RBCN: Rectified Binary Convolutional Networks for Enhancing the Performance of 1-bit DCNNs

Chunlei Liu[1] , **Wenrui Ding**[2] , **Xin Xia**[1] , **Yuan Hu**[1] , **Baochang Zhang**[3*] ,
Jianzhuang Liu[4] , **Bohan Zhuang**[5] and **Guodong Guo**[6,7]

[1] School of Electronic and Information Engineering, Beihang University
[2] Unmanned System Research Institute, Beihang University
[3] School of Automation Science and Electrical Engineering, Beihang University
[4] Huawei Noah's Ark Lab
[5] The University of Adelaide, Australia
[6] Institute of Deep Learning, Baidu Research
[7] National Engineering Laboratory for Deep Learning Technology and Application
{liuchunlei, ding, xinxia, zy1602hy, bczhang}@buaa.edu.cn, liu.jianzhuang@huawei.com,
bohan.zhuang@adelaide.edu.au, guoguodong01@baidu.com

Abstract

Binarized convolutional neural networks (BCNNs) are widely used to improve memory and computation efficiency of deep convolutional neural networks (DCNNs) for mobile and AI chips based applications. However, current BCNNs are not able to fully explore their corresponding full-precision models, causing a significant performance gap between them. In this paper, we propose rectified binary convolutional networks (RBCNs), towards optimized BCNNs, by combining full-precision kernels and feature maps to rectify the binarization process in a unified framework. In particular, we use a GAN to train the 1-bit binary network with the guidance of its corresponding full-precision model, which significantly improves the performance of BCNNs. The rectified convolutional layers are generic and flexible, and can be easily incorporated into existing DCNNs such as WideResNets and ResNets. Extensive experiments demonstrate the superior performance of the proposed RBCNs over state-of-the-art BCNNs. In particular, our method shows strong generalization on the object tracking task.

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. They usually involve millions of parameters and billions of FLOPs during computation. This is critical because models of vision applications may consume very large amounts of memory and computation, making them impractical for most embedded platforms.

Binary filters instead of using full-precision filter weights have been investigated in DCNNs to compress the deep models to handle the aforementioned problems. Many works attempt to quantize the weights of a network while keeping the activations (feature maps) to 32-bit floating points [Zhou *et al.*, 2017; Zhu *et al.*, 2016; Wang *et al.*, 2018]. Although this scheme leads to less performance decrease compared to its full-precision counterpart, it still needs a substantial amount of computational resource to handle the full-precision activations. Therefore, the so-called 1-bit DCNNs, which target the problem of training the networks with both 1-bit quantized weights and 1-bit activations, become more promising and significant in the field of DCNNs compression. As presented in [Rastegari *et al.*, 2016], by reconstructing the full-precision filters with a single scaling factor, XNOR provides an efficient implementation of convolutional operations. More recently, Bi-Real Net [Liu *et al.*, 2018] explores a new variant of residual structure to preserve the real activations before the sign function. And the researchers in [Hou *et al.*, 2016] propose a new value approximation method that considers the effect of binarization on the loss to further obtain binarized weights. PCNN [Gu *et al.*, 2019] learns a set of diverse quantized kernels by exploiting multiple projections with discrete back propagation.

The investigation into prior arts reveals that how to use the full-precision models is the key issue to obtain the optimized BCNNs. Most existing methods use the full-precision models as an initialization [Rastegari *et al.*, 2016] [Liu *et al.*, 2018], or for kernel approximation [Gu *et al.*, 2019] [Rastegari *et al.*, 2016]. Besides, knowledge distillation uses a teacher model (e.g., a full-precision model) to provide a guidance to quantize the network [Polino *et al.*, 2018; Zhuang *et al.*, 2018; Mishra and Marr, 2017]. While these methods generally use a regularization term to minimize the difference between the student's and teacher's posterior probabilities or intermediate feature representations, they fail to consider the full-precision
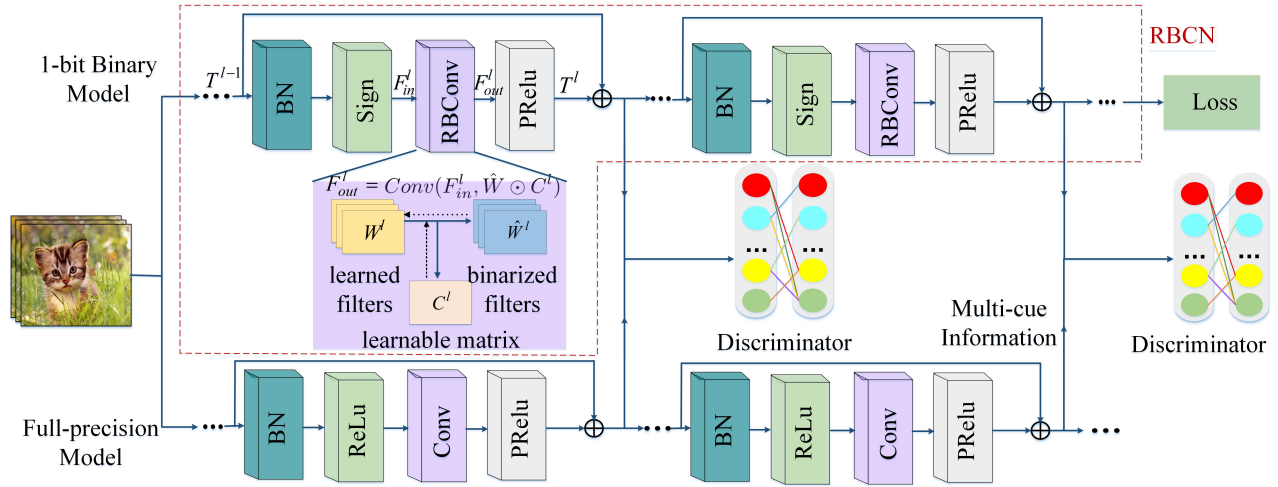
---

*Contact Author

Figure 1: The framework of Rectified Binary Convolutional Network (RBCN). The full-precision model provides the "real" feature maps, while the 1-bit model (as the generator) provides the "fake" feature maps, to the discriminators that try to distinguish the "real" from the "fake". Meanwhile, the generator tries to make the discriminators unable to work well. By repeating this process, both the full-precision feature maps and kernels (across all the convolutional layers) are sufficiently employed to enhance the capacity of the 1-bit binary model. Note that (1) the full-precision model is used only in learning but not in inference; (2) after training, the full-precision learned filters W are discarded, and only the binarized filters $\hat{W}$ and the shared learnable matrixs C (C^*) are kept in RBCN for the calculation of the feature maps in inference.

feature maps (activations) in a comprehensive way. This might be the reason why the knowledge distillation methods have not been employed to obtain the extreme 1-bit CNNs yet. To narrow down the performance gap between a BCNN and its full-precision model, we propose that the full-precision kernels and feature maps should be considered in a more comprehensive way, in order to fully exploit the multi-cue information.

In this paper, we introduce a rectified binary convolutional network (RBCN) to calculate an optimized BCNN in which a novel learning architecture is introduced to combine the full-precision feature maps and the kernels approximation in an end-to-end manner. Based on the powerful probability fitting ability of generative adversarial network (GAN), we discover that training a BCNN network with GAN, a better performance can be obtained by fitting the distribution of feature maps between full-precision and 1-bit binary networks. By doing so, GAN is introduced to distill RBCN from full-precision network by exploiting their full-precision feature maps. To the best of our knowledge, we are the first to use a GAN to do binary approximation of the full-precision model. The whole process is illustrated in Fig. 1, where the full-precision model and the 1-bit binary model (generator) respectively provide "real" and "fake" feature maps to the discriminators. The discriminators try to distinguish the "real" from the "fake", and the generator tries to make the discriminators unable to work well. By repeating this process, the multi-cue information (full-precision kernels and feature maps) is sufficiently employed in the training process to enhance the representational ability of the 1-bit binary model. Besides, kernel (filter) approximation (RBConv in Fig. 1) is integrated in the framework. Also, multiple discriminators are used to further improve the performance of RBCN. This

process involving the GAN and the kernel approximation is a rectified process, which can lead to a unique architecture with more precise estimation of the full-precision model. The contributions of this paper are summarized as follows.

(1) A novel BCNN learning architecture, referred to as rectified binary convolutional network (RBCN), is proposed, which employs the full-precision kernels and feature maps to rectify the binarization process in a comprehensive framework.

(2) To the best of our knowledge, we are the first to use a GAN to calculate a BCNN. Besides, we discover that using multiple discriminators in the GAN can significantly improve the performance of the 1-bit binary model.

(3) Extensive experiments demonstrate the superior performance of the proposed RBCNs over state-of-the-art BCNNs on the object classification and tracking tasks.

2 Rectified Binary Convolutional Networks (RBCNs)

We design RBCNs via kernel approximation and training with GANs to rectify BCNNs in a unified framework. During this process, the multi-cue information of the full-precision feature maps and kernels[1] is exploited to improve the performance degraded by binarization. The rectified convolutional layers are generic and flexible, which can be easily incorporated into existing CNNs, such as WideResNets and ResNets. First of all, Table 1 gives the main notation used in this paper.

---

[1] In this paper, the terms "filter" and "kernel" are exchangeable.

$\mathscr{L}$: loss function	$\hat{W}$: binarized filters	T : feature maps from RBCN to $D(\cdot)$
W : learned filters	C : learnable matrixs	R : feature maps from the full-precision model
$D(\cdot)$: discriminators	δ_C : gradient of C	
i : filter index	η : learning rate	F : feature maps before and after convolution in RBCN
t : t^{th} iteration	L : number of layers	Y : filters of the discriminators
l : layer index	δ_W : gradient of W	

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

2.1 Loss Function of RBCNs

The rectified process combines the full-precision kernels and feature maps to rectify the binarization process. It includes kernel approximation and adversarial learning. This learnable kernel approximation can lead to an unique architecture with more precise estimation of the original convolutional filters through minimizing a kernel loss. The discriminators $D(\cdot)$ with filters Y are introduced to distinguish the feature maps R of the full-precision model from those T of RBCN. The generator (RBCN) with filters W and learnable matrixs C is learned together with Y by using the knowledge from the supervised feature maps R. Therefore, W, C and Y are learned by solving the following optimization problem:

$$
\arg \min_{W,\hat{W},C} \max_{Y} \quad \mathscr{L} = \mathscr{L}_{Adv}(W,\hat{W},C,Y) \\
+ \mathscr{L}_{Kernel}(W,\hat{W},C) + \mathscr{L}_S(W,\hat{W},C), \tag{1}
$$

where $\mathscr{L}_{Adv}(W,\hat{W},C,Y)$ is the adversarial loss:

$$
\mathscr{L}_{Adv}(W,\hat{W},C,Y) = log(D(R;Y)) + log(1 - D(T;Y)), \tag{2}
$$

where $D(\cdot)$ consists of four basic blocks, each of which has a linear layer and a LeakyRelu layer.

In addition, $\mathscr{L}_{Kernel}(W,\hat{W},C)$ is the kernel loss between the learned full-precision filters W and the binarized filters $\hat{W}$, which is expressed by MSE:

$$
\mathscr{L}_{Kernel}(W,\hat{W},C) = \frac{1}{2}\lambda_1 ||W - C\hat{W}||^2. \tag{3}
$$

Finally, $\mathscr{L}_S(W,\hat{W},C)$ is a traditional problem-dependent loss such as the softmax loss.

For simplicity, the update of the discriminators is omitted in the following description until Algorithm 1. Besides, we find that the log in Equ. 2 has little effect during training and so it is omitted too. Then, based on the Lagrangian method, the optimization problem in Equ. 1 is rewritten as:

$$
\min \quad \mathscr{L}_S(W,\hat{W},C) \\
+ \lambda_1/2 \sum_l \sum_i ||W_i^l - C^l\hat{W}_i^l||^2 \\
+ \lambda_2/2 \sum_l \sum_i ||1 - D(T_i^l;Y)||^2. \tag{4}
$$

In Equ. 4, the target is to obtain W, $\hat{W}$ and C with Y fixed, which is why the term $D(R;Y)$ in Equ. 2 can be ignored. The update of Y can be found in Algorithm 1. The advantage of our formulation in Equ. 4 lies in that the loss function is trainable, meaning that it can be easily incorporated into existing learning frameworks.

2.2 Forward Propagation in RBCNs

In RBCNs, a binary filter $\hat{W}_i^l$ is calculated as:

$$
\hat{W}_i^l = \text{sign}(W_i^l), \tag{5}
$$

where W_i^l is the corresponding full-precision filter, and the values of $\hat{W}_i^l$ are 1 or -1. Both W_i^l and $\hat{W}_i^l$ are jointly obtained in the end-to-end learning.

In RBCNs, the convolution is implemented based on C^l and F_{in}^l to calculate the feature maps F_{out}^l:

$$
F_{out}^l = RBConv(F_{in}^l; \hat{W}^l, C^l) \\
= Conv(F_{in}^l, \hat{W} \odot C^l), \tag{6}
$$

where $RBConv$ denotes the convolution operation implemented as a new module, F_{in}^l and F_{out}^l are the feature maps before and after the convolution, respectively, and $\odot$ is the element-by-element product. Note that F_{in}^l is binary after the sign operation (see Fig. 1), and C is actually C^*, which will be elaborated at the end of section 3.3.

2.3 Backward Propagation in RBCNs

In RBCNs, what need to be learned and updated are the full-precision filters W and the learnable matrixs C. These two sets of parameters are jointly learned. In each convolutional layer, an RBCN updates W first and then C.

First we updates the full-precision filters W. Let $\delta_{W_i^l}$ be the gradient of the full-precision filter W_i^l. During backpropagation, the gradients pass to $\hat{W}_i^l$ first and then to W_i^l. Thus:

$$
\delta_{W_i^l} = \frac{\partial \mathscr{L}}{\partial W_i^l} = \frac{\partial \mathscr{L}}{\partial \hat{W}_i^l} \frac{\partial \hat{W}_i^l}{\partial W_i^l}, \tag{7}
$$

where

$$
\frac{\partial \hat{W}_i^l}{\partial W_i^l} = \begin{cases} 2 + 2W_i^l, & -1 \le W_i^l < 0, \\ 2 - 2W_i^l, & 0 \le W_i^l < 1, \\ 0, & \text{otherwise}, \end{cases} \tag{8}
$$

which is an approximation of the 2×dirac-delta function [Liu et al., 2018]. Furthermore,

$$
\frac{\partial \mathscr{L}}{\partial \hat{W}_i^l} = \frac{\partial \mathscr{L}_S}{\partial \hat{W}_i^l} + \frac{\partial \mathscr{L}_{Kernel}}{\partial \hat{W}_i^l} + \frac{\partial \mathscr{L}_{Adv}}{\partial \hat{W}_i^l}, \tag{9}
$$

and

$$
W_i^l \leftarrow W_i^l - \eta_1 \delta_{W_i^l}, \tag{10}
$$

where η_1 is a learning rate. Then:

$$
\frac{\partial \mathscr{L}_{Kernel}}{\partial \hat{W}_i^l} = -\lambda_1(W_i^l - C^l\hat{W}_i^l)C^l, \tag{11}
$$

$$\frac{\partial \mathcal{L}_{Adv}}{\partial \hat{W}_i^l} = -\lambda_2 (1 - D(T_i^l; Y)) \frac{\partial D}{\partial \hat{W}^l}. \tag{12}$$

We further update the learnable matrix C^l with W^l fixed. Let δ_{C^l} be the gradient of C^l. Then we have:

$$\delta_{C^l} = \frac{\partial \mathcal{L}_S}{\partial C^l} + \frac{\partial \mathcal{L}_M}{\partial C^l} + \frac{\partial \mathcal{L}_{Adv}}{\partial C^l}, \tag{13}$$

and

$$C^l \leftarrow C^l - \eta_2 \delta_{C^l}, \tag{14}$$

where η_2 is another learning rate. Further,

$$\frac{\partial \mathcal{L}_{Kernel}}{\partial C^l} = -\lambda_1 \sum_i (W_i^l - C^l \hat{W}_i^l) \hat{W}_i^l, \tag{15}$$

$$\frac{\partial \mathcal{L}_{Adv}}{\partial C^l} = -\lambda_2 \sum_i (1 - D(T_i^l; Y)) \frac{\partial D}{\partial C^l}. \tag{16}$$

The above derivations show that the rectified process is trainable in an end-to-end manner. The complete training process is summarized in Algorithm 1, including the update of the discriminators. Besides, in the implementation, the batch normalization (BN) layers are updated with W and C fixed after each epoch.

We note that in our implementation, the value of C will be replaced by its average during the forward process, resulting into a new matrix denoted by C^{*2}. By doing so, only a scalar instead of a matrix involve into the convolution which thus speed up the calculation.

3 Experiments

Our RBCNs are evaluated first on object classification using MNIST [Lecun *et al.*, 1998], CIFAR10/100 [Krizhevsky and Hinton, 2009] and ILSVRC12 ImageNet datasets [Russakovsky *et al.*, 2015], and then on object tracking. For object classification, WideResNet (WRN) [Zagoruyko and Komodakis, 2016] and ResNet [He *et al.*, 2016] are employed as the backbone networks to build our RBCNs. Also, binarizing the neuron activations is carried out in all of our experiments.

3.1 Datasets and Implementation Details

MINIST. The MINIST [Lecun *et al.*, 1998] 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.

CIFAR. CIFAR10 [Krizhevsky and Hinton, 2009] 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.

ImageNet. ImageNet object classification dataset [Russakovsky *et al.*, 2015] 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 RBCNs.

---

[2]its elements are equal

Algorithm 1 RBCN Training

Input: The training dataset, the feature maps R from the full-precision model, and the hyper-parameters such as initial learning rate, weight decay, convolution stride and padding size.

Output: A binary 1-bit model RBCN with weights $\hat{W}$ and learnable matrixs C.

1: Initialize W randomly;
2: **repeat**
3:     Randomly sample a mini-batch data;
4:     // Forward propagation
5:     **for all** $l = 1$ to L convolutional layer **do**
6:         $F_{out}^l = Conv(F_{in}^l, \hat{W}^l \odot C^l)$;
7:     **end for**
8:     // Back propagation
9:     **for all** $l = L$ to 1 **do**
10:         Update the discriminators $D^l(\cdot)$ by ascending their stochastic gradients:
11:         $\nabla_{D^l}(log(D^l(R^l; Y)) + log(1 - D^l(T^l; Y)))$;
12:         Calculate the gradients δ_{W^l}; // Using Eq. 7
13:         $W^l \leftarrow W^l - \eta \delta_{W^l}$; // Update the weights
14:         Calculate the gradient δ_{C^l}; // Using Eq. 13
15:         $C^l \leftarrow C^l - \eta_2 \delta_{C^l}$; // Update the learnable matrixs
16:     **end for**
17:     Update all the parameters of the batch normalization layers
18: **until** the maximum epoch
19: $\hat{W} = sign(W)$.

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 [Zagoruyko and Komodakis, 2016]. λ_1 and λ_2 are set as 0.01 with a degradation of 10% for every 60 epochs before reaching the maximum epoch of 200 for CIFAR10/100. For example, WRN22 is a network with 22 convolutional layers and similarly for WRN40.

ResNet18 Backbone. SGD is used as the optimization algorithm with a momentum of 0.9 and a weight decay 1e-4. λ_1 and λ_2 are set as 0.1 with a degradation of 10% for every 20 epochs before reaching the maximum epoch of 70 on ImageNet, while on CIFAR10/100, λ_1 and λ_2 are set as 0.01 with a degradation of 10% for every 60 epochs before reaching the maximum epoch of 200.

3.2 Ablation Study

In this section, we study the performance contributions of the components in RBCNs, which include kernel approximation, GAN, and the update of the BN layers. CIFAR100 and ResNet18 with different kernel stages are used in this experiment. The details are given below.

    1) We only replace the convolution in Bi-Real Net with our

	Kernel Stage	Bi	R	R+G	R+G+B
RBCN	32-32-64-128	54.92	56.54	59.13	**61.64**
RBCN	32-64-128-256	63.11	63.49	64.93	**65.38**
RBCN	64-64-128-256	63.81	64.13	65.02	**66.27**

Table 2: Performance (accuracy, %) contributions of the components in RBCNs on CIFAR100, where Bi, R, G, and B denote the Bi-Real Net, $RBConv$, GAN, update of the BN layers, respectively. The bold numbers represent the best results.

Model	Kernel Stage	Dataset	
		CIFAR-10	CIFAR-100
ResNet18	32-32-64-128	92.67	67.07
ResNet18	32-64-128-256	93.88	72.51
ResNet18	64-64-128-256	94.57	72.89
RBCN (ResNet18)	32-32-64-128	89.03	**61.09**
RBCN (ResNet18)	32-64-128-256	90.67	**65.38**
RBCN (ResNet18)	64-64-128-256	90.40	**66.27**
WRN22	64-64-128-256	95.19	76.38
WRN40	64-64-128-256	94.92	74.91
RBCN (WRN22)	64-64-128-256	93.28	**72.06**
RBCN (WRN40)	64-64-128-256	93.69	**73.08**
XNOR (ResNet18)	32-32-64-128	71.01	43.58
XNOR (WRN22)	64-64-128-256	86.90	58.05
Bi-Real (ResNet18)	32-32-64-128	85.34	54.92
Bi-Real (WRN22)	64-64-128-256	90.65	68.51
PCNN (ResNet18)	32-32-64-128	85.50	55.66
PCNN (WRN22)	64-64-128-256	91.62	70.32
Scheme-A (ResNet18)	32-64-128-256	75.45	46.32
Scheme-A (WRN22)	64-64-128-256	87.83	59.54

Table 3: Classification accuracy (%) based on ResNet18 and WRN40 on CIFAR10/100. The bold represent the best results among the binary networks.

kernel approximation ($RBConv$) and compare the results. As shown in the R column in Table 2, RBCN achieves 1.62% accuracy improvement over Bi-Real Net (56.54% vs. 54.92%) using the same network structure as in ResNet18 with 32-32-64-128. This significant improvement verifies the effectiveness of the learnable matrixs.

2) In RBCNs, if we use the GAN to help binarization, we can find a more significant improvement from 56.54% to 59.13% with the kernel stage of 32-32-64-128, which shows that the GAN can really enhance the binarized networks.

3) We find that a training trick can also improve RBCNs, which is to update the BN layers with W and C fixed after each epoch (line 17 in Algorithm 1). This trick makes RBCN boost 2.51% (61.64% vs. 59.13%) in CIFAR100 with 32-32-64-128.

3.3 Accuracy Comparison with State-of-the-Art

CIFAR10/100. The same parameter settings are used in RBCNs on both CIFAR10 and CIFAR100. We first compare our RBCNs with the original ResNet18 with different stage kernels, followed by a comparison with the original WRNs with the initial channel dimension 64 in Table 3. Thanks

to the rectified process, our results on both the datasets are close to the full-precision networks ResNe18 and WRN40. Then, we compare our results with other state-of-the-arts such as Bi-Real Net [Liu *et al.*, 2018], PCNN [Gu *et al.*, 2019], Scheme-A [Mishra and Marr, 2017] and XNOR [Rastegari *et al.*, 2016]. All these BCNNs have both binary filters and binary activations. It is observed that at most 6.17% (= 61.09%−54.92%) accuracy improvement is gained with our RBCN, and in other cases, larger margins are achieved.

ImageNet. Five state-of-the-art methods on ImageNet are chosen for comparison: Bi-Real Net [Liu *et al.*, 2018], BinaryNet [Courbariaux *et al.*, 2016], XNOR [Rastegari *et al.*, 2016], PCNN [Gu *et al.*, 2019] and ABC-Net [Lin *et al.*, 2017]. Again, these networks are representative methods of binarizing both network weights and activations and achieve state-of-the-art results. All the methods in Table 4 perform the binarization of ResNet18. The results in Table 4 are quoted directly from their papers, except that the result of BinaryNet is from [Lin *et al.*, 2017]. The comparison clearly indicates that the proposed RBCN outperforms the five binary networks by a considerable margin in terms of both the top-1 and top-5 accuracies. Specifically, for top-1 accuracy, RBCN outperforms BinaryNet and ABC-Net with a gap over 16%, achieves 7.9% improvement over XNOR, 3.1% over the very recent Bi-Real Net, and 2.2% over the latest PCNN. In Fig. 2, we plot the training and testing loss curves of XNOR and RBCN. It clearly shows that using our rectified process, RBCN converges faster than XNOR.

3.4 Experiments on Object Tracking

The key message conveyed in the proposed method is that although the conventional binary training method has a limited model capability, the proposed rectified process can lead to a powerful model. In this section, we show that this framework can also be used in object tracking. In particular, we consider the problem of tracking an arbitrary object in videos, where the object is identified solely by a rectangle in the first frame. For object tracking, it is necessary to update the weights of the network online, severely compromising the speed of the system. To directly apply the proposed framework to this application, we can construct a binary convolution with the same structure to reduce the convolution time. In this way, RBCN can be used to binarize the network further to guarantee the tracking performance.

In this paper, we use SiamFC Network as the backbone for object tracking. We binarize SiamFC as Rectified Binary Convolutional SiamFC Network (RB-SF). We evaluate RB-SF on four datasets, GOT-10K [Huang *et al.*, 2018], OTB50 [Wu *et al.*, 2013], OTB100 [Wu *et al.*, 2015], and UAV123 [Mueller *et al.*, 2016], using accuracy occupy (AO) and success rate (SR). The results are shown in Table 5. Intriguingly, our framework achieves about 7% AO improvement over XNOR, both using the same network architecture as in SiamFC Network on GOT-10k. Further, our framework brings so much benefit that Bi-SF performs almost as well as the full-precision SiamFC Network.

		Full-Precision	XNOR	ABC-Net	BinaryNet	Bi-Real	PCNN	RBCN
ResNet18	Top-1	69.3	51.2	42.7	42.2	56.4	57.3	**59.5**
	Top-5	89.2	73.2	67.6	67.1	79.5	79.8	**81.6**

Table 4: Classification accuracy (%) on ImageNet. The bold represents the best result among the binary networks.

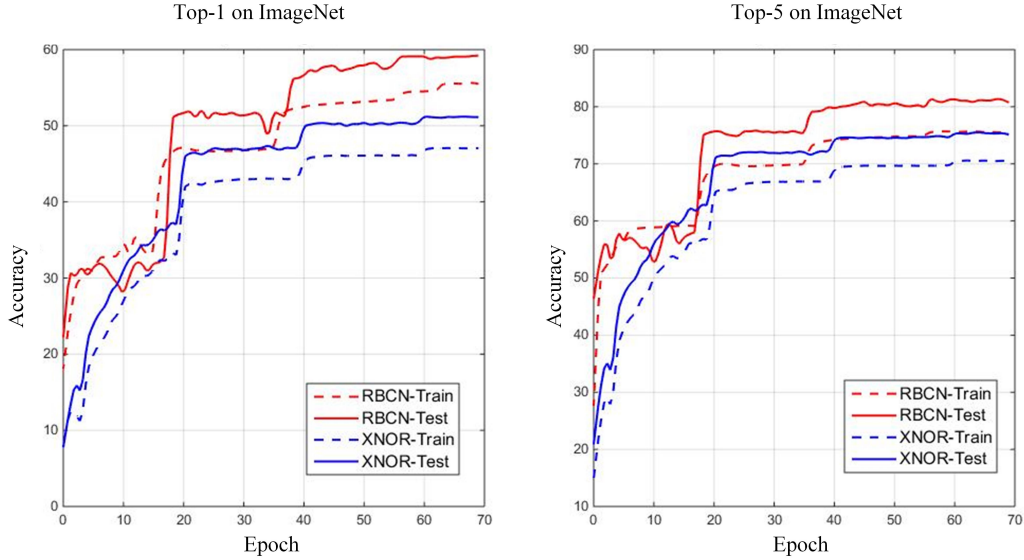

Figure 2: Training and Testing error curves of RBCN and XNOR based on the ResNet18 backbone on ImageNet.

Dataset	Index	SiamFC	XNOR	RB-SF
GOT-10K	AO	0.348	0.251	0.327
	SR	0.383	0.230	0.343
OTB50	Precision	0.761	0.457	0.706
	SR	0.556	0.323	0.496
OTB100	Precision	0.808	0.541	0.786
	SR	0.602	0.394	0.572
UAV123	Precision	0.745	0.547	0.688
	SR	0.528	0.374	0.497

Table 5: Tracking performance comparison between XNOR and RB-SF on different datasets.

	RBCN	XNOR-Net	ResNet18
Memory usage	33.7Mbits	33.7Mbits	374.1Mbits
Memory saving	11.10×	11.10×	-
FLOPs	1.67×10^8	1.67×10^8	1.81×10^9
Speedup	10.86×	10.86×	-

Table 6: Comparison of memory usage and FLOPs calculation.

4 Conclusion

In this paper, we introduce rectified binary convolutional networks (RBCNs), towards optimized BCNNs, by exploiting the full-precision kernels and feature maps in an end-to-end manner. In particular, we use a GAN to train the 1-bit binary network with the guidance of its corresponding full-precision model, which significantly improves the performance of the BCNN. Furthermore, as a general model, RBCNs can be used not only in object classification but also in other tasks such as object tracking. The experiments on both object classification and object tracking demonstrate the superior performance of the proposed RBCNs over state-of-the-art binary models.

3.5 Efficiency Analysis

The memory usage is computed as the summation of 32 bits times the number of real-valued parameters and 1 bit times the number of binary parameters in the network. Further, we use FLOPs to measure the speed. The results are given in Table 6. The FLOPs are calculated as the amount of real-valued floating point multiplications plus 1/64 of the amount of 1-bit multiplications [Liu *et al.*, 2018]. As shown in Table 6, the proposed RBCN, along with XNOR, reduces the memory usage of the full-precision ResNet18 by 11.10 times. For efficiency, both RBCN and XNOR gain 10.86× speedup over ResNet18. Note the computational and storage costs brought by learnable scalar C^* can be negligible.

author.

References

[Courbariaux *et al.*, 2016] Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or -1. *arXiv preprint arXiv:1602.02830*, 2016.

[Gu *et al.*, 2019] Jiaxin Gu, Baochang Zhang, and Jianzhuang Liu. Projection convolutional neural networks. In *AAAI*, 2019.

[He *et al.*, 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[Hou *et al.*, 2016] Lu Hou, Quanming Yao, and James T Kwok. Loss-aware binarization of deep networks. *arXiv preprint arXiv:1611.01600*, 2016.

[Huang *et al.*, 2018] Lianghua Huang, Xin Zhao, and Kaiqi Huang. Got-10k: A large high-diversity benchmark for generic object tracking in the wild. *arXiv preprint arXiv:1810.11981*, 2018.

[Krizhevsky and Hinton, 2009] Nair Krizhevsky and Hinton. The cifar-10 dataset. *online: http://www. cs. toronto. edu/kriz/cifar. html*, 2009.

[Lecun *et al.*, 1998] Yann Lecun, Leon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[Lin *et al.*, 2017] Xiaofan Lin, Cong Zhao, and Wei Pan. Towards accurate binary convolutional neural network. In *Advances in Neural Information Processing Systems*, pages 345–353, 2017.

[Liu *et al.*, 2018] Zechun Liu, Baoyuan Wu, Wenhan Luo, Xin Yang, Wei Liu, and Kwang-Ting Cheng. Bi-real net: Enhancing the performance of 1-bit cnns with improved representational capability and advanced training algorithm. In *Proceedings of the European Conference on Computer Vision*, pages 722–737, 2018.

[Mishra and Marr, 2017] Asit Mishra and Debbie Marr. Apprentice: Using knowledge distillation techniques to improve low-precision network accuracy. *arXiv:1711.05852v1*, 2017.

[Mueller *et al.*, 2016] Matthias Mueller, Neil Smith, and Bernard Ghanem. A benchmark and simulator for uav tracking. In *European conference on computer vision*, pages 445–461. Springer, 2016.

[Polino *et al.*, 2018] Antonio Polino, Razvan Pascanu, and Alistarh Dan. Model compression via distillation and quantization. *arXiv:1802.05668v1*, 2018.

[Rastegari *et al.*, 2016] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European Conference on Computer Vision*, pages 525–542, 2016.

[Russakovsky *et al.*, 2015] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, and Michael Bernstein. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[Wang *et al.*, 2018] Xiaodi Wang, Baochang Zhang, Ce Li, Rongrong Ji, Jungong Han, Xianbin Cao, and Jianzhuang Liu. Modulated convolutional networks. In *The IEEE Conference on Computer Vision and Pattern Recognition*, June 2018.

[Wu *et al.*, 2013] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Online object tracking: A benchmark. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2411–2418, 2013.

[Wu *et al.*, 2015] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Object tracking benchmark. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1834–1848, 2015.

[Zagoruyko and Komodakis, 2016] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

[Zhou *et al.*, 2017] Aojun Zhou, Anbang Yao, Yiwen Guo, Lin Xu, and Yurong Chen. Incremental network quantization: Towards lossless cnns with low-precision weights. *arXiv preprint arXiv:1702.03044*, 2017.

[Zhu *et al.*, 2016] Chenzhuo Zhu, Song Han, Huizi Mao, and William J Dally. Trained ternary quantization. *arXiv preprint arXiv:1612.01064*, 2016.

[Zhuang *et al.*, 2018] Bohan Zhuang, Chunhua Shen, Mingkui Tan, Lingqiao Liu, and Ian Reid. Towards effective low-bitwidth convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7920–7928, 2018.