XSepConv: Extremely Separated Convolution

Jiarong Chen[1], Zongqing Lu[1], Jing-Hao Xue[2], and Qingmin Liao*[1]

[1] Tsinghua University
cjr18@mails.tsinghua.edu.cn,luzq@sz.tsinghua.edu.cn,
liaoqm@tsinghua.edu.cn
[2] University College London
jinghao.xue@ucl.ac.uk
* Corresponding author

Abstract. Depthwise convolution has gradually become an indispensable operation for modern efficient neural networks and larger kernel sizes (≥ 5) have been applied to it recently. In this paper, we propose a novel extremely separated convolutional block (XSepConv), which fuses spatially separable convolutions into depthwise convolution to further reduce both the computational cost and parameter size of large kernels. Furthermore, an extra 2×2 depthwise convolution coupled with improved symmetric padding strategy is employed to compensate for the side effect brought by spatially separable convolutions. XSepConv is designed to be an efficient alternative to vanilla depthwise convolution with large kernel sizes. To verify this, we use XSepConv for the state-of-the-art architecture MobileNetV3-Small and carry out extensive experiments on four highly competitive benchmark datasets (CIFAR-10, CIFAR-100, SVHN and Tiny-ImageNet) to demonstrate that XSepConv can indeed strike a better trade-off between accuracy and efficiency.

Keywords: depthwise convolution, spatially separable convolutions, trade-off, efficiency

1 Introduction

Convolutional neural networks (CNNs) are becoming increasingly ubiquitous in numerous computer vision tasks, such as object detection and image classification, due to their more and more outstanding performance over time. Consequently, it has stimulated the desire to deploy these top-performing CNNs on resource-constrained platforms, e.g., on mobile phones, drones, self-driving cars, robots and Internet-of-Things (IOT) devices. However, most top-performing CNNs are in need of tremendous computational resources, severely impeding their practical deployment on these devices with constrained computing power.

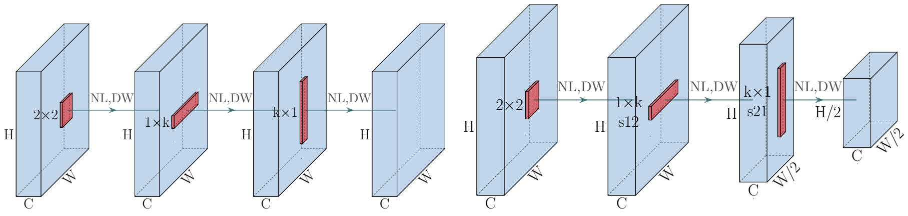
Regarding the issue mentioned above, a lot of research work has been dedicated to the design of efficient CNN architectures, leading to the emergence of a variety of architectures with outstanding performance in terms of accuracy and efficiency trade-off, including Xception [2], MobileNets [7,25,6], ShuffleNets [39,18], NASNet [40], MnasNet [31], EfficientNet [32] and IGCV family

[38,37,27], to name a few. Among these top-performing architectures, the CNNs built upon depthwise convolution, which are represented by the family of MobileNets [7,25,6], are increasingly becoming the mainstream attributing to their better trade-off between accuracy and efficiency.

A standard depthwise convolution can be viewed as a special case of group convolution, where the number of groups is equal to the number of channels. With an input tensor of N channels, it contains N kernels, each of which is independently applied to one channel of the input tensor, thereby reducing both the computational cost and parameter count by a factor of N. Since its extensive adoption in the milestone efficient CNN architecture MobileNets [7], depthwise convolution has attracted a lot of research interest in utilizing it to design efficient CNNs. In the early stage, the design of efficient CNNs is mainly through manual effort, and the major research effort is devoted to devising the basic building blocks as well as overall architectures of neural networks. As a result, the kernel size of depthwise convolution is simply specified as 3 in most cases [2,7,25,39,18,37,17]. Recently, neural architecture search (NAS) has been exploited to automatically design efficient CNNs, leading to the adoption of larger depthwise convolutional kernel sizes [40,22,16,23,1,34,31,6,32,33]. Moreover, by adopting symmetric padding [36], the conventional convolution with even-sized kernels can also achieve competitive accuracy compared with depthwise convolution.

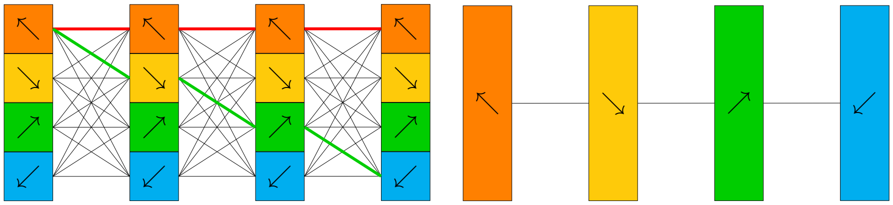
Recognizing that the CNNs built upon depthwise convolution mainly focus on exploration in the dimension of channels to reach the aim of reduction in computational cost, potentially more reduction can be achieved through further decomposition in the spatial dimension of orthogonal space, especially for large depthwise convolutional kernels. Spatially separable convolutions can be viewed as a decomposition at spatial level where a width-wise convolution followed by a height-wise convolution is composed to be an approximate replacement of original two-dimensional spatial convolution, thereby shrinking the computational complexity. However, it will suffer significant information loss if spatially separable convolutions are directly employed in network architectures [14].

In this paper, we propose an extremely separated convolutional block, dubbed XSepConv, which mixes depthwise convolution with spatially separable convolutions to form spatially separated depthwise convolutions, further reducing the parameter size and computational burden of large depthwise convolutional kernels. Considering that spatially separable convolutions lack sufficient ability to capture information except in vertical and horizontal directions, additional operations are required to capture information in other directions (e.g. diagonal direction) to avoid significant information loss. Here we employ a simple but effective operation, 2×2 depthwise convolution with improved symmetric padding strategy, to compensate for the above-mentioned side effect to a certain degree. Fig. 1(a) shows the basic structure of XSepConv, which is composed of 2×2 depthwise convolution followed by spatially separated depthwise convolutions. For spatial downsampling, the structure is illustrated in Fig. 1(b), which is divided into two downsampling phases, width-wise and height-wise, to pre-



(a) (b)

Fig. 1. (a) the basic XSepConv block. (b) the XSepConv block for spatial down sampling (2×). **NL**: nonlinearity. **DW**: depthwise convolution. **s12**: stride=(1,2). **s21**: stride=(2,1). (Better viewed in color)



(a) (b)

Fig. 2. Two symmetric padding strategies performed on four consecutive layers. (a) Symmetric padding strategy proposed in [36], different colors represent different groups in a single layer. (b) Our proposed improved symmetric padding strategy, different colors represent different layers. The direction of the arrow indicates the direction of position offset, e.g., the arrow in the left-top direction illustrates the shift in the left-top direction, which is caused by padding one more zero on the right and bottom sides of feature maps. The red line and green line represent "bad" and "good" information flow paths respectively, please refer to Section 3.2 for details. (Best viewed in color)

serve as much information as possible during downsampling. Fig. 2(b) illustrates our proposed improved symmetric padding strategy, which performs symmetric padding within four successive even-sized convolution layers instead of a single even-sized convolution layer [36] as shown in Fig. 2(a). Extensive experiments on four highly competitive benchmark datasets (CIFAR-10, CIFAR-100, SVHN and Tiny-ImageNet) show that simply replacing large depthwise convolution kernels with XSepConv in MobileNetV3-Small achieves a solid accuracy improvement with fewer parameters and FLOPs (floating point operations). Therefore, we demonstrate that XSepConv is a simple while efficient replacement of vanilla depthwise convolution with large kernel sizes, which can strike a better trade-off between accuracy and efficiency.

2 Related Work

2.1 Efficient CNNs

In recent years, with the increasing demand for deploying CNNs on resource-constrained platforms, a series of studies have been conducted in designing efficient CNNs to strike an optimal trade-off between accuracy and efficiency. SqueezeNet [11] extensively utilizes 1×1 convolution in the Fire module and achieves AlexNet-level accuracy with $50\times$ fewer parameters. Xception [2] obtains the performance improvement due to a more efficient use of model parameters by adopting depthwise separable convolutions. MobileNetV1 [7] is built on depthwise separable convolutions composed of a depthwise convolution followed by a pointwise convolution, greatly improving the computational efficiency. MobileNetV2 [25] introduces resource-efficient inverted residuals with linear bottlenecks. Based on MobileNetV2, MobileNetV3 [6] integrates squeeze-and-excitation module [8] into the bottleneck structure and applies modified swish nonlinearities, subsequently these architecture advances are blended with hardware-aware neural architecture search to build efficient models. ShuffleNets [39,18] further reduce computational cost by utilizing pointwise group convolution and channel shuffle. CondenseNet [9] combines dense connectivity with learned group convolution to promote feature re-use while eliminating redundant connections. IGCNets[38,37,27] propose interleaved group convolutions which are efficient in terms of parameter and computation. ShiftNet [35] introduces shift operations to replace expensive spatial convolutions.

Recently, neural architecture search has been leveraged to automate the model design process, thus spawning a series of efficient models, such as NASNet [40], PNASNet [16], AmoebaNet [23], FBNet [34], MnasNet [31], EfficientNet [32] and MixNets [33].

2.2 Separable Convolutions

Depthwise separable convolutions, first introduced in [26], factorize a standard convolution into a depthwise convolution followed by a pointwise convolution and can be viewed as a decomposition along the channel domain. Since Xception [2] and MobileNets [7] which are built upon depthwise separable convolutions achieved great success in terms of accuracy and efficiency trade-off, depthwise separable convolutions have attracted a lot of research efforts. Therefore, a variety of efficient models have emerged, from hand-crafted models [2,7,25,39,18,37] to models found by NAS [17,40,22,16,23,1,34,31,6,32,33].

Compared with depthwise separable convolutions, spatially separable convolutions or called asymmetric convolutions, which can be regarded as a decomposition in the spatial dimension, have been applied earlier in CNNs [19]. Spatially separable convolutions factorize a traditional two-dimensional ($k \times k$) convolution into a width-wise ($k \times 1$) convolution and a height-wise ($1 \times k$) convolution to reduce the number of parameters and the computation. However, it will result in significant information loss if the separation is directly applied to filters [14]. Several methods have been proposed to tackle this problem, e.g., by deriving an appropriate low-rank approximation using SVD [3], by minimizing the L_2 reconstruction error [13], and by applying structural constraints [14]. In addition, spatially separable convolutions are widely applied as a building unit in CNNs such as Inception-v3 [30] and Inception-v4 [28].

2.3 Kernel Sizes

In the hand-crafted efficient models, the most commonly used kernel size of depthwise convolution is 3 [2,7,25,39,18,37]. Recently, by adding depthwise convolution with various kernel sizes into the search space, larger kernels are applied to CNNs found by NAS [40,22,16,23,1,34,31,6,32,33], e.g., 5×5 kernels in MobileNetV3 [6] and 7×7 kernels in ProxylessNAS [1], which have shown their potential abilities to improve accuracy and efficiency. Besides, direct implementation of even-sized kernels ($2{\times}2$, $4{\times}4$) will encounter performance degradation due to the shift problem, and applying symmetric padding can eliminate the problem, thus improving the generalization abilities of even-sized kernels [36].

3 XSepConv

The main idea of XSepConv is to utilize spatially separable convolutions to reduce the parameter size and computational complexity of large depthwise convolution kernels, and an extra 2×2 depthwise convolution with improved symmetric padding strategy is used to compensate the side effect brought by spatially separable convolutions. In this section, we will describe the advantages of XSepConv and the improved symmetric padding strategy.

3.1 Wider and Deeper

A standard depthwise convolutional kernel $\mathbf{W} \in \mathbb{R}^{k\times k\times c}$ performs a 2D convolution on each individual channel of the input tensor $\mathbf{X} \in \mathbb{R}^{h\times w\times c}$, and produces the output tensor $\mathbf{Y} \in \mathbb{R}^{h\times w\times c}$ with the same shape, where k denotes the kernel size, h denotes the spatial height, w denotes the spatial width and c denotes the number of channels. It can be formulated as

$$\mathbf{Y}_{x,y,z} = \sum_{-\frac{k}{2}\leq i,j\leq \frac{k}{2}} \mathbf{W}_{i,j,z}\mathbf{X}_{x+i,y+j,z}. \tag{1}$$

The computational cost of the vanilla depthwise convolution is k^2hwc and the parameter size is k^2c. It can be obviously seen that the computational cost and parameter size will increase in quadratic as the kernel size increases. In addition, the receptive field size of the depthwise convolution is equal to its kernel size k.

As shown in Fig. 1(a), XSepConv consists of three parts: 2×2 depthwise convolutional kernel $\hat{\mathbf{W}}' \in \mathbb{R}^{2\times 2\times c}$, $1 \times k$ depthwise convolutional kernel $\hat{\mathbf{W}}'' \in \mathbb{R}^{1\times k\times c}$ and $k\times 1$ depthwise convolutional kernel $\hat{\mathbf{W}}''' \in \mathbb{R}^{k\times 1\times c}$. The $1\times k$

and $k \times 1$ depthwise convolutions together form what we call spatially separated depthwise convolutions, and the 2×2 depthwise convolution plays an important role in capturing information that may be missed by spatially separated depthwise convolutions due to their inherent structural defects. Taking padding one more zero on the right and bottom sides of the input tensor $\hat{\mathbf{X}} \in \mathbb{R}^{h \times w \times c}$ before 2×2 depthwise convolution as an example, the output tensor $\hat{\mathbf{Y}} \in \mathbb{R}^{h \times w \times c}$ is calculated as

$$\hat{\mathbf{Y}}_{x,y,z} = \sum_{-\frac{k}{2} \le m \le \frac{k}{2}} \sum_{-\frac{k}{2} \le n \le \frac{k}{2}} \sum_{0 \le i,j \le 1} \hat{\mathbf{W}}'_{i,j,z} \hat{\mathbf{W}}''_{0,n,z} \hat{\mathbf{W}}'''_{m,0,z} \hat{\mathbf{X}}_{x+i+m,y+j+n,z}. \tag{2}$$

XSepConv has the computational cost of $4hwc + 2khwc$ and the parameter size of $4c + 2kc$. Then the ratio of the computational cost (and also parameter size) of XSepConv to the vanilla depthwise convolution is calculated as

$$\frac{4hwc + 2khwc}{k^2 hwc} = \frac{4 + 2k}{k^2}. \tag{3}$$

Since we are focusing on replacing large depthwise convolutional kernel ($k \geq 5$) with XSepConv, the ratio is consistently less than 1, which means that XSepConv requires less computation and fewer parameters. For instance, when $k = 5$, XSepConv can save 44% of the computational cost and parameter size, and the reduction will be greater as kernel size increases. More than that, as described in Section 4, XSepConv achieves better accuracy than vanilla depthwise convolution, which we believe is because XSepConv is wider and deeper.

From Eq. 2, we can see that the receptive field size of XSepConv is $k+1$, thus XSepConv has a wider receptive field. In effect, the widening comes from the 2×2 depthwise convolution. Fig. 1(a) shows the structure of XSepConv, which is composed of three consecutive layers, making it deeper. Moreover, all three layers are followed by batch normalization [12] and nonlinearity, so XSepConv can also increase the nonlinearities of the network, thereby enhancing its representation ability.

As for the downsampling layer, the structure of XSepConv is described in Fig. 1(b). The downsampling is split into two stages: first along the width direction with stride (1,2) and then along the height direction with stride (2,1). Similarly, the ratio of the computational complexity and parameter count of downsampling XSepConv to downsampling depthwise convolution is computed as

$$\frac{4hwc + khwc/2 + khwc/4}{k^2 hwc/4} = \frac{16 + 3k}{k^2}. \tag{4}$$

Since the ratio is less than 1 only when $k \geq 7$, it is only recommended to use downsampling XSepConv to replace downsampling depthwise convolution with kernel size no less than 7.

3.2 Improved Symmetric Padding Strategy

When using even-sized convolutions, asymmetric padding, such as padding on the right and bottom sides only, is often used in order to maintain the size of feature maps, thus the activated values are shifted to the left-top corner of the spatial location, which is identified as the shift problem in [36]. This problem limits the generalization abilities of even-sized kernels and a symmetric padding strategy which introduces symmetric padding within a single convolution layer is proposed in [36] to eliminate the problem.

Fig. 2(a) illustrates the symmetric padding strategy proposed in [36], it focuses on introducing symmetry to the output of a single convolution layer. Through dividing the input tensor into four groups and padding in the left-top, right-bottom, left-bottom and right-top directions respectively, the location offset in the output feature maps of a single convolution layer is eliminated.

However, in most computer vision tasks such as image classification, the output of the last layer rather than the intermediate single layer is the most significant. As shown in Fig. 2(a), the symmetric padding strategy of [36] will encounter asymmetry at some outputs of the last layer. If the information flows along a path where all four shifted directions are different, such as the green line in Fig. 2(a), the position offset will be eliminated in the final output. Otherwise, position offsets will accumulate in at least one direction, e.g., the red line in Fig. 2(a) indicates that the position offset in the left-top direction will accumulate in the final output, which will eventually squeeze features to the left-top corner of the spatial position and result in performance degradation.

Therefore, we propose an improved symmetric padding strategy as illustrated in Fig. 2(b), which aims at the final output of the network instead of any intermediate single layer. We introduce a set of padding directions

$$\{\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3, \mathcal{D}_0\} \tag{5}$$

that in turn contains four directions: right-bottom, left-top, left-bottom and right-top. Let N be the number of layers with even-sized kernels and $p(i)$ be the padding direction of the i-th layer with even-sized kernels. If N is an integer multiple of 4, the symmetry of the final output is strictly achieved by specifying the padding directions:

$$p(i) = \mathcal{D}_{i\%4}, \tag{6}$$

where % denotes the modulo operation. Even if N is not an integer multiple of 4, we can also ensure that the final position offset does not exceed 1 pixel in any direction. Here the order of padding directions counts, for example, in case of $N\%4 = 2$, the two padding directions of the last two layers with even-sized kernels are opposite, so as to eliminate the shift as much as possible. In addition, if $N\%4 = 1$, the last layer with even-sized kernels can use the original symmetric padding strategy to achieve slightly better symmetry.

4 Experiments

We conduct extensive experiments on four widely used image classification benchmark datasets: CIFAR-10 and CIFAR-100 [15], SVHN [20] and Tiny-ImageNet[3],

[3] `https://tiny-imagenet.herokuapp.com/`

based on the state-of-the-art models MobileNetV3-Small [6]. Each experiment was repeated 5 times to eliminate the effect brought by random initialization. In addition, all the architectures are implemented by using PyTorch [21].

4.1 Datasets and Training Settings

CIFAR: The two CIFAR datasets [15], i.e., CIFAR-10 and CIFAR-100, both contain 60,000 colored natural images with a size of 32 × 32, of which 50,000 images are for training and 10,000 images are for test. The major difference between CIFAR-10 and CIFAR-100 is that they consist of different numbers of classes. CIFAR-10 contains 10 classes, which means that each class consists of 5,000 training images and 1,000 test images. Similarly, CIFAR-100 contains 100 classes, each of which consists of 500 training images and 100 test images. We follow the most common data augmentation scheme [4,10]: first pad 4 zeros on each sides of the images and then randomly crop them to the size of 32 × 32, followed by randomly flipping the images horizontally. We finally normalize the images with the channel means and standard deviations.

SVHN: The Street View House Numbers (SVHN) [20] dataset consists of 10 classes, each of which corresponding to a certain number between 0 and 9. It contains real-world colored digit images of resolution 32 × 32. There are 73,257 training images, 26,032 test images and 531,131 additional training images. We use both the training and additional data for training without any data augmentation [10]. Normalization with the channel means and standard deviations is also performed.

Tiny-ImageNet: The Tiny-ImageNet dataset, which consists of 200 classes drawn from 1,000 classes of ImageNet [24], is actually a subset of ImageNet dataset. There are 500 training images, 50 validation images and 50 test images per class. The images are resized to 64×64, making Tiny-ImageNet more difficult to learn due to the loss of detailed information during downsampling. We follow the data augmentation scheme for training: crop the image with the size no less than 8% of the image area and the aspect ratio limited to the interval $[3/4, 4/3]$ as in [29], resize to 56×56 and randomly flip the image horizontally. Normalization with the channel means and standard deviations is used in the end.

Training Settings: All networks are optimized using stochastic gradient descent (SGD) with momentum 0.9. We use batch normalization after every convolution layer, and the weight decay is set to 6e-5 and batch size to 128. Following [5], we use cosine learning rate decay and a gradual learning rate warmup strategy for the first 5 epochs. On CIFAR and SVHN we train for 400 and 20 epochs, respectively, with an initial learning rate of 0.35. For Tiny-ImageNet, the initial learning rate is set to 0.15 and we train models for 200 epochs.

Table 1. Performance comparison of XSepConv with vanilla depthwise convolution on 4 datasets. "Params" refers to the number of parameters

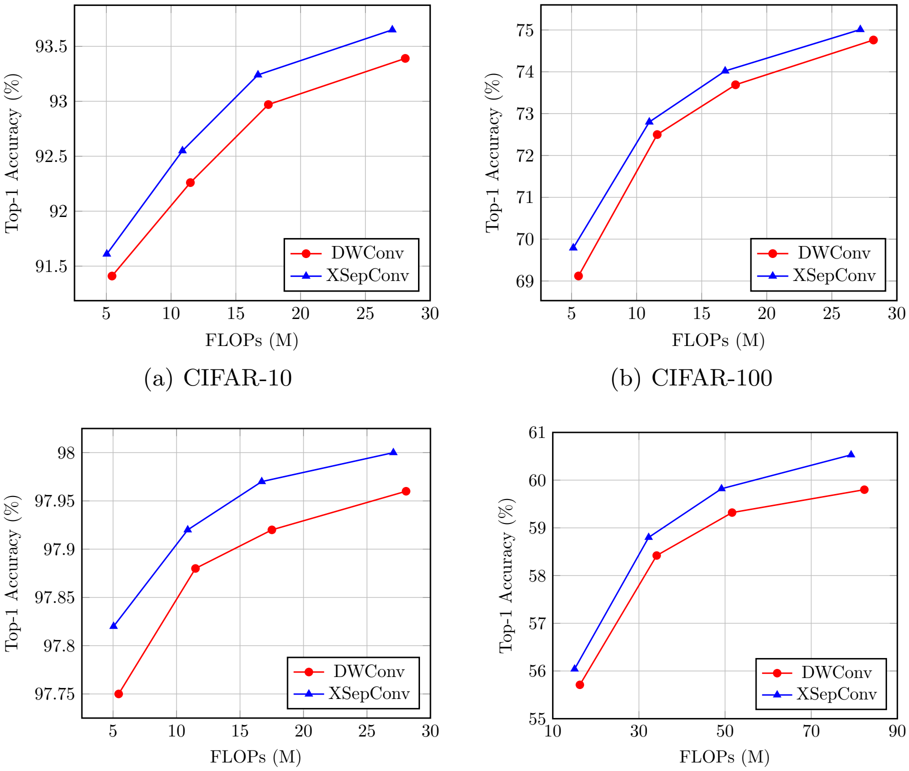
Convolution	Datasets	FLOPs	Params	Top-1 accuracy (%)
DWConv	CIFAR-10	17.51M	1.52M	92.97
XSepConv		**16.71M**	**1.50M**	**93.24**
DWConv	CIFAR-100	17.60M	1.61M	73.69
XSepConv		**16.80M**	**1.59M**	**74.02**
DWConv	SVHN	17.51M	1.52M	97.92
XSepConv		**16.71M**	**1.50M**	**97.97**
DWConv	Tiny-ImageNet	51.63M	1.71M	59.32
XSepConv		**49.18M**	**1.70M**	**59.82**

4.2 XSepConv Performance

Trade-off between Accuracy and Efficiency: To verify that XSepConv is an efficient replacement of vanilla depthwise convolution with large kernel, we evaluate its performance on 4 highly competitive image classification datasets with the state-of-the-art architecture MobileNetV3-Small [6], which extensively utilizes 5 × 5 depthwise convolution. We re-implement the MobileNetV3-Small and replace the 5 × 5 depthwise convolutions in the middle stage of the network with our proposed XSepConv of $k = 5$. In particular, we replace the 5 × 5 depthwise convolutions in the last stage (after the last downsampling layer) with XSepConv of $k = 3$ to gain more computational reduction. All downsampling layers keep using the original depthwise convolutions because the kernel size is less than 7. In addition, all the training details are the same for fair comparison.

Table 1 shows the classification performance of XSepConv compared with vanilla depthwise convolution (DWConv). We compare the computational complexity (indicated by FLOPs), parameter size and top-1 accuracy on 4 datasets. It can be seen that XSepConv consistently obtains higher accuracy with fewer parameters and smaller computational complexity than vanilla depthwise convolution on all 4 datasets, illustrating that XSepConv is more efficient. For example, XSepConv acquires an additional performance improvement of 0.5% while saving 4.75% of the computational cost on Tiny-ImageNet. For more intuitive comparison, we provide the trade-off curves between top-1 accuracy and FLOPs on all 4 datasets in Fig. 3. Without bells and whistles, simply replacing ordinary depthwise convolution with XSepConv reliably improves the accuracy with fewer FLOPs under various computational complexity, indicating that XSepConv is able to achieve a more excellent trade-off between accuracy and efficiency.

Downsampling: To evaluate the effectiveness of downsampling XSepConv as shown in Fig. 1(b), we enlarge the kernel size of the first 5 × 5 depthwise con-



(a) CIFAR-10

(b) CIFAR-100

(c) SVHN

(d) Tiny-ImageNet

Fig. 3. Top-1 accuracy versus FLOPs on 4 datasets with various width multipliers. For each curve we use 4 different width multipliers 0.5, 0.75, 1.0 and 1.25. Curves further to the top left are more efficient in terms of accuracy and FLOPs trade-off. XSepConv outperforms its counterpart on all 4 datasets

volution of stride 2 to 7 × 7 and then replace it with downsampling XSepConv of $k = 7$. The trade-off curves on Tiny-ImageNet are displayed in Fig. 4. With fewer FLOPs, downsampling XSepConv obtains higher accuracy, proving that downsampling XSepConv indeed strikes a better trade-off between accuracy and efficiency than vanilla downsampling depthwise convolution.

Larger Kernels: Furthermore, to confirm that XSepConv can still outperform depthwise convolution with even larger kernels, we increase the kernel size of 5 × 5 depthwise convolution in the middle stage to 7 × 7 and then replace it with XSepConv of $k = 7$. As shown in Fig. 5, we observe that XSepConv gains more reductions in FLOPs as the kernel size increases from 5 to 7 compared to Fig. 3(d), but still achieves better accuracy than vanilla depthwise convolution, suggesting that XSepConv is a more efficient alternative to depthwise

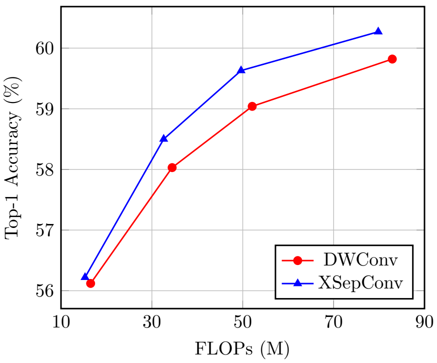


Fig. 4. Comparison of downsampling XSepConv with vanilla downsampling depthwise convolution of kernel size 7 on Tiny-ImageNet

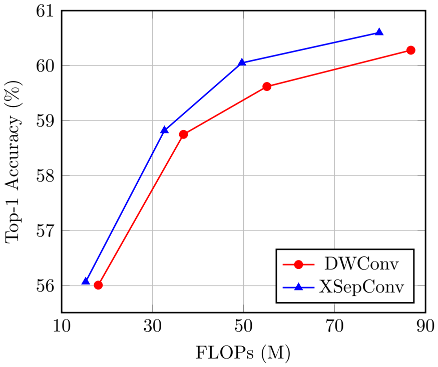


Fig. 5. Comparison of XSepConv with vanilla depthwise convolution of larger kernel size 7 on Tiny-ImageNet

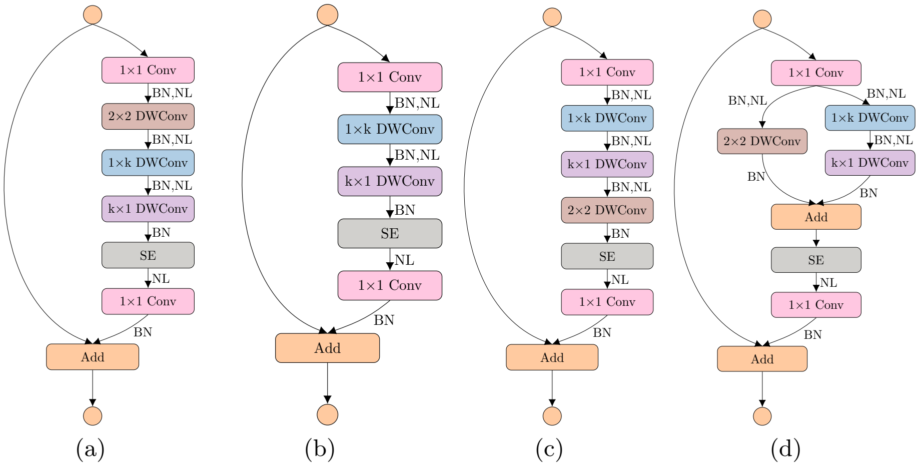


Fig. 6. Various building blocks based on the building block of MobileNetV3-Small, where the depthwise convolution is replaced with different structural designs of XSepConv: (a) standard XSepConv; (b) without 2×2 depthwise convolution; (c) 2×2 depthwise convolution placed behind spatially separated depthwise convolutions; (d) 2×2 depthwise convolution in parallel to spatially separated depthwise convolutions. **BN**: batch normalization [12]. **SE**: Squeeze-and-Excitation module [8]

convolution. Interestingly, combining Fig. 3(d) and Fig. 5, we observe that both XSepConv and depthwise convolution with larger kernels exhibit higher accuracy, indicating the potentiality of larger kernels to improve model accuracy.

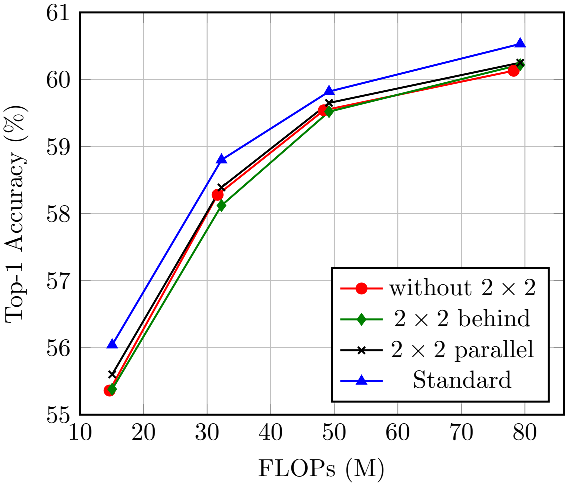


Fig. 7. Trade-off curves of different structural designs in Fig. 6 on Tiny-ImageNet

4.3 Ablation Studies

In this section, we conduct a series of ablation experiments to shed light on the impact of different structural designs as shown in Fig. 6. We also perform ablation experiments to investigate the impact of different symmetric padding strategies.

Importance of 2 × 2 Depthwise Convolution: To assess whether 2 × 2 depthwise convolution plays an important part in boosting the performance, we remove the 2 × 2 depthwise convolution in XSepConv as shown in Fig. 6(b). Table 2 illustrates the consistent decrease in accuracy on all 4 datasets although fewer parameters and less computational cost are required due to the removal of 2 × 2 depthwise convolution. For instance, removing 2 × 2 depthwise convolution results in an accuracy drop of about 0.25% with quite little decrease in FLOPs on CIFAR-10. In order to provide an intuitive perception of the impact of the removal of 2 × 2 depthwise convolution, the trade-off curves are displayed in Fig. 7, clearly revealing the worse trade-off without 2 × 2 depthwise convolution. Despite reducing the computational complexity, removing 2 × 2 depthwise convolution leads to a more significant reduction in accuracy and eventually makes the trade-off worse. We also observe that the accuracy gap becomes more pronounced with fewer FLOPs. This can be attributed to the inherent structural defects of spatially separated depthwise convolutions, which will miss information that may be more important as the model becomes smaller and less redundant. Consequently, the 2 × 2 depthwise convolution in XSepConv plays an indispensable role in capturing information that may be missed by spatially separated depthwise convolutions but is important to the performance of the model.

Table 2. Performance study for 2 × 2 depthwise convolution on 4 datasets

XSepConv	Dataset	FLOPs	Params	Top-1 accuracy (%)
w/ 2×2	CIFAR-10	16.71M	1.50M	**93.24**
w/o 2×2		**16.45M**	**1.49M**	92.99
w/ 2×2	CIFAR-100	16.80M	1.59M	**74.02**
w/o 2×2		**16.54M**	**1.58M**	73.89
w/ 2×2	SVHN	16.71M	1.50M	**97.97**
w/o 2×2		**16.45M**	**1.49M**	97.92
w/ 2×2	Tiny-ImageNet	49.18M	1.70M	**59.82**
w/o 2×2		**48.37M**	**1.68M**	59.62

Table 3. Performance comparison of different structural designs and padding strategy. FLOPs and parameters are the same and therefore not reported. "Original-Padding" refers to the original symmetric padding strategy proposed in [36]

Design	Top-1 Accuracy (%) on 4 datasets			
	CIFAR-10	CIFAR-100	SVHN	Tiny-ImageNet
XSepConv	**93.24**	**74.02**	**97.97**	**59.82**
XSepConv-B	93.03	73.53	97.94	59.62
XSepConv-P	93.06	73.14	97.91	59.65
Original-Padding	93.05	73.81	97.94	58.58

Location of 2 × 2 Depthwise Convolution: In order to evaluate the impact of the location of 2 × 2 depthwise convolution, we consider two other variants of XSepConv: (1) XSepConv-B, where the 2 × 2 depthwise convolution is located behind spatially separated depthwise convolutions as shown in Fig. 6(c); (2)XSepConv-P, in which the 2 × 2 depthwise convolution is placed in parallel to spatially separated depthwise convolutions as described in Fig. 6(d). The performance of each variant on 4 datasets is reported in Table 3 and the trade-off curves are displayed in Fig. 7. We observe that XSepConv-B and XSepConv-P both result in a drop in accuracy on all 4 datasets, with the same parameters and computational cost as standard XSepConv.

Fig. 7 shows a similar trade-off between XSepConv-B and the XSepConv without 2 × 2 depthwise convolution. The reason for the performance degradation of XSepConv-B is that the 2 × 2 depthwise convolution loses its role in capturing missing information since the information has been irreversibly partially lost after flowing through spatially separated depthwise convolutions. As a result, XSepConv-B performs similarly to the XSepConv without 2×2 depthwise convolution.

As for XSepConv-P, it also encounters performance degradation on all 4 datasets as illustrated in Table 3, especially on CIFAR-100, where an accuracy decrease of 0.88% is observed. The performance degradation can be attributed to the loss of the advantages of wider receptive field and greater depth. These advantages are replaced by a greater width, which in this experiment is inferior to the combination of a wider receptive field and a greater depth.

Symmetric Padding Strategies: As mentioned in Section 3.2, the original symmetric padding strategy will still encounter asymmetry in the final output of the network and we propose an improved symmetric padding strategy to overcome this shortcoming. The comparison of the two padding strategies is reported in Table 3. Compared with original symmetric padding strategy, our proposed improved symmetric padding strategy takes the final output of the network as the major concern, thereby achieving consistent performance improvement of image classification tasks on various datasets. For example, an accuracy improvement of 0.21% and 0.24% are reported on CIFAR-100 and Tiny-ImageNet, respectively. In addition, it should be noticed that in this experiments, $N\%4 = 2$, where N is the number of 2×2 depthwise convolution layers. This is not an ideal situation, but we can still attain stable performance improvement. Therefore, we have reason to believe that the performance improvement would be even greater in the case of $N\%4 = 0$.

5 Conclusions

In this paper, we aim to achieve a better trade-off between accuracy and efficiency for large depthwise convolutional kernels, which have shown their tendency to be employed in an increasing number of models, especially those found by NAS. To this end, we introduce XSepConv, which fuses spatially separable convolutions into depthwise convolution and adopt an extra 2×2 depthwise convolution coupled with improved symmetric padding strategy.

A wide range of experiments on multiple datasets show that compared to vanilla depthwise convolution with large kernels, XSepConv attains a further decrease in both computational budget and parameter size while achieves a solid performance improvement for image classification on various datasets. Moreover, we carry out a series of ablation experiments to further verify the effectiveness of the proposed XSepConv and the improved symmetric padding strategy.

Our proposed XSepConv is a more efficient alternative to depthwise convolution and can reliably boost the performance by simply replacing depthwise convolution in models such as MobileNetV3-Small with XSepConv. Furthermore, XSepConv can be adopted in NAS to enrich the search space and develop more efficient models.

Acknowledgements

This work is supported by the Special Foundation for the Development of Strategic Emerging Industries of Shenzhen under Grant JCYJ20170817161056260.

References

1. Cai, H., Zhu, L., Han, S.: ProxylessNAS: Direct neural architecture search on target task and hardware. In: International Conference on Learning Representations (2019), `https://openreview.net/forum?id=HylVB3AqYm`
2. Chollet, F.: Xception: Deep learning with depthwise separable convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1251–1258 (2017)
3. Denton, E.L., Zaremba, W., Bruna, J., LeCun, Y., Fergus, R.: Exploiting linear structure within convolutional networks for efficient evaluation. In: Advances in neural information processing systems. pp. 1269–1277 (2014)
4. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
5. He, T., Zhang, Z., Zhang, H., Zhang, Z., Xie, J., Li, M.: Bag of tricks for image classification with convolutional neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 558–567 (2019)
6. Howard, A., Sandler, M., Chu, G., Chen, L.C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., et al.: Searching for MobileNetV3. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 1314–1324 (2019)
7. Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: MobileNets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861 (2017)
8. Hu, J., Shen, L., Sun, G.: Squeeze-and-excitation networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 7132–7141 (2018)
9. Huang, G., Liu, S., Van der Maaten, L., Weinberger, K.Q.: CondenseNet: An efficient DenseNet using learned group convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2752–2761 (2018)
10. Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4700–4708 (2017)
11. Iandola, F.N., Han, S., Moskewicz, M.W., Ashraf, K., Dally, W.J., Keutzer, K.: SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and$<$ 0.5 mb model size. arXiv preprint arXiv:1602.07360 (2016)
12. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167 (2015)
13. Jaderberg, M., Vedaldi, A., Zisserman, A.: Speeding up convolutional neural networks with low rank expansions. arXiv preprint arXiv:1405.3866 (2014)
14. Jin, J., Dundar, A., Culurciello, E.: Flattened convolutional neural networks for feedforward acceleration. arXiv preprint arXiv:1412.5474 (2014)
15. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images. Tech. rep. (2009)
16. Liu, C., Zoph, B., Neumann, M., Shlens, J., Hua, W., Li, L.J., Fei-Fei, L., Yuille, A., Huang, J., Murphy, K.: Progressive neural architecture search. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 19–34 (2018)

17. Liu, H., Simonyan, K., Yang, Y.: DARTS: Differentiable architecture search. arXiv preprint arXiv:1806.09055 (2018)
18. Ma, N., Zhang, X., Zheng, H.T., Sun, J.: ShuffleNet V2: Practical guidelines for efficient CNN architecture design. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 116–131 (2018)
19. Mamalet, F., Garcia, C.: Simplifying convnets for fast learning. In: International Conference on Artificial Neural Networks. pp. 58–65. Springer (2012)
20. Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., Ng, A.Y.: Reading digits in natural images with unsupervised feature learning. In: NIPS Workshop on Deep Learning and Unsupervised Feature Learning. vol. 2011, p. 5 (2011)
21. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: PyTorch: An imperative style, high-performance deep learning library. In: Advances in Neural Information Processing Systems. pp. 8024–8035 (2019)
22. Pham, H., Guan, M., Zoph, B., Le, Q., Dean, J.: Efficient neural architecture search via parameter sharing. In: International Conference on Machine Learning. pp. 4092–4101 (2018)
23. Real, E., Aggarwal, A., Huang, Y., Le, Q.V.: Regularized evolution for image classifier architecture search. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 33, pp. 4780–4789 (2019)
24. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al.: ImageNet large scale visual recognition challenge. International journal of computer vision **115**(3), 211–252 (2015)
25. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: MobileNetV2: Inverted residuals and linear bottlenecks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4510–4520 (2018)
26. Sifre, L., Mallat, S.: Rigid-motion scattering for image classification. Ph. D. dissertation (2014)
27. Sun, K., Li, M., Liu, D., Wang, J.: IGCV3: Interleaved low-rank group convolutions for efficient deep neural networks. arXiv preprint arXiv:1806.00178 (2018)
28. Szegedy, C., Ioffe, S., Vanhoucke, V., Alemi, A.A.: Inception-v4, Inception-ResNet and the impact of residual connections on learning. In: Thirty-First AAAI Conference on Artificial Intelligence (2017)
29. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1–9 (2015)
30. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2818–2826 (2016)
31. Tan, M., Chen, B., Pang, R., Vasudevan, V., Sandler, M., Howard, A., Le, Q.V.: MnasNet: Platform-aware neural architecture search for mobile. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2820–2828 (2019)
32. Tan, M., Le, Q.: EfficientNet: Rethinking model scaling for convolutional neural networks. In: International Conference on Machine Learning. pp. 6105–6114 (2019)
33. Tan, M., Le, Q.V.: MixConv: Mixed depthwise convolutional kernels. ArXiv, abs/1907.09595 **7** (2019)
34. Wu, B., Dai, X., Zhang, P., Wang, Y., Sun, F., Wu, Y., Tian, Y., Vajda, P., Jia, Y., Keutzer, K.: FBNet: Hardware-aware efficient ConvNet design via differentiable neural architecture search. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 10734–10742 (2019)

35. Wu, B., Wan, A., Yue, X., Jin, P., Zhao, S., Golmant, N., Gholaminejad, A., Gonzalez, J., Keutzer, K.: Shift: A zero flop, zero parameter alternative to spatial convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 9127–9135 (2018)
36. Wu, S., Wang, G., Tang, P., Chen, F., Shi, L.: Convolution with even-sized kernels and symmetric padding. arXiv preprint arXiv:1903.08385 (2019)
37. Xie, G., Wang, J., Zhang, T., Lai, J., Hong, R., Qi, G.J.: Interleaved structured sparse convolutional neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 8847–8856 (2018)
38. Zhang, T., Qi, G.J., Xiao, B., Wang, J.: Interleaved group convolutions. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 4373–4382 (2017)
39. Zhang, X., Zhou, X., Lin, M., Sun, J.: ShuffleNet: An extremely efficient convolutional neural network for mobile devices. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 6848–6856 (2018)
40. Zoph, B., Vasudevan, V., Shlens, J., Le, Q.V.: Learning transferable architectures for scalable image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 8697–8710 (2018)