International Journal of Computer Vision (2021) 129:998–1012
https://doi.org/10.1007/s11263-020-01417-9

Rectified Binary Convolutional Networks with Generative Adversarial Learning

Chunlei Liu[1] · Wenrui Ding[1] · Yuan Hu[1] · Baochang Zhang[1] · Jianzhuang Liu[2] · Guodong Guo[3] · David Doermann[4]

Received: 19 December 2019 / Accepted: 4 December 2020 / Published online: 6 January 2021

Abstract
Binarized convolutional neural networks (BNNs) are widely used to improve the memory and computational efficiency of deep convolutional neural networks for to be employed on embedded devices. However, existing BNNs fail to explore their corresponding full-precision models' potential, resulting in a significant performance gap. This paper introduces a Rectified Binary Convolutional Network (RBCN) by combining full precision kernels and feature maps to rectify the binarization process in a generative adversarial network (GAN) framework. We further prune our RBCNs using the GAN framework to increase the model efficiency and promote flexibly in practical applications. Extensive experiments validate the superior performance of the proposed RBCN over state-of-the-art BNNs on tasks such as object classification, object tracking, face recognition, and person re-identification.

Keywords Binary convolutional neural network (BNN) · Rectified binary convolutional network (RBCN) · Generative adversarial network (GAN)

Communicated by Mei Chen.

Chunlei Liu and Wenrui Ding have contributed equally to this work.

✉ Baochang Zhang
bczhang@buaa.edu.cn

Chunlei Liu
liuchunlei@buaa.edu.cn

Wenrui Ding
ding@buaa.edu.cn

Yuan Hu
huyuan1248@gmail.com

Jianzhuang Liu
jz.liu@siat.ac.cn

Guodong Guo
guoguodong01@baidu.com

David Doermann
doermann@buffalo.edu

1 Beihang University, Beijing, China

2 Shenzhen Institutes of Advanced Technology, University of Chinese Academy of Sciences, Shenzhen, China

3 Baidu Research and National Engineering Laboratory for Deep Learning Technology and Application, Institute of Deep Learning, Beijing, China

4 University at Buffalo, Buffalo, USA

1 Introduction

Deep convolutional neural networks (CNNs) have been successfully demonstrated on many computer vision tasks. However, CNNs still face many challenges when applied to practical applications. CNNs typically involve millions of parameters and billions of FLOPs of computation. This is critical because vision application models consume large amounts of memory and computation, making them impractical for object recognition outside of high-performance computing environments. Existing work focuses on network pruning (He et al. 2017; Li et al. 2019), low-bit quantization (Zhang et al. 2018; Zhou et al. 2016, 2017), decomposition/factorization (Jaderberg et al. 2014; Denton et al. 2014), distillation (Xu et al. 2018; Changyong et al. 2019), or efficient architecture design (Sandler et al. 2018; Zhang et al. 2018) to address the problem of inefficient models. In particular, quantization approaches represent network weights and activations with low bit-width fixed-point integers, enabling computation by efficient bit-wise operations. Binarization (Rastegari et al. 2016; Zhang et al. 2019; Liu et al. 2018) is an extreme quantization approach where both weights and activations are $+1$ or -1, represented by a single bit. This paper designs highly compact binary neural networks (BNNs) from both quantization and network pruning perspectives.

Despite the progress made in 1-bit quantization and network pruning, few works have combined the two in a unified framework to reinforce each other. It is clearly necessary to introduce pruning techniques into 1-bit CNNs since not all filters and kernels are equally important or worth quantizing in the same way. One potential solution is to prune the network and then conduct 1-bit quantization over the remaining weights to produce a more compressed network. However, this solution fails to consider the difference between the binarized and full precision parameters during pruning. Therefore, a promising alternative is to prune the quantized network. However, designing a unified framework to combine quantization and pruning is still an open question.

To address these issues, we introduce a rectified binary convolutional network (RBCN) to train a BNN, in which a novel learning architecture is presented in a generative adversarial network (GAN) framework. Our motivation is based on the fact that GANs have the powerful ability to match two data distributions (the full precision and 1-bit networks). This can also be viewed as distilling/exploiting the full precision model for the benefit of its 1-bit counterpart. For training RBCN, the main process for binarization is illustrated in Fig. 1, where the full-precision model and the 1-bit 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. The result is a rectified process and a unique architecture with a more precise estimation of the full precision model. Pruning is also explored to improve the 1-bit models applicability in practical applications in the GAN framework. The main contributions of this paper are summarized as follows:

- We introduce a novel BNN learning architecture, the Rectified Binary Convolutional Network (RBCN), to rectify the binarization process based on the full precision kernels and feature maps in a unified framework.
- After binarization, we perform network pruning to further compress the deep models in a GAN framework.
- We perform a comprehensive experiment on image classification, object tracking, face recognition, and person re-identification. Our models achieve state-of-the-art performance compared with other 1-bit CNNs on ImageNet, CIFAR, and LFW datasets.

This paper is built upon our previous conference paper (Liu et al. 2019) by significantly extending our work, including: (1) improving our method by extending pruned 1-bit CNNs using GANs, which remains largely unexplored in the existing literature, (2) conducting extensive experiments on the tasks of classification, tracking, face recognition, person re-identification, and pruning. All of these contributions strongly support the effectiveness of our models.

2 Related Work

In this paper, we exploit Generative Adversarial Networks (GANs) to facilitate binarization and pruning training process, using GANs' ability to match feature map distributions. In the next two subsections, we briefly review GANs and network compression methods.

2.1 Generative Adversarial Networks (GANs)

GANs (Goodfellow et al. 2014) are one of the most successful and widely used methods for training CNNs. The main idea of a GAN is to train two competitive networks, referred to as the generator $G(x, z)$ and discriminator $D(x)$, based on the game theory principles. The goal of GANs is to train the generator G to match a data distribution x by transforming a noise vector z. The discriminator D is trained to distinguish the 'fake' samples (z) generated by G from the 'real' samples (x). The GAN training process is formulated as follows

$$\arg\min_{G}\max_{D} V(D, G) = E[log(D(x))] \\ +E[1 - log(D(G(x, z)))], \quad (1)$$

where E is the expectations and V is the loss function.

Recently, GANs have been harnessed in various domains, including image generation, compression, and domain transfer. Since GANs were first introduced, researchers have studied them vigorously. As a result, researchers propose a lot of loss functions, regularization and normalization methods, and neural architectures choices (Goodfellow et al. 2014; Salimans et al. 2016; Tschannen et al. 2018; Gulrajani et al. 2017; Odena et al. 2017; Mao et al. 2017; Arjovsky et al. 2017). These all show the power of GANs and their ability to match two data distributions. This has inspired us to explore GANs to match the feature maps distribution between full precision and 1-bit models.

2.2 Network Compression

This section briefly reviews the primary approaches for deep network compression, dividing them into five categories: quantization, network pruning, decomposition/factorization, distillation, and compact network design.

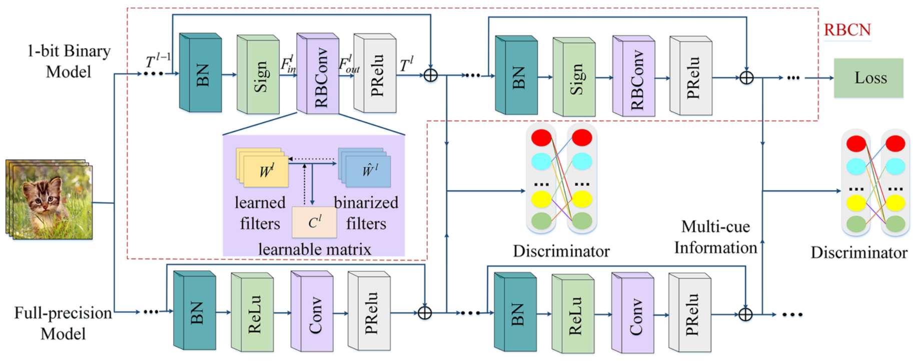

Fig. 1 This figure shows the framework integrating the rectified binary convolutional network (RBCN) with generative adversarial network (GAN) learning. The full-precision model provides the "real" feature maps, while the 1-bit model (as the generator) provides the "fake" feature maps, to the discriminators 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 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 matrices C are kept in RBCN for the calculation of the feature maps in inference

2.2.1 Quantization

Recently, the superior memory and computational efficiency of quantization networks stimulates the corresponding research. The quantization-based methods represent network weights and/or activations with low precision, typically from 32-bit to 1, 2, or 4 bits. This reduces the model storage and computation cost, thereby generating highly compact CNN models. Our paper aims for extreme binary quantization. Among the state-of-the-art binarization methods, XNOR-Net (Rastegari et al. 2016) provides an efficient binarization method by reconstructing the full precision filters with a single scaling factor. Bi-Real Net (Liu et al. 2018) introduces a new structure modification method to preserve the real activations before the sign function. Besides, by exploiting multiple projections with discrete backward propagation, projection convolutional neural networks (PCNNs) (Gu et al. 2019) learn a set of diverse quantized kernels to reconstruct a full-precision kernel. HORQ (Li et al. 2017) proposes a high-order binarization scheme by recursively performing residual quantization. Recently, ReActNet (Liu et al. 2020) proposes a baseline network by modifying and binarizing a compact real-valued network MobileNet with parameter-free shortcuts, bypassing all the intermediate convolutional layers to largely improve the representational capability of BNNs. Our analysis of the prior arts suggests that how full precision information is used is the key issue for optimizing BNNs.

2.2.2 Network Pruning

Neural network pruning focuses on removing the less-important network connections in an unstructured or structured manner. The paper in Hassibi and Stork (1993) proposes a saliency-like measurement to remove redundant weights, which is determined by the loss function's second-order derivative matrix. Moreover, in order to remove less-important weights with small avsolute values, Han et al. (2015) propose an iterative thresholding method. In addition, Guo et al. (2016) provide a connection splicing method in order to avoid incorrect weights pruning. Besides, through minimizing the reconstruction error, the paper in Yang et al. (2017) introduces an energy-aware pruning method to prune the less-important weights layer-by-layer.

By contrast, in the aspect of structured pruning, Li et al. (2016) propose to calculate the ℓ_1-norm of filters in a layer-wise manner to remove filters and their corresponding feature maps. Besides, the paper in Molchanov et al. (2016) proposes a Taylor expansion-based criterion to iteratively prune filters. Unlike the above multi-stage and layer-wise pruning methods, Huang and Wang (2018) propose to prune the network in an end-to-end manner by using an uncontrolled sparse soft mask. Furthremore, the method in Lin et al. (2019) effectively solves the pruning optimization problem using generative adversarial learning to learn a sparse soft mask in a label-free, end-to-end manner. Inspired by this, we perform pruning for 1-bit CNNs with a GAN framework.

2.2.3 Decomposition/Factorization

The main idea of decomposition/factorization is to construct a low-rank basis of filters to reduce the weights effectively. By exploiting the redundancy that exists between different feature channels and filters, Jaderberg et al. (2014) propose an agnostic approach for rank-1 filters in the spatial domain. Moreover, Rigamonti et al. (2013) preserve the performance of CNNs while drastically reducing their cost by learning and using separable filters that approximate the non-separable filters. Also, Lebedev et al. (2014) propose a simple two-step approach for speeding up convolutional layers based on tensor decomposition and discriminative fine-tuning. In addition, Yang et al. (2015) introduce a deep fried convnet to re-parameterize the matrix-vector multiplication of fully connected layers. Denton et al. (2014) present several techniques designed for object recognition tasks to speed up the evaluation of large convolutional networks.

2.2.4 Distillation

Distillation is a knowledge transfer method that provides guidance information through large teacher models to guide the training of small student models. Belagiannis and Farshad (2018) propose a knowledge transfer method based on adversarial learning, which teaches a smaller student network with limited capacity to mimic the larger teacher network to obtain better performance. Furthermore, inspired by the assumption that the small network can not perfectly mimic a large one due to the large network scale gap, Changyong et al. (2019) consider effective intermediate supervision under the adversarial training framework to learn the student network. Xu et al. (2018) propose to use conditional adversarial networks to learn the loss function to transfer knowledge from the teacher to the student. Gao et al. (2020) propose residual knowledge distillation (RKD) to remedy the performance degradation caused by the substantial gap between the learning capacities of the student and the teacher network. Specifically, RKD introduces an assistant network to aid the training process by learning the residual error between the student and teacher models. Ahn et al. (2019) propose an information-theoretic framework for knowledge transfer, where the student network effectively learns the target task by minimizing the cross-entropy loss while maintaining high mutual information with the teacher network. By learning the distribution of activation in the teacher network, the mutual information can be maximized.

Like the methods described above, we adopt the idea of distillation to transfer information from a teacher model to a student model. But different with those methods, we are the first to explore 1-bit network distillation using a GAN framework. In our approach, a full precision network is used as the teacher model, and the 1-bit network is the student model. Through the information transfer, the 1-bit student network mimics the output of the full precision teacher network to obtain better performance. Another difference lies in our method adopts multiple discriminators, which fit the distributions between a student network and a full precision network at different stages. This is similar to the method in Ahn et al. (2019), but our learnable discriminators avoid the manual tuning required by the handcrafted module in Ahn et al. (2019).

2.2.5 Efficient Network Design

There has been a rising interest in designing efficient architectures to meet the requirement of efficiency and accuracy. Recently, some approaches [e.g., GoogLeNet (Szegedy et al. 2015) and SqueezeNet (Iandola et al. 2016)] propose to replace 3×3 convolutional kernels with 1×1 kernels to reduce the computational complexity. Besides, Xception (Chollet 2017), MobileNet (Howard et al. 2017), and ShuffleNet (Zhang et al. 2018) employ depth-wise separable convolutions to build light weight deep neural networks. Meanwhile, Neural Architecture Search (NAS) has attracted a lot of attention due to its considerable performance gains while avoiding handcrafted design. Instead of adopting the conventional proxy-based framework, ProxylessNAS (Cai et al. 2018) introduces latency loss to search architectures for the target task. Also, EfficientNet (Tan and Le 2019) uniformly scales the dimensions of depth, width, and resolution to obtain more compact networks by employing a simple but highly effective compound coefficient. Furthermore, Binarized Neural Architecture Search (BNAS) (Chen et al. 2020) provides a more promising method for efficient network architectures according to the performance feedback.

3 Rectified Binary Convolutional Networks (RBCNs)

We design RBCNs with GANs to rectify BNNs in a unified framework. In our RBCNs, multi-cue information from full precision feature maps and kernels[1] are exploited to improve the performance of BNNs. The rectified convolutional layers are generic, flexible, and can be easily incorporated into existing CNNs such as WideResNets and ResNets. We furthermore prune RBCN with minimal performance degradation using GANs. To begin, Table 1 summarizes some of the notation used in this paper.

3.1 Optimized RBCN

This section describes the optimization process, including changing the loss function and updating the learnable parameters.

3.1.1 Loss Function

The rectification 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 leads to a unique architecture

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

Table 1 This figure contains a brief description of the variables and operators used in the paper

$\mathscr{L}$: loss function	$\hat{W}$: binarized filters	T : feature maps from RBCN to $D(\cdot)$
W : learned filters	C : learnable matrices	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/after convolution in RBCN
t : tth iteration	L : number of layers	Y : filters of the discriminators
l : layer index	δ_W : gradient of W	

with a precise estimation of the 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 RBCN generator with filters W and learnable matrices C is trained together with Y using the knowledge from the supervised feature maps R. In summary, W, C and Y are learned by solving the following optimization problem:

$$\arg \min_{W,\hat{W},C} \max_{Y} \mathscr{L} = \mathscr{L}_{Adv}(W, \hat{W}, C, Y) + \mathscr{L}_S(W, \hat{W}, C) + \mathscr{L}_{Kernel}(W, \hat{W}, C), \tag{2}$$

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

$$\mathscr{L}_{Adv}(W, \hat{W}, C, Y) = log(D(R; Y)) + log(1 - D(T; Y)), \tag{3}$$

where $D(\cdot)$ consists of a series of basic blocks, each containing a linear layer and a LeakyRelu layer. We also have multiple discriminators to rectify the training process of binarization.

In addition, $\mathscr{L}_{Kernel}(W, \hat{W}, C)$ denotes the kernel loss between the learned full precision filters W and the binarized filters $\hat{W}$ and is defined as:

$$\mathscr{L}_{Kernel}(W, \hat{W}, C) = \lambda_1/2||W - C\hat{W}||^2, \tag{4}$$

where λ_1 is a balance parameter. Finally, $\mathscr{L}_S$ is a traditional problem-dependent loss, such as softmax loss. The adversarial loss, kernel loss, and softmax loss are considered to be regularizations on $\mathscr{L}$.

For simplicity, the update of the discriminators is omitted in the following description until Algorithm 1. We also have omitted $log(\cdot)$ and rewrite the optimization in Eq. 2 as Eq. 5 for simplicity.

$$\min_{W,\hat{W},C} \mathscr{L}_S(W, \hat{W}, C) + \lambda_1/2 \sum_l \sum_i ||W_i^l - C^l \hat{W}_i^l||^2 + \sum_l \sum_i ||1 - D(T_i^l; Y)||^2. \tag{5}$$

where i represents the ith channel and l represents the lth layer. In Eq. 5, the target is to obtain W, $\hat{W}$ and C with Y fixed, which is why the term $D(R; Y)$ in Eq. 2 can be ignored. The update process for Y is found in Algorithm 1. The advantage of our formulation in Eq. 5 lies in that the loss function is trainable, meaning that it can be easily incorporated into existing learning frameworks.

Algorithm 1 RBCN Training

Input: The training dataset, the feature maps from the full precision model, and the hyper-parameters including initial learning rate, weight decay, convolution stride and padding size.
Output: A binary 1-bit model RBCN with weights $\hat{W}$ and learnable matrices C.

```
1: Initialize $W$ randomly;
2: repeat
3:   Randomly sample a mini-batch of 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:   // Backward propagation
9:   for all $l = L$ to 1 do
10:    Update the discriminators $D^l(\cdot)$ of GAN 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 $\delta_{W^l}$; // Using Eq. 7-12
13:    $W^l \leftarrow W^l - \eta_1 \delta_{W^l}$; // Update the weights
14:    Calculate the gradient $\delta_{C^l}$; // Using Eq. 13-16
15:    $C^l \leftarrow C^l - \eta_2 \delta_{C^l}$; // Update the learnable matrix
16:  end for
17:  Fix the parameters in the convolutional layers, and only update the parameters of other layers;
18: until the maximum epoch
19: $\hat{W} = \mathrm{sign}(W)$.
```

3.1.2 Learning RBCNs

In RBCNs, the convolution is implemented using W^l, C^l and F_{in}^l to calculate the output feature maps F_{out}^l as

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

where $RBConv$ denotes the convolution operation implemented as a new module, F^l_{in} and F^l_{out} are the feature maps before and after the convolution, respectively. W^l are full precision filters, the values of $\hat{W}^l$ are 1 or -1, and $\odot$ is the element-by-element product operation.

During the backward propagation process of RBCNs, what needs to be learned and updated are the full precision filters W and the learnable matrices C. These two sets of parameters are jointly learned. In each convolutional layer, we update W first and then C.

Update W Let $\delta_{W^l_i}$ be the gradient of the full precision filter W^l_i. During backpropagation, the gradients are passed to $\hat{W}^l_i$ first and then to W^l_i. Thus,

$$\delta_{W^l_i} = \frac{\partial \mathcal{L}}{\partial W^l_i} = \frac{\partial \mathcal{L}}{\partial \hat{W}^l_i} \frac{\partial \hat{W}^l_i}{\partial W^l_i}, \tag{7}$$

where

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

which is an approximation of the $2\times$ the Dirac delta function (Liu et al. 2018). Furthermore,

$$\frac{\partial \mathcal{L}}{\partial \hat{W}^l_i} = \frac{\partial \mathcal{L}_S}{\partial \hat{W}^l_i} + \frac{\partial \mathcal{L}_{Kernel}}{\partial \hat{W}^l_i} + \frac{\partial \mathcal{L}_{Adv}}{\partial \hat{W}^l_i}, \tag{9}$$

and

$$W^l_i \leftarrow W^l_i - \eta_1 \delta_{W^l_i}, \tag{10}$$

where η_1 is the learning rate. Then,

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

$$\frac{\partial \mathcal{L}_{Adv}}{\partial \hat{W}^l_i} = -2(1 - D(T^l_i; Y)) \frac{\partial D}{\partial \hat{W}^l_i}. \tag{12}$$

Update C 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

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

and

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

where η_2 is another learning rate. Furthermore,

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

$$\frac{\partial \mathcal{L}_{Adv}}{\partial C^l} = -\sum_i 2(1 - D(T^l_i; Y)) \frac{\partial D}{\partial C^l}. \tag{16}$$

These derivations show that the rectified process is trainable in an end-to-end manner. The complete training process is summarized in Algorithm 1, including how to update of the discriminators. As described in line 17 of Algorithm 1, we independently update other parameters while fixing convolutional layer's parameters to enhance the feature maps' variety in every layer. In this way, we speed up the training convergence and fully explore the potential of 1-bit networks. In our implementation, all the values of C^l are replaced by their average during the forward process. A scalar, not a matrix, is involved in the inference, thus speeding up the computation.

3.2 Network Pruning

We further prune the 1-bit CNNs to increase the model efficiency and further improve the flexibility of RBCNs in practical scenarios. This section considers the pruning process for optimization, including changing loss function and updating learnable parameters.

3.2.1 Loss Function

After binarizing the CNNs, we further prune the resulting 1-bit CNNs under the generative adversarial learning framework using the method described in Lin et al. (2019). We use a soft mask to remove the corresponding structures such as filters while obtaining close to the baseline accuracy. The discriminator $D_p(\cdot)$ with weights Y_p is introduced to distinguish baseline networks output R_p from those T_p of the pruned 1-bit network. The pruned network with weights W_p, $\hat{W}_p$, C_p and soft mask M_p, is learned together with Y_p using knowledge from the supervised features of baseline. W_p, $\hat{W}_p$, C_p, M_p and Y_p are therefore learned by solving the optimization problem as follows:

$$\begin{aligned} \arg \min_{W_p, \hat{W}_p, C_p, M_p} \max_{Y_p} \mathcal{L}_p = {} & \mathcal{L}_{Adv\_p}(W_p, \hat{W}_p, C_p, M_p, Y_p) \\ & + \mathcal{L}_{Kernel\_p}(W_p, \hat{W}_p, C_p) \\ & + \mathcal{L}_{S\_p}(W_p, \hat{W}_p, C_p) \\ & + \mathcal{L}_{Data\_p}(W_p, \hat{W}_p, C_p, M_p) + \mathcal{L}_{Reg\_p}(M_p, Y_p), \end{aligned} \tag{17}$$

where $\mathcal{L}_p$ is the loss function of pruning, the form of $\mathcal{L}_{Adv\_p}(W_p, \hat{W}_p, C_p, M_p, Y_p)$ and $\mathcal{L}_{Kernel\_p}(W_p, \hat{W}_p, C_p)$

are

$$\begin{aligned}&\mathscr{L}_{Adv\_p}(W_p, \hat{W}_p, C_p, M_p, Y_p)\\&\quad = log(D_p(R_p; Y_p)) + log(1 - D_p(T_p; Y_p)), \quad (18)\\&\mathscr{L}_{Kernel\_p}(W_p, \hat{W}_p, C_p)\\&\quad = \lambda_1/2||W_p - C_p\hat{W}_p||^2. \quad (19)\end{aligned}$$

$\mathscr{L}_{S\_p}$ is a traditional problem-dependent loss such as the softmax loss. $\mathscr{L}_{Data\_p}$ is the data loss between output features from both the baseline and the pruned network and used to align of these two networks' outputs. The data loss can then be expressed by MSE loss.

$$\mathscr{L}_{Data\_p}(W_p, \hat{W}_p, C_p, M_p) = \frac{1}{2n}\left\| R_p - T_p \right\|^2, \quad (20)$$

where n is the size of the mini-batch.

$\mathscr{L}_{Reg\_p}(M_p, Y_p)$ is a regularizer on W_p,$\hat{W}_p$,M_p and Y_p, which can be split into two parts as follows:

$$\mathscr{L}_{Reg\_p}(M_p, Y_p) = \mathscr{R}_\lambda(M_p) + \mathscr{R}(Y_p), \quad (21)$$

where $\mathscr{R}(Y_p) = log(D_p(T_p; Y_p))$, $\mathscr{R}_\lambda(M_p)$ is a sparsity regularizer form with parameter λ and $\mathscr{R}_\lambda(M_p) = \lambda||M_p||_{l_1}$.

As with the process in binarization, the update of the discriminators is omitted in the following description until Algorithm 2. We also have omitted $log(\cdot)$ for simplicity and rewrite the optimization of Eq. 17 as

$$\begin{aligned}\min_{W_p,\hat{W}_p,C_p,M_p} \quad & \lambda_1/2\sum_l\sum_i ||W^l_{p,i} - C^l_p\hat{W}^l_{p,i}||^2\\ & + \sum_l\sum_i ||1 - D(T^l_{p,i}; Y_p)||^2\\ & + \mathscr{L}_{S\_p}(W_p, \hat{W}_p, C_p) + \frac{1}{2n}\left\| R_p - T_p \right\|^2 + \lambda||M_p||_{l_1}. \quad (22)\end{aligned}$$

3.2.2 Learning Pruned RBCNs

In pruned RBCNs, the convolution is implemented as

$$\begin{aligned}F^l_{out,p} &= RBConv(F^l_{in,p}; \hat{W}^l_p \circ M^l_p, C^l_p)\\ &= Conv(F^l_{in,p}, (\hat{W}_p \circ M^l_p) \odot C^l_p), \quad (23)\end{aligned}$$

where $\circ$ is an operator that obtains the pruned weight with mask M_p. The other part of the forward propagation in the pruned RBCNs is the same with RBCNs.

In pruned RBCNs, what needs to be learned and updated are the full precision filters W_p, learnable matrices C_p, and soft mask M_p. In each convolutional layer, these three sets of parameters are jointly learned.

Update M_p M_p is updated by FISTA (Lin et al. 2018) with the initialization of $\alpha_{(1)} = 1$. Then we obtain

$$\alpha_{(k+1)} = \frac{1}{2}\left(1 + \sqrt{1 + 4\alpha^2_{(k)}}\right), \quad (24)$$

$$y_{(k+1)} = M_{p,(k)} + \frac{a_{(k)} - 1}{a_{(k+1)}}(M_{p,(k)} - M_{p,(k-1)}), \quad (25)$$

$$\begin{aligned}&M_{p,(k+1)} = prox_{\eta_{(k+1)}\lambda||\cdot||_1}\\&\left(y_{(k+1)} - \eta_{(k+1)}\frac{\partial(\mathscr{L}_{Adv\_p} + \mathscr{L}_{Data\_p})}{\partial(y_{(k+1)})}\right), \quad (26)\end{aligned}$$

where $\eta_{(k+1)}$ is the learning rate at the iteration $k + 1$ and $prox_{\eta_{(k+1)}\lambda||\cdot||_1}(z_i) = sign(z_i)\cdot(|z_i| - \eta_{(k+1)}\lambda)_+$, more detail can be found in Lin et al. (2019).

Update W_p Let $\delta_{W^l_{p,i}}$ be the gradient of the full precision filter $W^l_{p,i}$. During backpropagation, the gradients pass to $\hat{W}^l_{p,i}$ first and then to $W^l_{p,i}$. Furthermore,

$$\begin{aligned}\delta_{W^l_{p,i}} = \frac{\partial\mathscr{L}_p}{\partial\hat{W}^l_{p,i}} &= \frac{\partial\mathscr{L}_{S\_p}}{\partial\hat{W}^l_{p,i}} + \frac{\partial\mathscr{L}_{Adv\_p}}{\partial\hat{W}^l_{p,i}}\\ &+ \frac{\partial\mathscr{L}_{Kernel\_p}}{\partial\hat{W}^l_{p,i}} + \frac{\partial\mathscr{L}_{Data\_p}}{\partial\hat{W}^l_{p,i}}, \quad (27)\end{aligned}$$

and

$$W^l_{p,i} \leftarrow W^l_{p,i} - \eta_{p,1}\delta_{W^l_{p,i}}, \quad (28)$$

where $\eta_{p,1}$ is the learning rate, $\frac{\partial\mathscr{L}_{Kernel\_p}}{\partial\hat{W}^l_{p,i}}$ and $\frac{\partial\mathscr{L}_{Adv\_p}}{\partial\hat{W}^l_{p,i}}$ are

$$\frac{\partial\mathscr{L}_{Kernel\_p}}{\partial\hat{W}^l_{p,i}} = -\lambda_1(W^l_{p,i} - C^l_p\hat{W}^l_{p,i})C^l_p, \quad (29)$$

$$\frac{\partial\mathscr{L}_{Adv\_p}}{\partial\hat{W}^l_{p,i}} = -2(1 - D_p(T^l_{p,i}; Y_p))\frac{\partial D_p}{\partial\hat{W}^l_{p,i}}. \quad (30)$$

And

$$\frac{\partial\mathscr{L}_{Data\_p}}{\partial\hat{W}^l_{p,i}} = -\frac{1}{n}(R_p - T_p)\frac{\partial T_p}{\partial\hat{W}^l_{p,i}}, \quad (31)$$

Update C_p We further update the learnable matrix C^l_p with W^l_p and M^l_p fixed. Let $\delta_{C^l_p}$ be the gradient of C^l_p. Then we have

$$\begin{aligned}\delta_{C^l_p} = \frac{\partial\mathscr{L}_p}{\partial\hat{C}^l_p} &= \frac{\partial\mathscr{L}_{S\_p}}{\partial\hat{C}^l_p} + \frac{\partial\mathscr{L}_{Adv\_p}}{\partial\hat{C}^l_p}\\ &+ \frac{\partial\mathscr{L}_{Kernel\_p}}{\partial\hat{C}^l_p} + \frac{\partial\mathscr{L}_{Data\_p}}{\partial\hat{C}^l_p}, \quad (32)\end{aligned}$$

and

$$C^l_p \leftarrow C^l_p - \eta_{p,2}\delta_{C^l_p}. \tag{33}$$

and $\frac{\partial \mathscr{L}_{Kernel\_p}}{\partial C^l_p}$ and $\frac{\partial \mathscr{L}_{Adv\_p}}{\partial C^l_p}$ are

$$\frac{\partial \mathscr{L}_{Kernel\_p}}{\partial C^l_p} = -\lambda_1 \sum_i (W^l_{p,i} - C^l_p \hat{W}^l_{p,i})\hat{W}^l_{p,i}, \tag{34}$$

$$\frac{\partial \mathscr{L}_{Adv\_p}}{\partial C^l_p} = -\sum_i 2(1 - D_p(T^l_{p,i}; Y_p))\frac{\partial D_p}{\partial C^l_p}. \tag{35}$$

Further,

$$\frac{\partial \mathcal{L}_{Data\_p}}{\partial C^l_p} = \frac{1}{n}\sum_i (R_{p,i} - T_{p,i})\frac{\partial T_{p,i}}{\partial C^l_p}. \tag{36}$$

The complete training process is summarized in Algorithm 2, including the update of the discriminators.

Algorithm 2 Pruned RBCN

```
Input: The training dataset, the pre-trained 1-bit CNNs model, the feature maps $R_p$ from the pre-trained model, the pruning rate and the hyper-parameters including the initial learning rate, weight decay, convolution stride and padding size.
Output: The pruned RBCN with updated parameters $W_p$, $\hat{W}_p$, $M_p$ and $C_p$.
1: repeat
2:    Randomly sample a mini-batch;
3:    // Forward propagation
4:    Training a pruned architecture // Using Eq.17-22
5:    for all $l = 1$ to $L$ convolutional layer do
6:       $F^l_{out,p} = Conv(F^l_{in,p}, (\hat{W}^l_p \circ M_p) \odot C^l_p)$;
7:    end for
8:    // Backward propagation
9:    for all $l = L$ to 1 do
10:      Update the discriminators $D^l_p(\cdot)$ by ascending their stochastic gradients:
11:      $\nabla_{D^l_p}(log(D^l_p(R^l_p; Y_p)) + log(1 - D^l_p(T^l_p; Y_p)) + log(D^l_p(T_p; Y_p)))$;
12:      Update soft mask $M_p$ by FISTA // Using Eq. 24-26
13:      Calculate the gradients $\delta_{W^l_p}$; // Using Eq. 27-31
14:      $W^l_p \leftarrow W^l_p - \eta_{p,1}\delta_{W^l_p}$; // Update the weights
15:      Calculate the gradient $\delta_{C^l_p}$; // Using Eq. 32-36
16:      $C^l_p \leftarrow C^l_p - \eta_{p,2}\delta_{C^l_p}$; // Update the learnable matrix
17:   end for
18: until the maximum epoch
19: $\hat{W} = \mathrm{sign}(W)$.
```

4 Experiments

Our RBCNs are evaluated first on object classification using the CIFAR10/100 (Krizhevsky 2009) and ILSVRC12 ImageNet (Russakovsky et al. 2015) datasets. They are then evaluated on various tasks such as object tracking, face recognition (FR), and person re-identification (Re-id). For object classification, WideResNet (WRN) (Zagoruyko and Komodakis 2016) and ResNet (He et al. 2016) are employed as the backbone networks.

4.1 Datasets and Implementation Details

Datasets CIFAR10 (Krizhevsky 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. The dataset contains ten classes, including airplanes, dogs, birds, automobiles, cats, deers, frogs, ships, horses, and trucks, while CIFAR100 has 100 classes.

The ImageNet object classification dataset (Russakovsky et al. 2015) is more challenging due to its larger scale and greater diversity. This dataset contains 1000 classes, 1.2 million training images, and 50k validation images.

WRN Backbone WRN is a network structure similar to ResNet with a depth factor of k to control the feature map depth dimension expansion through three stages, within which the dimensions remain unchanged. For simplicity, we fix the depth factor to 1. Each WRN has a parameter i that indicates the channel dimension of the first stage that we set it to 16. η_1 and η_2 are both set to 0.01 with a degradation of 10% for every 60 epochs before reaching the maximum epoch of 200 for CIFAR10/100. WRN22 is a network with 22 convolutional layers, and WRN40 is a network with 40 convolutional layers.

ResNet18 Backbone SGD is used as the optimization algorithm with a momentum of 0.9. On ImageNet, η_1 and η_2 are both set to 0.1 with a degradation of 10% for every 20 epochs before reaching the maximum epoch of 70. On CIFAR10/100, η_1 and η_2 are both set to 0.01 with a degradation of 10% for every 60 epochs before reaching the maximum epoch of 200. For the hyper-parameters specific for pruning, more details can be found in Lin et al. (2019).

4.2 Ablation Study

This section studies the performance contributions of the kernel approximation, the GAN, and the update strategy (we fix the parameters of convolutional layers, and update other layers). CIFAR100 and ResNet18 with different kernel stages are used in these experiments.

(1) We replace the convolution in Bi-Real Net with our kernel approximation ($RBConv$) and compare the results. As shown in the column of "Bi" and "R" in Table 2, RBCN achieves 1.62% accuracy improvement over Bi-Real Net (56.54% vs. 54.92%) using the same network structure

Table 2 Performance (accuracy, %) contributions of the components in RBCNs on CIFAR100, where Bi = Bi-Real Net, R = $RBConv$, G = GAN, and B = update strategy

	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**

The bold numbers represent the best results

as in ResNet18. This significant improvement verifies the effectiveness of the learnable matrices.

(2) Using GAN makes RBCN improve 2.59% (59.13% vs. 56.54%) with the kernel stage of 32-32-64-128, which shows that the GAN can help mitigate the problem of being trapped in poor local minima.

(3) We further improve RBCNs by updating the BN layers with W and C fixed after each epoch (line 17 in Algorithm 1). This further boosts our accuracy by 2.51% (61.64% vs. 59.13%) in CIFAR100 with 32-32-64-128.

4.3 Accuracy Comparison with the 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 using different stage kernels, followed by a comparison with the original WRNs with the initial channel dimension 64 (Table 3). The rectified process leads to results on both the datasets that are close to the full precision networks ResNe18 and WRN40. We compare our results with other state-of-the-art approaches such as Bi-Real Net (Liu et al. 2018), PCNN (Gu et al. 2019), and XNOR-Net (Rastegari et al. 2016). All of these BNNs have both binary filters and binary activations. We observed that there is an accuracy improvement of up to 6.72% (= 61.64–54.92%). We also plot the training and testing loss curves of XNOR-Net and RBCN in Fig. 2 and show that RBCN converges faster than XNOR-Net.

ImageNet Five methods are chosen for comparison on ImageNet: Bi-Real Net (Liu et al. 2018), BinaryNet (Courbariaux et al. 2016), XNOR-Net (Rastegari et al. 2016), PCNN (Gu et al. 2019), and ABC-Net (Lin et al. 2017). These networks binarize both network weights and activations and achieve state-of-the-art results. All the methods in Table 4 perform the binarization of ResNet18. We quoted the results directly from the corresponding papers, except BinaryNet from Lin et al. (2017). The comparison clearly indicates that the proposed RBCN outperforms the five binary networks by a considerable margin in both the Top-1 and Top-5 accuracies. For Top-1 accuracy, RBCN outperforms BinaryNet and ABC-Net by over 16%, achieves 8.3% improvement over XNOR-Net, 3.1% over Bi-Real Net, and 2.2% over PCNN. In Fig. 3, we plot the training and testing loss curves of XNOR-Net and RBCN. It clearly shows that using our rectified process, RBCN converges faster than XNOR-Net.

Table 3 Classification accuracy (%) based on ResNet18 and WRN40 on CIFAR10/100

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	88.68	**61.64**
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-Net (ResNet18)	32-32-64-128	71.01	43.58
XNOR-Net (WRN22)	64-64-128-256	86.90	58.05
Bi-Real Net (ResNet18)	32-32-64-128	85.34	54.92
Bi-Real Net (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

Bold values represent the best results among the binary networks

4.4 Experiments on Different Tasks

In this section, we explore the effect of RBCNs for different tasks, including object tracking, person re-identification (reID), and face recognition (FR). These tasks can help to establish binary networks in other fields.

4.4.1 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. 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 only by a rectangle in the first frame. For object tracking, it is necessary to update the weights of the network online, severely compromising the systems speed. To directly apply the framework to tracking, we 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 and guarantee the tracking performance.

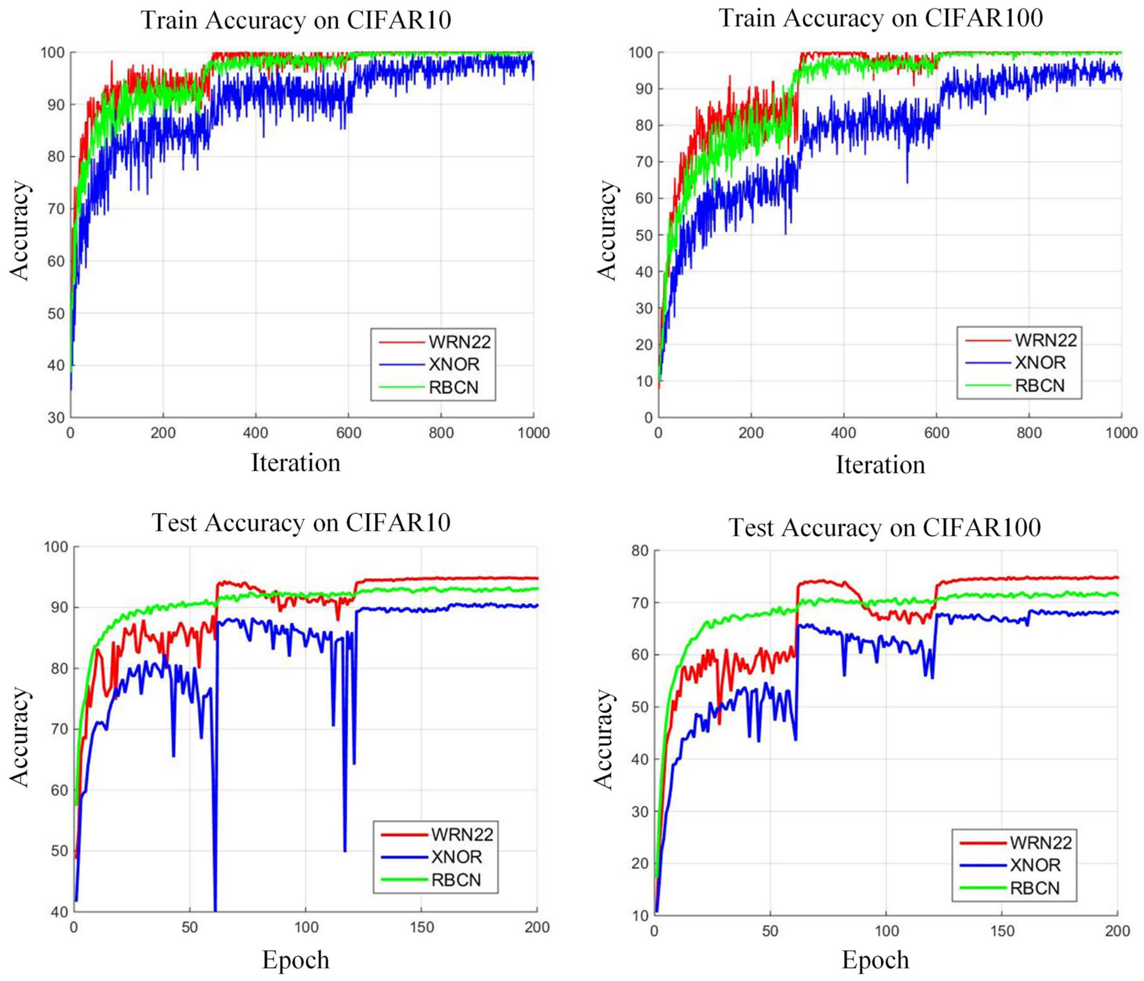

Fig. 2 Training and testing error curves of RBCN and XNOR-Net based on WRN40 for the CIFAR10/100 experiments

Table 4 Classification accuracy (%) on ImageNet

	FP	XNOR-Net	ABC-Net	BinaryNet	Bi-Real Net	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**

The bold represents the best result among the binary networks

For these experiments, we use the SiamFC Network as the backbone as a Rectified Binary Convolutional SiamFC Network (RB-SF). We evaluate RB-SF on four datasets, UAV123 (Mueller et al. 2016), GOT-10K (Huang et al. 2018), OTB50 (Wu et al. 2013), and OTB100 (Wu et al. 2015). We use accuracy occupy (AO) and success rate (SR) as metrics. In the following, we will briefly introduce the datasets used in our experiments.

The long-term aerial tracking dataset UAV123 consists of 123 videos with more than 110K frames. All of the sequences are from an aerial viewpoint. The dataset contains a wide variety of scenes including buildings, urban landscape, fields, roads, beaches and a harbor/marina, targets including cars, trucks, boats, persons, groups, and aerial vehicles, and activities including walking, cycling, wakeboarding, driving, swimming, and flying.

GOT-10k includes 563 object classes and 87 motion patterns in the real-world. It has over 10,000 segments and 1.5 million bounding boxes. The dataset is also split into unified training, validation, and testing subsets, which expands five subtrees: animal, artifact, person, natural object, and part. We use the testing subset containing 180 videos, 84 classes of moving objects, and 32 forms of motion.

The dataset OTB100 for tracker evaluation has 100 moving objects, containing basketball, birds, biker, etc. The full dataset consists of 100 sequences with more than 59K frames.

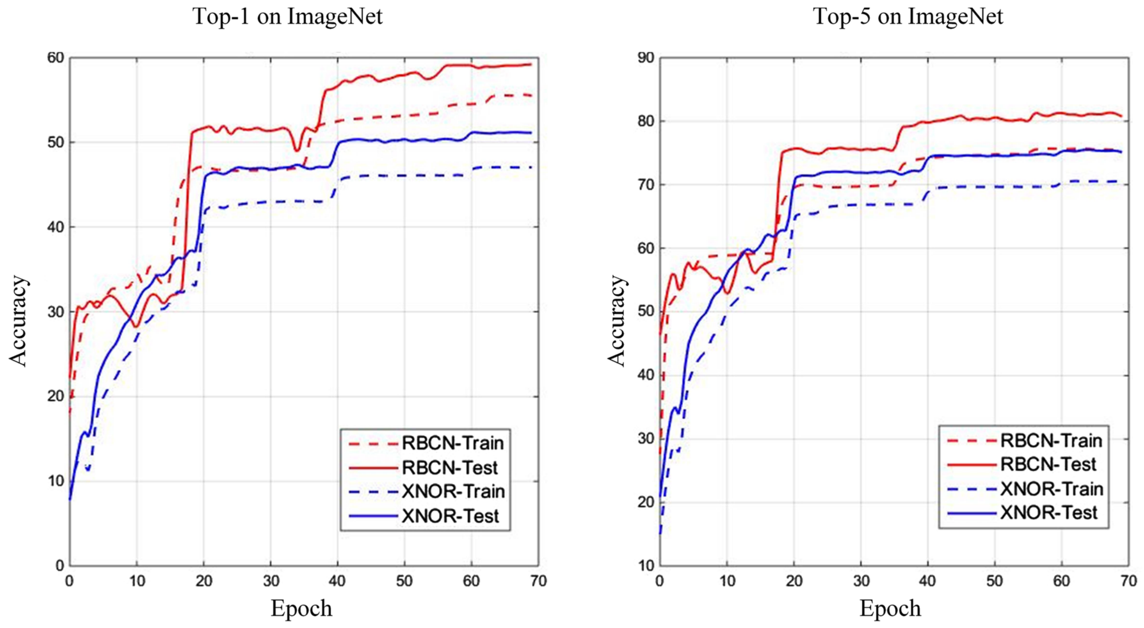

Fig. 3 Training and testing error curves of RBCN and XNOR-Net based on the ResNet18 backbone on ImageNet

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

Dataset	Index	SiamFC	XNOR-Net	Bi-real Net	PCNN	RB-SF
GOT-10K	AO	0.348	0.251	0.275	0.308	0.327
	SR	0.383	0.230	0.261	0.323	0.343
OTB50	Precision	0.761	0.457	0.464	0.648	0.706
	SR	0.556	0.323	0.340	0.470	0.496
OTB100	Precision	0.808	0.541	0.559	0.746	0.786
	SR	0.602	0.394	0.427	0.548	0.572
UAV123	Precision	0.745	0.547	0.572	0.622	0.688
	SR	0.528	0.374	0.407	0.449	0.497

OTB50 is a subset of OTB100 and consists of 51 sequences with more than 29k frames.

The results are shown in Table 5. Intriguingly, our framework achieves about 7% AO improvement over XNOR-Net, using the same network architecture as in SiamFC Network on GOT-10k.

4.4.2 Person Re-Identification

The task of person re-identification is to determine whether two images belong to the same subject. Specifically, the two images are usually captured by two disjoint cameras in practical applications. The performance of person re-identification is closely related to many other applications, such as behavior analysis, object retrieval, and cross-camera tracking. Although it has good performance on many benchmarks, the re-identification task is still difficult to be applied in the real-world due to the large amount of memory and computation costs. Therefore, we use the rectified method to compress the reID models in this paper to further improve the efficiency of CNNs. In the following, we will briefly introduce the person re-identification datasets used in our paper.

Market-1501 (Zheng et al. 2015) contains of 1501 identities captured from 6 different viewpoints, including 32668 labeled bounding boxes, which is currently the largest image-based reID benchmark dataset. Specifically, the dataset contains 12936 images with 751 identities for training, and 19732 images with 750 identities for testing.

DukeMTMC-reID (Zheng et al. 2017) contains 8 85-min high-resolution videos from 8 different cameras with available hand-drawn pedestrian bounding boxes.

The iLIDS dataset (Wang et al. 2014) is constructed from video images in a airport arrival hall. It has 119 pedestrians with 479 normalized images from non-overlapping cameras. In particular, the subjects in the iLIDS dataset are exposed to the chanllenge of large illumination changes and occlusions.

As shown in Table 6, our framework achieves 8.8% (= 67.4–58.6%) precision improvement over XNOR-Net, both

Table 6 Person re-identification (reID) accuracy (%) based on ResNet50, ResNet50_MID and DesNet121 on different datasets

	fp	XNOR-Net	PCNN	RBCN
Market-1501				
ResNet50	83.3	61.8	63.7	**65.6**
ResNet50_MID	87.9	68.3	72.6	**75.1**
DesNet121	87.5	58.6	65.6	**67.4**
DukeMTMC-reID				
ResNet50	75.9	45.0	47.2	**49.1**
ResNet50_MID	81.6	55.7	61.7	**64.2**
DesNet121	79.1	44.6	50.8	**52.4**
iLIDS				
ResNet50	56.0	41.3	44.7	**47.3**
ResNet50_MID	61.3	51.3	56.0	**57.3**
DesNet121	54.7	45.7	48.7	**50.7**

Bold is used to mark the best results among the binary networks

using the same network architecture as in DesNet121 Network on Market-1501. Besides, RBCN also outperforms XNOR-Net by 5.0% (= 50.7–45.7%) in DesNet121 Network on video dataset iLIDS, which confirms that our rectified network RBCNs can also obtain better performance on re-identification video dataset.

4.4.3 Face Recognition

Face recognition (FR) is important in recent days because it can be used to identity authentication. Though the face recognition technology has already obtained very high performance, it still meets the challenge of being used in light-weight devices because of the large memory and computation cost. Therefore, we employ our RBCN on the models in face recognition to improve their efficiency. In the following, We will briefly introduce the datasets used in our paper.

Training Dataset We use CASIA-WebFace (Yi et al. 2014) to train our RBCN models. Specifically, CASIA-WebFace has 494,414 face images belonging to 10,575 different individuals, which are horizontally flipped for data augmentation.

Testing Dataset LFW dataset (Huang et al. 2007) contains 13,323 web photos of 5749 celebrities, which are divided into 6000 face pairs in 10 splits. CFP dataset (Sengupta et al. 2016) contains 7000 images of 500 subjects, which is mainly used for evaluating pose variation. AgeDB (Moschoglou et al. 2017) contains 16,488 images of various 568 distinct subjects (e.g., actors, writers, scientists).

As shown in Table 7, RBCN achieves up to an 11.51% (= 83.63–72.12%) precision improvement over XNOR-Net using ResNet50 on AgeDB. RBCN with only weights binarized outperforms XNOR-Net with a gap over 18% (i.e., 90.33% vs. 72.12%) in ResNet50 on AgeDB. Furthermore, our framework with only binarized weights performs almost as well as the full precision ResNet50 Network.

Table 7 Face recognition accuracy (%) based on ResNet50 on different datasets

CASIA-Webface				
	fp	XNOR-Net	RBCN	RBCN*
ResNet50				
LFW	99.07	92.03	97.17	98.83
CFP	93.73	75.50	86.80	92.11
AgeDB	91.58	72.12	83.63	90.33

*Represents only weights are binarized

Table 8 Pruning results of ResNet18 on CIFAR10. RBCN (0.6) means that we prune 40% parameters on 1-bit networks

Model	Kernel stage	Param (M)	Accuracy
RBCN	16-16-32-64	0.19	82.16
RBCN	32-32-64-128	0.74	88.02
RBCN	32-64-128-256	2.80	89.91
RBCN (0.8)	16-16-32-64	0.15	81.04
RBCN (0.8)	32-32-64-128	0.59	87.47
RBCN (0.8)	32-64-128-256	2.24	89.77
RBCN (0.7)	16-16-32-64	0.13	80.35
RBCN (0.7)	32-32-64-128	0.52	87.31
RBCN (0.7)	32-64-128-256	1.96	89.89
RBCN (0.6)	16-16-32-64	0.11	79.29
RBCN (0.6)	32-32-64-128	0.44	85.59
RBCN (0.6)	32-64-128-256	1.68	89.80

4.5 Results on Network Pruning

In this section, we verify the performance of network pruning. Since BNNs have much information redundancy, we hope to use network pruning to obtain a more compact model without sacrificing much accuracy. In this implementation, we use ResNet18 as our backbone and change the pruning rates to validate the effectiveness of pruning on CIFAR10 dataset. The results are shown in Table 8. For ResNet18 with kernel stage 32-64-128-256, when we prune 20% channels of RBCN, we only sacrifice 0.14% accuracy compared with the baseline (i.e., 89.91–89.77%). For ResNet18 with kernel stage 32-32-64-128, when we prune 20% of the channels of RBCN, we obtain the result of 87.47% with significant memory saving and computational efficiency. From the results, we see that when we prune the redundancy in RBCNs, our models tend to be more compact while maintaining performance.

Table 9 Comparison of memory usage & FLOPs calculation

	RBCN	XNOR-Net	ResNet18
Memory usage	33.7 Mbits	33.7 Mbits	374.1 Mbits
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 10 Memory usage, FLOPs and accuracy comparison for different model compression methods

	Params (Mbits)	FLOPs	Accuracy
Full precision	374.1	1.81×10^9	69.30
FPGM (He et al. 2020)	261.9	1.05×10^9	68.41
SFP (He et al. 2018)	261.9	1.05×10^9	67.10
RKD (Gao et al. 2020)	261.9	1.81×10^9	71.46
RBCN	33.7	1.67×10^8	59.50

4.6 Efficiency Analysis

The memory usage is computed as the summation of 32× the number of real-valued parameters and 1× the number of binary parameters in BNNs. Furthermore, followed Liu et al. (2018), we use FLOPs to measure the speed. The FLOPs are calculated as the number of real-valued floating-point multiplications plus 1/64 of the number of 1-bit multiplications. From the results in Table 9, we can see that RBCN and XNOR-Net reduce the memory usage of the full precision ResNet18 by 11.10 times. For efficiency, both RBCN and XNOR-Net gain 10.86× speedup over ResNet18. Note that the computational and storage costs brought by the learnable matrices C are negligible. Considering that we only introduce a new training strategy (i.e., GAN) to better calculate 1-bit CNNs, the network architecture remains the same as other works Liu et al. (2018), Gu et al. (2019). Therefore, we have the same real acceleration ratio as other methods.

Furthermore, we compare the our RBCN with other model compression methods to evaluate our advantages in Table 10. For example, the distillation based method RKD (Gao et al. 2020) obtains higher accuracy with the same parameters and flops than the baseline. The pruning methods SFP and FPGM can reduce the number of parameters and flops with a little performance drop. Compared with these compression methods, the advantage of binarization is that the parameters and flops are extremely small, which has more potential to be used in resource limited devices.

5 Conclusion

This paper introduces Rectified Binary Convolutional Networks (RBCNs), which optimize BNNs by exploiting full precision kernels and feature maps in an end-to-end manner. We introduce GANs to train the binarized network with the guidance of its corresponding full precision model, significantly improving the performance of BNNs. RBCNs can not only be used for object classification but also other tasks as well, including object tracking, face recognition, and person re-identification. Experiments demonstrate the superior performance of RBCNs over state-of-the-art binarized models. We also achieve promising performance on model pruning, which validate the generality of our method. In the future, we will combine our method with neural architecture search (NAS) to build data-adaptive 1-bit CNNs.

Acknowledgements This work is supported by National Natural Science Foundation of China U20B2042, National Natural Science Foundation of China 62076019, and Science and Technology Innovation 2030-Key Project of "New Generation Artificial Intelligence" under Grant 2020AAA0108200. The work was supported in part by National Natural Science Foundation of China under Grants 62076016 and 61672079. This work is also supported by Shenzhen Science and Technology Program KQTD2016112515134654. Chunlei Liu and Wenrui Ding contribute equally. Baochang Zhang is the corresponding author who is also with Shenzhen Academy of Aerospace Technology, Shenzhen, China.

References

Ahn, S., Hu, S. X., Damianou, A., Lawrence, N. D., & Dai, Z. (2019). Variational information distillation for knowledge transfer. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 9163–9171).

Arjovsky, M., Chintala, S., & Bottou, L. (2017, August). Wasserstein generative adversarial networks. In *Proceedings of the 34th international conference on machine learning* (Vol. 70, pp. 214–223).

Belagiannis, V., Farshad, A., & Galasso, F. (2018). Adversarial network compression. In *Proceedings of the European conference on computer vision (ECCV)*.

Cai, H., Zhu, L., & Han, S. (2018). Proxylessnas: Direct neural architecture search on target task and hardware. arXiv preprint arXiv:1812.00332.

Changyong, S., Peng, L., Yuan, X., Yanyun, Q., Longquan, D., & Lizhuang, M. (2019). Knowledge squeezed adversarial network compression. arXiv preprint arXiv:1904.0510.

Chen, H., Lian Zhuo, B. Z., Zheng, X., Liu, J., Doermann, D., & Ji, R. (2020). Binarized neural architecture search. *Identity*, *2*, 3.

Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1251–1258).

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

Denton, E., Zaremba, W., Bruna, J., Lecun, Y., & Fergus, R. (2014). Exploiting linear structure within convolutional networks for efficient evaluation. arXiv preprint arXiv:1404.0736.

Gao, M., Shen, Y., Li, Q., & Loy, C. C. (2020). Residual knowledge distillation. arXiv preprint arXiv:2002.09168.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative adversarial nets. In *Advances in neural information processing systems* (pp. 2672–2680).

Gu, J., Zhang, B., & Liu, J. (2019). Projection convolutional neural networks. In *AAAI*.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., & Courville, A. C. (2017). Improved training of Wasserstein GANs. In *Advances in neural information processing systems* (pp. 5767–5777).

Guo, Y., Yao, A., & Chen, Y. (2016). Dynamic network surgery for efficient DNNs. In *Advances in neural information processing systems* (pp. 1379–1387).

Han, S., Pool, J., Tran, J., & Dally, W. (2015). Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems* (pp. 1135–1143).

Hassibi, B., & Stork, D. G. (1993). Second order derivatives for network pruning: Optimal brain surgeon. In *Advances in neural information processing systems* (pp. 164–171).

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *IEEE conference on computer vision and pattern recognition* (pp. 770–778).

He, Y., Kang, G., Dong, X., Fu, Y., & Yang, Y. (2018). Soft filter pruning for accelerating deep convolutional neural networks. arXiv preprint arXiv:1808.06866.

He, Y., Liu, P., Wang, Z., Hu, Z., & Yang, Y. (2020). Filter pruning via geometric median for deep convolutional neural networks acceleration. In *2019 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*.

He, Y., Zhang, X., & Sun, J. (2017). Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE international conference on computer vision* (pp. 1389–1397).

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., & Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861.

Huang, G. B., Ramesh, M., Berg, T., & Learned-Miller, E. (2007). Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical report 07-49. Amherst: University of Massachusetts.

Huang, L., Zhao, X., & Huang, K. (2018). Got-10k: A large high-diversity benchmark for generic object tracking in the wild. arXiv preprint arXiv:1810.11981.

Huang, Z., & Wang, N. (2018). Data-driven sparse structure selection for deep neural networks. In *Proceedings of the European conference on computer vision (ECCV)* (pp. 304–320).

Iandola, F. N., Han, S., Moskewicz, M. W., Ashraf, K., Dally, W. J., & Keutzer, K. (2016). Squeezenet: Alexnet-level accuracy with 50× fewer parameters and <0.5 mb model size. arXiv preprint arXiv:1602.07360.

Jaderberg, M., Vedaldi, A., & Zisserman, A. (2014). Speeding up convolutional neural networks with low rank expansions. arXiv preprint arXiv:1405.3866.

Krizhevsky, N. (2009). Hinton: The cifar-10 dataset. Online http://www.cs.toronto.edu/kriz/cifar.html.

Lebedev, V., Ganin, Y., Rakhuba, M., Oseledets, I., & Lempitsky, V. (2014). Speeding-up convolutional neural networks using fine-tuned cp-decomposition. arXiv preprint arXiv:1412.6553.

Li, H., Kadav, A., Durdanovic, I., Samet, H., & Graf, H. P. (2016). Pruning filters for efficient convnets. arXiv preprint arXiv:1608.08710.

Lin, S., Ji, R., Li, Y., Wu, Y., Huang, F., & Zhang, B. (2018). Accelerating convolutional networks via global & dynamic filter pruning. In *IJCAI* (pp. 2425–2432).

Lin, S., Ji, R., Yan, C., Zhang, B., Cao, L., Ye, Q., Huang, F., & Doermann, D. (2019). Towards optimal structured CNN pruning via generative adversarial learning. In *Proceedings of CVPR* (pp. 2790–2799).

Lin, X., Zhao, C., & Pan, W. (2017). Towards accurate binary convolutional neural network. In *Advances in neural information processing systems* (pp. 345–353).

Liu, C., Ding, W., Xia, X., Hu, Y., Zhang, B., Liu, J., Zhuang, B., & Guo, G. (2019). RBCN: Rectified binary convolutional networks for enhancing the performance of 1-bit DCNNs. In *International joint conference on artificial intelligence*.

Liu, Z., Shen, Z., Savvides, M., & Cheng, K. T. (2020). Reactnet: Towards precise binary neural network with generalized activation functions. arXiv preprint arXiv:2003.03488.

Liu, Z., Wu, B., Luo, W., Yang, X., Liu, W., & Cheng, K. T. (2018). 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* (pp. 722–737).

Li, Y., Lin, S., Zhang, B., Liu, J., Doermann, D., Wu, Y., Huang, F., & Ji, R. (2019). Exploiting kernel sparsity and entropy for interpretable CNN compression. In *Proceedings of CVPR* (pp. 2800–2809).

Li, Z., Ni, B., Zhang, W., Yang, X., & Gao, W. (2017). Performance guaranteed network acceleration via high-order residual quantization. In *Proceedings of the IEEE international conference on computer vision* (pp. 2584–2592).

Mao, X., Li, Q., Xie, H., Lau, R. Y., Wang, Z., & Paul Smolley, S. (2017). Least squares generative adversarial networks. In *Proceedings of the IEEE international conference on computer vision* (pp. 2794–2802).

Molchanov, P., Tyree, S., Karras, T., Aila, T., & Kautz, J. (2016). Pruning convolutional neural networks for resource efficient inference. arXiv preprint arXiv:1611.06440.

Moschoglou, S., Papaioannou, A., Sagonas, C., Deng, J., Kotsia, I., & Zafeiriou, S. (2017). Agedb: The first manually collected, in-the-wild age database. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops* (pp. 51–59).

Mueller, M., Smith, N., & Ghanem, B. (2016). A benchmark and simulator for UAV tracking. In *European conference on computer vision* (pp. 445–461).

Odena, A., Olah, C., & Shlens, J. (2017). Conditional image synthesis with auxiliary classifier GANs. In *Proceedings of the 34th international conference on machine learning* (Vol. 70, pp. 2642–2651).

Rastegari, M., Ordonez, V., Redmon, J., & Farhadi, A. (2016). XNOR-net: ImageNet classification using binary convolutional neural networks. In *European conference on computer vision*.

Rigamonti, R., Sironi, A., Lepetit, V., & Fua, P. (2013). Learning separable filters. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2754–2761).

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., et al. (2015). ImageNet large scale visual recognition challenge. *International Journal of Computer Vision*, *115*(3), 211–252.

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., & Chen, X. (2016). Improved techniques for training GANs. In *Advances in neural information processing systems* (pp. 2234–2242).

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 4510–4520).

Sengupta, S., Chen, J. C., Castillo, C., Patel, V. M., Chellappa, R., & Jacobs, D. W. (2016). Frontal to profile face verification in the wild. In *2016 IEEE winter conference on applications of computer vision (WACV)* (pp. 1–9).

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1–9).

Tan, M., & Le, Q. V. (2019). Efficientnet: Rethinking model scaling for convolutional neural networks. arXiv preprint arXiv:1905.11946.

Tschannen, M., Agustsson, E., & Lucic, M. (2018). Deep generative models for distribution-preserving lossy compression. In *Advances in neural information processing systems* (pp. 5933–5944).

Wang, T., Gong, S., Zhu, X., & Wang, S. (2014). Person re-identification by video ranking. In *European conference on computer vision* (pp. 688–703).

Wu, Y., Lim, J., & Yang, M. H. (2013). Online object tracking: A benchmark. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2411–2418).

Wu, Y., Lim, J., & Yang, M. H. (2015). Object tracking benchmark. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *37*(9), 1834–1848.

Xu, Z., Hsu, Y. C., & Huang, J. (2018). Training student networks for acceleration with conditional adversarial networks. In *BMVC* (p. 61).

Yang, T. J., Chen, Y. H., & Sze, V. (2017). Designing energy-efficient convolutional neural networks using energy-aware pruning. In *Proceedings of CVPR* (pp. 5687–5695).

Yang, Z., Moczulski, M., Denil, M., de Freitas, N., Smola, A., Song, L., & Wang, Z. (2015). Deep fried convnets. In *Proceedings of the IEEE international conference on computer vision* (pp. 1476–1483).

Yi, D., Lei, Z., Liao, S.,& Li, S. Z. (2014). Learning face representation from scratch. arXiv preprint arXiv:1411.7923.

Zagoruyko, S., & Komodakis, N. (2016). Wide residual networks. arXiv preprint arXiv:1605.07146.

Zhang, D., Yang, J., Ye, D., & Hua, G. (2018). LQ-nets: Learned quantization for highly accurate and compact deep neural networks. In *Proceedings of the European conference on computer vision (ECCV)* (pp. 365–382).

Zhang, J., Pan, Y., Yao, T., Zhao, H., & Mei, T. (2019). DABNN: A super fast inference framework for binary neural networks on arm devices. In *Proceedings of the 27th ACM international conference on multimedia* (pp. 2272–2275).

Zhang, X., Zhou, X., Lin, M., & Sun, J. (2018). Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 6848–6856).

Zheng, L., Shen, L., Tian, L., Wang, S., Wang, J., & Tian, Q. (2015). Scalable person re-identification: A benchmark. In *Proceedings of the IEEE international conference on computer vision* (pp. 1116–1124).

Zheng, Z., Zheng, L., & Yang, Y. (2017). Unlabeled samples generated by GAN improve the person re-identification baseline in vitro. CoRR arXiv:1701.07717.

Zhou, A., Yao, A., Guo, Y., Xu, L., & Chen, Y. (2017). Incremental network quantization: Towards lossless CNNs with low-precision weights. arXiv preprint arXiv:1702.03044.

Zhou, S., Wu, Y., Ni, Z., Zhou, X., Wen, H., & Zou, Y. (2016). Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. arXiv preprint arXiv:1606.06160.