Research Article

A Lightweight Binarized Convolutional Neural Network Model for Small Memory and Low-Cost Mobile Devices

Xuan Qi, Zegang Sun, Xue Mei, and Ryad Chellali

College of Electrical Engineering and Control Science, Nanjing Tech University, Nanjing 211816, China

Correspondence should be addressed to Xue Mei; mx@njtech.edu.cn

Received 9 November 2022; Revised 4 March 2023; Accepted 14 March 2023; Published 12 April 2023

Academic Editor: Yugen Yi

Copyright © 2023 Xuan Qi et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

In recent years, the high cost of implementing deep neural networks due to their large model size and parameter complexity has made it a challenging problem to design lightweight models that reduce application costs. The existing binarized neural networks suffer from both the large memory occupancy and the big number of trainable params they use. We propose a lightweight binarized convolutional neural network (CBCNN) model to address the multiclass classification/identification problem. We use both binary weights and activation. We show experimentally that a model using only 0.59 M trainable params is sufficient to reach about 92.94% accuracy on the GTSRB dataset, and it has similar performances compared to other methods on MNIST and Fashion-MNIST datasets. According to these results, we show that despite simplifications and color information removal, our network achieves similar performances compared to classical CNNs with lower costs in both in training and embedded deployment.

1. Introduction

Deep neural networks (DNNs) are making remarkable progresses every day and involved in many application fields. Computer vision, natural language processing and many other domains benefit from these progresses opening doors to new solutions to hard problems. Convolutional neural networks (CNNs) are the most common method in use these days. CNNs solve various visual problems such as image classification, recognition, or detection. New CNN models are constantly proposed and improved, such as ResNeXt [1] and SK-Net [2]; however, their architecture does not change much during the last decade. The main improvements were possible thanks to the computational power availability: the use of GPU-based machines as well as the increase of the associated memories allows CNNs to achieve outstanding performances. A natural question aroused that is it possible to reach similar or better performances at a lower computational cost? That is to say, is it possible to have CNNs or equivalent, running or cheaper machines with less memory (typically mobile phones for instance) and having comparable results? This new open problem has been addressed recently and people started to develop methods based on model compression and binarization. Yoshua Bengio in his seminal work [3] introduced a method for training binarized neural networks (BNNs). Indeed, in the training phase, binary weights and activations replace the real ones in the gradients operations as for CNNs. This greatly reduces the used memory size, the access times to it, and replaces most arithmetic operations with bitwise operations, which fits exactly with the initial quest of keeping the same effectiveness at a lower cost.

Our work takes inspiration from BNN [3]. We focused mostly on two points: (i) finding the conditions to reduce the number of trainable params and (ii) deriving the best preprocessing operation, all together with keeping the highest performances possible at the lowest cost possible. Moreover, we show also the color information is not necessary and the brightness in images is sufficient to reach similar performances to classical CNNs in executing the
same tasks. We obtain good experimental results. This double compression method has the following three advantages:

1. Effectively reducing the amount of calculation in the training process and accelerate the training speed of the model.
2. Greatly reducing the memory space occupied by the model.
3. Greatly reducing the actual application deployment cost.

2. Related Work

2.1. Information Loss. Unlike optimizing the binary process directly at the convolution layer, LAB2 [4] directly considers the binary loss and applies the near-end Newton algorithm to the binary weights. CI-BCNN [5], through learning to strengthen graph models, mining channel-level interaction, and iterating pop count, reduces symbol inconsistency in binary feature graphs and retains the input sample information. LNS [6] proposes to predict binary weight by monitoring noise learning and training binary functions. ProxyBNN [7] utilizes basis and coordinate submatrices to form the weight matrix prior to binary conversion, while IR-NET [8], RBNN [9], IA-BNN [10], SLB [11], and BBG-NET [12] optimize, reshape, activate, and allocate weights for the binary conversion process.

2.2. Network Structures. In the existing binary research on classical networks, there are some problems, such as too much memory, more parameters, complex network model structure, and relatively high application cost due to inheriting the structure of classical neural networks. Moreover, different binarized neural network architectures will not only affect the performance of binarized convolutional neural networks, but also affect the actual hardware deployment cost. Subsequent researchers have made a series of enhancements to BNN [3], such as BBG-Net [12] and Real-to-Bin [13], which aim to improve the accuracy of ResNet and other high-performance conventional networks. DMS [14] has effectively narrowed the precision gap between full-precision networks. BATS [15], BNAS [16], NASB [17], and high-capacity-expert [18] have proposed specialized NAS approaches to design architectures for searching BNN and comparing the accuracy of similar network models with some binarized conventional networks, such as ResNet. Meanwhile, high-capacity-expert [18] applies a conditional calculation method called expert convolution in BNN, combining the convolution group with the above methods. MobilNet-Mid [19] and Binarized MobileNet [20] propose a new, lighter BNN structure with better precision performance in reference to MobileNet-V1. MeliusNet [21] and ReActNet [22] design a new BNN model structure with a less floating point and binary operations (FLOPs/BOPs) calculation cost, which has better accuracy than full-precision lightweight MobileNet. BNN-BN-free [23] incorporates the BN-free [24] concept and presents a method of constructing a network architecture without batch normalization, which has been replaced by the scaling factor. FracBNN [25] reasonably extends the topology of ReActNet, reconstructs the network block. BCNN [26] designs a specific network structure specifically for the ImageNet data classification task, and its model is more lightweight than MeliusNet and ReActNet. The binary operation based on the classical network sometimes wastes computational resources while dealing with some practical small-scale engineering applications. At the same time, the model needs more memory space and increased application cost. We get a lot of inspiration on the basis of previous research, and then we design a lightweight CBCNN model to meet the hardware deployment problem in reality.

2.3. Training Strategy. The choice of training schemes and technique also affects the best accuracy of the neural network. Main/subsidiary [27] proposes a method for pruning BNN filters. Bop [28] and UniQ [29] each propose a new optimizer for training BNN. Referring to the lottery ticket hypothesis [30], MPT [31] designs a simpler scheme to learn BNN with high precision pruning and quantifying the full precision CNN with random weighting. Real-to-bin [13] designs a two-step training strategy using the method of transfer learning to train BNN by learning the real-value retraining network. By implementing this training strategy, highly accurate models such as ReActNet [22], high-capacity-expert [18], and BCNN [26] are ultimately trained. Additionally, BNN-stochastic [32] proposes a relaxed approach to stochastic methods that enhances the accuracy of the CIFAR-10 dataset. The above research has laid a solid foundation for the development of the binarized network, which greatly reduces the computational complexity and gradually increases the accuracy. Facing the needs of hardware deployment in application, a sequential model dual compression binarized convolutional neural network structure CBCNN is studied to make the network structure lighter.

2.4. CBCNN (Compress Binarized Convolutional Neural Network). CBCNN is a sequential model structure, which makes the model simpler than others. We binarize the network weights and activation functions to participate in the errors back-propagation. During training, binarized weights and activation values are involved in the calculation of gradients. When making predictions, the weights and activation values of the network are binary (−1/+1).

This section describes our proposed compress binarized convolutional neural network (CBCNN) framework and the related training details.

2.5. Model. The core target in our CBCNN is to reasonably compress the params of BCNN to make the model more lightweight. CBCNN contains three types of blocks, which we named Binary Block 1, Binary Block 2, and Image Compression Block as shown in Figure 1, where Binary Block 1 contains a Binary_C (the binary convolution layer),
a MaxPooling layer as well as a batch normalization layer, and Binary Block 2 contains a Binary_D (the binary dense layer) and a batch normalization layer. In addition, we design Image Compression Block to effectively compress the dataset. Our network architecture is shown in Figure 2, where different blocks are set for different input size. We evaluated our model with three datasets of two different sizes, 32 * 32 * 1, 28 * 28 * 1, and 28 * 28 * 1, respectively.

The Binary_C layer is designed to extract features. We carry out a series of experiments on the configuration of different blocks, and finally choose the model structure reasonably according to experimental results. The kernel size is 3 * 3, and the pool size is 2 * 2.

2.6. Training. As the GTSRB [33] dataset contains RGB color images, we use Image Compression Block to carry out some data preprocessing before training. The input images are converted from RGB to YUV, and the two color channels U and V are removed, while the brightness channel Y is kept and used as the input of the network. Meanwhile, we use the histogram equalization and the standardize features methods for training. In addition, the final mapping of histogram equalization is shown in equation (1), where \( S_k \) is the target pixel value, \( r_k \) is the original pixel value, \( L \) is the gray level, \( P_r (r_j) \) is the probability of \( r_j \) in the original image, \( MN \) is the total number of pixels in the image, and \( n_j \) represents the number of pixels with \( j \) in the original image.

\[
S_k = T(r_k) = (L - 1) \sum_{j=0}^{k} P_r (r_j) = \frac{L - 1}{MN} \sum_{j=0}^{k} n_j, \quad k = 1, 2, 3, \ldots L - 1, \quad (1)
\]

\[
\text{Std} = \frac{m - \mu}{\max (\sigma, 1.0/\sqrt{P})}. \quad (2)
\]

We introduce the implementation principles of Binary_act (the binary activation function), Binary_C (the binary convolutional layer), and Binary_D (the binary dense layer), respectively. The calculation rules of the single-layer gradient in the CBCNN model we defined are shown in Algorithm 1, where \( x \) is the weight of the input, \( g_x \) is the current gradient of the input, \( y \) is the weight of the output, and \( g_y \) is the gradient of the output. The process of Algorithm 1 is shown in Figure 3.

In order to train each layer of the CBCNN model according to Algorithm 1, we use the “hard_sigmoid” function as follows:

\[
\text{cross}(x) = x + S_G \left( \left| x + \frac{1}{2} \right| - x \right). \quad (4)
\]

We define the activation function of the CBCNN model as Binary_act (equation (5)), where “S_G” means the stop gradient and “h_s” means the “hard_sigmoid” function in equation (3).

\[
\text{Binary}_\text{act}(x) = 2 \left\{ h_s(x) + S_G \left[ \left| h_s(x) + \frac{1}{2} \right| - h_s(x) \right] \right\} - 1. \quad (5)
\]

As for Binary_C, we propose the function Binary_kC (equation (6)) to binarize the kernel of the convolutional layers in the CBCNN model. \( W_C \) is the value of the kernel in convolutional layers, where “S_G” means the stop
Figure 1: (a) Binary Block 1. (b) Binary Block 2. (c) Image Compression Block.

Figure 2: CBCNN architecture. (a) The size of neural network input is $32 \times 32 \times 1$ on GTSRB. (b) The size of neural network input is $28 \times 28 \times 1$ on fashion-MNIST and MNIST.

Figure 3: (a) Forward propagation process in Algorithm 1. (b) Backward propagation process in Algorithm 1. The parameters in the left square are input, and the parameters in the right square are output. Their positions in the square correspond one to one.
gradient and “h_s” means the “hard_sigmoid” function (equation (3)). Through function equation (6), we convert the value between $[-H_1, H_1]$ to $-H_1$ or $H_1$.

\[
\text{Binary}_{kC}(W_C, H_1) = 2 \ast H_1 \ast \left\{ h_s\left(\frac{W_C}{H_1}\right) + S_G\left[ h_s\left(\frac{W_C}{H_1}\right) + \frac{1}{2} \right] - h_s\left(\frac{W_C}{H_1}\right) \right\} - H_1. \tag{6}
\]

As for Binary_D, we propose the function Binary_{kD} (equation (7)) to binarize the kernel of the dense layers in the CBCNN model, $W_D$ is the value of the kernel in dense layers, where “S_G” means the stop gradient and “h_s” means the “hard_sigmoid” function (equation (3)). Through function equation (7), we convert the value between $[-H_2, H_2]$ to $-H_2$ or $H_2$.

\[
\text{Binary}_{kD}(W_D, H_2) = 2 \ast H_2 \ast \left\{ h_s\left(\frac{W_D}{H_2}\right) + S_G\left[ h_s\left(\frac{W_D}{H_2}\right) + \frac{1}{2} \right] - h_s\left(\frac{W_D}{H_2}\right) \right\} - H_2. \tag{7}
\]

3. Experimental Results

We tested our models on three different datasets (GTSRB [33], Fashion-MNIST [34], and MNIST [35]) and compared them to other neural network models that use convolution and binary methods.

3.1. GTSRB Test and Analysis. In order to better simulate the classification problems in actual engineering, in this article, we choose a more challenging and practical dataset (43 classes of traffic signs) to evaluate the performance of our model. We choose German Traffic Sign Recognition Benchmark (GTSRB) [33], a database for traffic sign recognition provided by the INI Institute of Neural Computation in Germany. Finally, 51840 images, more than 1700 instances, a total of 43 classes were obtained. According to the number of traffic sign pictures of each class, we reasonably divide them into a training set and a validation set. We have a training set with 39209 samples and a validation set with 12630 samples. To our knowledge, this article is the second to evaluate binarized neural networks on the GTSRB dataset. Our data of each class and their number distribution in the training set are shown in Figure 4. Our data of each class and their number distribution in the validation set are shown in Figure 5. On the GTSRB dataset, we compare against methods [36–40], Faster R-CNN [41], and 5 traditional methods [42] on test (12630 images). The result is shown in Table 1.
Table 1: The accuracy performance of different methods is compared on the GTSRB dataset.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Accuracy (%)</th>
<th>Params (M)</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOG + SVM [42]</td>
<td>77.6</td>
<td>—</td>
</tr>
<tr>
<td>LBP + SVM [42]</td>
<td>71.1</td>
<td>—</td>
</tr>
<tr>
<td>LBP + RF [42]</td>
<td>69.7</td>
<td>—</td>
</tr>
<tr>
<td>PI + LDA + SVM [42]</td>
<td>82.3</td>
<td>—</td>
</tr>
<tr>
<td>LDA + RF [42]</td>
<td>82.3</td>
<td>—</td>
</tr>
<tr>
<td>Faster R-CNN [41]</td>
<td>91.8</td>
<td>—</td>
</tr>
<tr>
<td>Multiscale CNN [36]</td>
<td>95.4</td>
<td>—</td>
</tr>
<tr>
<td>MobileNet [39]</td>
<td>88.15</td>
<td>—</td>
</tr>
<tr>
<td>ShuffleNet [39]</td>
<td>88.99</td>
<td>—</td>
</tr>
<tr>
<td>EffNet [39]</td>
<td>91.79</td>
<td>—</td>
</tr>
<tr>
<td>Multicolumn [37]</td>
<td>99.46</td>
<td>90</td>
</tr>
<tr>
<td>Weighted multiconvolutional [38]</td>
<td>99.59</td>
<td>75.3</td>
</tr>
<tr>
<td>DCR (bitwise operation) [40]</td>
<td>92.86</td>
<td>0.83</td>
</tr>
<tr>
<td>CBCNN (ours, bitwise operation)</td>
<td>92.94</td>
<td>0.59</td>
</tr>
</tbody>
</table>

(a) Train Samples (one per class)

(b) Train Distribution

Figure 4: (a) Number of training set classes. (b) Number of training set quantity distribution of each class.

(a) Test Samples (one per class)

(b) Test Distribution

Figure 5: (a) Number of validation set classes. (b) Number of validation set quantity distribution of each class.
We set some training parameters, the epoch is 1000. We use batch normalization with a minibatch of size 200 to speed up the training. The optimizer used is “Adam” and the loss function used is “squared hinge”. We use the learning rate as an initial value of $10^{-3}$ and an end value of $10^{-4}$. The accuracy and loss we obtained are shown in Figure 6, and the accuracy of the model reaches 92.94%. In Table 1, we can clearly see that CBCNN is superior to the five traditional methods in [42], the accuracy is 1.14% higher than that of Faster R-CNN [41] and only 6.65% lower than the current state-of-the-art result [37]. However, the memory of our model is only 6.81 MB, trainable params are only 0.59 M, and bitwise operation can be performed at the same time.

3.2. Fashion-MNIST Test and Analysis. Fashion-MNIST [34] is a dataset composed of objects related to clothing, shoes, and bags. The training set and test set of Fashion-MNIST have a consistent distribution with the training set and test set of MNIST. To our knowledge, this article is the first to evaluate binarized neural networks on the Fashion-MNIST dataset. We compare the test (10000 images) with other
advanced methods [43–48] on the Fashion-MNIST dataset, and the result is shown in Table 2. We set some training parameters, the epoch is 500. We use batch normalization with a minibatch of size 50 to speed up the training. The optimizer used is "Adam" and the loss function used is "squared hinge." We use the learning rate as an initial value of $10^{-3}$ and an end value of $10^{-4}$. The accuracy and loss we obtained are shown in Figure 7, and the accuracy of the model reaches 92.86%. In Table 2, we can clearly see that the accuracy of CBCNN is only 4.05% lower than that of the current best method [48]. However, the memory of our model is only 1.89 MB, trainable params are only 0.48 M, and bitwise operation can be performed at the same time.

3.3. MNIST Test and Analysis. MNIST is a benchmark image classification dataset [35]. It is made up of $28 \times 28$ grayscale images, representing numbers between 0 and 9, and contains 60000 training sets and 10000 test sets. In BNN [3], the binary MLP method is used to obtain the best accuracy of 99.04% on MNIST, but the design of MLP makes the model occupy a large amount of memory. We compare the results tested by the CBCNN method with other methods, and the result is shown in Table 3.

Table 2: The accuracy performance of different methods is compared on the Fashion-MNIST dataset.

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Accuracy (%)</th>
<th>Params (M)</th>
<th>Search methods</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNeXt-8-64 + random erasing [43]</td>
<td>96.2 ± 0.06</td>
<td>34.4</td>
<td>Manual</td>
</tr>
<tr>
<td>ResNet-110 + random erasing [43]</td>
<td>95.9 ± 0.13</td>
<td>1.7</td>
<td>Manual</td>
</tr>
<tr>
<td>VGG8B [44]</td>
<td>95.47</td>
<td>7.3</td>
<td>Manual</td>
</tr>
<tr>
<td>WRN-28-10 + random erasing [43]</td>
<td>96.3 ± 0.03</td>
<td>36.5</td>
<td>Manual</td>
</tr>
<tr>
<td>DARTS (2nd order) + cutout + random erasing [47]</td>
<td>96.57</td>
<td>2.6</td>
<td>Gradient-based</td>
</tr>
<tr>
<td>Fine-tuning DARTS [48]</td>
<td>96.91</td>
<td>3.2</td>
<td>Gradient-based</td>
</tr>
<tr>
<td>Neupde [46]</td>
<td>92.40</td>
<td>0.4</td>
<td>Manual</td>
</tr>
<tr>
<td>CBCNN (ours, bitwise operation)</td>
<td>92.86</td>
<td>0.48</td>
<td>Gradient-based</td>
</tr>
</tbody>
</table>

Our experimental parameter configuration is the same as the Fashion-MNIST test. The accuracy and loss we obtained are shown in Figure 8, and the accuracy of the model reaches 99.32%. In Table 3, we can see that the best accuracy of CBCNN is 0.28% higher than that of the current best method [3] in binarized neural networks. Moreover, the memory of our model is only 1.89 MB, trainable params are only 0.48 M, and bitwise operation can be performed at the same time.

4. Discussion

We analyze the model performance of CBCNN as follows.
4.1. Memory Size and Accesses. Compared to 32-bit DNNs, CBCNN has 32 times less memory and 32 times less memory access. This will effectively reduce energy consumption by more than 32 times. At the same time, after the network layers and training data are effectively compressed, the used memory will be greatly reduced, which is more suitable for embedded deployment.

4.2. Forward Pass Efficiency. During forward pass (run time and training time), CBCNN drastically reduces memory size and access and replaces most arithmetic operations with bitwise operations. We think CBCNN can reduce the time complexity by 60% on dedicated hardware [3].

4.3. Binary Activations and Weights. In CBCNN, the network’s activations and weights are both limited to $-1$ or $+1$. Thus, a large number of 32-bit floating-point operations are replaced by 1-bit operations and consequently the actual application cost is drastically reduced.

4.4. Binary Filters. CBCNN has binary filters, and 2D filters of size $3 \times 3$ repeat themselves. If dedicated hardware and software are used, we can apply unique 2D filters on each feature map and add the results to obtain the convolutional results of each 3D filter [3].

According to the experimental results, the memory sizes of our models in the three datasets are 6.81 MB, 1.89 MB, and 1.89 MB, respectively, which is enough for our models to be deployed on the chip with very limited memory. In addition, our models work in the way of bitwise operation, which is enough to be applied to hardware circuits containing only low memory elements. The operation of reducing model parameters greatly reduces the application cost of the models in mobile terminal devices. At the same time, we provide developers with an idea of the binarized convolutional neural network model after double compression, so that developers can reasonably configure the number of binary blocks according to their own project requirements.

5. Conclusions

In this article, we propose a lightweight neural network CBCNN (compress binarized convolutional neural network) to solve the problem of image multiclassification recognition. We compress both the datasets and the binarized convolutional neural network structures when dealing with the multiclassification problem. CBCNN obtain the most advanced results in binarized convolutional neural networks when dealing with the multiclassification problem. CBCNN obtain the most advanced results in binarized convolutional neural networks on GTSRB [33], Fashion-MNIST [34], and MNIST [35]. In addition, in the process of the forward pass (running and training), the CBCNN model replaces most arithmetic operations with bitwise operations and reduces the memory size and memory access 32 times. Furthermore, the dual compression method (the network structure and dataset) greatly reduces the memory space occupied by the model and enables the potential for loading neural networks onto portable devices that have severely limited memory, which is more conducive to neural network embedded deployment. Experimental results show that CBCNN has a slightly lower accuracy than the convolutional neural network when dealing with multiclassification problems, but CBCNN has a lower cost in hardware deployment. High-performance neural network architecture sometimes causes waste of computing resources when dealing with some practical engineering problems. Moreover, excessive reliance on high-performance hardware increases the application cost. In the future, we will continue to work on improving the performance of binarized neural networks by changing the network structures and training strategies.

Data Availability

Our code is available at https://github.com/AI-Xuan/CBCNN. All experimental datasets are public datasets.
Conflicts of Interest

The authors declare that they have no conflicts of interest.

Authors’ Contributions

Xuan Qi and Zegang Sun contributed equally to this work.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant no. 61973334).

References


