doi:10.3772/j.issn.1006-6748.2022.03.001

Multi-layer dynamic and asymmetric convolutions①

LUO Chunjie (罗纯杰), ZHAN Jianfeng②
(Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, P. R. China)
(University of Chinese Academy of Sciences, Beijing 100049, P. R. China)

Abstract

Dynamic networks have become popular to enhance the model capacity while maintaining efficient inference by dynamically generating the weight based on over-parameters. They bring much more parameters and increase the difficulty of the training. In this paper, a multi-layer dynamic convolution (MDConv) is proposed, which scatters the over-parameters over multi-layers with fewer parameters but stronger model capacity compared with scattering horizontally; it uses the expanding form where the attention is applied to the features to facilitate the training; it uses the compact form where the attention is applied to the weights to maintain efficient inference. Moreover, a multi-layer asymmetric convolution (MAConv) is proposed, which has no extra parameters and computation cost at inference time compared with static convolution. Experimental results show that MDConv achieves better accuracy with fewer parameters and significantly facilitates the training; MAConv enhances the accuracy without any extra cost of storage or computation at inference time compared with static convolution.

Key words: neural network, dynamic network, attention, image classification

0 Introduction

Deep neural networks have received great successes in many areas of machine intelligence. Many researchers have shown rising interest in designing light-weight convolutional networks[1-8]. Light-weight networks improve the efficiency by decreasing the size of the convolutions. That also leads to the decrease of the model capacity.

Dynamic networks[9-12] have become popular to enhance the model capacity while maintaining efficient inference by applying attention on the weight. Conditionally parameterized convolutions (CondConv)[9] and dynamic convolution (DYConv)[10] were proposed to use a dynamic linear combination of n experts as the kernel of the convolution. CondConv and DYConv bring much more parameters. WeightNet[11] used a grouped fully-connected layer applied to the attention vector to generate the weight in a group-wise manner, achieving comparable accuracy with fewer parameters than CondConv and DYConv. Dynamic convolution decomposition (DCD)[12] replaced dynamic attention over channel groups with dynamic channel fusion, resulting in a more compact model. However, DCD brings new problems: it increases the depth of the weight, thus it hinders error back-propagation; it increases the dynamic coefficients since it uses a full dynamic matrix. More dynamic coefficients make the training more difficult.

To reduce the parameters and facilitate the training, a multi-layer dynamic convolution (MDConv) is proposed, which scatters the over-parameters over multi-layers with fewer parameters but stronger model capacity compared with scattering horizontally; it uses the expanding form where the attention is applied to the features to facilitate the training; it uses the compact form where the attention is applied to the weights to maintain efficient inference. In CondConv and DYConv, the over-parameters are scattered horizontally. The key foundation for the success of deep learning is that deeper layers have stronger model capacity. Unlike CondConv and DYConv, MDConv scatters over-parameters over multi-layers, enhancing the model capacity with fewer parameters. Moreover, MDConv brings fewer dynamic coefficients thus is easier to train compared with DCD. There are two additional mechanisms to facilitate the training of deeper layers in the expanding form. One is batch normalization (BN) af-

① Supported by the National Key Research and Development Program of China (No. 2016YFB1000601) and the Standardization Pilot Research Project of Chinese Academy of Sciences (No. 20194620).
② To whom correspondence should be addressed. E-mail: zhanjianfeng@ ict.ac.cn.
Received on Sep. 9, 2021

ter each convolution. BN can significantly accelerate and improve the training of deeper networks. The other mechanism that helps the training is the bypass convolution with the static kernel. The bypass convolution shortens the path of error back-propagation.

At training time, the attention in MDConv is applied to features. While at inference time, the attention becomes weight attention. Batch normalization can be fused into the convolution. Squeeze-and-excite (SE) attention can be viewed as a diagonal matrix. Then, the three convolutions and SE attention can be further fused into a single convolution with dynamic weight for efficient inference. After fusion, the weight of the final convolution for inference is dynamically generated, and only one convolution needs to be performed. When implementing, there is no need to construct the diagonal matrix. After generating the dynamic coefficients, broadcasting multiply can be used instead of matrix multiply. Thus MDConv costs fewer memories and computational resources than DCD, which generates a dense matrix.

Although dynamic attention could significantly enhance the model capacity, it brings extra parameters and the number of float point operations (FLOPs) at inference time. Besides multi-layer dynamic convolution, a multi-layer asymmetric convolution (MAConv) is proposed, which removes the dynamic attention from multi-layer dynamic convolution. After the training, the weights need to be fused just one time and re-parameterized as new static kernels since they do not depend on the input anymore. As a result, there are no extra parameters and FLOPs at inference time compared with static convolution.

The experiments show that:

(1) MDConv achieves better accuracy with fewer parameters compared with other dynamic convolutions (DYConv and DCD). For example, for MobileNetV2 _ x1.0 and ShuffleNetV2 _ x1.0, multi-layer dynamic convolution achieves better accuracy with about half of the parameters compared with dynamic convolution. Moreover, MDConv converges faster than the state-of-the-art DCD and achieves higher accuracy, especially on the smaller training dataset.

(2) MAConv enhances the accuracy without any extra cost of storage or computation at inference time compared with static convolution. For example, MAConv improves the accuracy of MobileNetV2 _ x0.5 by 0.908% and MobileNetV2 _ x1.0 by 0.982%. These improvements are decent since there is no extra cost of storage or computation at inference time.

The remainder of this paper is structured as follows. Section 1 briefly presents related work. Section 2 describes the details of multi-layer dynamic convolution. Section 3 introduces multi-layer asymmetric convolution. In Section 4, the experiment settings and the results are presented. Conclusions are made in Section 5.

1 Related work

1.1 Dynamic networks

CondConv[9] and DYConv[10] compute convolutional kernels as a function of the input instead of using static convolutional kernels. In particular, the convolutional kernels are over-parameterized as a linear combination of n experts. Although largely enhancing the model capacity, CondConv and DYConv bring much more parameters, thus are prone to over-fitting. Besides, more parameters require more memory resources. Moreover, the dynamics make the training more difficult. To avoid over-fitting and facilitate the training, these two methods apply additional constraints. For example, CondConv shares routing weights between layers in a block. DYConv uses the Softmax with a large temperature instead of Sigmoid on the output of the routing network. WeightNet[11] uses a grouped fully-connected layer applied to the attention vector to generate the weight in a group-wise manner. WeightNet achieves comparable accuracy with fewer parameters than CondConv and DYConv. To further compact the model, DCD[12] decomposes the convolutional weight, which reduces the latent space of the weight matrix and results in a more compact model.

Although dynamic weight attentions enhance the model capacity, they increase the difficulty of the training since they introduce dynamic factors. Extremely, dynamic filter network[13] generates all the convolutional filters dynamically conditioned on the input. On the other hand, SE[14] is an effective and robust module by applying attention to the channel-wise features. Other dynamic networks[15-20] try to learn dynamic network structure with static convolution kernels.

1.2 Re-parameterization

ExpandNet[21] expands convolution into multiple linear layers without adding any nonlinearity. The expanding network can benefit from over-parameterization during training and can be compressed back to the compact one algebraically at inference. For example, a $k \times k$ convolution is expanded by three convolutional layers with kernel size 1×1, $k \times k$ and 1×1, respectively. ExpandNet increases the network depth, thus makes the training more difficult. ACNet[22] uses asymmetric convolution to strengthen the kernel skeletons for powerful networks. At training time, it uses three

branches with 3×3, 1×3, and 3×1 kernels respectively. At inference time, the three branches are fused into one static kernel. RepVGG[23] constructs the training-time model using branches consisting of identity map, 1×1 convolution and 3×3 convolution. After training, RepVGG constructs a single 3×3 kernel by re-parameterizing the trained parameters. ACNet and RepVGG can only be used for $k \times k (k > 1)$ convolution.

2 Multi-layer dynamic convolution

The main problem of CondConv[9] and DYConv[10] is that they bring much more parameters. DCD[12] reduces the latent space of the weight matrix by matrix decomposition and results in a more compact model. However, DCD brings new problems: (1) it increases the depth of the weight, thus hinders error back-propagation; (2) it increases the dynamic coefficients since it uses a full dynamic matrix. More dynamic coefficients make the training more difficult. In the extreme situation, e. g., dynamic filter network[13], all the convolutional weights are dynamically conditioned on the input. It is hard to train and can not be applied in modern deep architecture successfully.

To reduce the parameters and facilitate the training, MDConv is proposed. As shown in Fig. 1, MDConv has two branches: (1) the dynamic branch consists of a $k \times k (k >= 1)$ convolution, a SE module, and a 1×1 convolution; (2) the bypass branch consists of a $k \times k$ convolution with a static kernel. The output of MDConv is the addition of the two branches.

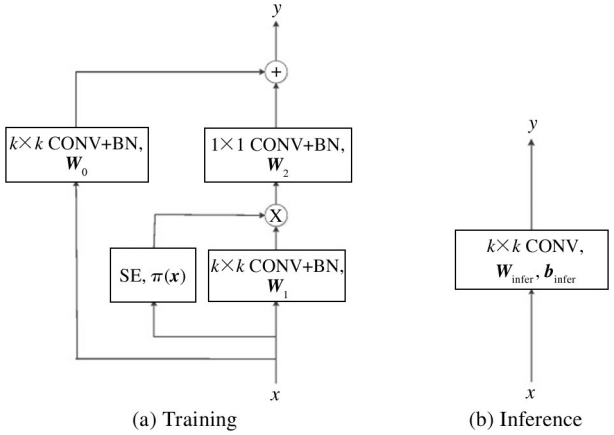

Fig. 1 Training and inference of MDConv

Unlike CondConv and DYConv, MDConv encapsulates the dynamic information into multi-layer convolutions by applying SE attention between two convolutional layers. By scattering the over-parameters over multi-layers, MDConv increases the model capacity with fewer parameters than horizontally scattering. Moreover, MDConv facilitates the training of dynamic networks. In MDConv, SE can be viewed as a diagonal matrix $\boldsymbol{A}$.

$$\boldsymbol{A} = \begin{bmatrix} \pi_1(\boldsymbol{x}) & 0 & \cdots & 0 \\ 0 & \pi_2(\boldsymbol{x}) & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & \pi_m(\boldsymbol{x}) \end{bmatrix} \tag{1}$$

where the diagonal element $\pi(\boldsymbol{x})$ is computed as

$$\pi(\boldsymbol{x}) = \text{Sigmoid}(F(\text{GlobalAveragePool}(\boldsymbol{x}))) \tag{2}$$

where F is a multi-layer fully-connected attention network. Compared with DCD, which uses a full dynamic matrix, MDConv brings fewer dynamic coefficients thus is easier to train. There are two additional mechanisms to facilitate the training of deeper layers in MDConv. One is batch normalization after each convolution. Batch normalization can significantly accelerate and improve the training of deeper networks. Another mechanism that helps the training is the bypass convolution with the static kernel. The bypass convolution shortens the path of error back-propagation.

MDConv uses two layer convolutions in the dynamic branch. Three or more layers bring the following problems: (1) more convolutions bring more computation FLOPs; (2) more dynamic layers are harder to train and need more training data.

Although the expanding form of MDConv facilitates the training, it is more expensive since there are three convolutional operators at training time. The compact form of MDConv can be used for efficient inference. MDConv can be defined as

$$\boldsymbol{y} = \frac{\boldsymbol{W}_2 \otimes \left(\boldsymbol{A}\left(\frac{\boldsymbol{W}_1 \otimes \boldsymbol{x} - \boldsymbol{\mu}_1}{\boldsymbol{\sigma}_1}\right)\right) - \boldsymbol{\mu}_2}{\boldsymbol{\sigma}_2} + \frac{\boldsymbol{W}_0 \otimes \boldsymbol{x} - \boldsymbol{\mu}_0}{\boldsymbol{\sigma}_0} \tag{3}$$

where $\boldsymbol{\mu}_i$ and $\boldsymbol{\sigma}_i (i = \{1,2,3\})$ are the mean and the standard deviation (STD) used in batch normalization. The normalization is channel-wise and broadcastable. For simplicity, the broadcast notation is ignored in the equation. At inference time, $\boldsymbol{\mu}_i$ and $\boldsymbol{\sigma}_i$ are constant and they can be fused into the convolutional weight and bias.

$$\boldsymbol{y} = \frac{\boldsymbol{w}_2}{\boldsymbol{\sigma}_2} \otimes \left(\boldsymbol{A}\left(\frac{\boldsymbol{W}_1}{\boldsymbol{\sigma}_1} \otimes \boldsymbol{x} - \frac{\boldsymbol{\mu}_1}{\boldsymbol{\sigma}_1}\right)\right) - \frac{\boldsymbol{\mu}_2}{\boldsymbol{\sigma}_2} + \frac{\boldsymbol{W}_0}{\boldsymbol{\sigma}_0} \otimes \boldsymbol{x} - \frac{\boldsymbol{\mu}_0}{\boldsymbol{\sigma}_0} \tag{4}$$

Then the three convolutions of MDConv can be fused into a single convolution for efficient inference.

$$\boldsymbol{y} = \left(\frac{\boldsymbol{W}_2}{\boldsymbol{\sigma}_2}\boldsymbol{A}\frac{\boldsymbol{W}_1}{\boldsymbol{\sigma}_1} + \frac{\boldsymbol{W}_0}{\boldsymbol{\sigma}_0}\right) \otimes \boldsymbol{x} - \frac{\boldsymbol{W}_2}{\boldsymbol{\sigma}_2}\boldsymbol{A}\frac{\boldsymbol{\mu}_1}{\boldsymbol{\sigma}_1} - \frac{\boldsymbol{\mu}_2}{\boldsymbol{\sigma}_2} - \frac{\boldsymbol{\mu}_0}{\boldsymbol{\sigma}_0} \tag{5}$$

$$\boldsymbol{W}_{\text{infer}} = \frac{\boldsymbol{W}_2}{\boldsymbol{\sigma}_2}\boldsymbol{A}\frac{\boldsymbol{W}_1}{\boldsymbol{\sigma}_1} + \frac{\boldsymbol{W}_0}{\boldsymbol{\sigma}_0} \tag{6}$$

$$\boldsymbol{b}_{\text{infer}} = -\frac{\boldsymbol{W}_2}{\boldsymbol{\sigma}_2}\boldsymbol{A}\frac{\boldsymbol{\mu}_1}{\boldsymbol{\sigma}_1} - \frac{\boldsymbol{\mu}_2}{\boldsymbol{\sigma}_2} - \frac{\boldsymbol{\mu}_0}{\boldsymbol{\sigma}_0} \tag{7}$$

where $\boldsymbol{W}_{\text{infer}}$ and $\boldsymbol{b}_{\text{infer}}$ are the new weight and bias of the convolution after re-parameterization.

Assume the input $\boldsymbol{x}$ has the size $p \times h \times w$, the output $\boldsymbol{y}$ has the size $q \times h \times w$, where p is the input channel, q is the output channel, h and w are the height and width of the feature. The size of convolutional weight $\boldsymbol{W}_1$ is defined as $m \times p \times k \times k$, where m is the intermediate channel, $k \times k$ is the receptive field size. It is important to set $\boldsymbol{W}_2$ as pointwise kernel with the size of $q \times m \times 1 \times 1$, otherwise the bias cannot pass forward and fused with the bias of the next layer. $\boldsymbol{W}_2$ can be reshaped into the shape of $q \times m$. Then $\frac{\boldsymbol{W}_2}{\boldsymbol{\sigma}_2}\boldsymbol{A}\frac{\boldsymbol{W}_1}{\boldsymbol{\sigma}_1}$ has the size of $q \times p \times k \times k$. $\boldsymbol{W}_0$ also has the size of $q \times p \times k \times k$. As a result, $\boldsymbol{W}_{\text{infer}}$ has the size of $q \times p \times k \times k$. For the term of bias, $\frac{\boldsymbol{\mu}_1}{\boldsymbol{\sigma}_1}$ has the size of $m \times 1$, then $\frac{\boldsymbol{W}_2}{\boldsymbol{\sigma}_2}\boldsymbol{A}\frac{\boldsymbol{\mu}_1}{\boldsymbol{\sigma}_1}$ has the size of $q \times 1$. $\frac{\boldsymbol{\mu}_0}{\boldsymbol{\sigma}_0}$ and $\frac{\boldsymbol{\mu}_2}{\boldsymbol{\sigma}_2}$ also have the size of $q \times 1$. As a result, $\boldsymbol{b}_{\text{infer}}$ has the size of $q \times 1$.

When implementing, the diagonal matrix $\boldsymbol{A}$ does not need be constructed. After generating the dynamic coefficients, the broadcasting multiply can be used instead of matrix multiply. Thus MDConv costs fewer memories and computational resources than DCD, which generates a dense matrix $\boldsymbol{A}$.

3 Multi-layer asymmetric convolution

Although dynamic attention could significantly enhance model capacity, it still brings extra parameters and FLOPs at inference time. Besides MDConv, this paper also proposes MAConv, which removes the dynamic attention in MDConv.

$$\boldsymbol{y} = \frac{\boldsymbol{W}_2 \otimes \left(\frac{\boldsymbol{W}_1 \otimes \boldsymbol{x} - \boldsymbol{\mu}_1}{\boldsymbol{\sigma}_1}\right) - \boldsymbol{\mu}_2}{\boldsymbol{\sigma}_2} + \frac{\boldsymbol{W}_0 \otimes \boldsymbol{x} - \boldsymbol{\mu}_0}{\boldsymbol{\sigma}_0} \tag{8}$$

After expanding training, the weights need to be fused just once and re-parameterized as new static kernels since they are not dependent on the input anymore. As a result, there are no extra parameters and FLOPs at inference time compared with static convolution.

$$\boldsymbol{y} = \left(\frac{\boldsymbol{W}_2}{\boldsymbol{\sigma}_2}\frac{\boldsymbol{W}_1}{\boldsymbol{\sigma}_1} + \frac{\boldsymbol{W}_0}{\boldsymbol{\sigma}_0}\right) \otimes \boldsymbol{x} - \frac{\boldsymbol{W}_2}{\boldsymbol{\sigma}_2}\frac{\boldsymbol{\mu}_1}{\boldsymbol{\sigma}_1} - \frac{\boldsymbol{\mu}_2}{\boldsymbol{\sigma}_2} - \frac{\boldsymbol{\mu}_0}{\boldsymbol{\sigma}_0} \tag{9}$$

$$\boldsymbol{W}_{\text{infer}} = \frac{\boldsymbol{W}_2}{\boldsymbol{\sigma}_2}\frac{\boldsymbol{W}_1}{\boldsymbol{\sigma}_1} + \frac{\boldsymbol{W}_0}{\boldsymbol{\sigma}_0} \tag{10}$$

$$\boldsymbol{b}_{\text{infer}} = -\frac{\boldsymbol{W}_2}{\boldsymbol{\sigma}_2}\frac{\boldsymbol{\mu}_1}{\boldsymbol{\sigma}_1} - \frac{\boldsymbol{\mu}_2}{\boldsymbol{\sigma}_2} - \frac{\boldsymbol{\mu}_0}{\boldsymbol{\sigma}_0} \tag{11}$$

In ExpandNet[21], a $k \times k (k >= 1)$ convolution is expanded vertically by three convolutional layers with kernel size 1×1, $k \times k$, 1×1, respectively. When $k > 1$, it cannot use BN in the intermediate layer. The bias caused by BN fusion cannot pass forward through the $k \times k$ kernel, thus cannot be fused with the bias of the next layer. MAConv avoids this problem, thus can use BN to facilitate the training. Besides, MAConv uses the bypass convolution for shortening the path of error back-propagation. BN and the bypass shortcut in MAConv help the training of deep layers. Both BN and the bypass shortcut can be compressed and re-parameterized to the compact one, thus without any extra cost at inference time.

ACNet[22] and RepVGG[23] horizontally expand the $k \times k (k > 1)$ convolution into convolutions with different kernel shape. That hinders its usage for lightweight networks, which heavily utilizes the 1×1 pointwise convolutions. MAConv uses asymmetric depth instead of asymmetric kernel shape and expands the convolution both vertically and horizontally. MAConv can be used for both 1×1 convolution and $k \times k (k > 1)$ convolution.

4 Multi-layer asymmetric convolution

4.1 ImageNet

ImageNet classification dataset[24] has 1.28×10^6 training images and 50 000 validation images with 1000 classes. The experiments are based on the official example of Pytorch.

The standard augmentation is used for training image as the same as the official example: (1) randomly cropped with the size of 0.08 to 1.0 and aspect ratio of 3/4 to 4/3, and then resized to 224×224; (2) randomly horizontal flipped. The validation image is resized to 256×256, and then center cropped with size 224×224. Each channel of the input image is normalized into 0 mean and 1 STD globally. The batch size is 256. Four TITAN Xp GPUs are used to train the models.

Five network architectures are employed on ImageNet dataset: MobileNetV2 _ x0.5, MobileNetV2 _ x1.0, ShuffleNetV2 _ x0.5, ShuffleNetV2 _ x1.5, and ResNet18. The training setups are as follows. (1) For MobileNetV2 and ShuffleNetV2, the initial learning rate is 0.1 and is scheduled to arrive at zero by using the linear-decay policy. The weight decay is 4e-5. All

models are trained by using stochastic gradient descent (SGD) optimizer with 0.9 momentum for 300 epochs. Dropout with 0.2 is used in the last fully connected layers of MobileNetV2. (2) For ResNet18, the initial learning rate is 0.1 and drops by 10 at epoch 30, 60, and 90. The weight decay is 1e-4. The model is trained using SGD optimizer with 0.9 momentum for 100 epochs.

Firstly, comparisons are made between static convolution, DYConv[10], MAConv, and MDConv on MobileNetV2 and ShuffleNetV2. For DYConv, the number of experts is set to the default value 4[10]. For MDConv and MAConv, the number of intermediate channels m is set to 20. For DYConv and MDConv, the attention network is a two-layer fully-connected network with the hidden units to be 1/4 input channels. As recommended in the original paper[10], the temperature of Softmax is set to 30 in DYConv. DYConv, MDConv, and MAConv are applied to the pointwise convolutional layer in the inverted bottlenecks of MobileNetV2 or the blocks of ShuffleNetV2. They are used to replace the static convolution.

Table1 shows that MAConv enhances the accuracy compared with static convolution. The parameters and FLOPs are the values at inference time. Gains refer to the improvement compared with static convolution. For example, MAConv improves the accuracy of MobileNetV2 _ x0.5 by 0.908% and MobileNetV2 _ x1.0 by 0.982%. These improvements are decent since there is no extra cost of storage or computation at inference time. Besides, DYConv and MDConv significantly increase the accuracy compared with static convolution. Moreover, MDConv achieves better accuracy with fewer parameters compared with DYConv. For example, MobileNetV2 _ x1.0 and ShuffleNetV2 _ x1.5 with MDConv achieve better accuracies with about half parameters compared with DYConv. For MobileNetV2 _ x1.0 and ShuffleNetV2 _ x1.5, the accuracies using MDConv are 1.108% and 0.662% higher than DYConv, respectively.

Table 1 Top-1 accuracies of lightweight networks on ImageNet validation dataset

	Parameters($\times 10^6$)	FLOPs($\times 10^6$)	Accuracies/%	Gains/%
MobileNetV2 _ ×0.5, static	2.0	97	62.712	-
MobileNetV2 _ ×0.5, DYConv	3.6	99	66.504	3.792
MobileNetV2 _ ×0.5, MAConv	2.0	97	63.620	0.908
MobileNetV2 _ ×0.5, MDConv	2.4	106	66.950	4.238
MobileNetV2 _ ×1.0, static	3.5	300	70.806	-
MobileNetV2 _ ×1.0,DYConv	9.8	308	72.806	2.000
MobileNetV2 _ ×1.0, MAConv	3.5	300	71.788	0.982
MobileNetV2 _ ×1.0,MDConv	5.0	335	73.914	3.108
ShuffleNetV2 _ ×0.5, static	1.4	41	59.068	-
ShuffleNetV2 _ ×0.5,DYConv	1.8	42	63.108	4.040
ShuffleNetV2 _ ×0.5,MAConv	1.4	41	59.316	0.248
ShuffleNetV2 _ ×0.5,MDConv	1.5	44	63.210	4.142
MobileNetV2 _ ×1.0, static	3.5	299	71.690	-
MobileNetV2 _ ×1.0,DYConv	9.0	306	73.430	1.740
MobileNetV2 _ ×1.0, MAConv	3.5	299	72.206	0.516
MobileNetV2 _ ×1.0,MDConv	4.2	333	74.092	2.402

Comparisons are also made on $k \times k(k > 1)$ convolution. DYConv, MDConv, and MAConv are applied in the 3×3 convolutional layer of ResNet18's residual block. Table 2 shows that MAConv is also effective on $k \times k(k > 1)$ convolution. MDConv increases the accuracy by 2.086% with only 1×10^6 additional parameters compared with static convolution. Moreover, it achieves higher accuracy with much fewer parameters than DYConv.

Comparisons are then made between MDConv and DCD[12] on MobileNetV2 _ ×1.0. DCD generates a full $m \times m$ attention matrix where m is set to 20, the same as MDConv. To verify that MDConv facilitates the training, the models are trained with different amounts of data (100%, 50%, and 10% original training data). As shown in Table 3, MDConv achieves higher accuracy with fewer parameters than DCD. As the training data decreases, the advantage of MDConv is

more obvious. The accuracy of MDConv is 0.934%, 3.262%, and 3.822% higher than DCD with 100%, 50%, and 10% training data, respectively. That implies DCD is more difficult to train, thus needs more training data. Fig. 2 compares MDConv with DCD in terms of training convergence. Without any additional parameter tuning, MDConv converges faster than DCD and achieves higher accuracy, especially on the smaller training dataset. Moreover, MDConv costs fewer memories and computational resources than DCD, which generates a dense matrix $\boldsymbol{A}$. Assume the size of matrix $\boldsymbol{A}$ is $m \times m$, then DCD brings $m \times m$ extra parameters and computation FLOPs, while MDConv only brings m extra parameters and computation FLOPs since $\boldsymbol{A}$ is diagonal matrix in MDConv.

Table 2 Top-1 accuracies of ResNet18 on ImageNet validation dataset

	Parameters ($\times 10^6$)	Accuracies/%
Static	11.7	70.342
DYConv	44.9	71.736
MAConv	11.7	70.722
MDConv	12.7	72.428

Table 3 Comparison of validation accuracies (%) between DCD and MDConv on MobileNetV2 _ x1.0 trained with different amounts of data

	Parameters	Training data		
		100%	50%	10%
DCD	5.9×10^6	72.980	67.306	47.764
MDConv	5.0×10^6	73.914	70.568	51.586

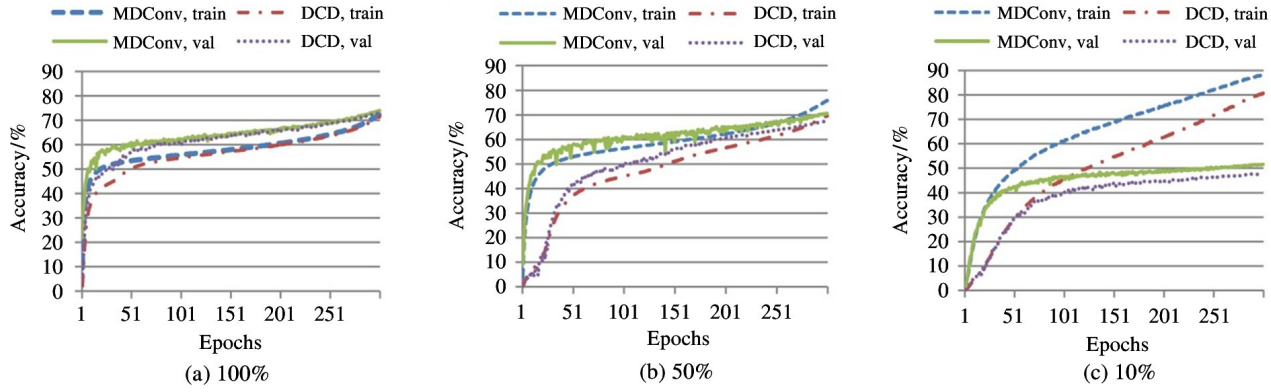

Fig. 2 Comparison of training and validation accuracy curves between DCD and MDConv on MobileNetV2 _ x1.0 trained with different amounts of data

Comparisons are further made between MDConv and the feature attention SE[13]. Table 4 shows that MDConv outperforms the feature attention SE. On MobileNetV2 _ x0.5 and MobileNetV2 _ x1.0, the accuracy of MDConv is 1.986% and 1.488% higher than SE respectively.

Table 4 Comparison of validation accuracies on ImageNet between MDConv and SE

	SE/%	MDConv/%
MobileNetV2 _ ×0.5	64.964	66.950
MobileNetV2 _ ×1.0	72.426	73.914

4.2 CIFAR-10

CIFAR-10 is a dataset of natural 32 × 32 RGB images in 10 classes with 50 000 images for training and 10 000 for testing. The training images are padded with 0 to 36 × 36 and then randomly cropped to 32 × 32 pixels. Then randomly horizontal flipping is made. Each channel of the input is normalized into 0 mean and 1 STD globally.

SGD with momentum 0.9 and weight decay 5e-4 are used. The batch size is set to 128. The learning rate is set to 0.1, and scheduled to arrive at zero using the cosine annealing scheduler. The networks are trained with 200 epochs.

MobileNetV2 with different width multipliers are evaluated on this small dataset. The setups for different attentions are the same as subsection 4.1. Each test is run 5 times. The mean and the STD of the accuracies are listed in Table 5. MAConv increases the accuracy compared with static convolution. MAConv even outperforms DYConv on this small dataset. MDConv further improves the performance and achieves the best performance, while DCD achieves the worst performance and has large variance. That implies again DCD brings more dynamics and is difficult to train on the small dataset.

Table 5 Test accuracies on CIFAR-10 with MobileNetV2

Width	×0.35		×0.5		×0.75		×1.0	
	mean/%	STD	mean/%	STD	mean/%	STD	mean/%	STD
Static	90.668	0.207	91.240	0.268	92.228	0.146	92.410	0.084
DYConv	90.974	0.268	91.614	0.249	92.506	0.102	92.694	0.100
DCD	84.178	3.707	87.286	2.265	88.346	2.116	88.706	2.848
MAConv	91.110	0.306	91.738	0.198	92.678	0.137	92.894	0.285
MDConv	91.352	0.261	92.118	0.225	92.638	0.129	93.060	0.349

4.3 CIFAR-100

CIFAR-100 is a dataset of natural 32 × 32 RGB images in 100 classes with 50 000 images for training and 10 000 for testing. The training images are padded with 0 to 36 × 36 and then randomly cropped to 32 × 32 pixels. Then randomly horizontal flipping is made. Each channel of the input is normalized into 0 mean and 1 STD globally. SGD with momentum 0.9 and weight decay 5e-4 are used. The batch size is set to 128. The learning rate is set to 0.1, and scheduled to arrive at zero using the cosine annealing scheduler. The networks are trained with 200 epochs.

MobileNetV2 _ x0.35 is evaluated on this dataset. The setups for different attentions are the same as subsection 4.1. Each test is run 5 times. The mean and the standard deviation of the accuracies are reported in Table 6. Results show that dynamic networks do not improve the accuracy compared with the static network. Moreover, more dynamic factors lead to worse performance. For example, DCD is worse than MDConv, and MDConv is worse than DyConv. That is because dynamic networks are harder to train and need more training data and CIFAR-100 has 100 classes, and each class has fewer training examples than CIFAR-10. MAConv achieves the best performance, 70.032%. When the training dataset is small, MAConv is still effective to enhance the model capacity instead of dynamic ones.

Table 6 Test accuracies on CIFAR-100

	Accuracies	
	mean/%	STD
Static	69.574	0.358
DYConv	69.564	0.281
DCD	57.920	4.814
MAConv	70.032	0.404
MDConv	68.470	0.862

4.4 SVHN

The street view house numbers (SVHN) dataset includes 73 257 digits for training, 26 032 digits for testing, and 531 131 additional digits. Each digit is a 32 × 32 RGB image. The training images are padded with 0 to 36 × 36 and then randomly cropped to 32 × 32 pixels. Then randomly horizontal flipping is made. Each channel of the input is normalized into 0 mean and 1 STD globally. SGD with momentum 0.9 and weight decay 5e-4 are used. The batch size is set to 128. The learning rate is set to 0.1, and scheduled to arrive at zero using the cosine annealing scheduler. The networks are trained with 200 epochs.

MobileNetV2 _ x0.35 is used on this dataset. The setups for different attentions are the same as subsection 4.1. Each test is run 5 times. The mean and the standard deviation of the accuracies are reported in Table 7. Results show that DCD decreases the performance compared with the static one. DCD is hard to train on the small dataset. DYConv, MAConv, and MDConv increase the accuracy compared with the static one. Among them, MAConv and MDConv achieve similar performance, better than DYConv.

Table 7 Test accuracies on SVHN

	Accuracies	
	mean/%	STD
Static	95.681	0.085
DYConv	95.829	0.079
DCD	95.108	0.352
MAConv	95.911	0.074
MDConv	95.937	0.475

4.5 Ablation study

Ablation experiments are carried out on CIFAR-10 using two network architectures. One is MobileNetV2 _ x0.35. To make the comparison more distinct, a smaller and simpler network named SmallNet is also used. SmallNet has the first convolutional layer with 3 × 3 kernels and 16 output channels, followed by three blocks. Each block comprises of a 1 × 1 pointwise convolution with 1 stride and no padding, and a 3 × 3 depthwise convolution with 2 strides and 1 padding. These 3 blocks have 16, 32 and 64 output channels, respectively. Each convolutional layer is followed by batch normalization, and ReLU activation. The output

of the last layer is passed through a global average pooling layer, followed by a Softmax layer with 10 classification output. Other experiment settings are the same as subsection 4.2.

The effect of the bypass shortcut in the MAConv and MDConv is investigated firstly. To show the effect of dynamic attention, ReLU activation is further used instead of dynamic attention in the multi-layer branch. The results are shown in Table 8, w/ means with bypass shortcut, w/o means without bypass shortcut. Results show that the bypass shortcut improves the accuracy, especially in deeper networks (MobileNetV2 _ x0.35). Moreover, MDConv (with dynamic attention) increases the capabilities of models compared with MAConv (without dynamic attention). Using ReLU activation instead of dynamic attention can further increase the capabilities. However, the weights cannot be fused into compact one anymore because of the non-linear activation function. Thus the costs of storage and computation are much higher than single-layer convolution at inference time.

Table 8 Effect of bypass shortcut

		MobileNetV2/%	SmallNet/%
MAConv	w/	91.110	74.458
	w/o	85.450	73.276
MDConv	w/	91.352	77.998
	w/o	86.332	76.164
ReLU	w/	91.742	78.500
	w/o	87.986	75.250

Next, the networks are trained by using the compact form directly. Table 9 shows the comparison between the expanding training and the compact training. Results show that expanding training improves the performance of MAConv and MDConv. To evaluate the benefit of BN in expanding training, BN is applied after the addition of two branches instead of after each convolution (without BN after each convolution). Results show that BN in expanding form helps the training since it achieves better performance than that without BN. Moreover, the expanding form without BN helps training itself, since it achieves better performance than compact training.

Table 9 Effect of expanding training

		MobileNetV2/%	SmallNet/%
MAConv	Expanding	91.110	74.458
	Compact	90.768	73.260
	Expending w/o BN	90.564	74.092
MDConv	Expanding	91.352	77.998
	Compact	88.474	77.024
	Expending w/o BN	91.096	77.500

The effect of different input/output channels and different intermediate channels are also investigated. Different width multipliers are applied on all layers except the first layer of SmallNet. The results are shown in Table 10. Results show that the gains of MAConv and MDConv are higher with fewer channels. These results imply that over-parameterization is more effective in smaller networks. SmallNet are then trained with different intermediate channels. The results are shown in Table 11. Results show that the gains of MAConv and MDConv are trivial when increasing the intermediate channel on the CIFAR-10 dataset. Using more intermediate channels means that the dynamic part takes more influence, thus increases the difficulty of training and needs more training data.

Table 10 Effect of input/output channel width

Width multipliers	Static/%	MAConv/%	MDConv/%
x1	73.936	74.458	77.998
x2	81.010	81.452	84.204
x3	84.356	84.368	86.398
x4	85.650	85.872	87.424
x5	86.912	86.874	88.140
x6	87.290	87.608	88.552

Table 11 Effect of different intermediate channel width

Width	Static/%	MAConv/%	MDConv/%
20	73.936	74.458	77.998
30		74.924	77.798
40		74.216	78.026
50		74.294	77.970
60		74.156	78.330

Finally, different setups for the attention network are investigated in SmallNet with MDConv. Different numbers of the hidden units are used, ranging from 1/4 times of the input channel to 4 times of the input channel. Table 12 shows that increasing hidden units can improve the performance until 4 times of the input channel. Softmax and Softmax with different temperatures, as proposed in Ref. [10], are used as the gate function in the last layer of the attention network. As shown in Table 13, Softmax achieves better accuracy than Sigmoid. However, the temperature does not improve the performance.

Table 12 Effect of the hidden layer in the attention network

Hidden layer	Accuracy/%
1/4 × input channel	77.998
1/2 × input channel	78.096
1 × input channel	78.550
2 × input channel	79.074
4 × input channel	78.472

Table 13 Effect of the gate function in the last layer of the attention network

Gate function	Accuracy/%
Sigmoid	77.998
Softmax	78.148
Softmax temp = 5	77.160
Softmax temp = 10	76.612
Softmax temp = 20	76.486
Softmax temp = 30	76.326

5 Conclusions

Two powerful convolutions are proposed to increase the model's capacity: MDConv and MAConv. MDConv expands the static convolution into multi-layer dynamic one, with fewer parameters but stronger model capacity than horizontally expanding. MAConv has no extra parameters and FLOPs at inference time compared with static convolution. MDConv and MAConv are evaluated on different networks. Experimental results show that MDConv and MAConv improve the accuracy compared with static convolution. Moreover, MDConv achieves better accuracy with fewer parameters and facilitates the training compared with other dynamic convolutions.

References

[1] HOWARD A G, ZHU M, CHEN B, et al. Mobilenets: efficient convolutional neural networks for mobile vision applications [EB/OL]. https://arxiv.org/pdf/1704.04861.pdf:arxiv,(2017-04-17),[2021-09-09]

[2] SANDLER M, HOWARD A, ZHU M, et al. Mobilenetv2: inverted residuals and linear bottlenecks [C]// Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, USA, 2018: 4510-4520

[3] HOWARD A, SANDLER M, CHU G, et al. Searching for MobileNetV3 [C]// Proceedings of the IEEE/CVF International Conference on Computer Vision, Long Beach, USA, 2019: 1314-1324

[4] TAN M, CHEN B, PANG R, et al. Mnasnet: platform-aware neural architecture search for mobile [C]// Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, USA, 2019: 2820-2828

[5] ZHANG X, ZHOU X, LIN M, et al. Shufflenet: an extremely efficient convolutional neural network for mobile devices [C]// Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, USA, 2018: 6848-6856

[6] ZHANG T, QI G J, XIAO B, et al. Interleaved group convolutions [C]// Proceedings of the IEEE International Conference on Computer Vision, Venice, Italy, 2017: 4373-4382

[7] XIE G, WANG J, ZHANG T, et al. Interleaved structured sparse convolutional neural networks [C]// Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, USA, 2018: 8847-8856

[8] SUN K, LI M, LIU D, et al. IGCV3: interleaved low-rank group convolutions for efficient deep neuralnetworks [EB/OL]. https://arxiv.org/pdf/1806.00178v2.pdf: arXiv,(2018-07-20),[2021-09-09]

[9] YANG B, BENDER G, LE Q V, et al. Condconv: conditionally parameterized convolutions for efficient inference [C]// Advances in Neural Information Processing Systems, Vancouver, Canada, 2019:1307-1318

[10] CHEN Y, DAI X, LIU M, et al. Dynamic convolution: attention over convolution kernels [C]// Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, USA, 2020: 11030-11039

[11] MA N, ZHANG X, HUANG J, et al. Weightnet: revisiting the design space of weight networks [C]// The 16th European Conference on Computer Vision-ECCV 2020, Glasgow, UK, 2020: 776-792

[12] LI Y, CHEN Y, DAI X, et al. Revisiting dynamic convolution via matrix decomposition [EB/OL]. https://arxiv.org/pdf/2103.08756.pdf: arXiv, (2021-03-15), [2021-09-09]

[13] JIA X, DE BRABANDERE B, TUYTELAARS T, et al. Dynamic filter networks [C]// Advances in Neural Information Processing Systems, Barcelona, Spain, 2016: 667-675

[14] HU J, SHEN L, SUN G. Squeeze-and-excitation networks [C]// Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, USA, 2018: 7132-7141

[15] LIN J, RAO Y, LU J, et al. Runtime neural pruning [C]// Proceedings of the 31st International Conference on Neural Information Processing Systems, Long Beach, USA, 2017: 2178-2188

[16] LIU L L, DENG J. Dynamic deep neural networks: optimizing accuracy-efficiency trade-offs by selective execution [C]// The 32nd AAAI Conference on Artificial Intelligence, New Orleans, USA, 2018: 1-12

[17] WANG X, YU F, DOU Z Y, et al. Skipnet: learning dynamic routing in convolutional networks [C]// Proceedings of the European Conference on Computer Vision, Munich, Germany, 2018: 409-424

[18] WU Z, NAGARAJAN T, KUMAR A, et al. Blockdrop: dynamic inference paths in residual networks [C]// Pro-

ceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, USA, 2018: 8817-8826

[19] YU J, YANG L, XU N, et al. Slimmable neural networks[EB/OL]. https://arxiv.org/pdf/1812.08928.pdf;arXiv,(2018-12-21),[2021-09-09]

[20] HUANG G, CHEN D, LI T, et al. Multi-scale dense networks for resource efficient image classification[EB/OL]. https://arxiv.org/pdf/1703.09844.pdf;arXiv,(2018-06-08),[2021-09-09]

[21] GUO S, ALVAREZ J M, SALZMANN M. Expandnets: linear over-parameterization to train compact convolutional networks[EB/OL]. https://arxiv.org/pdf/1811.10495.pdf;arxiv,(2021-04-14),[2021-09-09]

[22] DING X, GUO Y, DING G, et al. ACNet: strengthening the kernel skeletons for powerful CNN via asymmetric convolution blocks[C]// Proceedings of the IEEE/CVF International Conference on Computer Vision, Long Beach, USA, 2019: 1911-1920

[23] DING X, ZHANG X, MA N, et al. RepVGG: making VGG-style ConvNets great again[C]// Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Nashville, USA, 2021: 13733-13742

[24] RUSSAKOVSKY O, DENG J, SU H, et al. Imagenet large scale visual recognition challenge[J]. *International Journal of Computer Vision*, 2015, 115(3): 211-252

LUO Chunjie, born in 1987. He is a Ph. D. candidate of University of Chinese Academy of Sciences. He received his M. S. degree and B. S. degree from Huazhong University of Science and Technology in 2009 and from University of Chinese Academy of Sciences in 2012 respectively. His research interests include benchmarking and machine learning.