All You Need is a Few Shifts: Designing Efficient Convolutional Neural Networks for Image Classification

Weijie Chen, Di Xie, Yuan Zhang, Shiliang Pu
Hikvision Research Institute, Hangzhou, China
{chenweijie5, xiedi, zhangyuan, pushiliang}@hikvision.com

Abstract

Shift operation is an efficient alternative over depthwise separable convolution. However, it is still bottlenecked by its implementation manner, namely memory movement. To put this direction forward, a new and novel basic component named Sparse Shift Layer (SSL) is introduced in this paper to construct efficient convolutional neural networks. In this family of architectures, the basic block is only composed by 1x1 convolutional layers with only a few shift operations applied to the intermediate feature maps. To make this idea feasible, we introduce shift operation penalty during optimization and further propose a quantization-aware shift learning method to impose the learned displacement more friendly for inference. Extensive ablation studies indicate that only a few shift operations are sufficient to provide spatial information communication. Furthermore, to maximize the role of SSL, we redesign an improved network architecture to Fully Exploit the limited capacity of neural Network (FE-Net). Equipped with SSL, this network can achieve 75.0% top-1 accuracy on ImageNet with only 563M M-Adds. It surpasses other counterparts constructed by depthwise separable convolution and the networks searched by NAS in terms of accuracy and practical speed.

1. Introduction

Owing to the amazing performance of convolutional neural networks (CNNs), it becomes a big trend to apply CNNs to practical application scenarios. However, it is hindered by their substantial computational cost and storage overhead, which motivates lots of researchers and engineers to gush into this subject.

One of the useful solutions to tackle this problem is to design accurate and compact neural network architectures directly. A well-designed network topology as well as a hardware-friendly basic component can bring about surprising breakthroughs. Recently, a popular basic component named depthwise separable convolution is welcomed to design lightweight architectures, such as MobileNet [10] and ShuffleNet [40]. Despite its lower float point operations (FLOPs), it is inefficient to implement in practice because of the fragmented memory footprints. To jump out of the constraint of depthwise separable convolution, ShiftNet [37] propose another alternative, say shift operation, to construct architectures cooperated with point-wise convolution. In this network, shift operation provides spatial information communication through shifting feature maps, which makes the followed point-wise convolution layer not only available for channel information aggregation but also available for spatial information aggregation.

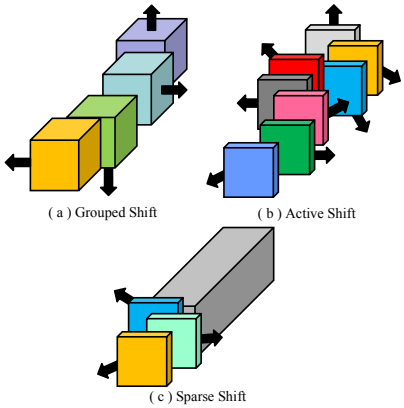

Figure 1. The comparison of different shift operations applied to feature maps.

In order to compare these two basic components, we decompose the occupied time of each basic component of ShiftNet for detailed analysis on both compute-bound and memory-bound computation platforms. As illustrated in Fig.2 (a) and (b), shift operation occupies 3.6% of runtime on CPU, but occupies 28.1% on GPU, indicating that shift operation still occupies considerable runtime on memory-bound computation platforms due to memory movement. As for depthwise separable convolution, in MobileNetV2, it occupies about 36% runtime on GPU. However, it is unfair to compare these two components in two different architec-

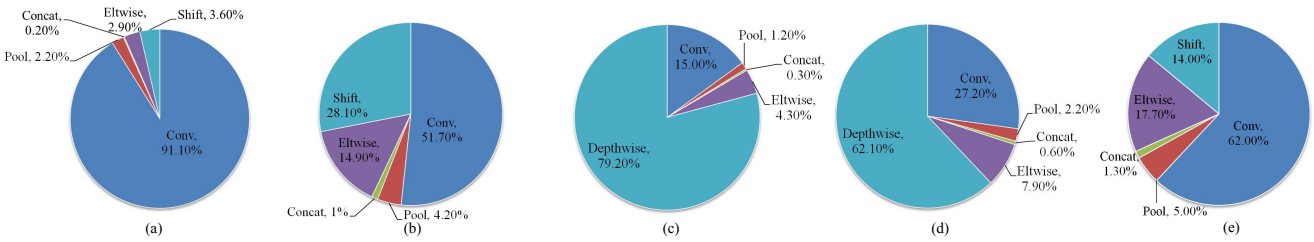

Figure 2. The practical runtime analysis. For clear comparison, both batch-normalization and ReLU layers are neglected since they can be merged into convolutional layer for inference. Also data feeding and preprocessing time are not considered here. Results are achieved under Caffe with mini-batch 32. They are averaged from 100 runs. (a) ShiftNet-A [37] on CPU (Intel Xeon E5-2650, atlas). (b) ShiftNet-A on GPU (TITAN X Pascal, CUDA8 and cuDNN5). (c) Shift layers in ShiftNet-A are replaced by depthwise separable convolution layers. (d) Depthwise separable convolution layers with kernel size 5 are replaced by the ones with kernel size 3. (e) ShiftNet-A with 80% shift sparsity on GPU (Shift sparsity denotes the ratio of unshifted feature maps).

tures. For fair comparison, we use the same architecture with ShiftNet and only replace shift operation by depthwise separable convolution to test its inference time. As shown in Fig.2 (c), it occupies 79.2% of runtime on GPU which seriously mismatches its theoretical FLOPs. From this viewpoint, shift operation is significantly superior to depthwise separable convolution. Also, another attractive characteristic of shift operation is its irrelevance of computational cost to kernel size, while the practical runtime of depthwise separable convolution is strongly influenced by kernel size. As illustrated in Fig.2 (c) and (d), the occupied runtime of depthwise separable convolution is lowered to 62.1% after decreasing the kernel size 5[1] to 3.

Despite the superiority of shift operation in terms of practical runtime to depthwise separable convolution, it is still bottlenecked by its implementation, namely memory movement. Here naturally comes a question, *is each shift operation really necessary?* Those moving memory can be reduced if the meaningless shifts are eliminated. Bringing this question, we make a further study about shift operation. To suppress redundant shift operation, penalty is added during its optimization. We surprisingly find that a few shift operations are actually sufficient to provide spatial information communication. It can provide comparable performance with shifting a small portion of feature maps. We name this type of shift layer as *Sparse Shift Layer* (SSL) in order to distinguish from other types of shift layers as shown in Fig.1. As shown in Fig.2 (e), it can significantly reduce the occupied time of shift operation after inducing sparsity.

The prerequisite of SSL is to ensure shift operation learnable. A common solution is to relax the displacement from integer to real-value and relax shift operation to bilinear interpolation so as to make it differentiable [16]. However, interpolation cannot bring the same inference benefit as shift operation. Borrowing the idea from QNN [13], we propose a quantization-aware shift learning method to enable shift operation differentiable while avoiding interpolation during inference.

When designing the compact network architecture, a straightforward guideline is to ensure the information flow while maintaining the feature maps diversity. We hope it can contain label-related information as abundant as possible in the limited feature space. However, the feature maps usually tend to collapse into a small subset, which does not fully exploit the limited feature space. To ease this problem, we design a novel network architecture FE-Net as shown in Fig.3, which involves feature maps into computation progressively as layer increases to impose diversity while avoiding redundant overhead.

In this paper, we mainly conduct experiments on image classification benchmarks. Extensive ablation studies on CIFAR-10 and CIFAR100 validate the impact of SSL. Furthermore, we carry out experiments on a large-scale image classification dataset ImageNet to confirm the efficiency and the generalization of SSL. With network architecture improvement, we surpass ShiftNet and AS-ResNet [16] by a large margin. *It is worth highlighting that our network even surpasses other counterparts composed by depthwise separable convolution.* We achieve **75.0%** top-1 accuracy on ImageNet with **563M** M-Adds. This is the first time the compact networks can achieve such high accuracy in this level of computational cost without using depthwise separable convolution. Equipped with *Squeeze-and-Excitation* module [11] in a proper way, our network can be further boosted to **76.5%** top-1 accuracy with **566M** M-Adds.

In summary, our main contributions are listed as follows:

- A new basic component named *Sparse Shift Layer* is introduced to build fast and accurate neural networks, which can eliminate meaningless memory movement. Beyond this, through extensive ablation studies, we find that only a few shift operations are sufficient to provide spatial information communication, which

[1] In ShiftNet, there are 11 shift layers with kernel size 5.

will inspire more exploration in the development of compact neural networks.

- A quantization-aware shift learning method is proposed to ensure shift operation learnable while avoiding interpolation during inference.
- An improved compact network architecture is designed to fully exploit the capacity of the limited feature space. Combining it with SSL, we achieve state-of-the-art results in classification benchmarks in terms of both accuracy and inference speed.

2. Related Works

Over the past few years, more and more approaches are proposed to lighten neural networks in terms of storage, computation and the practical inference time, while keeping their performance powerful. We divide these related methods into the following two parts from the view of whether a pretrained model is given.

2.1. Neural Networks Compression

To compress a given pretrained model into a lightweight one, there exist four different approaches: 1) Pruning [6, 20, 5, 35, 21, 27, 8, 25, 24, 1, 2] aims to remove unimportant parameters and turns weight matrices into sparse ones. 2) Tensor decomposition [3, 15, 19, 17, 36, 34] exploits the channel or spatial redundancy of weight matrices and seeks their low-rank approximations. 3) Quantization [4, 13, 28] adopts low bits instead of float point representation for each weight parameter. 4) Knowledge distillation [9, 31] transfers the knowledge from teacher models to lightweight student models.

These methods can effectively compress neural networks into small ones. However, their performances heavily depend on the given pretrained model. Without architecture improvement, the accuracy cannot go a step further.

2.2. Compact Networks Development

How to design a compact neural architecture is a popular research topic recently. Some related works [14, 39] used group convolution to construct compact networks. The most famous work, MobileNet [10], adopts depthwise separable convolution to build an accurate and lightweight network, which moves forward a big step in this field. After that, lots of researchers follow these works and design more compact and powerful architectures, such as ShuffleNet, MobileNetV2, ShuffleNetV2, IGCV2 and so on [40, 30, 26, 38]. However, even though depthwise separable convolution only needs little theoretical computation cost, it is difficult to implement efficiently in practice since the arithmetic intensity is too low.

[37] provide an alternative named shift operation which only shifts feature maps without computation. A compact network can be constructed by interleaving this operation with point-wise convolutions. Before that, a random shift operation [42] is applied to pooling layer to enhance the generalization of networks, serving as an alternative of data augmentation. [16] propose a method to make shift operation learnable, which means the receptive field of each layer can be learnt automatically. The existing problem is that this operation still occupies considerable inference time because it is implemented by memory movement. This is exactly what we want to solve in this paper.

3. Background

We first review the standard shift operation, which can be formulated as follows:

$$O_{c,i,j} = I_{c,i+\alpha_c,j+\beta_c} \tag{1}$$

where I and O are the input and output feature maps, respectively. c is the channel index. i and j denote the spatial position. α_c and β_c denote the horizontal and vertical displacement assigned to c_{th} input feature map. The parameter number of α and β is separately equivalent to the channel number of input feature maps, which is almost negligible compared with the parameters of convolution layer.

Grouped shift. In the work of [37], for a shift operation with kernel size K, the input feature maps are evenly divided into K^2 groups, and each group is assigned one displacement as illustrated in Fig.1(a). This displacement assignment can be formulated as follows:

$$\begin{aligned} \alpha_c &= \lfloor \lfloor c/K^2 \rfloor \ / \ K \rfloor - \lfloor K/2 \rfloor, \\ \beta_c &= \lfloor c/K^2 \rfloor \bmod K - \lfloor K/2 \rfloor, \end{aligned} \tag{2}$$

where $\lfloor \cdot \rfloor$ denotes floor function. However, the heuristic assignment is not task-driven. The kernel size of each shift operation is set through lots of trial-and-error experiments, and the uniform distribution of the displacement is not generally suitable for every task.

Active shift. To solve this problem, [16] proposes a method to make α and β differentiable, which relaxes the integer constraint of α and β to real value and relaxes shift operation to bilinear interpolation. In this manner, Eqn.1 can be relaxed as follows:

$$O_{c,i,j} = \sum_{(n,m)\in\Omega} I_{c,n,m} \cdot (1 - |i + \alpha_c - n|)(1 - |j + \beta_c - m|) \tag{3}$$

where Ω is the neighbor set of $(i + \alpha_c, j + \beta_c)$ composed by four nearest integer points. Hence, α and β can be optimized adaptively by gradient descent optimizers through backpropagation. This shift pattern is illustrated in Fig.1(b).

4. Designing Efficient Convolutional Neural Networks with a Few Shifts

It demonstrates in the works of [16, 37] that shift operation can provide receptive field for spatial information com-

munication in ConvNets. However, not each feature map is required to shift. Redundant shift operation will bring redundant memory movement and further impact the inference time of neural network. Starting from this point, we develop a method in this section to build efficient ConvNets with fewer shift operations.

4.1. Sparsifying Shift Operation

To avoid meaningless memory movement, we add displacement penalty to eliminate useless shift operation in loss function. Also, it can avoid diffusion of shift learning since a big displacement will induce useful boundary information loss especially for those feature maps with lower resolution. To this end, we add L1 regularization to α and β to penalize redundant shifts, which is formulated as follows:

$$\begin{aligned} \mathcal{L}_{total}{=}\textstyle\sum_{(x,y)} \mathcal{L}(f(x \mid W, \alpha, \beta), y) + \lambda \mathcal{R}(\alpha, \beta) \\ \mathcal{R}(\alpha, \beta) = \parallel \alpha \parallel_1 + \parallel \beta \parallel_1 \end{aligned} \tag{4}$$

where (x, y) is the input data and its corresponding label, W denotes the trainable parameters except α and β, $f(\cdot)$ outputs the predicted label, $\mathcal{L}(\cdot)$ is the loss function of neural networks, and λ balances these two terms.

With such sparsity-induced penalty, we can adopt minimum memory movement to build an accurate and fast neural network. We name this new component as sparse shift layer (SSL), which is illustrated in Fig.1(c), to distinguish from the previous shift operations .

4.2. Quantization-aware Shift Learning

Despite flexibility and sparsity are introduced, some problems remain unsolved. Although the integer constraint of α and β is relaxed to real value for the sake of learning shift operation, it weakens the inference advantage of shift operation to some extent since interpolation still needs multiplications while standard shift operation only needs memory movement during inference.

Inspired by the method of training quantization neural networks [13], we propose a quantization-aware shift learning approach to make these problems tractable. In this approach, we aim to quantize the displacement back to integer during feed-forward, while keeping shift operation still learnable.

Feed-forward. We use integer approximation of α and β to recover shift operation instead of interpolation, which can be formulated as follows:

$$O_{c,i,j} = I_{c,i+\lfloor \alpha_c \rceil_\dagger, j+\lfloor \beta_c \rceil_\dagger} \tag{5}$$

where $\lfloor \cdot \rceil_\dagger$ denotes the rounding approximation of real value. In this way Eqn.3 is actually converted back to Eqn.1 through quantization, meaning that we apply shift operation instead of interpolation to compute the loss of network.

Back-propagation. Different from feed-forward phase, real-valued shift is required to compute their gradients and optimized through Stochastic Gradient Descent (SGD). According to Eqn.3, the gradients of loss with respect to α and β are formulated as follows:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \alpha_c} = & \sum_i^w \sum_j^h \frac{\partial \mathcal{L}}{\partial O_{c,i,j}} \sum_{(n,m)\in\Omega} I_{c,n,m} \cdot \\ & (1 - |j + \beta_c - m|) \cdot \mathrm{Sign}(n - i - \alpha_c) \\ \frac{\partial \mathcal{L}}{\partial \beta_c} = & \sum_i^w \sum_j^h \frac{\partial \mathcal{L}}{\partial O_{c,i,j}} \sum_{(n,m)\in\Omega} I_{c,n,m} \cdot \\ & (1 - |i + \alpha_c - n|) \cdot \mathrm{Sign}(m - j - \beta_c) \end{aligned} \tag{6}$$

where w and h are the spatial size of input feature maps. And $\mathrm{Sign}(\cdot)$ is a function to output $+1$ or -1 according to the sign of input value.

As to back-propagate the gradients of loss with respect to the feature maps from higher layers to shallow layers, both Eqn.3 and 5 work to compute the partial derivation. Consider Eqn.5 is the actual feed-forward process we apply instead of Eqn.3. It is more reasonable and efficient to adopt Eqn.5 to compute the gradients, which is formulated as:

$$\frac{\partial \mathcal{L}}{\partial I_{c,i,j}} = \frac{\partial \mathcal{L}}{\partial O_{c,i-\lfloor \alpha_c \rceil_\dagger, j-\lfloor \beta_c \rceil_\dagger}} \tag{7}$$

This is an inverse memory movement compared with Eqn.5.

Discussion. After training, the rounding approximation of displacement is preserved and only shift operation is executed during inference. What's more, another surprising by-product is that this method turns L1 regularizer into a *truncated* regularizer, which will shrinks more small displacement towards exact zero.

4.3. Network Architecture Improvement

The network capacity is not always fully exploited. As demonstrated in the works of cross-channel decomposition [41], feature maps usually tend to collapse into a small subset. From this perspective, not each feature map is necessary to be involved into the next layer's convolution. According to this insight, we redesign an improved network architecture to ease this problem.

Network architecture. In this section, we propose a Fully-Exploited Network (FE-Net) composed by the block as shown in Fig.3. In this block, only a subset of feature maps are involved into computation, and the remaining ones directly propagate to the next layer to ensure information flowing which can be formulated as follows:

$$\begin{aligned} & I_1, I_2 \Leftarrow I \\ & O = f(I_1) \parallel I_2 \end{aligned} \tag{8}$$

where I and O mean the input and output feature maps. $\Leftarrow$ denotes channel-wise split and $\parallel$ denotes channel-wise

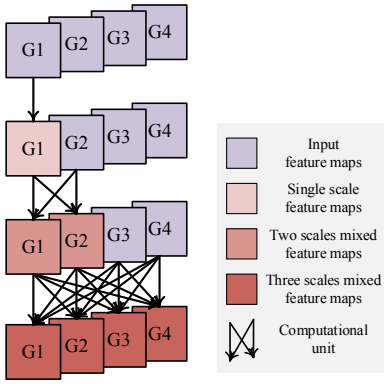

Figure 3. Fully-Exploited computational Block (FE-Block). Only a subset of feature maps is involved into optimization at each basic computational unit. For each resolution, the feature maps are progressively mixed in as layer increases. For a computational block with $n = 3$ basic units as shown in this figure, we evenly divide the input feature maps into 2^{n-1} parts, and involve $\frac{2^{l-1}}{2^{n-1}}(l = 1, \ldots, n)$ feature maps into optimization each layer. In this paper, the computational unit is implemented as *inverted bottlenecks* [30].

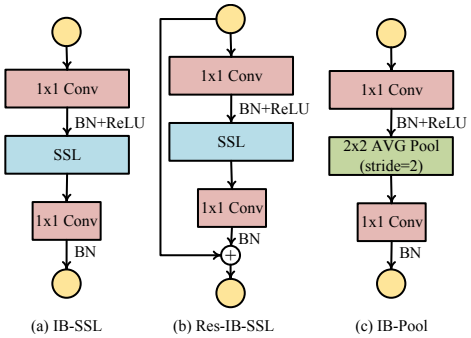

Figure 4. The basic computational units for FE-Block. (a): the basic unit without skipping connection; (b): the basic unit with skipping connection; (c): the basic unit for spatial down sampling ($2\times$). Note that in (a) and (b), it is SSL that provides receptive fields. (IB is short for inverted bottleneck.)

concatenation. In practice, this computational pattern can be implemented for efficiency as that I_1 joins into computation and its output $f(I_1)$ is rewritten back to the original memory position of I_1. The remaining feature maps I_2 do not require any operations. As layer increases, we mix more feature maps into computation. In this way, each input feature map is involved into optimization at last, and multi-scale feature maps are obtained for prediction. We empirically prove its validation in section 5.4.

Basic computational unit. In this paper, we adopt inverted bottlenecks [30] as basic computational units to build our efficient networks as shown in Fig.4. Without any specific statement, their expansion rate is always set to 6 by default, which means the first 1×1 Conv is always used to expand the input channel by 6 times. To combine the advantages of *residual learning* [7], for each computation block as shown in Fig.3, we mainly adopt Fig.4(b) as the basic computational unit except the last one. For the last computational unit at each computation block, we use Fig.4(a) to change the channel number for the next computational block, or use Fig.4(c) for spatial down sampling.

5. Experiments

In this section, we first carry out several ablation experiments on CIFAR10 and CIFAR100 [18] to demonstrate the effect of SSL. In these experiments, we prove that it is enough to provide spatial information communication and build compact ConvNets with only a few shift operations. Then we conduct experiments on ILSVRC-2012 [29] to assess its generalization ability to a large-scale dataset.

5.1. Benchmarks and Training Settings

CIFAR10 / CIFAR100 [18] are the datasets for 10-categories and 100-categeories image classification, respectively. Both of them consist of 50k images for training and 10k images for testing with resolution 32×32.

In the experiments on CIFAR, we choose ShiftResNet [37] which is built by *CSC* modules to evaluate the ability of SSL. Note that a *CSC* module is composed by a shift layer sandwiched between a 1×1 Conv layer for dimension ascending and a 1×1 Conv layer for dimension descending. Only the shift layer in this module is leveraged for spatial information communication. Replacing shift layer by SSL, through adjusting hyperparameter λ in Eqn.4, we study how many shift operations are required at least to maintain the performance of ShiftResNet.

We use ShiftResNet-20 and ShiftResNet-56 with expansion rate 6 as two representatives for ablation study. We train these networks by two GPUs with mini-batch 128 and base learning rate 0.1. As the same with [37], the learning rate decays by a factor of 10 after 32k and 48k iterations, and the training stops after 64k iterations. Specifically, we stop the training of SSL after 48k iterations in order to fix the learned shift pattern. For data augmentation, only horizontal flipping and random cropping are adopted. We use L2 regularization to shift values in the following experiments since we find that the result of L2 regularization is slightly better than L1.

ImageNet2012 [29] is a large-scale image classification benchmark with 1.28 million images for training and 50k images for validation. As is known, it is challenging to perform well on such large-scale dataset with lightweight neural networks. In order to boost the performance of the networks built with SSL by a further step, we redesign the neural network architecture as shown in Fig.3 to fully exploit the limited network capacity.

depth	Networks	λ	Accuracy CIFAR10 / CIFAR100	Params / FLOPs	Shift Sparsity CIFAR10 / CIFAR100
20	ResNet [37]	-	91.4% / 66.3%	0.27M / 81M	-
	ShiftResNet (GroupedShift) [37]	-	90.6% / 68.6%	0.16M / 53M	11.1%
	ShiftResNet (SSL)	0	91.7% / 69.2%		12.1% / 10.3%
		1e-4	91.1% / 69.2%		66.6% / 41.2%
		4e-4	90.4% / 67.7%		91.7% / 80.0%
		5e-4	89.8% / 67.7%		93.5% / 86.1%
	ShiftResNet (1x1 only)	-	81.5% / 56.7%		100%
56	ResNet [37]	-	92.0% / 69.3%	0.86M / 251M	-
	ShiftResNet (GroupedShift) [37]	-	92.7% / 72.1%	0.55M / 166M	11.1%
	ShiftResNet (SSL)	0	93.8% / 72.4%		12.8% / 11.4%
		1e-4	92.9% / 71.7%		87.8% / 73.8%
		4e-4	91.9% / 71.1%		97.4% / 94.6%
		5e-4	91.8% / 69.9%		98.0% / 96.1%
	ShiftResNet (1x1 only)	-	82.5% / 56.1%		100%

Table 1. The analysis of SSL on CIFAR10 and CIFAR100

In the experiments on ImageNet, we use SGD to train the networks with mini-batch 1024, weight decay 0.00004 and momentum 0.9. Training is started by a learning rate 0.6 with linear decaying policy and is stopped after 480 epochs, while the training of SSL is stopped after 240 epochs. The entire training iteration is comparable with [32, 22, 30, 26]. For data augmentation, we scale the short-side of images to 256 and adopt 224×224 random crop as well as horizontal flip to augment the training dataset. Also, to further rich the training images, more image of distortions are provided as used in Inception training [33, 10]. But it will be withdrawn in last several epochs. At the validation phase, we only center crop the feeding resized images to 224×224 and present the results with single-view approach.

5.2. Ablation Study

We explore the characteristic of SSL from three terms: (i) grouped shift vs. sparse shift; (ii) deep networks vs. shallow networks; (iii) the settings of λ.

Grouped shift vs. sparse shift. As shown in Tab.1, without shift penalty, the results of shift learning are superior to that of heuristic setting on both CIFAR10 and CIFAR100. Through shift learning, the network can adaptively adjust the displacement and direction of shift operation according to different tasks and different datasets. With shift penalty, it can eliminate a great portion of shift operations while keeping the accuracy of the network comparable with original network. Even with more than 90% sparsity to shift operation, the network can maintain a quite good performance, which suggests that only a few shift operations play crucial roles on communicating spatial information for image classification.

Deep networks vs. shallow networks. We analyze the sparsity of SSL on CIFAR10 / CIFAR100 with a shallow network and a deeper one, say ShiftResNet-20 and ShiftResNet-56. As shown in Tab.1, the shift sparsity on ShiftResNet-56 is more than ShiftResNet-20. It can provide good performance on CIFAR10 / CIFAR100 with even over 95% sparsity on ShiftResNet-56. Increasing depth brings more redundancy in the shift layer.

Different settings of λ. We increase λ from 0 to 5e-4 and find that a majority of shift operation is eliminated progressively while the accuracy of the networks decline a little. Here SSL (λ=0) is actually equivalent to quantization-aware Active Shift. When we increase λ significantly, we shrink all displacement to zero, which means the basic modules are all composed by 1×1 Conv layers and only three pooling layers in the network provide for spatial information communication. In this case, the accuracy drops a lot, which reflects from another side that such a few shifts really matter a lot for spatial information communication. Let us take ShiftResNet-56 on CIFAR100 as an example. Its accuracy can be boosted from 56.1% to 69.9% with only 3.9% feature maps shifted.

5.3. Case Study

We take ShiftResNet-20 on CIFAR10 and CIFAR100 with λ = 5e-4 for more detailed study. In Tab.2, we show the shift sparsity of each layer in detail. In some of the blocks, almost all feature maps stay unshifted which indicates the shift layers in these positions are unimportant. Actually, the sparsity of shift layer can be taken as a metric to measure the importance of these layers. It can decide which shift layer is unimportant and can be removed without accuracy decline. For examples, the shift layer in block2_1 is the most unimportant while the one in block2_2 is the most important in ShiftResNet-20. We take the shift layer in block2_2 for visualization as shown in Fig.5. Although the major-

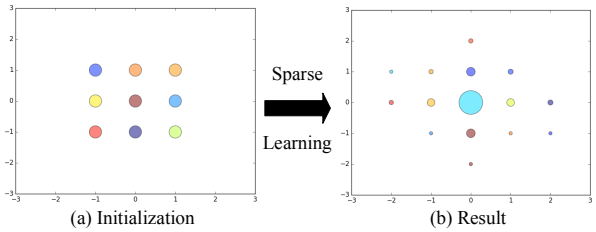

Figure 5. The visualization of shift values in the shift layer from block2_2 of ShiftResNet-20 on CIFAR100. The area of each point denotes the channel number of the feature maps with the same shift pattern. x-axis and y-axis denote the horizontal and vertical displacement, respectively. (Best viewed in color)

Block	CIFAR10		CIFAR100	
	Unshifts/ Channels	Shift Sparsity	Unshifts/ Channels	Shift Sparsity
block1_1	93 / 96	96.9%	82 / 96	85.4%
block1_2	87 / 96	90.6%	84 / 96	87.5%
block1_3	94 / 96	97.9%	92 / 96	95.8%
block2_1	96 / 96	100%	96 / 96	100%
block2_2	161 / 192	83.9%	146 / 192	76.0%
block2_3	181 / 192	94.3%	164 / 192	85.4%
block3_1	190 / 192	99.0%	189 / 192	98.4%
block3_2	331 / 384	86.2%	316 / 384	82.3%
block3_3	382 / 384	99.5%	319 / 384	83.1%
Total	1615 / 1728	93.5%	1488 / 1728	86.1%

Table 2. The shift sparsity of each layer in ShiftResNet-20 ($\lambda = 0.0005$) on CIFAR10 and CIFAR100.

Removed shift layer number (ShiftResNet20-SSL, λ=0)	Accuracy CIFAR10/CIFAR100
0	91.7% / 69.2%
4	91.5% / 68.2%
6	91.4% / 67.0%
8	89.4% / 66.0%
9 (All removed)	81.5% / 56.7%

Table 3. The performance of ShiftResNet-20 on CIFAR10 and CIFAR100 after removing the most unimportant shift layers.

ity of channels stay unshifted, the remaining ones can learn a meaningful shift pattern and provide multiple receptive fields. Actually, a shift layer cooperated with point-wise convolution can take a role of an Inception module. This is also a major advantage of shift layer over conventional convolution layer.

To take a further analysis, we carry out several experiments with ShiftResNet-20 on CIFAR10 and CIFAR100 by removing the most unimportant shift layers according to their sparsity in Tab.2. As shown in Tab.3, when we progressively remove the most unimportant shift layers, the accuracy only declines a little. Even when we preserve only one shift layer in block2_2, the accuracy still maintains in a considerable level.

Input	Operator	t	n	c	s
$224^2 \times 3$	conv 3×3+BN	-	-	16	2
$112^2 \times 16$	IB-SSL	4	-	16	1
$112^2 \times 16$	IB-Pool 2×2	5	-	32	2
$56^2 \times 32$	FE-Block	6	3	64	2
$28^2 \times 64$	FE-Block	6	4	128	2
$14^2 \times 128$	FE-Block	6	4	128	1
$14^2 \times 128$	FE-Block	6	4	256	2
$7^2 \times 256$	FE-Block	6	3	256	1
$7^2 \times 256$	conv1×1+BN+ReLU	-	-	1380	1
$7^2 \times 1380$	GAP 7×7	-	-	1380	-
$1^2 \times 1380$	Dropout 0.2	-	-	1380	-
$1^2 \times 1380$	conv 1×1	-	-	1000	1

Table 4. Network configuration. t denotes expansion rate. n means the computational unit number of FE-Block. c denotes the output channels. And s means stride.

Networks	MAdds	Params	Top-1
MobileNetV1 0.75x [10]	325M	2.6M	68.4%
MobileNetV2 1.0x[30]	300M	3.4M	72.0%
ShuffleNetV1 1.5x(g=3)[40]	292M	3.4M	69.0%
ShuffleNetV2 1.5x[26]	299M	3.5M	72.6%
IGCV3-D [32]	318M	3.6M	72.2%
CondenseNet(G=C=8)[12]	274M	2.9M	71.0%
ShiftNet-B [37]	371M	1.1M	61.2%
AS-ResNet-w50 [16]	404M	1.96M	69.9%
FE-Net (ours) 1.0x	301M	3.7M	**72.9%**
MobileNetV1 1.0x[10]	569M	4.2M	70.6%
MobileNetV2 1.4x[30]	585M	6.9M	74.7%
ShuffleNetV1 2x[40]	524M	5.4M	70.9%
ShuffleNetV2 2x[26]	591M	7.4M	74.9%
IGCV3-D 1.4x[32]	610M	7.2M	74.55%
CondenseNet(G=C=4)[12]	529M	4.8M	73.8%
PNASNet[22]	588M	5.1M	74.2%
DARTS [23]	595M	4.9M	73.1%
ShiftNet-A [37]	1400M	4.1M	70.1%
AS-ResNet-w68 [16]	729M	3.42M	72.2%
FE-Net (ours) 1.375x	563M	5.9M	**75.0%**

Table 5. The performance comparison of several compact neural architectures on ImageNet.

5.4. Performance on ImageNet

Our redesigned network architecture for ImageNet2012 classification task is described in Tab.4, which is mainly composed by FE-Block equipped with SSL. We use width multiplier as a hyperparameter to tune the tradeoff between accuracy and computational cost.

Comparison with other counterparts. As shown in Tab.5, with network architecture improvement, our results surpass ShiftNet and AS-ResNet by a large margin. What's more, before our work, the best performance in terms of FLOPs/accuracy is always dominated by the networks built by depthwise separable convolution in the past few years. We are the first one to build a compact neural network without using depthwise separable convolution which can achieve superior results to other counterparts constructed by depthwise separable convolution. As shown in Tab.5, our network surpasses MobileNet series networks and ShuffleNet series networks, as well as the networks automatically searched by NAS technique [22, 23], indicating that SSL can be taken as an alternative choice over depthwise separable convolution. This can provide a new basic component for NAS and inspire more exploration in this direction.

As for practical runtime, we mainly compare our network with MobileNetV2, which is the most representative compact network constructed by depthwise separable convolution. As illustrated in Tab.6, our networks achieve higher accuracy with significantly faster inference time on GPU and CPU, which proves that SSL is a more friendly basic component for practical application scenarios.

An ablation study of FE-Net. We also train the FE-Nets equipped with depthwise separable convolution (DW) on ImageNet so as to decompose the benifit of SSL from the improved network design. As shown in Tab.6, the gap of accuracy between SSL and DW based FE-Net is small while their practical runtime is significant larger, which further validates the superiority of SSL and FE-Net.

Compatibility with other methods. Our network can also be combined with other methods for further performance exploration. For instance, our network can be equipped with SE module (*Squeeze-and-Excitation* [11]) for channel attention. However, we find it matters to place SE module in different position of basic block. Here we only discuss the position of SE module in inverted bottleneck. As illustrated in Fig.6, there are two different placement manners. The first manner is the conventional one, which places SE module in the output position of the basic block. However, as for inverted bottleneck, the most redundant information exists in the expansion part. Since SE module is used for channel attention, it is more reasonable to place SE module in the expansion part of inverted bottleneck as shown in Fig.6(b). The results in Tab.7 empirically validates this idea. Moreover, we note that the shift sparsity increases a lot after equipping SE module as shown in Tab.8. Through channel-wise feature recalibration, it imposes more unshifted feature maps, since SE module encodes global information which lowers the need of shifting for spatial information communication.

Networks	Top1	Top5	GPU	CPU
FE-Net	72.9%	91.2%	16.1ms	1.9s
MobileNetV2 (DW)	72.0%	-	21.4ms	2.9s
FE-Net (DW)	73.2%	91.4%	21.8ms	2.7s
FE-Net 1.375x	75.0%	92.4%	23.1ms	3.8s
MobileNetV2 1.4x (DW)	74.7%	-	30.6ms	5.8s
FE-Net 1.375x (DW)	75.2%	92.8%	30.4ms	5.3s

Table 6. An ablation study of FE-Net with shift operation (SSL) vs. depthwise convolution (DW) on ImageNet (batchsize 32).

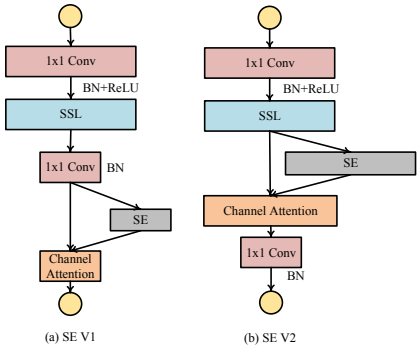

Figure 6. Two different placement manners of SE module.

Networks	MAdds	Params	Top1
MobileNetV1 1.0x + SE [11]	572M	4.7M	74.7%
ShuffleNetV2 2X + SE [26]	597M	-	75.4%
FE-Net 1.375x + SE V1	564M	6.1M	75.6%
FE-Net 1.375x + SE V2	566M	8.2M	**76.5%**

Table 7. The performance comparison of several compact neural architectures equipped with SE modules on ImageNet.

Networks	Top1	Shift Sparsity
FE-Net 1.0x	72.9%	60.0%
FE-Net 1.375x	75.0%	69.5%
FE-Net 1.375x + SE V1	75.6%	77.7%
FE-Net 1.375x + SE V2	76.5%	80.2%

Table 8. The shift sparsity of FE-Net with different accuracy.

6. Conclusions

In this paper, we mainly study the feasibility of SSL to build a compact and accurate neural network. Extensive experiments prove that only a few shift operations are sufficient for spatial information communication. We also show that SSL can be taken as an efficient alternative over depthwise separable convolution. A well-designed network equipped with SSL can surpass other counterparts equipped with depthwise separable convolution in terms of accuracy, FLOPs and practical inference time. Our work will inspire more exploration for network design and searching.

References

[1] A. Aghasi, N. Nguyen, and J. K. Romberg. Net-trim: Convex pruning of deep neural networks with performance guarantee. In *NIPS*, 2017.

[2] W. Chen, Y. Zhang, D. Xie, and S. Pu. A layer decomposition-recomposition framework for neuron pruning towards accurate lightweight networks. In *AAAI*, 2019.

[3] E. Denton, W. Zaremba, J. Bruna, Y. Lecun, and R. Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. In *NIPS*, 2014.

[4] Y. Gong, L. Liu, M. Yang, and L. D. Bourdev. Compressing deep convolutional networks using vector quantization. *CoRR*, abs/1412.6115, 2014.

[5] S. Han, J. Pool, J. Tran, and W. J. Dally. Learning both weights and connections for efficient neural networks. In *NIPS*, 2015.

[6] B. Hassibi and D. G. Stork. Second order derivatives for network pruning: Optimal brain surgeon. In *NIPS*, 1993.

[7] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[8] Y. He, X. Zhang, and J. Sun. Channel pruning for accelerating very deep neural networks. In *ICCV*, 2017.

[9] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. In *NIPS, workshop*, 2014.

[10] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861, 2017.

[11] J. Hu, L. Shen, and G. Sun. Sqyeeze-and-excitation networks. In *CVPR*, 2018.

[12] G. Huang, S. Liu, L. V. Der Maaten, and K. Q. Weinberger. Condensenet: An efficient densenet using learned group convolutions. In *CVPR*, 2018.

[13] I. Hubara, M. Courbariaux, D. Soudry, R. Elyaniv, and Y. Bengio. Quantized neural networks: Training neural networks with low precision weights and activations. *Journal of Machine Learning Research*, 18(187):1–30, 2016.

[14] Y. Ioannou, D. Robertson, R. Cipolla, and A. Criminisi. Deep roots: Improving cnn efficiency with hierarchical filter groups. In *CVPR*, 2017.

[15] M. Jaderberg, A. Vedaldi, and A. Zisserman. Speeding up convolutional neural networks with low rank expansions. In *BMVC*, 2014.

[16] Y. Jeon and K. Junmo. Constructing fast network through deconstruction of convolution. In *NIPS*, 2019.

[17] Y. Kim, E. Park, S. Yoo, T. Choi, L. Yang, and D. Shin. Compression of deep convolutional neural networks for fast and low power mobile applications. In *ICLR*, 2016.

[18] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. 2009.

[19] V. Lebedev, Y. Ganin, M. Rakhuba, I. V. Oseledets, and V. S. Lempitsky. Speeding-up convolutional neural networks using fine-tuned cp-decomposition. In *ICLR*, 2015.

[20] Y. Lecun, J. S. Denker, and S. A. Solla. Optimal brain damage. In *NIPS*, 1990.

[21] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf. Pruning filters for efficient convnets. In *ICLR*, 2017.

[22] C. Liu, M. Neumann, B. Zoph, J. Shlens, W. Hua, L. Li, L. Feifei, A. L. Yuille, J. Huang, and K. P. Murphy. Progressive neural architecture search. In *ECCV*, 2017.

[23] H. Liu, K. Simonyan, and Y. Yang. DARTS: differentiable architecture search. *CoRR*, abs/1806.09055, 2018.

[24] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang. Learning efficient convolutional networks through network slimming. In *ICCV*, 2017.

[25] J. H. Luo, J. Wu, and W. Lin. Thinet: A filter level pruning method for deep neural network compression. In *ICCV*, 2017.

[26] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *ECCV*, 2018.

[27] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz. Pruning convolutional neural networks for resource efficient inference. In *ICLR*, 2017.

[28] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *ECCV*, 2016.

[29] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, and M. Bernstein. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[30] M. Sandler, A. G. Howard, M. Zhu, A. Zhmoginov, and L. Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*, 2018.

[31] B. B. Sau and V. N. Balasubramanian. Deep model compression: Distilling knowledge from noisy teachers. *CoRR*, abs/1610.09650, 2016.

[32] K. Sun, M. Li, D. Liu, and J. Wang. Igcv3: Interleaved low-rank group convolutions for efficient deep neural networks. In *BMVC*, 2018.

[33] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.

[34] C. Tai, T. Xiao, Y. Zhang, X. Wang, and E. Weinan. Convolutional neural networks with low-rank regularization. In *ICLR*, 2016.

[35] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li. Learning structured sparsity in deep neural networks. In *NIPS*, 2016.

[36] W. Wen, C. Xu, C. Wu, Y. Wang, Y. Chen, and H. Li. Coordinating filters for faster deep neural networks. In *ICCV*, 2017.

[37] B. Wu, A. Wan, X. Yue, P. H. Jin, S. Zhao, N. Golmant, A. Gholaminejad, J. E. Gonzalez, and K. Keutzer. Shift: A zero flop, zero parameter alternative to spatial convolutions. In *CVPR*, 2018.

[38] G. Xie, J. Wang, T. Zhang, J. Lai, R. Hong, and G. Qi. Interleaved structured sparse convolutional neural networks. In *CVPR*, 2018.

[39] T. Zhang, G. Qi, B. Xiao, and J. Wang. Interleaved group convolutions. In *ICCV*, 2017.

[40] X. Zhang, X. Zhou, M. Lin, and J. Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *CVPR*, 2018.

[41] X. Zhang, J. Zou, K. He, and J. Sun. Accelerating very deep convolutional networks for classification and detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(10):1943–1955, 2016.

[42] G. Zhao, J. Wang, and Z. Zhang. Random shifting for cnn: a solution to reduce information loss in down-sampling layers. In *IJCAI*, 2017.