Circulant Binary Convolutional Networks: Enhancing the Performance of 1-bit DCNNs with Circulant Back Propagation

Chunlei Liu,[1] Wenrui Ding,[1] Xin Xia,[1] Baochang Zhang,[1]*Jiaxin Gu,[1]
Jianzhuang Liu,[2] Rongrong Ji,[3] David Doermann[4]

[1] Beihang University [3] Huawei Noah's Ark Lab
[2] Xiamen University [4] University at Buffalo
{liuchunlei, ding, xiaxin, bczhang}@buaa.edu.cn

Abstract

The rapidly decreasing computation and memory cost has recently driven the success of many applications in the field of deep learning. Practical applications of deep learning in resource-limited hardware, such as embedded devices and smart phones, however, remain challenging. For binary convolutional networks, the reason lies in the degraded representation caused by binarizing full-precision filters. To address this problem, we propose new circulant filters (CiFs) and a circulant binary convolution (CBConv) to enhance the capacity of binarized convolutional features via our circulant back propagation (CBP). The CiFs can be easily incorporated into existing deep convolutional neural networks (DCNNs), which leads to new Circulant Binary Convolutional Networks (CBCNs). Extensive experiments confirm that the performance gap between the 1-bit and full-precision DCNNs is minimized by increasing the filter diversity, which further increases the representational ability in our networks. Our experiments on ImageNet show that CBCNs achieve 61.4% top-1 accuracy with ResNet18. Compared to the state-of-the-art such as XNOR, CBCNs can achieve up to 10% higher top-1 accuracy with more powerful representational ability.

1. Introduction

Deep convolutional neural networks (DCNNs) have been successfully demonstrated on many computer vision tasks such as object detection and image classification. DCNNs deployed in practical environments, however, still face many challenges. It is particularly true when the portability and real time performance are required. This is critical because models of vision applications can require very large memory, making them impractical for most embedded platforms. Besides, floating-point inputs and network weights along with forward or backward data flow can result in a significant computational burden.

*Baochang Zhang is the corresponding author.

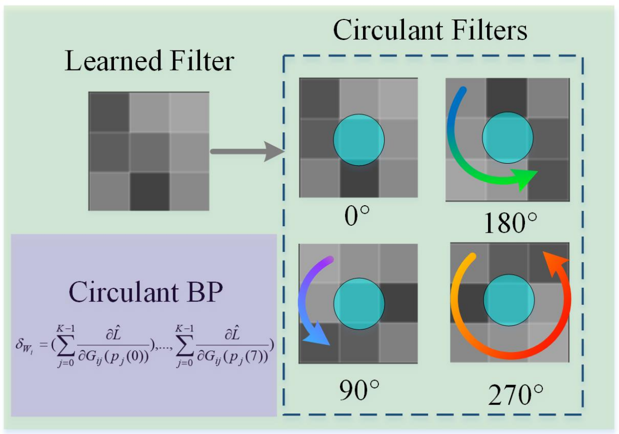

Circulant Filters
Learned Filter
Circulant BP
$\delta_{W_i} = (\sum_{j=0}^{K-1} \frac{\partial \hat{L}}{\partial G_{ij}(p_j(0))}), \ldots, \sum_{j=0}^{K-1} \frac{\partial \hat{L}}{\partial G_{ij}(p_j(7))})$
0°
180°
90°
270°

Figure 1. Circulant back propagation (CBP). We manipulate the learned convolution filters using the circulant transfer matrix, which is employed to build our CBP. By doing so, the capacity of the binarized convolutional features are significantly enhanced, e.g., robustness to the orientation variations in objects, and the performance gap between the 1-bit and full-precision DCNNs is minimized. In the example, 4 CiFs are produced based on the learned filter and the circular matrix.

Binary filters instead of using full-precision weights have been investigated in DCNNs to compress the deep models to handle the aforementioned problems. They are also called 1-bit DCNNs, as each weight parameter and activation can be represented by a single bit. As presented in [10], XNOR has both the convolution weights and inputs attached to the convolution be approximated with binary

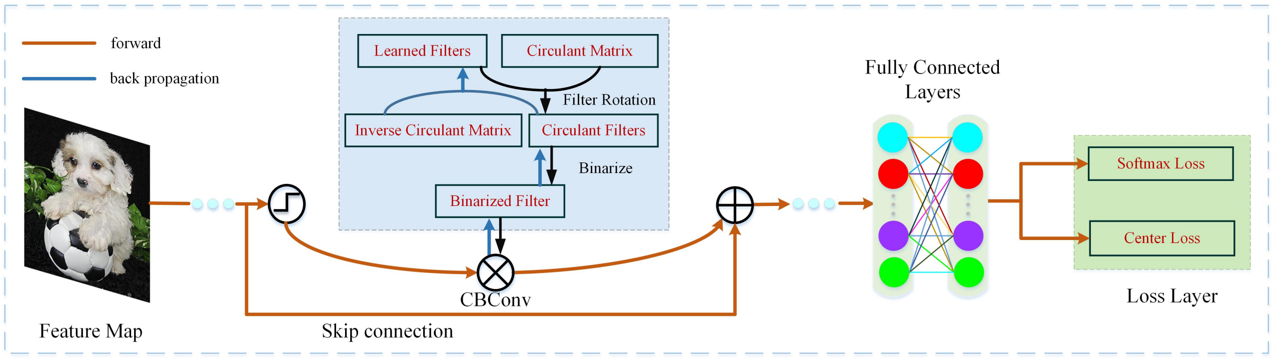


Figure 2. Circulant Binary Convolutional Networks (CBCNs) are designed based on circulant and binary filters to variate the orientations of the learned filters in order to increase the representational ability. By considering the center loss and softmax loss in a unified framework, we achieve much better performance than state-of-the-art binarized models. Most importantly, our CBCNs also achieve the performance comparable to well-known full-precision ResNets and WideResNets. The circulant binary filters are only shown for demonstrating the computation procedure, which are not saved for testing.

values, providing an efficient implementation of convolutional operations, particularly by reconstructing the unbinarized filters with a single scaling factor. More recently, Bi-Real Net [20] explores a new variant of residual structure to preserve the real activations before the sign function and TBN [16] replaces the sign function with a threshold-based ternary function to obtain ternary input tensor. Both provide an optimal tradeoff among memory, efficiency and performance. A warm-restart learning-rate schedule in [9] is adopted to accelerate the training for 1-bit-per-weight networks. Furthermore, a method called WAGE [17] is proposed to discretize both training and inference, where not only weights and activations but also gradients and errors are quantized. In these previous methods, however, the binarization of the filters often degrades the representational ability of the models for the rotation variations in objects.

In this paper, we introduce circulant filters (CiFs) and the circulant binary convolution (CBConv) to actively calculate diverse feature maps, which can improve the representational ability of the resulting binarized DCNNs. The key insight of producing CiFs to help back propagation is shown in Fig. 1. Compared to previous binarized DCNN filters, CiFs are defined based on a circulant operation on each learned filter. A new circulant back propagation (CBP) algorithm is also introduced to develop an end-to-end trainable DCNN. During the convolution, CiFs are used to produce diverse feature maps which provide the binarized DCNNs with the ability to capture variations previously unseen. Instead of introducing extra functional modules or new network topologies, our method implements CBConv onto the most basic element of DCNNs, the convolution operator. Thus, it can be naturally fused with modern DCNN architectures, upgrading them to more expressive and compact Circulant Binary Convolutional Networks (CBCNs) for resource limited applications. We design a simple and unique variation process, which is deployed at each layer and can be solved within the same pipeline of the new CBP algorithm. In addition, we consider the center loss to further enhance the performance of CBCNs as shown in Fig. 2. Thanks to the low model complexity, such an architecture is less prone to over-fitting and suitable for resource-constrained environments. Our CBCNs reduce the required storage space of full-precision models by a factor of 32, while achieving better performance than existing binarized filters based DCNNs. The contributions of this paper are summarized as follows:

(1) CiFs are used to obtain more robust feature representation, e.g., orientation variations in objects, which minimize the performance gap between full-precision DCNNs and binarized DCNNs.

(2) We develop a CBP algorithm to reduce the loss during back propagation and make convolutional networks more compact and efficient. Experimental results show that CBP is not only effective, but also converged quickly.

(3) The presented circulant convolution is generic, and can be easily used on existing DCNNs, such as ResNets and conventional DCNNs. Our highly compressed models outperform state-of-the-art binarized models by a large margin on MNIST, CIFAR and ImageNet databases.

2. Related Work

DCNNs with a large number of parameters consume considerable computational resources. From our practical study, about half of the power consumption of a DCNN is related to the model size. Many attempts have been made to accelerate and simplify DCNNs while avoiding the increase of the errors. Network binarization is one of the most popular approaches, which is briefly reviewed below.

The research in [6] demonstrates that the storage of real-valued deep neural networks such as AlexNet [4], GoogLeNet [15] and VGG-16 [13] can be reduced signif-

Table 1. A brief description of variables and operators used in the paper.

M :circulant transfer matrix	G : circulant filter	W : learned filter	F : feature map
δ_W : the gradient of the learned filter W	P : inverse circulant transfer matrix	$\hat{G}$: binarized filter	L: loss function
K : number of orientations for each filter	i : filter index	j : orientation index	l : layer index
g : input feature map index	h : output feature map index	η : learning rate	

icantly when their 32-bit parameters are quantized to 1-bit. Expectation BackPropagation (EBP) in [14] uses a variational Bayesian approach to achive high performance with a network with binary weights and activations. BinaryConnect (BC) [1] extends the idea of EBP by employing 1-bit precision weights (1 and -1). Later, Courbariaux et al. [2] propose BinaryNet (BN) that is an extension of BC and further constrains activations to +1 and -1, binaring the input (except the first layer) and the output of each layer. BC and BN both achieve sufficiently high accuracy on small datasets such as MNIST, CIFAR10 and SVHN. According to Rastegari et al. [11], BC and BN are not very successful on large-scale data sets. XNOR [10] has a different binarization method and a network architecture. Both the weights and inputs attached to the convolution are approximated with binary values, which results in an efficient implementation of the convolutional operations by reconstructing the unbinarized filters with a single scaling factor. Recent studies such as MCN [18] and Bi-Real Net [20] have been conducted to explore new network structures and training techniques for binarizing both weights and activations while minimizing accuracy degradation using a concept similar to XNOR. MCN introduces modulated filters to recover the unbinarized filters and leads to a new architecture to calculate the network model. Bi-Real Net connects the real activations to the activations of consecutive blocks through an identity shortcut.

The results of these studies are encouraging, but due to the weight binarization process, the representational ability of the networks can be degraded. This inspires us to seek a way to increase the filter variations in order to increase the network representation ability. In particular, for the first time, we use the circulant matrix to build CiFs for our binarized CNNs. We also develop a CBP algorithm to make the DCNNs more compact and effective in an end-to-end framework.

3. Methodology

We design a specific architecture in CBCNs based on CiFs, and train it with a new BP algorithm. Attempting to increase the representational ability reduced by the binarization process, CiFs are designed to enrich the binarized filters for the enhancement of the network performance. As shown in the experiments, the performance drop is marginal even when the learned network parameters are highly compressed. First of all, Table 1 gives the main notation used in

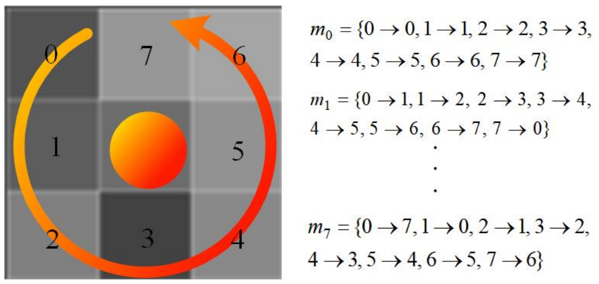

0
7
6
1
5
2
3
4
m_0 = {0 → 0, 1 → 1, 2 → 2, 3 → 3, 4 → 4, 5 → 5, 6 → 6, 7 → 7}
m_1 = {0 → 1, 1 → 2, 2 → 3, 3 → 4, 4 → 5, 5 → 6, 6 → 7, 7 → 0}
m_7 = {0 → 7, 1 → 0, 2 → 1, 3 → 2, 4 → 3, 5 → 4, 6 → 5, 7 → 6}

Figure 3. Illustration of the circulant transfer matrix M for $K = 8$. The center position stays unchanged, and the remaining numbers are circled in a counter-clockwise direction. Each column of M is obtained from m_0 with a rotation angle $\in \{0°, 45°, ..., 315°\}$. It clearly shows that a circulant filter explicitly encodes the position and orientation.

this paper.

3.1. Circulant Transfer Matrix M

A circulant matrix M is defined by a single vector in the first column, with cyclic permutations of the vector with offset equal to the column index in the remaining columns. An important property of the circulant matrix is that it can produce different representations using simple vectors or matrices. With this unique characteristic, we define the circulant transfer matrix of K columns as $M = (m_0, ..., m_j, ..., m_{K-1}), j = \{0, 1, .., K-1\}$:

$$M = \begin{pmatrix} 0 & 7 & 6 & 5 & 4 & 3 & 2 & 1 \\ 1 & 0 & 7 & 6 & 5 & 4 & 3 & 2 \\ 2 & 1 & 0 & 7 & 6 & 5 & 4 & 3 \\ 3 & 2 & 1 & 0 & 7 & 6 & 5 & 4 \\ 4 & 3 & 2 & 1 & 0 & 7 & 6 & 5 \\ 5 & 4 & 3 & 2 & 1 & 0 & 7 & 6 \\ 6 & 5 & 4 & 3 & 2 & 1 & 0 & 7 \\ 7 & 6 & 5 & 4 & 3 & 2 & 1 & 0 \end{pmatrix}, \quad (1)$$

where $K = 8$ and 8 vector rotations are used to form M. The first column m_0 corresponds to the numbers in Fig. 3, and the other columns are obtained by a counter-clockwise rotation of the numbers. Each column of M represents one rotation angle $\in \{0°, 45°, 90°, 135°, 180°, 225°, 270°, 315°\}$. We set m_0 to correspond to the learned filter and m_{1-7} to the derived rotated versions of m_0.

3.2. Circulant Filters (CiFs)

We now design the specific convolutional filters CiFs used in our CBCNs. A CiF is a $4D$ tensor of size

$K \times K \times H \times H$, generated based on a learned filter and M. These CiFs are deployed across all convolutional layers. As shown in Fig. 4(a), the original $2D$ $H \times H$ learned filter is modified to $3D$ by replicating it three times and concatenating them to obtain the $4D$ CiF. For $K = 4$, every channel of the network input is replicated as a group of four elements. By doing so, we can easily implement our CBCNs using PyTorch. One example of the CBConv is illustrated in Fig. 4(b), and the comparison between traditional convolution and CBConv at a network layer is shown in Fig. 5.

To facilitate the math description below, we represent a $2D$ $H \times H$ learned filter as a $1D$ vector of size H^2 so that its corresponding $4D$ CiF can be represented using a $2D$ matrix of size $H^2 \times K$ (see Fig. 4(a)). Let G_i be such a matrix representing the i^{th} CiF. Then

$$\begin{aligned} G_i &= (G_{i0}, ..., G_{ij}, ..., G_{i(K-1)}) \\ &= (W_i \circ m_0, ..., W_i \circ m_j, ..., W_i \circ m_{K-1}), \end{aligned} \tag{2}$$

where W_i is a $1D$ vector containing the i^{th} learned filter's weights (including the unchanged central one during rotation), and $\circ$ denotes the rotation operation of W_i with m_j (see Fig. 3). G_{i0} corresponds to the i^{th} learned filter and the other columns of G_i are introduced to increase the representational ability.

3.3. Forward Propagation of CBCNs based on the CBConv Module

In CBCNs, a binary CiF denoted by $\hat{G}_i$ is calculated as:

$$\hat{G}_i = sign(G_i), \tag{3}$$

where G_i is the corresponding full-precision CiF, and the values of $\hat{G}_i$ are 1 or -1. Both G_i and $\hat{G}_i$ are jointly obtained in the end-to-end learning framework. Let the set of all the binarized filters in the l^{th} layer be $\hat{G}^l$. Then the output feature maps F^{l+1} are obtained by:

$$F^{l+1} = CBConv(sign(F^l); \hat{G}^l), \tag{4}$$

where $CBConv$ is the convolution operation implemented as a new module including the CiF generation process (the blue part in Fig. 2). Fig. 4(b) shows a simple example of a forward convolution where there is one input feature map with one generated output feature map. In the CBConv, the channels of an output feature map are generated as follows:

$$F^{l+1}_{h,j} = \sum_{i,g} F^l_g * \hat{G}^l_{ij}, \tag{5}$$

$$F^{l+1}_h = \{F^{l+1}_{h,0}, F^{l+1}_{h,1}, ..., F^{l+1}_{h,K-1}\}, \tag{6}$$

where $*$ denotes the convolution operation. $F^{l+1}_{h,k}$ is the k^{th} channel of the h^{th} feature map, and F^l_g denotes the g^{th} feature map of the input in the l^{th} convolutional layer. In Fig. 4(b), $h = 1$ and $g = 1$, where after the CBConv with one binary CiF, the number of the channels of the output feature map is the same as that of the input feature map.

Fig. 5(b) illustrates another example of CBConv with multiple feature maps. One output feature map is the sum of the convolution between all of the 10 input feature maps and the 10 binary CiFs in one group (different groups are colored differently). For example, for the first output feature map, $h = 1$, $i = 1, ..., 10$, $g = 1, ..., 10$, and for the second output feature map, $h = 2$, $i = 11, ..., 20$, $g = 1, ..., 10$.

One advantage of our CBConv module lies in that the numbers of the input and output channels in all of the feature maps are the same, so it can easily be replicated for every layer.

3.4. Circulant Back Propagation (CBP)

During the back-propagation, what needs to be learned and updated are the learned filters only. And the inverse transformation of the circulant transfer matrix M is involved in the process of BP to further enhance the representational ability of our CBCNs. To facilitate the math description below, we define the inverse circulant matrix P of K columns as $P = (p_0, ..., p_j, ..., p_{K-1})$, $j = \{0, 1, .., K-1\}$:

$$P = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 0 \\ 2 & 3 & 4 & 5 & 6 & 7 & 0 & 1 \\ 3 & 4 & 5 & 6 & 7 & 0 & 1 & 2 \\ 4 & 5 & 6 & 7 & 0 & 1 & 2 & 3 \\ 5 & 6 & 7 & 0 & 1 & 2 & 3 & 4 \\ 6 & 7 & 0 & 1 & 2 & 3 & 4 & 5 \\ 7 & 0 & 1 & 2 & 3 & 4 & 5 & 6 \end{pmatrix}, \tag{7}$$

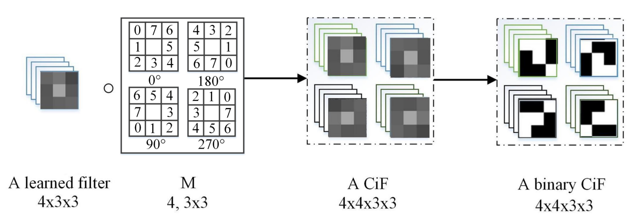
where $K = 8$ and 8 vector inverse rotations are used to form P. Let δ_{W_i} be the gradient of a learned filter W_i. Note that we need to sum up the gradients of the K sub-filters in the corresponding CiF, G_i. Thus:

$$\delta_{W_i} = (\sum_{j=0}^{K-1} \frac{\partial \hat{L}}{\partial G_{ij}(p_j(0))}, ..., \sum_{j=0}^{K-1} \frac{\partial \hat{L}}{\partial G_{ij}(p_j(7))}), \tag{8}$$

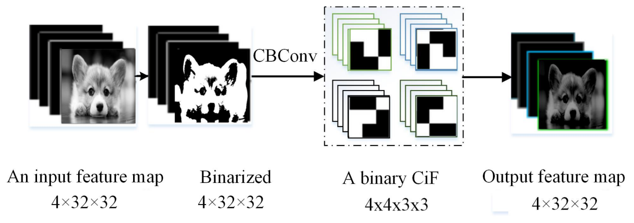
$$W_i \leftarrow W_i - \eta \delta_{W_i}, \tag{9}$$

where $\hat{L}$ is the network loss function, and η is the learning rate. Note that since the central weights of CiFs are not rotated, their gradients are obtained as in the common BP procedure and are not presented in Eq. 8. As is shown, the circular operation involves in our BP process, which makes CBP be adaptive to orientation variations in objects.

For the gradient of the sign function, some special process is necessary due to its discontinuity property. In [2] and [20], the sign function is approximated by the clip function and the piecewise polynomial function, respectively,

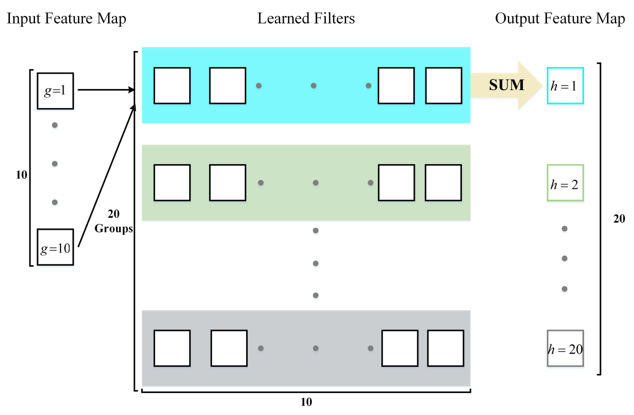


(a) Traditional convolution

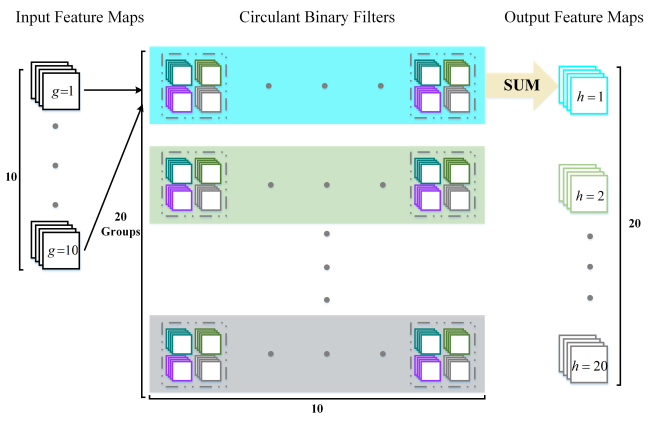


(b) CBConv

Figure 4. CiF and CBConv examples for $K = 4$ orientations (0°, 90°, 180°, 270°) and $H = 3$. (a) The generation of a CiF and its corresponding binary CiF based on a learned filter and M. To obtain the $4D$ CiF, the original $2D$ $H \times H$ learned filter is modified to $3D$ by copying it 3 times. (b) CBConv on an input feature map. Note that in this paper, a feature map is defined as $3D$ with K channels, and these channels are usually not the same.



(a) Traditional convolution



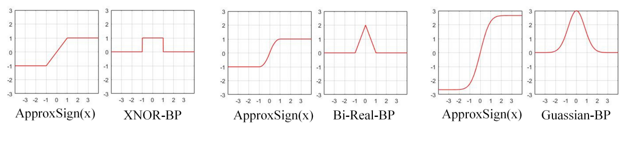
(b) CBConv

Figure 5. Comparison between traditional convolution and CBConv at a network layer. (a) There are 10 and 20 channels in the input and the output feature maps, respectively. (b) The corresponding CBConv where the binary $4D$ CiFs are generated based on their corresponding learned $2D$ filters in (a) and the circulant transfer matrix M.

as shown in Fig. 6(a) and Fig. 6(b) where their corresponding derivatives are also given. Since the derivative of the sign function (an impulse) can be represented as $\lim_{\sigma \to 0} \frac{1}{\sigma\sqrt{\pi}} \exp(-\frac{G^2}{\sigma^2})$, in this work, we use this Gaussian function (Fig. 6(c)) as the approximation of the gradient:

$$\frac{\partial \hat{G}_i}{\partial G_i} = \frac{A}{\sigma\sqrt{\pi}} \exp(-\frac{G_i^2}{\sigma^2}), \tag{10}$$

where A and σ are the amplitude gain and variance of the Gaussian function, respectively, which are determined empirically. In our experiments, we find that our approximation in Fig. 6(c) is better than those in Fig. 6(a) and Fig. 6(b). From the equations above, we can see that the BP process can be easily implemented. Thus only updating the learned convolution filters with the help of CiFs, our CBCNs are significantly compact and efficient, reducing the memory storage by 32. Finally, the learning algorithm to train CBCNs is given in Algorithm 1.



(a) (b) (c)

Figure 6. Three approximations of the sign function for its gradient computation. (a) The clip function and its derivative in [2]. (b) The piecewise polynomial function and its derivative in [20]. (c) Our proposed function and its derivative.

4. Experiments

Our CBCNs are evaluated on object classification using MNIST [7], CIFAR10/100 [5] and ILSVRC12 ImageNet datasets [12]. LeNet [7], WideResNet (WRN) [19] and ResNet18 [3] are employed as the backbone networks to build our CBCNs simply by replacing the full-precision convolution with CBConv. Also, binarizing the neuron activations is carried out in all of our experiments.

Table 2. Error rates on the MNIST and CIFAR10 and their variants. 'fp' denotes the full precision result. The bold denotes the best result among the binary networks.

Dataset	Backbone	kernel stage	original (%)			rot (%)		
			fp	XNOR	CBCN	fp	XNOR	CBCN
MNIST	LeNet	5-10-20-40	0.91	3.76	**1.91**	2.77	17.26	**5.76**
		10-20-40-80	0.69	1.50	**1.24**	1.89	7.77	**4.95**
CIFAR10	ResNet18	16-16-32-64	8.94	22.88	**10.9**	19.07	40.75	**19.68**
		32-32-64-128	6.63	15.55	**8.13**	12.96	33.69	**16.2**
		32-64-128-256	5.27	13.43	**8.09**	10.47	21.93	**15.11**

Algorithm 1 CBCN Training.

```
Require: The training dataset; the full-precision learned filters $W$; the circulant transfer matrix $M$; the number of orientations $K$; hyper-parameters such as initial learning rate, weight decay, convolution stride and padding size.
Ensure: A CBCN based on the CiFs.
1:  Initialize $W$ randomly;
2:  repeat
3:      // Forward propagation
4:      for all $l = 1$ to $L$ convolutional layer do
5:          Use Eqs. 1 and 2 to obtain $G^l$;
6:          $F^{l+1} = CBConv(\text{sign}(F^l), \text{sign}(G^l))$;
7:      end for
8:      // Back propagation
9:      for all $l = L$ to 1 do
10:         Calculate the gradients $\delta_W$; // Using Eq. 8
11:         $W \leftarrow W - \eta\delta_W$; // Update the parameters
12:     end for
13: until the maximum epoch.
```

4.1. Datasets and Implementation Details

Datasets: The MINIST [7] dataset is composed of a training set of 60,000 and a testing set of 10,000 32×32 grayscale images of hand-written digits from 0 to 9. Each sample is randomly rotated in $[-45°, 45°]$ yielding MNIST-rot.

CIFAR10 [5] is a natural image classification dataset containing a training set of $50,000$ and a testing set of $10,000$ 32×32 color images across the following 10 classes: airplanes, automobiles, birds, cats, deers, dogs, frogs, horses, ships, and trucks, while CIFAR100 consists of 100 classes. And we randomly rotate each sample in the CIFAR10 dataset between $[0, 360°]$ to yield CIFAR10-rot.

ImageNet object classification dataset [12] is more challenging due to its large scale and greater diversity. There are 1000 classes and 1.2 million training images and 50k validation images in it. We compare our method with the state-of-the-art on the ImageNet dataset and we adopt ResNet18 to validate the superiority and effectiveness of CBCNs.

In the implementation, LeNet, WRN, and ResNet18 backbone networks are used to build CBCNs. We simply replace the full-precision convolution with CBConv, and keep other components unchanged. The parameters σ and A for the Gaussian function in the Eq. 10 are set to 1 and $3\sqrt{2\pi}$, respectively. More details are elaborated below.

LeNet Backbone: LetNet contains four simple convolutional layers. We adopt Max-pooling and ReLU after each convolution layer, and a dropout layer after the fully connected layer to avoid over-fitting. The initial learning rate is 0.01 with no degradation before reaching the maximum epoch of 50 for MNIST and MNIST-rot.

WRN Backbone: WRN is a network structure similar to ResNet with a depth factor k to control the feature map depth dimension expansion through 3 stages, within which the dimensions remain unchanged. For simplicity we fix the depth factor to 1. Each WRN has a parameter i which indicates the channel dimension of the first stage and we set it to 16 leading to a network structures 16-16-32-64. The training details are the same as in [19]. The initial learning rate is 0.01 with a degradation of 10% for every 60 epochs before it reaches the maximum epoch of 200 for CIFAR10/100 and CIFAR10-rot. For example, WRN22 is a network with 22 convolutional layers and similarly for WRN40.

ResNet18 Backbone: Fig. 7 respectively illustrates the architectures of ResNet18, XNOR and CBCNs. SGD is used as the optimization algorithm with a momentum of 0.9 and a weight decay 1e-4. The initial learning rate is 0.01 with a degradation of 10% for every 20 epochs before reaching the maximum epoch of 70 on ImageNet, while on CIFAR10/100, the initial rate is 0.01 with a degradation of 10% for every 60 epochs before reaching the maximum epoch of 200.

4.2. Rotation Invariance

With LeNet and ResNet18 backbones, we build XNOR and CBCNs and compare them on MNIST, MNIST-rot, CIFAR10, and CIFAR10-rot. K is set to 4 in CBCNs.

Table 2 gives the results in terms of error rates, and 'fp' represents the full-precision results. The state-of-the-

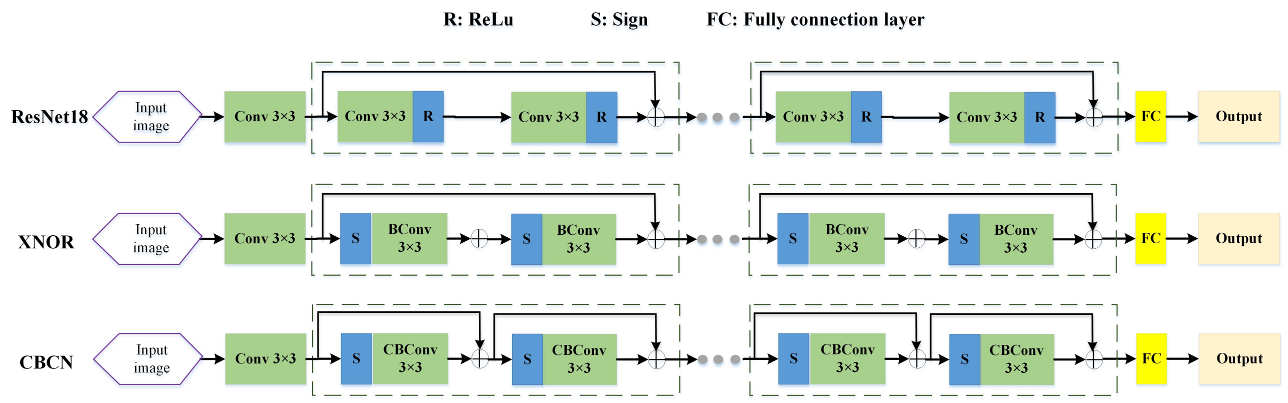


Figure 7. Network architectures of ResNet18, XNOR on ResNet18 and CBCN on ResNet18. Note that CBCN doubles the shortcuts.

Table 3. Performance (accuracy, %) contributions of the components in CBCNs on CIFAR10, where ConvComp, S, C, and G denote the convolution comparison between BConv in XNOR and CBConv, doubled shortcuts, using the center loss, and using the Gaussian gradient function, respectively. The bold number represents the best result.

	Conv -Comp	S	S+C	S+G	S+C +G
XNOR	76.3	80.53	80.97	81.65	**82.32**
CBCN (K=2)	81.84	85.79	86.23	86.67	**87.56**
CBCN (K=4)	84.79	89.10	89.6	90.22	**90.83**
CBCN (K=8)	86.79	90.80	91.27	91.53	**92.02**

art XNOR has a dramatical performance drop on the more challenging rotated datasets. On MNIST-rot, with the kernel stage 5-10-20-40, CBCN shows impressive performance improvement 11.5% over XNOR, while 1.85% improvement is achieved on MNIST. On CIFAR10-rot, with the kernel stage 16-16-32-64, CBCN has about 20% improvement over XNOR. From Table 2, we can also see that on CIFAR10-rot, the performance gap between CBCN and XNOR decreases from about 20% to 17% to 6% with the increase of the kernel stage (parameters), meaning that the improvement of CBCN over XNOR is more significant when they have fewer parameters. The results in Table 2 confirm that with the improved representation ability from the proposed CiFs, CBCNs are more robust than conventional binarization methods for rotation variations of input images.

4.3. Ablation Study

In this section, we study the performance contributions of the components in CBCNs, which include CBConv, center loss, additional shortcuts (Fig. 7), and the Gaussian gradient function (Eq. 10). CIFAR10 and ResNet18 with kernel stage 16-16-32-64 are used in this experiment. The details are given below.

1) We only replace the convolution BConv in XNOR with our CBConv convolution and compare the results. As shown in the ConvComp column in Table 3, CBCN (K=4) achieves about 8% accuracy improvement over XNOR (84.79% vs. 76.3%) using the same network structure and shortcuts as in ResNet18. This significant improvement verifies the effectiveness of our CBConv.

2) In CBCNs, if we double the shortcuts (Fig. 7), we can also find a decent improvement from 84.79% to 89.10% (see the column under S in Table 3), which shows that the increase of shortcuts can also enhance binarized deep networks.

3) Fine-tuning CBCN with the center loss can also improve the performance of CBCN by 0.5% as shown in the column under S+C in Table 3).

4) Replacing the piecewise polynomial function in [20] with the Gaussian function for back propagation, CBCN obtains 1.12% improvement (90.22% vs. 89.10%), which shows that the gradient function we use is a better choice.

5) From the column under S in Table 3, we can see that CBCN performs better using more orientations in CiFs. More orientations can better deal with the problem of degraded representation caused by network binarization.

4.4. Accuracy Comparison with State-of-the-Art

CIFAR10/100: The same parameter settings are used in CBCNs on both CIFAR10 and CIFAR100. We first compare our CBCNs with original ResNet18 with stage kernels as 16-16-32-64 and 32-64-128-256, followed by a comparison with the original WRNs with the initial channel dimension 16 in Table 4. Thanks to the multiple orientations in use, our results on both datasets are comparable to the full-precision networks ResNe18 and WRN40. Then, we compare our results with other state-of-the-arts such as BNN [1], BWN [10], and XNOR [10]. It is observed that at least 1.84% (= 93.42%-91.58%) accuracy improvement

Table 5. Classification accuracy (%) on ImageNet. The bold represents the best result among the binary networks. $K = 4$ in CBCN.

		Full-Precision	XNOR	ABC-Net	BinaryNet	Bi-Real	CBCN
ResNet18	Top-1	69.3	51.2	42.7	42.2	56.4	**61.4**
	Top-5	89.2	73.2	67.6	67.1	79.5	**82.80**

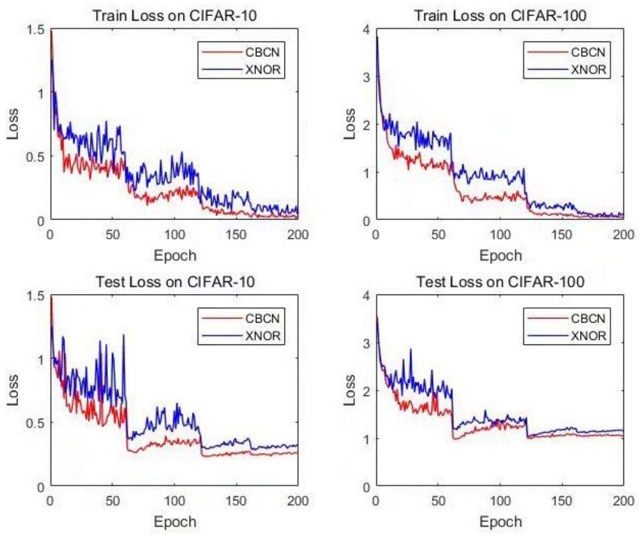


Figure 8. Training and Testing error curves of CBCN and XNOR based on WRN40 for the CIFAR10/100 experiments.

Table 4. Classification accuracy (%) based on ResNet18 and WRN40, respectively, on CIFAR10/100. The bold represents the best result among the binary networks. $K = 4$ in CBCN.

Model	Kernel Stage	Dataset	
		CIFAR -10	CIFAR -100
BNN	-	89.85	-
BWN	-	90.12	-
XNOR (ResNet18)	64-64-128-256	87.1	66.08
XNOR (WRN40)	64-64-128-256	91.58	73.18
ResNet18	16-16-32-64	94.84	75.37
CBCN	16-16-32-64	90.22	69.97
CBCN	32-64-128-256	**91.60**	**70.07**
WRN40	16-16-32-64	95.8	79.41
WRN22	16-16-32-64	90.32	67.19
CBCN	16-16-32-64	**93.42**	**74.80**

is gained with our CBCN, and in other cases, larger margins are achieved. Also, we plot the training and testing loss curves of XNOR and CBCN, respectively, in Fig. 8, which clearly show that CBCN (CBP) converges faster than XNOR (BP).

ImageNet: Four state-of-the-art methods on ImageNet are chosen for comparison: Bi-Real Net [20], BinaryNet [2], XNOR [10] and ABC-Net [8]. These four networks are representative methods of binarizing both network weights and activations and achieve state-of-the-art results. All the

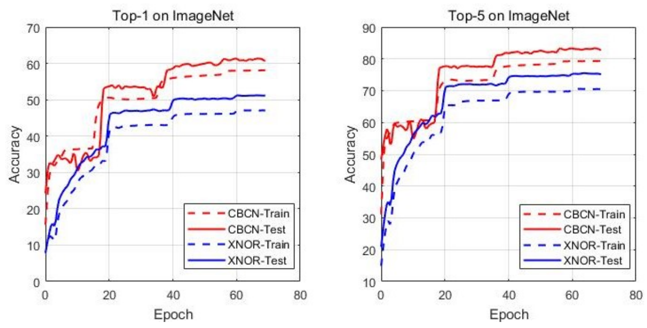


Figure 9. Training and Testing error curves of CBCN and XNOR based on the ResNet18 backbone on ImageNet.

methods in Table 5 perform the binarization of ResNet18. For a fair comparison, our CBCN contains the same amount of learned filters as ResNet18. The comparative results in Table 5 are quoted directly from the references, except that the result of BinaryNet is from [8]. The comparison clearly indicates that the proposed CBCN outperforms the four binary networks by a considerable margin in terms of both the top-1 and top-5 accuracies. Specifically, for top-1 accuracy CBCN outperforms BinaryNet and ABC-Net with a gap over 18%, achieves about 10% improvement over XNOR, and about 5% over the latest Bi-Real Net. In Fig. 9, we plot the training and testing loss curves of XNOR and CBCN, respectively. It clearly shows that using our CBP algorithm, CBCN converges faster than XNOR.

5. Conclusion

In this paper, we have proposed new circulant binary convolutional networks (CBCNs) that are implemented by a set of binary circulant filters (CiFs). The proposed CiFs and circulant binary convolution (CBConv) are used to enhance the representation ability of binary networks. CBCNs can be trained end-to-end with the developed circulant BP (CBP) algorithm. Our extensive experiments demonstrate that CBCNs have superiority over state-of-the-art binary networks, and obtain results that are more close to the full-precision backbone networks ResNets and WRNs, with a storage reduction of about 32 times. As a generic convolutional layer, CBConv can also be used on other deep models and various tasks for model compression, which is our future work.

References

[1] M. Courbariaux, Y. Bengio, and J. P. David. Binaryconnect: Training deep neural networks with binary weights during propagations. *In Advances in Neural Information Processing Systems*.

[2] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or -1. *arXiv preprint arXiv:1602.02830*, 2016.

[3] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *International Conference on Neural Information Processing Systems*, pages 1097–1105, 2012.

[5] N. Krizhevsky and Hinton. The cifar-10 dataset. *online: http://www. cs. toronto. edu/kriz/cifar. html*.

[6] L. Lai, N. Suda, and V. Chandra. Deep convolutional neural network inference with floating-point weights and fixed-point activations. *arXiv:1703.03073v1*.

[7] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[8] X. Lin, C. Zhao, and W. Pan. Towards accurate binary convolutional neural network. *arXiv:1711.11294*, 2017.

[9] M. D. McDonnell. Training wide residual networks for deployment using a single bit for each weight. In *International Conference on Learning Representations*, 2018.

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

[11] R. Rigamonti, A. Sironi, V. Lepetit, and P. Fua. Learning separable filters. In *Computer Vision and Pattern Recognition*, pages 2754–2761, 2013.

[12] 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.

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

[14] D. Soudry, I. Hubara, and R. Meir. Expectation backpropagation: parameter-free training of multilayer neural networks with continuous or discrete weights. In *International Conference on Neural Information Processing Systems*, pages 963–971, 2014.

[15] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[16] D. Wan, F. Shen, L. Liu, F. Zhu, J. Qin, L. Shao, and H. T. Shen. Tbn: Convolutional neural network with ternary inputs and binary weights. In *European Conference on Computer Vision*, pages 315–332, 2018.

[17] S. Wu, G. Li, F. Chen, and L. Shi. Training and inference with integers in deep neural networks. *International Conference on Learning Representations*, 2018.

[18] C. L. R. J. J. H. X. C. Xiaodi Wang, Baochang Zhang and J. Liu. Modulated convolutional networks. In *Computer Vision and Pattern Recognition*, 2018.

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

[20] W. L. X. Y. W. L. Zechun Liu, Baoyuan Wu and K.-T. Cheng. Bi-real net: Enhancing the performance of 1-bit cnns with improved representational capability and advanced training algorithm. In *European Conference on Computer Vision*, 2018.