WeightNet: Revisiting the Design Space of Weight Networks

Ningning Ma[1][0000−0003−4628−8831], Xiangyu Zhang[2⋆][0000−0003−2138−4608], Jiawei Huang[2][0000−0003−4839−1971], and Jian Sun[2][0000−0002−6178−4166]

[1] Hong Kong University of Science and Technology
[2] MEGVII Technology
nmaac@cse.ust.hk, {zhangxiangyu,huangjiawei,sunjian}@megvii.com

Abstract. We present a conceptually simple, flexible and effective framework for weight generating networks. Our approach is general that unifies two current distinct and extremely effective SENet and CondConv into the same framework on weight space. The method, called $WeightNet$, generalizes the two methods by simply adding one more grouped fully-connected layer to the attention activation layer. We use the WeightNet, composed entirely of (grouped) fully-connected layers, to directly output the convolutional weight. WeightNet is easy and memory-conserving to train, on the kernel space instead of the feature space. Because of the flexibility, our method outperforms existing approaches on both ImageNet and COCO detection tasks, achieving better Accuracy-FLOPs and Accuracy-Parameter trade-offs. The framework on the flexible weight space has the potential to further improve the performance. Code is available at https://github.com/megvii-model/WeightNet.

Keywords: CNN, weight network, conditional kernel

1 Introduction

Designing convolution weight is a key issue in convolution networks (CNNs). The weight-generating methods [14,6,24] using a network, which we call weight networks, provide an insightful neural architecture design space. These approaches are conceptually intuitive, easy and efficient to train. Our goal in this work is to present a simple and effective framework, in the design space of weight networks, inspired by the rethinking of recent effective conditional networks.

Conditional networks (or dynamic networks)[2,11,38], which use extra sample-dependent modules to conditionally adjust the network, have achieved great success. SENet [11], an effective and robust attention module, helps many tasks achieve state-of-the-art results [9,31,32]. Conditionally Parameterized Convolution (CondConv) [38] uses over-parameterization to achieve great improvements but maintains the computational complexity at the inference phase.

Both of the methods consist of two steps: first, they obtain an attention activation vector, then using the vector, SE scales the feature channels, while

⋆ Corresponding author

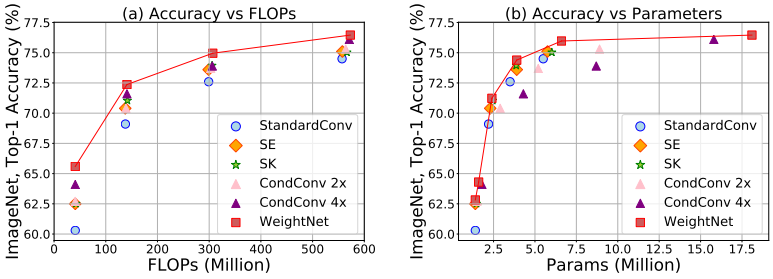

Fig. 1. Accuracy vs. FLOPs vs. Parameters comparisons on ImageNet, using ShuffleNetV2 [22]. (a) The trade-off between accuracy and FLOPs; (b) the trade-off between accuracy and number of parameters.

CondConv performs a mixture of expert weights. Despite they are usually treated as entirely distinct methods, they have some things in common. It is natural to ask: *do they have any correlations?* We show that we can link the two extremely effective approaches, by generalizing them in the weight network space.

Our methods, called *WeightNet*, extends the first step by simply adding one more layer for generating the convolutional weight directly (see Fig. 2). The layer is a grouped fully-connected layer applied to the attention vector, generating the weight in a group-wise manner. To achieve this, we rethink SENet and CondConv and discover that the subsequent operations after the first step can be cast to a *grouped fully-connected layer*, however, they are particular cases.

In that grouped layer, the output is direct the convolution weight, but the input size and the group number are variable. In CondConv the group number is discovered to be a minimum number of one and the input is small (4, 8, 16, etc.) to avoid the rapid growth of the model size. In SENet the group is discovered to be the maximum number equal to the input channel number.

Despite the two variants having seemingly minor differences, they have a large impact: they together control the parameter-FLOPs-accuracy tradeoff, leading to surprisingly different performance. Intuitively, we introduce two hyperparameters M and G, to control the input number and the group number, respectively. The two hyperparameters have not been observed and investigated before, in the additional grouped fully-connected layer. By simply adjusting them, we can strike a better trade-off between the representation capacity and the number of model parameters. We show by experiments on ImageNet classification and COCO detection the superiority of our method (Figure 1).

Our main contributions include: 1) First, we rethink the weight generating manners in SENet and CondConv, for the first time, to be complete fully-connected networks; 2) Second, only from this new perspective can we revisit the novel network design space in the weight space, which provides more effective structures than those in convolution design space (group-wise, point-wise, and depth-wise convolutions, etc). In this new and rarely explored weight space,

there could be new structures besides fully-connected layers, there could also be more kinds of sparse matrix besides those in Fig. 4. We believe this is a promising direction and hope it would have a broader impact on the vision community.

2 Related Work

Weight generation networks Schmidhuber et al. [28] incorporate the "fast" weights into recurrent connections in RNN methods. Dynamic filter networks [14] use filter-generating networks on video and stereo prediction. HyperNetworks [6] decouple the neural networks according to the relationship in nature: a genotype (the hypernetwork), and a phenotype (the main network), that uses a small network to produce the weights for the main network, which reduces the number of parameters while achieving respectable results. Meta networks [24] generate weights using a meta learner for rapid generalization across tasks. The methods [14,6,24,25] provide a worthy design space in the weight-generating network, our method follows the spirits and uses a WeightNet to generate the weights.

Conditional CNNs Different from standard CNNs [29,7,30,40,4,10,27,9], conditional (or dynamic) CNNs [17,20,37,39,15] use dynamic kernels, widths, or depths conditioned on the input samples, showing great improvement. Spatial Transform Networks [13] learns to transform to warp the feature map in a parametric way. Yang et al. [38] proposed conditional parameterized convolution to mix the experts voted by each sample's feature. The methods are extremely effective because they improve the Top-1 accuracy by more than 5% on the ImageNet dataset, which is a great improvement. Different from dynamic features or dynamic kernels, another series of work [35,12] focus on dynamic depths of the convolutional networks, that skip some layers for different samples.

Attention and gating mechanism Attention mechanism [33,21,1,34,36] is also a kind of conditional network, that adjusts the networks dependent on the input. Recently the attention mechanism has shown its great improvement. Hu et al. [11] proposed a block-wise gating mechanism to enhance the representation ability, where they adopted a squeeze and excitation method to use global information and capture channel-wise dependencies. SENet achieves great success by not only winning the ImageNet challenge [5], but also helping many structures to achieve state-of-the-art performance [9,31,32]. In GaterNet [3], a gater network was used to predict binary masks for the backbone CNN, which can result in performance improvement. Besides, Li et al. [16] introduced a kernel selecting module, where they added attention to kernels with different sizes to enhance CNN's learning capability. In contrast, WeightNet is designed on kernel space which is more time-conserving and memory-conserving than feature space.

3 WeightNet

The WeightNet generalizes the current two extremely effective modules in weight space. Our method is conceptually simple: both SENet and CondConv generate

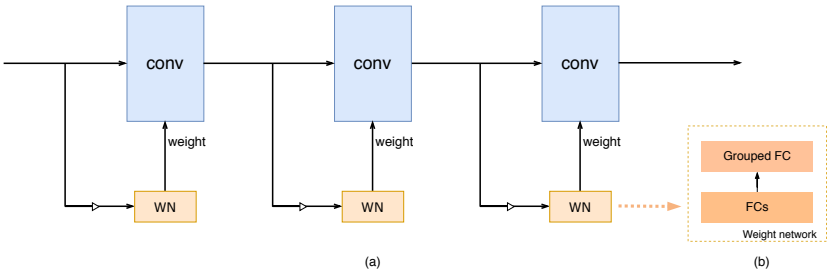

Fig. 2. The WeightNet structure. The convolutional weight is generated by WeightNet that is comprised entirely of (grouped) fully-connected layers. The symbol ($\triangleright$) represents the dimension reduction (global average pool) from feature space ($C \times H \times W$) to kernel space (C). The 'FC' denotes a fully-connected layer, 'conv' denotes convolution, and 'WN' denotes the WeightNet.

the *activation vector* using a global average pooling (GAP) and one or two fully-connected layers followed with a non-linear sigmoid operation; to this we simply add one more grouped fully-connected layer, to generate the weight *directly* (Fig. 2). This is different from common practice that applies the vector to feature space and we avoid the memory-consuming training period.

WeightNet is computationally efficient because of the dimension reduction (GAP) from $C \times H \times W$ dimension to a 1-D dimension C. Evidently, the WeightNet only consists of (grouped) fully-connected layers. We begin by introducing the matrix multiplication behaviors of (grouped) fully-connected operations.

Grouped fully-connected operation Conceptually, neurons in a fully-connected layer have full connections and thus can be computed with a matrix multiplication, in the form $Y = WX$ (see Fig. 3 (a)). Further, neurons in a grouped fully-connected layer have group-wise sparse connections with activations in the previous layer.

Formally, in Fig. 3 (b), the neurons are divided exactly into g groups, each group (with i/g inputs and o/g outputs) performs a fully-connected operation (see the red box for example). One notable property of this operation, which can be easily seen in the graphic illustration, is that the weight matrix becomes a sparse, *block diagonal matrix*, with a size of $(o/g \times i/g)$ in each block.

Grouped fully-connected operation is a general form of fully-connected operation where the group number is one. Next, we show how it generalizes CondConv and SENet: use the grouped fully-connected layer to replace the subsequent operations after the activation vector and directly output the generated weight.

Denotation We denote a convolution operation with the input feature map $\mathbf{X} \in \mathbb{R}^{C \times h \times w}$, the output feature map $\mathbf{Y} \in \mathbb{R}^{C \times h' \times w'}$, and the convolution weight $\mathbf{W'} \in \mathbb{R}^{C \times C \times k_h \times k_w}$. For simplicity, but without loss of generality, it is

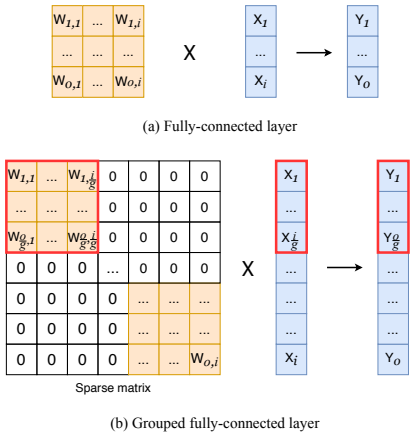

Fig. 3. The **matrix multiplication** behaviors of the (grouped) fully-connected operations. Here i, o and g denote the numbers of the input channel, output channel and group number. (a) A standard matrix multiplication representing a fully-connected layer. (b) With the weight in a block diagonal sparse matrix, it becomes a general grouped fully-connected layer. Each group (red box) is exactly a standard matrix multiplication in (a), with i/g input channels and o/g output channels. Fig. (a) is a special case of Fig. (b) where $g = 1$.

assumed that the number of the input channels equals to that of output channels, here $(h, w), (h', w'), (k_h, k_w)$ denote the 2-D heights and the widths for the input, output, and kernel. Therefore, we denote the convolution operation using the symbol $(*)$: $\mathbf{Y}_c = \mathbf{W}'_c * \mathbf{X}$. We use $\boldsymbol{\alpha}$ to denote the attention activation vector in CondConv and SENet.

3.1 Rethinking CondConv

Conditionally parameterized convolution (CondConv) [38] is a mixture of m experts of weights, voted by a m-dimensional vector $\boldsymbol{\alpha}$, that is sample-dependent and makes each sample's weight dynamic.

Formally, we begin with reviewing the first step in CondConv, it gets $\boldsymbol{\alpha}$ by a global average pooling and a fully-connected layer $\mathbf{W}_{fc1}$, followed by a sigmoid $\sigma(\cdot)$: $\boldsymbol{\alpha} = \sigma(\mathbf{W}_{fc1} \times \frac{1}{hw} \sum_{i \in h, j \in w} \mathbf{X}_{c,i,j})$, here $(\times)$ denotes the matrix multiplication, $\mathbf{W}_{fc1} \in \mathbb{R}^{m \times C}$, $\alpha \in \mathbb{R}^{m \times 1}$.

Next, we show the following mixture of expert operations in the original paper can essentially be replaced by a fully-connected layer. The weight is generated by multiple weights: $\mathbf{W}' = \alpha_1 \cdot \mathbf{W}_1 + \alpha_2 \cdot \mathbf{W}_2 + ... + \alpha_m \cdot \mathbf{W}_m$, here $\mathbf{W}_i \in \mathbb{R}^{C \times C \times k_h \times k_w}, (i \in \{1, 2, ..., m\})$. We rethink it as follows:

$$\begin{aligned} \mathbf{W}' &= \mathbf{W}^T \times \boldsymbol{\alpha} \\ where\ \mathbf{W} &= [\mathbf{W}_1 \mathbf{W}_2 ... \mathbf{W}_m] \end{aligned} \tag{1}$$

Table 1. Summary of the configure in the grouped fully-connected layer. λ is the proportion of input size to group number, representing the major increased parameters.

Model	Input size	Group number	λ	Output size
CondConv	m	1	m	$C \times C \times k_h \times k_w$
SENet	C	C	1	$C \times C \times k_h \times k_w$
WeightNet	$M \times C$	$G \times C$	M/G	$C \times C \times k_h \times k_w$

Here $\mathbf{W} \in \mathbb{R}^{m \times CCk_hk_w}$ denotes the matrix concatenation result, $(\times)$ denotes the matrix multiplication (fully-connected in Fig. 3a). Therefore, the weight is generated by simply adding one more layer ($\mathbf{W}$) to the activation layer. That layer is a fully-connected layer with m inputs and $C \times C \times k_h \times k_w$ outputs.

This is different from the practice in the original paper in the training phase. In that case, it is memory-consuming and suffers from the batch problem when increasing m (batch size should be set to one when $m > 4$). In this case, we train with large batch sizes efficiently.

3.2 Rethinking SENet

Squeeze and Excitation (SE) [11] block is an extremely effective "plug-n-play" module that is acted on the feature map. We integrate the SE module into the convolution kernels and discover it can also be represented by adding one more grouped fully-connected layer to the activation vector $\boldsymbol{\alpha}$. We start from the reviewing of the $\boldsymbol{\alpha}$ generation process. It has a similar process with CondConv: a global average pool, two fully-connected layer with non-linear ReLU (δ) and sigmoid (σ): $\boldsymbol{\alpha} = \sigma(\mathbf{W}_{fc2} \times \delta(\mathbf{W}_{fc1} \times \frac{1}{hw} \sum_{i \in h, j \in w} \mathbf{X}_{c,i,j}))$, here $\mathbf{W}_{fc1} \in \mathbb{R}^{C/r \times C}$, $\mathbf{W}_{fc2} \in \mathbb{R}^{C \times C/r}$, $(\times)$ in the equation denotes the matrix multiplication. The two fully-connected layers here are mainly used to reduce the number of parameters because $\boldsymbol{\alpha}$ here is a C-dimensional vector, a single layer is parameter-consuming.

Next, in common practice the block is used before or after a convolution layer, $\boldsymbol{\alpha}$ is computed right before a convolution (on the input feature $\mathbf{X}$): $\mathbf{Y}_c = \mathbf{W}'_c * (\mathbf{X} \cdot \boldsymbol{\alpha})$, or right after a convolution (on the output feature $\mathbf{Y}$): $\mathbf{Y}_c = (\mathbf{W}'_c * \mathbf{X}) \cdot \boldsymbol{\alpha}_c$, here $(\cdot)$ denotes dot multiplication broadcasted along the C axis. In contrast, on kernel level, we analyze the case that SE is acted on $\mathbf{W}'$: $\mathbf{Y}_c = (\mathbf{W}'_c \cdot \boldsymbol{\alpha}_c) * \mathbf{X}$. Therefore we rewrite the weight to be $\mathbf{W}' \cdot \boldsymbol{\alpha}$, the $(\cdot)$ here is different from the $(\times)$ in Equ. 1. In that case, a dimension reduction is performed; in this case, no dimension reduction. Therefore, it is essentially a grouped sparse connected operation, that is a particular case of Fig. 3 (b), with C inputs, $C \times C \times k_h \times k_w$ outputs, and C groups.

3.3 WeightNet Structure

By far, we note that the group number in the general grouped fully-connected layer (Fig. 3 b) has values range from 1 to the channel number. That is, the group

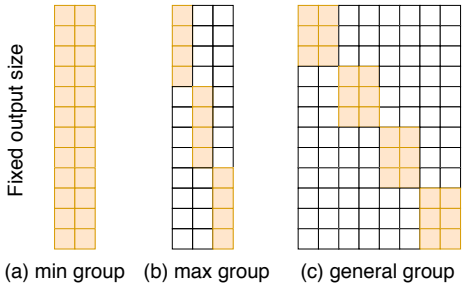

Fig. 4. The diagrams of the different cases in the **block diagonal matrix** (Fig. 3 b), that can represent the weights of the grouped fully-connected layer in CondConv, SENet and the general WeightNet. They output the same fixed size (convolution kernel's size $C \times C \times k_h \times k_w$), but have different group numbers: (a) the group number has a minimum number of one (CondConv), (b) the group number has a maximum number equals to the input size C (SENet), since (a) and (b) are extreme cases, (c) shows the general group number between 1 and the input size (WeightNet).

has a minimum number of one and has a maximum number of the input channel numbers. It, therefore, generalizes the CondConv, where the group number takes the minimum value (one), and the SENet, where it takes the maximum value (the input channel number). We conclude that they are two extreme cases of the general grouped fully-connected layer (Fig. 4).

We summarize the configure in the grouped fully-connected layer (in Table 1) and generalize them using two additional hyperparameters M and G. To make the group number more flexible, we set it by combining the channel C and a constant hyperparameter G, Moreover, another hyperparameter M is used to control the input number, thus M and G together to control the parameter-accuracy tradeoff. The layer in CondConv is a special case with $M = m/C, G = 1/C$, while for SENet $M = 1, G = 1$. We constrain $M \times C$ and $G \times C$ to be integers, M is divisible by G in this case. It is notable that the two hyperparameters are right there but have not been noticed and investigated.

Implementation details For the activation vector $\boldsymbol{\alpha}$'s generating step, since $\boldsymbol{\alpha}$ is a $(M \times C)$-dimensional vector, it may be large and parameter-consuming, therefore, we use two fully-connected layers with a reduction ratio r. It has a similar process with the two methods: a global average pool, two fully-connected layer with non-linear sigmoid (σ): $\boldsymbol{\alpha} = \sigma(\mathbf{W}_{fc_2} \times \mathbf{W}_{fc1} \times \frac{1}{hw} \sum_{i \in h, j \in w} \mathbf{X}_{c,i,j})$, here $\mathbf{W}_{fc1} \in \mathbb{R}^{C/r \times C}$, $\mathbf{W}_{fc2} \in \mathbb{R}^{MC \times C/r}$, $(\times)$ denotes the matrix multiplication, r has a default setting of 16.

In the second step, we adopt a grouped fully-connected layer with $M \times C$ input, $C \times C \times k_h \times k_w$ output, and $G \times C$ groups. We note that the structure is a straightforward design, and more complex structures have the potential to improve the performance further, but it is beyond the focus of this work.

Complexity analysis The structure of WeightNet decouples the convolution computation and the weight computation into two separate branches (see Fig. 2). Because the spatial dimensions ($h \times w$) are reduced before feeding into the weight branch, the computational amount (FLOPs) is mainly in the convolution branch. The FLOPs complexities in the convolution and weight branches are $O(hwCCk_hk_w)$ and $O(MCCk_hk_w/G)$, the latter is relatively negligible. The parameter complexities for each branch are zero and $O(M/G \times C \times C \times k_h \times k_w)$, which is M/G times of normal convolution. We notate λ to represent it (Table 1).

Training with batch dimension The weight generated by WeightNet has a dimension of batch size, here we briefly introduce the training method related to the batch dimension. We denote B as batch size and reshape the input $\mathbf{X}$ of the convolution layer to $(1, B \times C, h, w)$. Thus $\mathbf{X}$ has $B \times C$ channel numbers, which means we regard different samples in the same batch as different channels. Next, we reshape the generated weight $\mathbf{W}$ to (B, C, C, k_h, k_w). Then it becomes a group convolution, with a group number of B, the inputs and the outputs in each group are both equal to C. Therefore, we use the same memory-conserving method for both training and inference periods, and this is different from CondConv.

4 Experiments

In this section, we evaluate the WeightNet on classification and COCO detection tasks [19]. In classification task, we conduct experiments on a light-weight CNN model ShuffleNetV2 [22] and a deep model ResNet50 [7]. In the detection task, we evaluate our method's performance on distinct backbone models under RetinaNet. In the final analysis, we conduct ablation studies and investigate the properties of WeightNet in various aspects.

4.1 Classification

We conduct image classification experiments on ImageNet 2012 classification dataset, which includes 1000 classes [26]. Our models are first trained on the training dataset that consists of 1.28 million images and then evaluated over 50k images in the validation dataset. For the training settings, all the ShuffleNetV2 [22] models are trained with the same settings as [22]. For ResNet-50, we use a linear decay scheduled learning rate starting with 0.1, a batch size of 256, a weight decay of 1e-4, and 600k iterations.

ShuffleNetV2 To investigate the performance of our method on light-weight convolution networks, we construct experiments based on a recent effective network ShuffleNetV2 [22]. For a fair comparison, we retrain all the models by ourselves, using the same code base. We replace the standard convolution kernels in each bottleneck with our proposed WeightNet, and control FLOPs and the number of parameters for fairness comparison.

Table 2. ImageNet classification results of the WeightNet on ShuffleNetV2 [22]. For fair comparison, we control the values of λ to be $1\times$, to make sure that the experiments are under the same FLOPs and the same number of parameters.

Model	# Params	FLOPs	Top-1 err.
ShuffleNetV2 (0.5×)	1.4M	41M	39.7
+ WeightNet (1×)	1.5M	41M	**36.7**
ShuffleNetV2 (1×)	2.2M	138M	30.9
+ WeightNet (1×)	2.4M	139M	**28.8**
ShuffleNetV2 (1.5×)	3.5M	299M	27.4
+ WeightNet (1×)	3.9M	301M	**25.6**
ShuffleNetV2 (2×)	5.5M	557M	25.5
+ WeightNet (1×)	6.1M	562M	**24.1**

Table 3. ImageNet classification results of the WeightNet on ShuffleNetV2 [22]. The comparison is under the same FLOPs and regardless of the number of parameters. To obtain the optimum performance, we set the λ to {8×, 4×, 4×, 4×} respectively.

Model	# Params	FLOPs	Top-1 err.
ShuffleNetV2 (0.5×)	1.4M	41M	39.7
+ WeightNet (8×)	2.7M	42M	**34.0**
ShuffleNetV2 (1×)	2.2M	138M	30.9
+ WeightNet (4×)	5.1M	141M	**27.6**
ShuffleNetV2 (1.5×)	3.5M	299M	27.4
+ WeightNet (4×)	9.6M	307M	**25.0**
ShuffleNetV2 (2×)	5.5M	557M	25.5
+ WeightNet (4×)	18.1M	573M	**23.5**

As shown in Table 1, λ is utilized to control the number of parameters in a convolution. For simplicity, we fix $G = 2$ when adjusting λ. In our experiments, λ has several sizes $\{1\times, 2\times, 4\times, 8\times\}$. To make the number of channels conveniently divisible by G when scaling the number of parameters, we slightly adjust the number of channels for ShuffleNetV2 $1\times$ and $2\times$.

We evaluate the WeightNet from two aspects. Table 2 reports the performance of our method considering parameters. The experiments illustrate that our method has significant advantages over the other counterparts under the same FLOPs and the same number of parameter constraints. ShuffleNetV2 $0.5\times$ gains 3% Top-1 accuracy without additional computation budgets.

In Table 3, we report the advantages after applying our method on ShuffleNetV2 with different sizes. Considering in practice, the storage space is sufficient. Therefore, without the loss of fairness, we only constrain the Flops to be the same and tolerate the increment of parameters.

ShuffleNet V2 ($0.5\times$) gains 5.7% Top-1 accuracy which shows further significant improvements by adding a minority of parameters. ShuffleNet V2 ($2\times$) gains 2.0% Top-1 accuracy.

To further investigate the improvement of our method, we compare our method with some recent effective conditional CNN methods under the same FLOPs and the same number of parameters. For the network settings of CondConv [38], we replace standard convolutions in the bottlenecks with CondConv, and change the number of experts as described in CondConv to adjust parameters, as the number of experts grows, the number of parameters grows. To reveal the model capacity under the same number of parameters, for our proposed WeightNet, we control the number of parameters by changing λ. Table 4 describes the comparison between our method and other counterpart effective methods, from which we observe our method outperforms the other conditional CNN methods under the same budgets. The Accuracy-Parameters tradeoff and the Accuracy-FLOPs tradeoff are shown in Figure 1.

Table 4. Comparison with recently effective attention modules on ShuffleNetV2 [22] and ResNet50 [7]. We show results on ImageNet.

Model	# Params	FLOPs	Top-1 err.
ShuffleNetV2 [22] (0.5×)	1.4M	41M	39.7
+ SE [11]	1.4M	41M	37.5
+ SK [16]	1.5M	42M	37.5
+ CondConv [38] (2×)	1.5M	41M	37.3
+ WeightNet (1×)	1.5M	41M	**36.7**
+ CondConv [38] (4×)	1.8M	41M	35.9
+ WeightNet (2×)	1.8M	41M	**35.5**
ShuffleNetV2 [22] (1.5×)	3.5M	299M	27.4
+ SE [11]	3.9M	299M	26.4
+ SK [16]	3.9M	306M	26.1
+ CondConv [38] (2×)	5.2M	303M	26.3
+ WeightNet (1×)	**3.9M**	301M	**25.6**
+ CondConv [38] (4×)	8.7M	306M	26.1
+ WeightNet (2×)	**5.9M**	**303M**	**25.2**
ShuffleNetV2 [22] (2.0×)	5.5M	557M	25.5
+ WeightNet (2×)	10.1M	565M	**23.7**
ResNet50 [7]	25.5M	3.86G	24.0
+ SE [11]	26.7M	3.86G	22.8
+ CondConv [38] (2×)	72.4M	3.90G	23.4
+ WeightNet (1×)	31.1M	3.89G	**22.5**

From the results, we can see SE and CondConv boost the base models of all sizes significantly. However, CondConv has major improvements in smaller sizes especially, but as the model becomes larger, the smaller the advantage it has. For example, CondConv performs better than SE on ShuffleNetV2 0.5× but SE performs better on ShuffleNetV2 2×. In contrast, we find our method can be uniformly better than SE and CondConv.

To reduce the overfitting problem while increasing parameters, we add dropout [8] for models with more than 3M parameters. As we described in Section 3.3, λ represents the increase of parameters, so we measure the capacity of networks by changing parameter multiplier λ in $\{1\times, 2\times, 4\times, 8\times\}$. We further analyze the effect of λ and the grouping hyperparameter G on each filter in the ablation study section.

ResNet50 For larger classification models, we conduct experiments on ResNet50 [7]. We use a similar way to replace the standard convolution kernels in ResNet50 bottlenecks with our proposed WeightNet. Besides, we train the conditional CNNs utilizing the same training settings with the base ResNet50 network.

In Table 4, based on ResNet50 model, we compare our method with SE [11] and CondConv [38] under the same computational budgets. It's shown that our method still performs better than other conditional convolution modules. We perform CondConv (2×) on ResNet50, the results reveal that it does not have further improvement comparing with SE, although CondConv has a larger num-

Table 5. Object detection results comparing with baseline backbone. We show RetinaNet [18] results on COCO.

Backbone	# Params	FLOPs	mAP
ShuffleNetV2 [22] (0.5×)	1.4M	41M	22.5
+ WeightNet (4×)	2.0M	41M	**27.1**
ShuffleNetV2 [22] (1.0×)	2.2M	138M	29.2
+ WeightNet (4×)	4.8M	141M	**32.1**
ShuffleNetV2 [22] (1.5×)	3.5M	299M	30.8
+ WeightNet (2×)	5.7M	303M	**33.3**
ShuffleNetV2 [22] (2.0×)	5.5M	557M	33.0
+ WeightNet (2×)	9.7M	565M	**34.0**

Table 6. Object detection results comparing with other conditional CNN backbones. We show RetinaNet [18] results on COCO.

Backbone	# Params	FLOPs	mAP
ShuffleNetV2 [22] (0.5×)	1.4M	41M	22.5
+ SE [11]	1.4M	41M	25.0
+ SK [16]	1.5M	42M	24.5
+ CondConv [38] (2×)	1.5M	41M	25.8
+ CondConv [38] (4×)	1.8M	41M	25.0
+ CondConv [38] (8×)	2.3M	42M	26.4
+ WeightNet (4×)	2.0M	41M	**27.1**

ber of parameters. We conduct our method (1×) by adding limited parameters and it also shows further improvement comparing with SE. Moreover, ShuffleNetV2 [22] (2×) with our method performs better than ResNet50, with only 40% parameters and 14.6% FLOPs.

4.2 Object Detection

We evaluate the performance of our method on COCO detection [19] task. The COCO dataset has 80 object categories. We use the *trainval*35*k* set for training and use the *minival* set for testing. For a fair comparison, we train all the models with the same settings. The batch size is set to 2, the weight decay is set to 1e-4 and the momentum is set to 0.9. We use anchors for 3 scales and 3 aspect ratios and use a 600-pixel train and test image scale. We conduct experiments on RetinaNet [18] using ShuffleNetV2 [22] as the backbone feature extractor. We compare the backbone models of our method with the standard CNN models.

Table 5 illustrates the improvement of our method over standard convolution on the RetinaNet framework. For simplicity we set $G = 1$ and adjust the size of WeightNet to $\{2\times, 4\times\}$. As we can see our method improves the mAP significantly by adding a minority of parameters. ShuffleNetV2 (0.5×) with WeightNet (4×) improves 4.6 mAP by adding few parameters under the same FLOPs.

To compare the performance between WeightNet and CondConv [38] under the same parameters, we utilize ShuffleNetV2 0.5× as the backbone and investigate the performances of all CondConv sizes. Table 6 reveals the clear advantage of our method over CondConv. Our method outperforms CondConv uniformly under the same computational budgets. As a result, our method is indeed robust and fundamental on different tasks.

4.3 Ablation Study and Analysis

The influence of λ. By tuning λ, we control the number of parameters. We investigate the influence of λ on ImageNet Top-1 accuracy, conducting experiments on the ShuffleNetV2 structure. Table 7 shows the results. We find that the optimal λ for ShuffleNetV2 {0.5×, 1×, 1.5×, 2×} are {8,4,4,4}, respectively.

Table 7. Ablation on λ. The table shows the ImageNet Top-1 err. results. The experiments are conducted on ShuffleNetV2 [22]. By increasing λ in the range {1,2,4,8}, the FLOPs does not change and the number of parameters increases.

	λ			
Model	1	2	4	8
ShuffleNetV2 (0.5×)	36.7	35.5	34.4	**34.0**
ShuffleNetV2 (1.0×)	28.8	28.1	**27.6**	27.8
ShuffleNetV2 (1.5×)	25.6	25.2	**25.0**	25.3
ShuffleNetV2 (2.0×)	24.1	23.7	**23.5**	24.0

Table 8. Ablation on G. We tune the group hyperparameter G to {1,2,4}, we keep $\lambda = 1$. The results are ImageNet Top-1 err. The experiments are conducted on ShuffleNetV2 [22].

Model	G	# Params	FLOPs	Top-1 err.
ShuffleNetV2 (0.5×)	G=1	1.4M	41M	37.18
	G=2	1.5M	41M	36.73
	G=4	1.5M	41M	**36.37**
ShuffleNetV2 (1.0×)	G=1	2.3M	139M	29.09
	G=2	2.4M	139M	28.77
	G=4	2.6M	139M	**28.76**

Table 9. Ablation study on **different stages**. The ✓ means the convolutions in that stage is integrated with our proposed WeightNet.

Stage2	Stage3	Stage4	Top-1 err.
✓			39.13
	✓		36.82
		✓	36.43
✓	✓		35.73
✓		✓	36.44
	✓	✓	**35.30**
✓	✓	✓	35.47

Table 10. Ablation study on the number of the **global average pooling** operators in the whole network. We conduct ShuffleNetV2 0.5× experiments on ImageNet dataset. We compare the cases: one global average pooling in 1) each stage, 2) each block, and 3) each layer. GAP represents global average pooling in this table.

	Top-1 err.
Stage wise GAP	37.01
Block wise GAP	35.47
Layer wise GAP	35.04

The model capacity has an upper bounded as we increase λ, and there exists a choice of λ to achieve the optimal performance.

The influence of G. To investigate the influence of G, we conduct experiments on ImageNet based on ShuffleNetV2. Talbe 8 illustrates the influence of G. We keep λ equals to 1, and change G to {1,2,4}. From the result we conclude that increasing G has a positive influence on model capacity.

WeightNet on different stages. As WeightNet makes the weights for each convolution layer changes dynamically for distinct samples, we investigate the influence of each stage. We change the static convolutions of each stage to our method respectively as Table 9 shows. From the result, we conclude that the last stage influences much larger than other stages, and the performance is best when we change the convolutions in the last two stages to our method.

The number of the global average pooling operator. Sharing the global average pooling (GAP) operator contributes to improving the speed of the con-

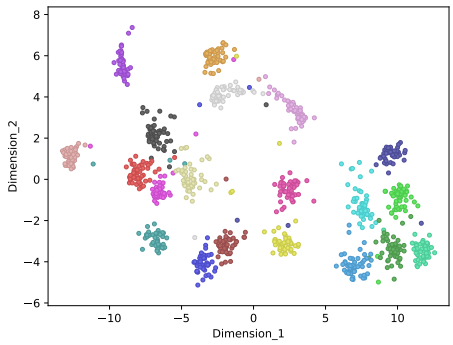

Fig. 5. Analysis for weights generated by our WeightNet. The figure illustrates of the weights of the 1,000 samples. The samples belong to 20 classes, which are represented by 20 different colors. Each point represents the weights of one sample.

ditional convolution network. We compare the following three kinds of usages of GAP operator: using GAP for each layer, sharing GAPs in a block, sharing GAPs in a stage. We conduct experiments on ShuffleNetV2 [22] (0.5×) baseline with WeightNet (2×). Table 10 illustrates the comparison of these three kinds of usages. The results indicate that by adding the number of GAPs, the model capacity improves.

Weight similarity among distinct samples. We randomly select 20 classes in the ImageNet validation set, which has 1,000 classes in total. Each class has 50 samples, and there are 1,000 samples in total. We extract the weights in the last convolution layer in Stage 4 from a well-trained ShuffleNetV2 (2×) with our WeightNet. We project the weights of each sample from a high dimensional space to a 2-dimension space by t-SNE [23], which is shown in Figure 5. We use 20 different colors to distinguish samples from 20 distinct classes.

We observe two characteristics. First, different samples have distinct weights. Second, there are roughly 20 point clusters and the weights of samples in the same classes are closer to each other, which indicates that the weights of our method capture more class-specific information than static convolution.

Channel similarity. We conduct the experiments to show the channel similarity of our method, we use the different filters' similarity in a convolution weight to represent the channel similarity. Lower channel similarity would improve the representative ability of CNN models and improve the channel representative capacities. Strong evidence was found to show that our method has a lower channel similarity.

We analyze the last convolution layer's kernel in the last stage of ShuffleNetV2 [22] (0.5×), where the channel number is 96, thus there are 96 filters

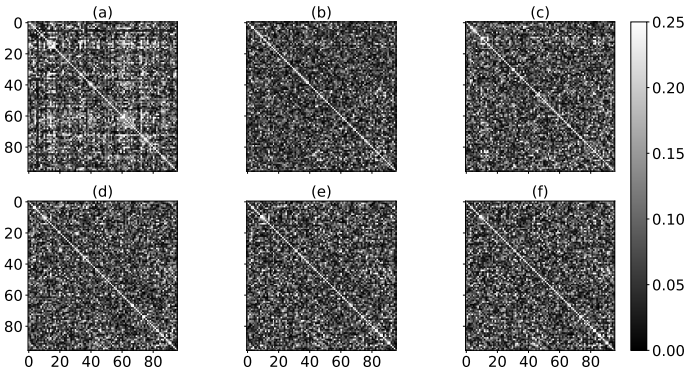

Fig. 6. Cosine similarity matrix. A 96×96 matrix represents 96 filters' pair-by-pair similarity, the smaller value (darker color) means the lower similarities. (a) Standard convolution kernel's similarity matrix, (b-f) WeightNet kernels' similarity matrixes. The colors in (b-f) are obviously much darker than (a), meaning lower similarity.

in that convolution kernel. We compute the cosine similarities of the filters pair by pair, that comprise a 96×96 cosine similarity matrix.

In Figure 6, we compare the channel similarity of WeightNet and standard convolution. We first compute the cosine similarity matrix of a standard convolution kernel and display it in Figure 6-(a). Then for our method, because different samples do not share the same kernel, we randomly choose 5 samples in distinct classes from the ImageNet validation set and show the corresponding similarity matrix in Figure 6-(b,c,d,e,f). The results clearly illustrate that our method has lower channel similarity.

5 Conclusion and Future Works

The study connects two distinct but extremely effective methods SENet and CondConv on weight space, and unifies them into the same framework we call WeightNet. In the simple WeightNet comprised entirely of (grouped) fully-connected layers, the grouping manners of SENet and CondConv are two extreme cases, thus involving two hyperparameters M and G that have not been observed and investigated. By simply adjusting them, we got a straightforward structure that achieves better tradeoff results. The more complex structures in the framework have the potential to further improve the performance, and we hope the simple framework in the weight space helps ease future research. Therefore, this would be a fruitful area for future work.

Acknowledgements This work is supported by The National Key Research and Development Program of China (No. 2017YFA0700800) and Beijing Academy of Artificial Intelligence (BAAI).

References

1. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473 (2014)
2. Cao, Y., Xu, J., Lin, S., Wei, F., Hu, H.: Gcnet: Non-local networks meet squeeze-excitation networks and beyond. In: Proceedings of the IEEE International Conference on Computer Vision Workshops. pp. 0–0 (2019)
3. Chen, Z., Li, Y., Bengio, S., Si, S.: Gaternet: Dynamic filter selection in convolutional neural network via a dedicated global gating network. arXiv preprint arXiv:1811.11205 (2018)
4. 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)
5. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition. pp. 248–255. Ieee (2009)
6. Ha, D., Dai, A., Le, Q.V.: Hypernetworks. arXiv preprint arXiv:1609.09106 (2016)
7. 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)
8. Hinton, G.E., Srivastava, N., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.R.: Improving neural networks by preventing co-adaptation of feature detectors. arXiv preprint arXiv:1207.0580 (2012)
9. 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)
10. 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)
11. 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)
12. Huang, G., Chen, D., Li, T., Wu, F., van der Maaten, L., Weinberger, K.Q.: Multi-scale dense networks for resource efficient image classification. arXiv preprint arXiv:1703.09844 (2017)
13. Jaderberg, M., Simonyan, K., Zisserman, A., et al.: Spatial transformer networks. In: Advances in neural information processing systems. pp. 2017–2025 (2015)
14. Jia, X., De Brabandere, B., Tuytelaars, T., Gool, L.V.: Dynamic filter networks. In: Advances in Neural Information Processing Systems. pp. 667–675 (2016)
15. Keskin, C., Izadi, S.: Splinenets: continuous neural decision graphs. In: Advances in Neural Information Processing Systems. pp. 1994–2004 (2018)
16. Li, X., Wang, W., Hu, X., Yang, J.: Selective kernel networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 510–519 (2019)
17. Lin, J., Rao, Y., Lu, J., Zhou, J.: Runtime neural pruning. In: Advances in Neural Information Processing Systems. pp. 2181–2191 (2017)
18. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection. In: Proceedings of the IEEE international conference on computer vision. pp. 2980–2988 (2017)
19. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: European conference on computer vision. pp. 740–755. Springer (2014)

20. Liu, L., Deng, J.: Dynamic deep neural networks: Optimizing accuracy-efficiency trade-offs by selective execution. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018)
21. Luong, M.T., Pham, H., Manning, C.D.: Effective approaches to attention-based neural machine translation. arXiv preprint arXiv:1508.04025 (2015)
22. 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)
23. Maaten, L.v.d., Hinton, G.: Visualizing data using t-sne. Journal of machine learning research **9**(Nov), 2579–2605 (2008)
24. Munkhdalai, T., Yu, H.: Meta networks. In: Proceedings of the 34th International Conference on Machine Learning-Volume 70. pp. 2554–2563. JMLR. org (2017)
25. Platanios, E.A., Sachan, M., Neubig, G., Mitchell, T.: Contextual parameter generation for universal neural machine translation. arXiv preprint arXiv:1808.08493 (2018)
26. 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)
27. 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)
28. Schmidhuber, J.: Learning to control fast-weight memories: An alternative to dynamic recurrent networks. Neural Computation **4**(1), 131–139 (1992)
29. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
30. 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)
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.V.: Efficientnet: Rethinking model scaling for convolutional neural networks. arXiv preprint arXiv:1905.11946 (2019)
33. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: Advances in neural information processing systems. pp. 5998–6008 (2017)
34. Wang, F., Jiang, M., Qian, C., Yang, S., Li, C., Zhang, H., Wang, X., Tang, X.: Residual attention network for image classification. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 3156–3164 (2017)
35. Wang, X., Yu, F., Dou, Z.Y., Darrell, T., Gonzalez, J.E.: Skipnet: Learning dynamic routing in convolutional networks. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 409–424 (2018)
36. Woo, S., Park, J., Lee, J.Y., So Kweon, I.: Cbam: Convolutional block attention module. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 3–19 (2018)
37. Wu, Z., Nagarajan, T., Kumar, A., Rennie, S., Davis, L.S., Grauman, K., Feris, R.: Blockdrop: Dynamic inference paths in residual networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 8817–8826 (2018)

38. Yang, B., Bender, G., Le, Q.V., Ngiam, J.: Condconv: Conditionally parameterized convolutions for efficient inference. In: Advances in Neural Information Processing Systems. pp. 1305–1316 (2019)
39. Yu, J., Yang, L., Xu, N., Yang, J., Huang, T.: Slimmable neural networks. arXiv preprint arXiv:1812.08928 (2018)
40. 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)