$,$\tilde{\mathbf{Q}}$);
$\frac{\partial L}{\partial \tilde{\mathbf{Q}}}$=ConvolutionBackward($\frac{\partial L}{\partial \mathbf{Y}}$,$\mathbf{X}$,$\tilde{\mathbf{Q}}$) ;
set $\frac{\partial L}{\partial \mathbf{Q}}$=$\frac{\partial L}{\partial \tilde{\mathbf{Q}}}$
Update($\mathbf{Q}$,$\frac{\partial L}{\partial \mathbf{Q}}$)
```

Algorithm 1: The training process of $\tilde{\mathbf{M}}_l$

Reduce the Redundancy of 1×1 Convolution

The filters $\tilde{\mathbf{W}} \in \mathcal{R}^{S \times C_{in} \times 1 \times 1}$ in the across channel-domain projection map the features to a high dimensional space (from the input channel number being C_{in} to the output channel number being $S = C_{out} \times K_h \times K_w$). Usually, we have $S > C_{in}$. For example, a convolution layer in ResNet-18 with setting $C_{out} = 512, K_h = K_w = 3$ will result in $S = 512 \times 3 \times 3 > C_{in} = 512$. That is, the across channel-domain projection of standard convolution is a mapping from a low dimensional space to a high dimensional space, which may cause a redundancy. To reduce the redundancy, we set $S = C_{in}$, which is the smallest projection dimension to provide lossless projection.

Analysis

We denote the proposed scheme as balance decoupling spatial convolution (BDSC), with an unaggressive 1×1 convolution of $\tilde{\mathbf{W}} \in \mathcal{R}^{S \times C_{in} \times 1 \times 1}$ by setting $S = C_{in}$, followed by a 3×3 convolution of a learned spatial configuration $\tilde{\mathbf{M}} \in \{0,1\}^{C_{out} \times S \times K_h \times K_w}$. In the following, we discuss the number of parameters and FLOPs compared BDSC with the standard convolution, where we assume the number of the input channels and the output channels are the same, i.e., $C_{out} = C_{in} = C$.

#Params. The number of parameters in a convolution layer with the filters being $\mathbf{W} \in \mathcal{R}^{C \times C \times K_h \times K_w}$ is $C \times C \times K_h \times K_w$ with float type. The balance decoupling spatial convolution layer in our network contains the projection filters $\tilde{\mathbf{W}} \in \mathcal{R}^{C \times C \times 1 \times 1}$ and the spatial configuration $\tilde{\mathbf{M}} \in \{0,1\}^{C \times C \times K_h \times K_w}$, where the number of parameters in $\tilde{\mathbf{W}}$ is $C \times C$ with float type and the number of parameters in $\tilde{\mathbf{M}}$ is $C \times C \times K_h \times K_w$ with binary value $\{0,1\}$, which takes $\frac{1}{32} C \times C \times K_h \times K_w$ with respect to float type. Thus the compression rate is,

$$r_p = \frac{C \times C \times K_h \times K_w}{\frac{1}{32} C \times C \times K_h \times K_w + C \times C}. \tag{13}$$

With a typical setting that $K_h = K_w = 3$, the compression rate is $r = \frac{1}{\frac{1}{32} + \frac{1}{9}} \approx 7$.

FLOPs. For a standard convolution layer, the FLOPs is

$$H \times W \times C \times C \times K_h \times K_w$$

with $H \times W$ being the spatial size of the output. For our network, $\tilde{\mathbf{M}}$ is a tensor with value $\{0,1\}$, and an entry may be valued as 1 with a probability q. The convolution of spatial configuration with value 0 and 1 contains only additions and no multiplications. Therefore the FLOPs of across spatial-domain convolution with spatial configuration $\tilde{\mathbf{M}}$ is $\frac{q}{2}$ FLOPs of the standard convolution. The FLOPs of both across channel-domain convolution and across spatial-domain convolution is

$$H \times W \times (C \times C + \frac{q}{2} C \times C \times K_h \times K_w). \tag{14}$$

In summary, the speed up rate is

$$r_f = \frac{H \times W \times C \times C \times K_h \times K_w}{H \times W \times (C \times C + \frac{q}{2} C \times C \times K_h \times K_w)}. \tag{15}$$

With a typical setting that $K_h = K_w = 3$ and $q = \frac{1}{2}$ (in the experiments, usually $q < \frac{1}{2}$), the speed up rate is $\frac{1}{\frac{1}{9} + \frac{1}{4}} \approx 2.77$.

Experiments

Datasets

We use three datasets to demonstrate our network. The first is the benchmark ImageNet dataset (ILSVRC2012) (Russakovsky et al. 2015) that consists of $1,000$ classes. ImageNet contains over 1.2 million training images and $50,000$ validation images. For testing, we report the top-1 accuracy of center crop of the validation dataset of ImageNet. The results reported are the best performance of model during training. The other two are CIFAR-100 dataset, which contains 50000 training images and 10000 test images, each labeled with 100 classes and CIFAR-10 dataset, which also consists of 50000 training images and 10000 test images, each labeled with 10 classes. We randomly resize the 32×32 image to scale within the range [32,40] and randomly crop a 32×32 patch with randomly horizontal mirroring for training. Then we test on the 10000 test images on the size 32×32.

Setup

We implement our model based on Caffe (Jia et al. 2014). For the classification task of 1000 classes of ImageNet, we train all the models for $500,000$ iterations with batch size

Model	ResNet32-1×		ResNet32-2×		ResNet32-3×		ResNet32-4×	
	Accuracy	Model size	Accuracy	Model size	Accuracy	Model size	Accuracy	Model size
Standard convolution	0.6839	$1.83MB$	0.7283	$7.17MB$	0.7450	$16.0MB$	0.7568	$28.4MB$
BDSC-1	0.6648	$0.36MB$	0.7199	$1.20MB$	0.7392	$2.58MB$	0.7502	$4.49MB$
BDSC-2	0.6824	$0.63MB$	0.7326	$2.25MB$	0.7458	$4.91MB$	0.7566	$8.61MB$
BDSC-3	**0.6969**	$0.90MB$	**0.7357**	$3.30MB$	**0.7499**	$7.24MB$	**0.7587**	$15.4MB$

Table 1: Comparison between standard convolution and BDSC-p with $S = pC_{in}$ based on ResNet32-$\alpha\times$ over CIFAR-100. BDSC-p achieves better performance with a larger p. Compared with the standard convolution, BDSC-3 achieves better performance with smaller model size.

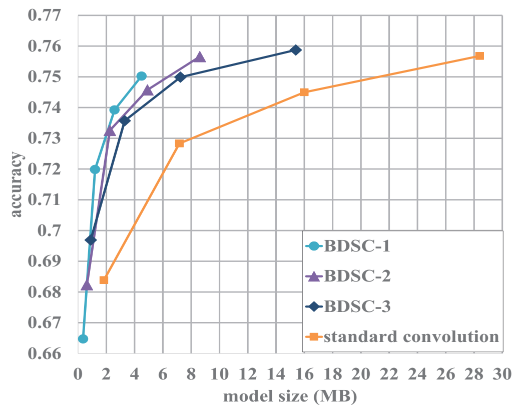

Figure 2: Illustrating the results of BDSC-p with $S = pC_{in}$ on CIFAR-100. When p decreases, the curve of BDSC-p moves to the left, which shows that we can reduce the redundancy by decreasing p while maintain the same level of accuracy.

256. For CIFAR-100 and CIFAR-10, we train $180,000$ iterations with batch size 64. The weight decay is set as 0.0001 and the momentum is 0.9. we set the initial learning rate as 0.1 and divided the learning rate by 10 for each $150,000$ iterations on ImageNet and for each $50,000$ iterations on CIFAR. On ImageNet, we use multi-scale (randomly resizing the image to scale within range [256,480]) and randomly crop with randomly horizontal mirroring for data augmentation. We initialize the weights with the MSRA initialization techniques introduced in (He et al. 2015) and train the model from scratch. We train all models by SGD with Nesterov momentum. We use a factor α to multiply the width of the network, e.g., ResNet32-$\alpha\times$, where the $\alpha\times$ means that we widen the width of each layers in ResNet32 by multiplying a factor α. In all models for ImangeNet and CIFAR, we keep the first layer and the last layer unchanged.

Empirical study

The Effect of Intermediate Feature Width. In the across channel-domain convolution, we set $S = C_{in}$ to reduce the redundancy and guarantee the lossless projection. In this experiment, we explore how the dimension of S reflect the redundancy of the standard convolution and use ResNet32-$\alpha\times$ on CIFAR-100 dataset. In addition, we view p of $S = pC_{in}$ as a variable, and investigate when p becomes larger, how the accuracy-against-model-size curve changes. The results are shown in Figure 2 with different width $S = pC_{in}$ where $p = 1, 2, 3$. We denote those models as BDSC-p with setting $S = pC_{in}$, e.g., BDSC-1 is the ResNet32-$\alpha\times$ by setting $p = 1$, BDSC-2 is the ResNet32-$\alpha\times$ by setting $p = 2$ and so on. The previous analysis shows that ResNet32-$\alpha\times$ with the standard convolution is equivalent to a decoupled model with a fixed sparse spatial configuration in the across spatial-domain convolution and $p = 9$ in the across channel-domain convolution. Thus ResNet32-$\alpha\times$ with the standard convolution can be viewed as an extreme case of our BDSC model with the most redundancy.

From Figure 2 where α changes from 1 to 4 (denoted as from left to right on each line), we found that the accuracy of ResNet32-$\alpha\times$ with the standard convolution grows slowly when the model size increases, while the accuracy of BDSC-p grows faster. For example, the curve of BDSC-2 reaches the highest accuracy at the point $(8.61, 0.756)$ (these numbers are shown in Table 1), while the curve of the standard convolution reaches to the highest accuracy at the point $(28.4, 0.7568)$, which indicates that about $28.4-8.6 = 19.8MB$ parameters of ResNet32-4× with the standard convolution are waste. This shows that there exist large redundancy in the models using the standard convolution. When p change from 3 to 1, the curve of BDSC-p gradually moves to the left. This phenomena shows that the redundancy is gradually reduced when p becomes smaller. It can be seen that BDSC-1 shows a good trade-off between model size and accuracy, and as a result, setting $p = 1$ is a suggested choice to design a model to reduce the redundancy of the standard convolution.

model	ResNet32-2×		ResNet32-4×	
	Acc.	Model size	Acc.	Model size
BDSC-float	0.7354	$8.35MB$	0.7597	$33.0MB$
BDSC	0.7199	$1.20MB$	0.7502	$4.49MB$

Table 2: Comparison between models using float-type spatial filters (denoted as BDSC-float) and our proposed BDSC which uses $\{0, 1\}$-type spatial filters. We use Acc. to represent accuracy. Our model achieves inferior performance, but largely reduces the model size.

The Effect of the Type of Spatial Configuration. In BDSC, the spatial configuration $\tilde{\mathbf{M}}_l$ is forced to be a tensor with value 0 or 1. To verify whether this setting is efficient, we compare two types of $\tilde{\mathbf{M}}_l$, one is with float type (32bits), and the other is with value $\{0, 1\}$ (1bit). We do experiments

Model	ResNet18		
	Standard convolution	Depthwise	BDSC
Accuracy	0.6944	0.6570	0.6898
Model size	$44.6MB$	$8.89MB$	$9.00MB$
Model	ResNet34		
	Standard convolution	Depthwise	BDSC
Accuracy	0.7294	0.6868	0.7219
Model size	$83.2MB$	$13.4MB$	$14.6MB$

Table 3: Comparison between standard convolution, depthwise separable convolution (denoted as Depthwise) and BDSC over ImageNet. Our BDSC with smaller model size performs slightly worse than standard convolution. Compared with depthwise separable convolution, our model achieves better performance with similar model size.

on ResNet32-2$\times$ and ResNet32-4$\times$ on CIFAR-100 and the comparison over the model size and accuracy is given in Table 2. We can see that models with $\{0, 1\}$-type spatial filters perform worse than models with float-type spatial filters, and the gaps are 1.55% and 0.95% on ResNet32-2$\times$ and ResNet32-4$\times$ respectively. These gaps are acceptable as models with $\{0, 1\}$-type spatial filters achieve smaller model size, saving $7.15MB$ and $28.51MB$ respectively. This shows that the spatial configuration in our network is reasonable.

Results

The experiments are conducted on three aspects. First we compare BDSC with the standard convolution based on ResNet (He et al. 2016). Then we show the advantage of BDSC over depthwise separable convolution (Chollet 2016). At last, we demonstrate the effectiveness of BDSC on densely connected network (Huang et al. 2016).

Comparison with Standard Convolution We use ResNet (He et al. 2016) as the baseline, and our models replace all the convolution layers with BDSC except the first convolution layer.
ImageNet. Table 3 shows the results of models with the standard convolution and our models with BDSC. We implement the baseline ResNet18 and ResNet34 by ourselves, and the performance is comparable to the results in the original paper (He et al. 2016). It can be seen that by reducing the redundancy, our BDSC models can achieve 5$\times$ compression rate. By learning flexible spatial configuration, our model ResNet18-BDSC gets top-1 accuracy of 0.6898 and ResNet34-BDSC gets top-1 accuracy of 0.7219, which is comparable to 0.6944 on ResNet18 and 0.7294 on ResNet34. On both models, the top-1 accuracy drops less than 0.75% but the model sizes are reduced about 5$\times$ rate. This demonstrates that our models can better explore the parameter effectiveness while maintain high accuracy.

The empirical rate number is about 5$\times$, which is smaller than the rate 7$\times$ in theoretical. This might be caused by the size of classifier, the first convolution layer and possibly other cost.
CIFAR. Similar to ImageNet, our models are formed by replacing the standard spatial convolution layer with BDSC on the ResNet32-2$\times$ and ResNet74-2$\times$. The results are shown in Table 4. Compared with standard convolution, the top-1 accuracy of ResNet32-2$\times$ with BDSC-1 is close to the accuracy of ResNet32-2$\times$ with standard convolution. For example on CIFAR-100, ResNet32-2$\times$ of BDSC-1 has 0.7199 top-1 accuracy, while ResNet32-2$\times$ of standard convolution has 0.7283 top-1 accuracy, about 0.8% drop of accuracy. While the model size of our network is $1.20MB$, achieves 6$\times$ less than the network with standard convolution.

We also show the accuracy-against-model-size curve on CIFAR-100 by varying α in ResNet32-$\alpha\times$ in Figure 3. It can be seen that our model with BDSC can largely boost the accuracy with the increasing model size. Note that the accuracy of ResNet32-1$\times$ with BDSC is 0.6648, which gets about 1.9% drop in performance compared to 0.6839 in ResNet32-1$\times$ with the standard convolution. The reason might be that the intermediate output dimension of 1×1 convolution of BDSC is too small here given the output channels of ResNet32-1$\times$ are $[16, 32, 64]$ for each stage respectively, which leads to that the intermediate representation is not sufficient to express enough information. In this case where the number of intermediate input channel of a model is small, we suggest to set p in $S = pC_{in}$ to be a bit larger, e.g., $p > 1$. From Table 1, we can see that setting $p = 2$ on ResNet32-1$\times$ leads to nearly no drop in top-1 accuracy, while the model size of BDSC is still smaller than the standard convolution model ($0.63MB$ for BDSC, vs $1.83MB$ for the standard convolution model).

Comparison with Depthwise Separable Convolution. Depthwise separable convolution (Chollet 2016) decouples the standard convolution into 3×3 depthwise convolution followed by a 1×1 convolution. Here we do experiments to compare BDSC with depthwise convolution.
ImageNet. We replace the standard convolution layer in ResNet with the corresponding depthwise separable convolution and set the decay on the depthwise 3×3 convolution to 0 during training. The comparison results are shown in Table 3, from which we can see that BDSC model has similar model size as model with depthwise separable convolution, but achieves higher accuracy higher, about 3% better on both ResNet18 and ResNet34.
CIFAR. The comparison on CIFAR datasets are shown in Table 4 and Figure 3. We can see that our model achieves higher accuracy than depthwise separable convolution in Table 4. Figure 3 shows the accuracy-against-model-size curve of BDSC and depthwise separable convolution on ResNet32-$\alpha\times$. Our model with BDSC, achieves higher accuracy compared with depthwise separable convolution at the same level of models size. For example, top-1 accuracy of ResNet32-1$\times$ of BDSC is about 1.8% higher than that of depthswise separable convolution, which shows the advantage of BDSC over depthwise separable convolution. We think that the superior performance of BDSC than depthwise separable convolution stems from the differences of the spatial relationship encoding ability. The 3×3 depthwise convolution is a channel-wise convolution, encoding the spatial relationship within one channel. BDSC uses the learned spatial configuration, which encodes the spatial relationship not

Model	CIFAR-100				CIFAR-10			
	ResNet32-2×		ResNet74-2×		ResNet32-2×		ResNet74-2×	
	Accuracy	Model size	Accuracy	Model size	Accuracy	Model size	Accuracy	Model size
Standard convolution	0.7283	$7.17MB$	0.7476	$17.5MB$	0.9369	$7.12MB$	0.9430	$17.5MB$
BDSC-3	0.7357	$3.30MB$	0.7515	$7.96MB$	0.9379	$3.25MB$	0.9394	$7.91MB$
Depthwise	0.6937	$1.08MB$	0.7201	$2.48MB$	0.9225	$1.04MB$	0.9309	$2.44MB$
BDSC-1	0.7199	$1.20MB$	0.7332	$2.83MB$	0.9334	$1.16MB$	0.9380	$2.78MB$

Table 4: Comparison between standard convolution, depthwise separable convolution (denoted as Depthwise) and BDSC over CIFAR. BDSC-3 with smaller model size achieves comparable performance to standard convolution. BDSC-1 achieves better performance at similar level of model size compared with depthwise separable convolution.

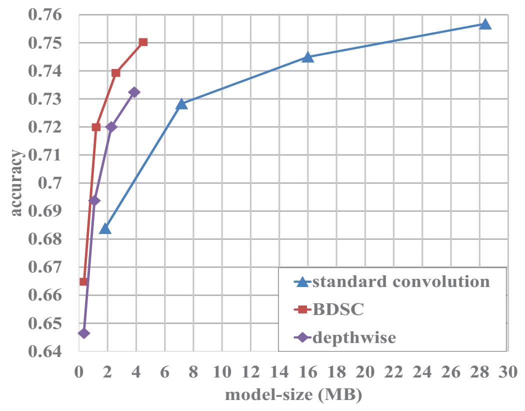

Figure 3: Comparison between standard convolution, depthwise separable convolution (denoted as depthwise) and BDSC based on ResNet32-$\alpha\times$ over CIFAR-100. Our model achieves better performance than both standard convolution and depthwise separable convolution with the same model size.

only within one channel, but also across channels.

Analysis. We compare the number of parameters and FLOPs between our BDSC and depthwise separable convolution. For depthwise separable convolution with input and output channels being C, the number of parameters is,

$$C \times K_h \times K_w + C \times C, \tag{16}$$

and the FLOPs is,

$$H \times W(C \times K_h \times K_w + C \times C). \tag{17}$$

Therefore, the compression rates of #params and FLOPs comparing BDSC with depthwise separable convolution, which are denoted as r_p and r_f respectively, are,

$$\begin{aligned} r_p &= \frac{\frac{1}{32}C \times C \times K_h \times K_w + C \times C}{C \times K_h \times K_w + C \times C} = \frac{\frac{1}{32}+\frac{1}{9}}{\frac{1}{C}+\frac{1}{9}}, \\ r_f &= \frac{H \times W(\frac{1}{4}C \times C \times K_h \times K_w + C \times C)}{H \times W(C \times K_h \times K_w + C \times C)} = \frac{\frac{1}{4}+\frac{1}{9}}{\frac{1}{C}+\frac{1}{9}}. \end{aligned} \tag{18}$$

When C is in range $[32, 2048]$, r_p is in range $[1, 1.27]$, and r_f is in range $[2.54, 3.22]$. This suggests that depthwise separable convolution is faster than BDSC with the number of parameters at a similar level. However, from Figure 3, one can see that at the same level of model size, our model BDSC achieves better performance. Accuracy, model size and speed are three things we consider to balance. So our BDSC achieves an alternative balance among speed, model size and accuracy, and it's a good choice in the case where accuracy is mostly considered as well as small model size and moderately speedup.

Model	CIFAR-10		CIFAR-100	
	Acc.	Model size	Acc.	Model size
DenseNet	0.9476	$3.98MB$	0.7558	$4.13MB$
DenseNet*	0.9433	$3.98MB$	0.7450	$4.13MB$
BDSC-6	0.9494	$2.95MB$	0.7629	$3.10MB$

Table 5: Comparison between standard convolution and BDSC based on DenseNet. We implement DenseNet by ourselves and report the results denoted as DenseNet*. We also report the results from the paper denoted as DenseNet. We set $S = 6C_{out}$ in BDSC-6. Acc. in the table means accuracy.

Comparison over Densely Connected Networks We also show the effectiveness of our BDSC over densely connected networks (Huang et al. 2016). DenseNet-40($k = 12$) is adopted to conduct experiments on CIFAR-100 and CIFAR-10. We use the same data augmentation as (Huang et al. 2016) and train for 400 epochs, and the results are shown in table 5. We also report the results number from the paper, which are denoted as DenseNet. We can see that although the results of DenseNet* implemented by ourselves performs worse than the numbers from the paper, our models with BDSC blocks still achieve better performance than DenseNet with a smaller model size.

Conclusion

In this paper, we present a novel two-step interpretation of convolution by decoupling it into an across channel-domain convolution and an across spatial-domain convolution. Based on the interpretation, we propose an effective approach by relaxing the sparsity of the fixed sparse filter in across spatial-domain convolution and by reducing the redundancy of 1×1 convolution. Empirical results on ImageNet and CIFAR datasets demonstrate that our proposed balance decoupling spatial convolution can achieve a model with small size but still performs comparable to standard convolution.

Acknowledgments

This project was supported by the NSFC (U1611461, 61573387).

References

Chollet, F. 2016. Xception: Deep learning with depthwise separable convolutions. *arXiv preprint arXiv:1610.02357*.

Denton, E. L.; Zaremba, W.; Bruna, J.; LeCun, Y.; and Fergus, R. 2014. Exploiting linear structure within convolutional networks for efficient evaluation. In *Advances in Neural Information Processing Systems*, 1269–1277.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, 1026–1034.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 770–778.

Huang, G.; Liu, Z.; Weinberger, K. Q.; and van der Maaten, L. 2016. Densely connected convolutional networks. *arXiv preprint arXiv:1608.06993*.

Ioannou, Y.; Robertson, D.; Shotton, J.; Cipolla, R.; and Criminisi, A. 2015. Training cnns with low-rank filters for efficient image classification. *arXiv preprint arXiv:1511.06744*.

Ioannou, Y.; Robertson, D.; Cipolla, R.; and Criminisi, A. 2016. Deep roots: Improving cnn efficiency with hierarchical filter groups. *arXiv preprint arXiv:1605.06489*.

Jaderberg, M.; Vedaldi, A.; and Zisserman, A. 2014. Speeding up convolutional neural networks with low rank expansions. *arXiv preprint arXiv:1405.3866*.

Jia, Y.; Shelhamer, E.; Donahue, J.; Karayev, S.; Long, J.; Girshick, R.; Guadarrama, S.; and Darrell, T. 2014. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*.

Kim, Y.-D.; Park, E.; Yoo, S.; Choi, T.; Yang, L.; and Shin, D. 2015. Compression of deep convolutional neural networks for fast and low power mobile applications. *arXiv preprint arXiv:1511.06530*.

Kolesnikov, A., and Lampert, C. H. 2016. Seed, expand and constrain: Three principles for weakly-supervised image segmentation. In *European Conference on Computer Vision*, 695–711. Springer.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. ImageNet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 1097–1105.

Long, J.; Shelhamer, E.; and Darrell, T. 2015. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3431–3440.

Mamalet, F., and Garcia, C. 2012. Simplifying convnets for fast learning. *Artificial Neural Networks and Machine Learning–ICANN 2012* 58–65.

Rastegari, M.; Ordonez, V.; Redmon, J.; and Farhadi, A. 2016. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European Conference on Computer Vision*, 525–542. Springer.

Redmon, J., and Farhadi, A. 2016. Yolo9000: better, faster, stronger. *arXiv preprint arXiv:1612.08242*.

Ren, S.; He, K.; Girshick, R.; and Sun, J. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, 91–99.

Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; Berg, A. C.; and Fei-Fei, L. 2015. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)* 115(3):211–252.

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

Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; and Rabinovich, A. 2015. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1–9.

Szegedy, C.; Ioffe, S.; Vanhoucke, V.; and Alemi, A. 2016. Inception-v4, inception-resnet and the impact of residual connections on learning. *arXiv preprint arXiv:1602.07261*.

Tai, C.; Xiao, T.; Zhang, Y.; Wang, X.; et al. 2015. Convolutional neural networks with low-rank regularization. *arXiv preprint arXiv:1511.06067*.

Zagoruyko, S., and Komodakis, N. 2016. Wide residual networks. *arXiv preprint arXiv:1605.07146*.

Zhang, X.; Zou, J.; Ming, X.; He, K.; and Sun, J. 2015. Efficient and accurate approximations of nonlinear convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1984–1992.

Zhang, T.; Guo-Jun, Q.; Bin, X.; and Jingdong, W. 2017. Interleaved group convolutions. *International Conference on Computer Vision*.