Hindawi
Mobile Information Systems
Volume 2023, Article ID 5870630, 11 pages
https://doi.org/10.1155/2023/5870630

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

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. Accordingly, most arithmetic operations with bitwise operations are simplified, thus both used memory size and memory accesses are reduced by 32 times. Moreover, the color information was removed, which also reduced drastically the training time. All these together allow our architecture to run effectively and in real time on simple CPUs (rather than GPUs). Through the results we obtained, 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 method. MoBiNet-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, $\mathrm{P_r}(r_j)$ 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. Then, the standardize features method is presented as shown in equation (2), where μ is the mean value of the image, m is the image matrix, σ is the standard variance, and P represents the pixel value of the image. For Fashion-MNIST [34] and MNIST [35], which are themselves grayscale images of a channel, we do not carry out additional processing before training.

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

$$\mathrm{Std} = \frac{m-\mu}{\max(\sigma, 1.0/\sqrt{P})}. \tag{2}$$

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:

$$\mathrm{hard\_sigmoid}(x) = h\_s(x) = \begin{cases} 0, & x \le 1, \\ 0.5 * x + 0.5, & -1 < x < 1, \\ 1, & \mathrm{x} \ge 1. \end{cases} \tag{3}$$

In the training process of the CBCNN model, in order to realize the forward propagation algorithm and backward propagation algorithm (Algorithm 1), we define the intermediate function "cross" in equation (4). "S_G" means the stop gradient.

$$\mathrm{cross}(x) = x + S\_G\left(\lfloor x + \frac{1}{2} \rfloor - x\right). \tag{4}$$

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).

$$\mathrm{Binary\_act}(x) = 2 * \left\{h\_s(x) + S\_G\left[\lfloor h\_s(x) + \frac{1}{2} \rfloor - h\_s(x)\right]\right\} - 1. \tag{5}$$

As for Binary_C, we propose the function Binary_k_C (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

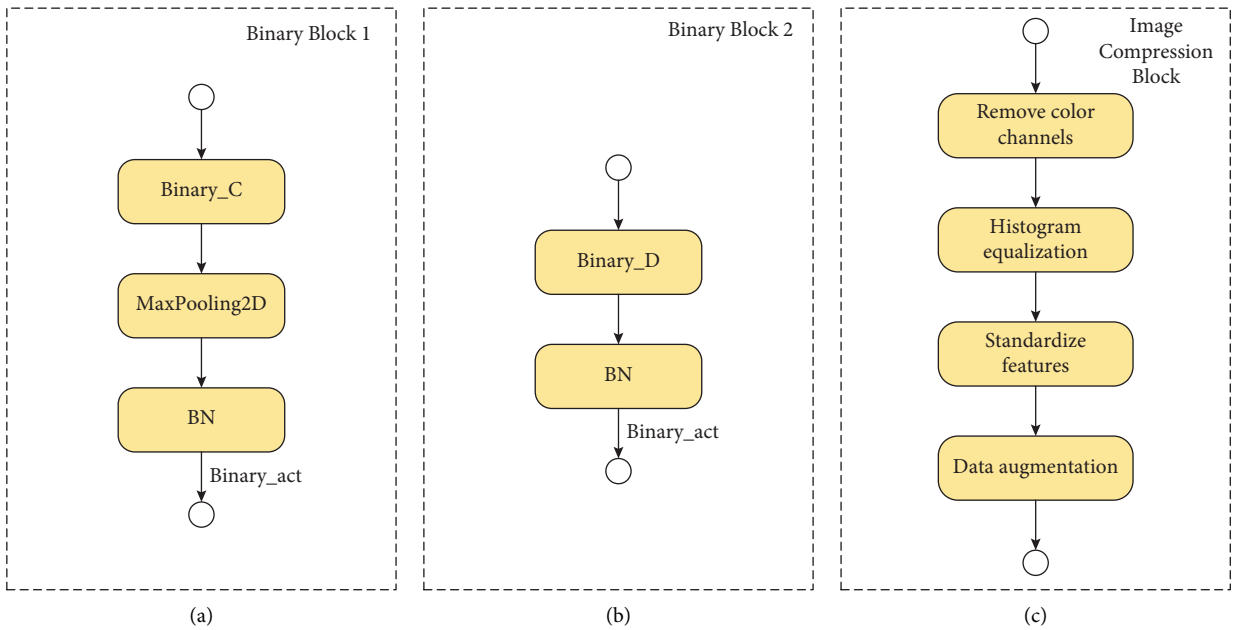


Figure 1: (a) Binary Block 1. (b) Binary Block 2. (c) Image Compression Block.

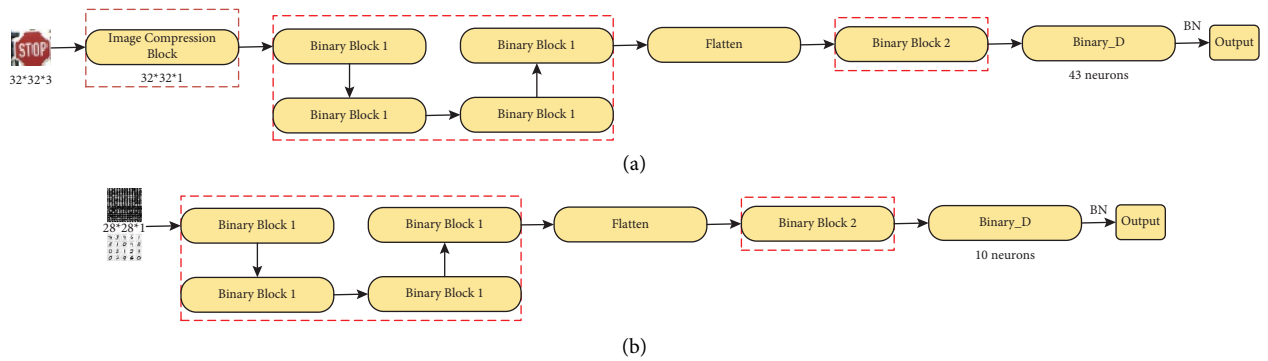


Figure 2: CBCNN architecture. (a) The size of neural network input is $32 \times 32 \times 1$ on GTSRB. (b) The size of neural network input is $28 \times 28 \times 1$ on fashion-MNIST and MNIST.

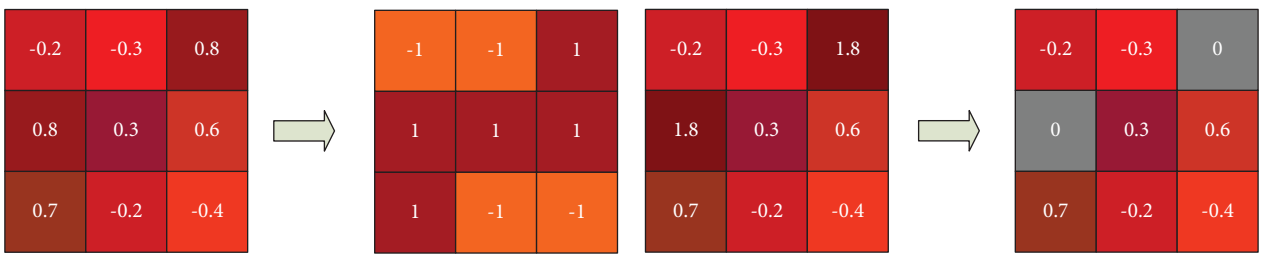


Figure 3: (a) Forward propagation process in Algorithm 1. (b) Backward propagation process in Algorithm 1. The parameters in the left square are input, and the parameters in the right square are output. Their positions in the square correspond one to one.

Input: x, g_x
Output: y, g_y
(1) Begin
(2) Case 1: Forward propagation
(3) if $x \leq 0$
(4) $y = -1$, $g_y = 0$
(5) end if
(6) else
(7) $y = 1$, $g_y = 0$
(8) end else
(9) Case 2: Backward propagation
(10) if $x \leq -1$
(11) $y = -1$, $g_y = 0$
(12) end if
(13) if $-1 < x < 1$
(14) $y = x$, $g_y = g_x$
(15) end if
(16) else
(17) $y = 1$, $g_y = 0$
(18) end else
(19) end

Algorithm 1: Rules for calculating the gradient of a single layer in CBCNN.

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.

$$\text{Binary\_}k_C(W_C, H_1) = 2 * H_1 * \left\{ h\_s\left(\frac{W_C}{H_1}\right) + S\_G\left[\lfloor h\_s\left(\frac{W_C}{H_1}\right) + \frac{1}{2} \rfloor - h\_s\left(\frac{W_C}{H_1}\right) \right] \right\} - H_1. \tag{6}$$

As for Binary_D, we propose the function Binary_k_D (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.

$$\text{Binary\_}k_D(W_D, H_2) = 2 * H_2 * \left\{ h\_s\left(\frac{W_D}{H_2}\right) + S\_G\left[\lfloor h\_s\left(\frac{W_D}{H_2}\right) + \frac{1}{2} \rfloor - h\_s\left(\frac{W_D}{H_2}\right) \right] \right\} - H_2. \tag{7}$$

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.

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

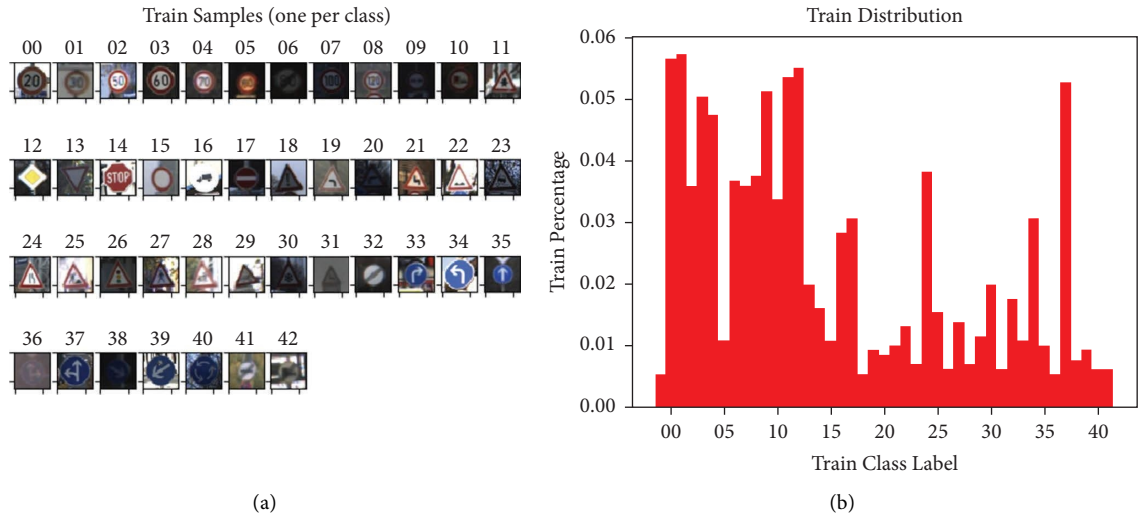


Figure 4: (a) Number of training set classes. (b) Number of training set quantity distribution of each class.

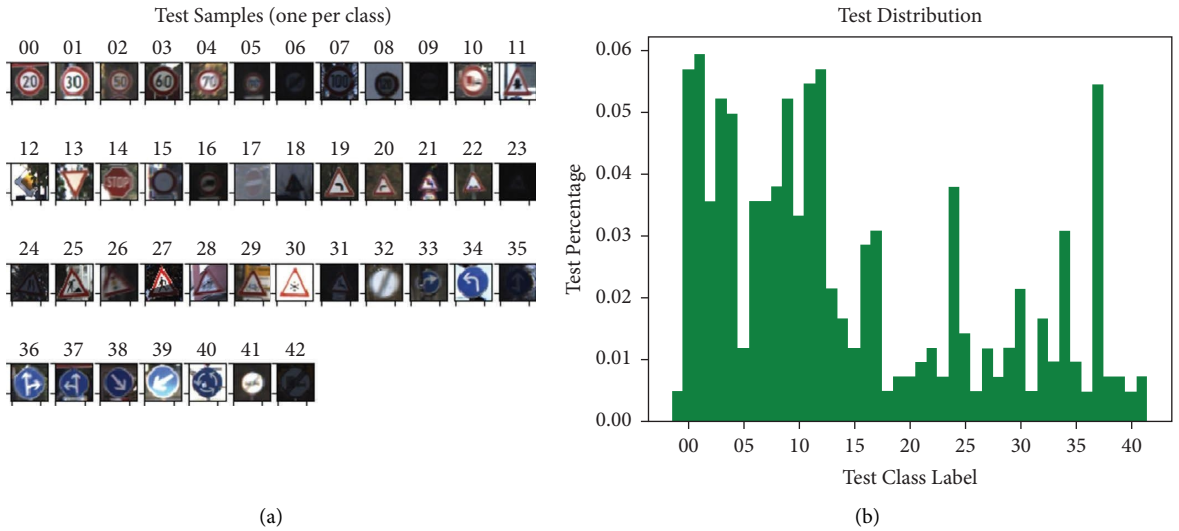


Figure 5: (a) Number of validation set classes. (b) Number of validation set quantity distribution of each class.

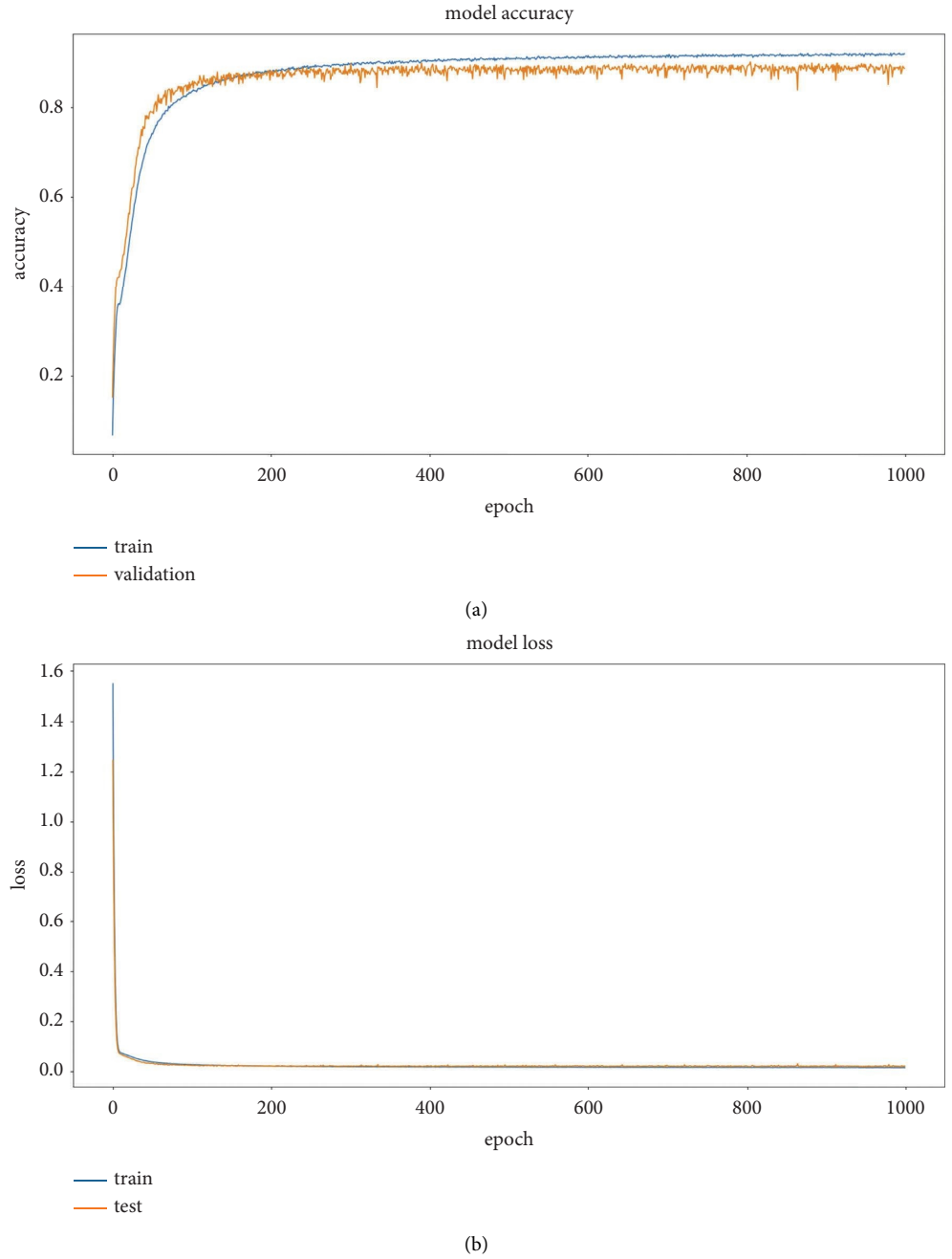


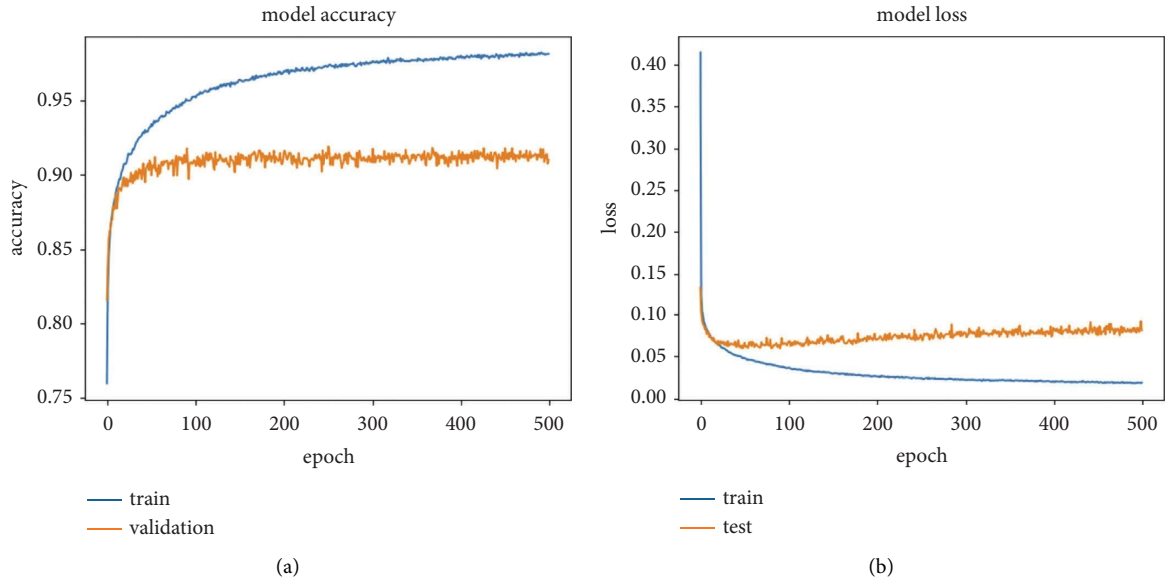
Figure 6: (a) Accuracy of CBCNN training and validation sets on GTSRB. (b) The loss of CBCNN on GTSRB.

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

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

Architecture	Accuracy (%)	Params (M)	Search methods
ResNeXt-8-64 + random erasing [43]	96.2 ± 0.06	34.4	Manual
ResNet-110 + random erasing [43]	95.9 ± 0.13	1.7	Manual
VGG8B [44]	95.47	7.3	Manual
DeepCaps [45]	94.46	7.2	Manual
WRN-28-10 + random erasing [43]	96.3 ± 0.03	36.5	Manual
DARTS (2nd order) + cutout + random erasing [47]	96.57	2.6	Gradient-based
Fine-tuning DARTS [48]	96.91	3.2	Gradient-based
Neupde [46]	92.40	0.4	Manual
CBCNN (ours, bitwise operation)	92.86	0.48	Gradient-based



(a) (b)

Figure 7: (a) Accuracy of CBCNN training and validation sets on Fashion-MNIST. (b) The loss of CBCNN on Fashion-MNIST.

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 × 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 3: The accuracy performance of different methods is compared on the MNIST dataset.

Methods	Accuracy (%)	Params (M)
SOPCNN [49]	99.83	1.4
No routing needed [50]	99.87	1.5
Efficient-CapsNet [51]	99.84	0.16
BNN (Torch7) [3]	98.60	—
BNN (Theano) [3]	99.04	—
CBCNN (ours, bitwise operation)	99.32	0.48

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.

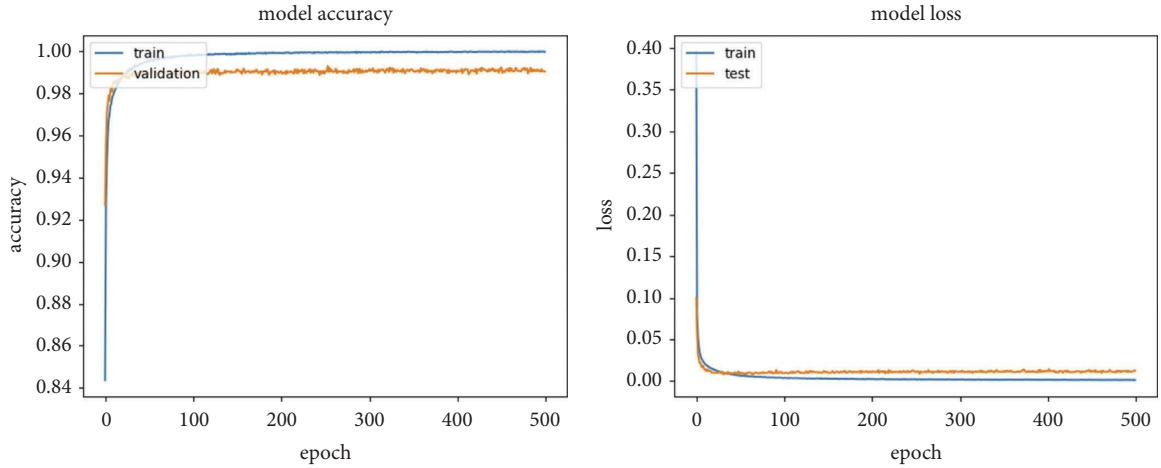


Figure 8: (a) Accuracy of CBCNN training and validation sets on MNIST. (b) The loss of CBCNN on MNIST.

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 3×3 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 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

[1] S. Xie, R. B. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5987–5995, Honolulu, Hawaii, July 2017.

[2] X. Li, W. Wang, X. Hu, and J. Yang, "Selective kernel networks," in *Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 510–519, Long Beach, CA, USA, June 2019.

[3] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized Neural Networks: Training Deep Neural Networks with Weights and Activations Constrained to +1 or −1," 2016, https://arxiv.org/abs/1602.02830.

[4] L. Hou, Q. Yao, and J. T.-Y. Kwok, "Loss-aware binarization of deep networks," 2017, https://arxiv.org/abs/1611.01600.

[5] Z. Wang, J. Lu, C. Tao, J. Zhou, and Q. Tian, "Learning channel-wise interactions for binary convolutional neural networks," in *Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 568–577, Long Beach, CA, USA, June 2019.

[6] K. Han, Y. Wang, and Y. Xu, "Training Binary Neural Networks through Learning with Noisy Supervision," 2020, https://arxiv.org/abs/2010.04871.

[7] X. He, Z. Mo, and K. Cheng, "Proxybnn: Learning Binarized Neural Networks via Proxy Matrices," *Computer Vision – ECCV*, vol. 12348, 2020.

[8] H. Qin, R. Gong, and X. Liu, "Forward and backward information retention for accurate binary neural networks," in *Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2247–2256, Washington, DC, USA, June 2020.

[9] M. Lin, R. Ji, and Z.-H. Xu, "Rotated binary neural network," 2020, https://arxiv.org/abs/2009.13055.

[10] H. Kim, J. Park, C.-H. Lee, and J.-J. Kim, "Improving accuracy of binary neural networks using unbalanced activation distribution," in *Proceedings of the 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7858–7867, Virtual, June 2021.

[11] Z. Yang, Y. Wang, and K. Han, "Searching for Low-Bit Weights in Quantized Neural Networks," 2020, https://arxiv.org/abs/2009.08695.

[12] M. Shen, X. Liu, and K. Han, "Balanced binary neural networks with gated residual," in *Proceedings of the ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4197–4201, Barcelona, Spain, May 2020.

[13] B. Martínez, J. Yang, A. Bulat, and G. Tzimiropoulos, "Training binary neural networks with real-to-binary convolutions," 2020, https://arxiv.org/abs/2003.11535.

[14] Y. Li, R. Gong, F. Yu, X. Dong, and X. Liu, "Dms: differentiable dimension search for binary neural networks," 2020, https://arxiv.org/pdf/2103.13630.pdf.

[15] A. Bulat, B. Martínez, and G. Tzimiropoulos, "Bats: Binary Architecture Search," 2020, https://arxiv.org/abs/2003.01711.

[16] K. P. Singh, D. Kim, and J. Choi, "Learning Architectures for Binary Networks," 2020, https://arxiv.org/abs/2002.06963.

[17] B. Zhu, Z. Al-Ars, and H. P. Hofstee, "Nasb: neural architecture search for binary convolutional neural networks," in *Proceedings of the 2020 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, Glasgow, UK, July 2020.

[18] A. Bulat, B. Martínez, and G. Tzimiropoulos, *High-capacity Expert Binary Networks*, 2021, https://arxiv.org/abs/2010.03558.

[19] H. T. Phan, D. T. Huynh, Y. He, M. Savvides, and Z. Shen, "Mobinet: a mobile binary network for image classification," in *Proceedings of the 2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 3442–3451, Snowmass Village, CO, USA, March 2020.

[20] H. T. Phan, Z. Liu, and D. T. Huynh, "Binarizing mobilenet via evolution-based searching," in *Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13417–13426, Washington, DC, USA, June 2020.

[21] J. Bethge, C. Bartz, H. Yang, Y. Chen, and C. Meinel, "Meliusnet: Can Binary Neural Networks Achieve Mobilenet-Level Accuracy?," 2020, https://arxiv.org/abs/2001.05936.

[22] Z. Liu, Z. Shen, M. Savvides, and K.-T. Cheng, "Reactnet: Towards Precise Binary Neural Network with Generalized Activation Functions," 2020, https://arxiv.org/abs/2003.03488.

[23] T. Chen, Z. A. Zhang, and X. Ouyang, "Bnn - bn = ?": training binary neural networks without batch normalization," in *Proceedings of the 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 4614–4624, Virtual, June 2021.

[24] A. Brock, S. De, S. L. Smith, and K. Simonyan, "High-performance large-scale image recognition without normalization," 2021, https://arxiv.org/abs/2102.06171.

[25] Y. Zhang, J. Pan, and X. Liu, "Fracbnn: accurate and fpga-efficient binary neural networks with fractional activations," 2021, https://arxiv.org/abs/2012.12206.

[26] A. J. Redfern, L. Zhu, and M. K. Newquist, "Bcnn: a binary cnn with all matrix ops quantized to 1 bit precision," 2021, https://arxiv.org/abs/2010.00704.

[27] Y. Xu, X. Dong, Y. Li, and H. Su, "A main/subsidiary network framework for simplifying binary neural networks," 2019, https://arxiv.org/abs/1812.04210.

[28] K. Helwegen, J. Y. Widdicombe, and L. Geiger, "Latent weights do not exist: rethinking binarized neural network optimization," 2019, https://arxiv.org/abs/1906.02107.

[29] P. Pham, J. A. Abraham, and J. Chung, "Training multi-bit quantized and binarized networks with a learnable symmetric quantizer," *IEEE Access*, vol. 9, pp. 47194–47203, 2021.

[30] J. Frankle and M. Carbin, "The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks," 2019, https://arxiv.org/abs/1803.03635.

[31] J. Diffenderfer and B. Kailkhura, "Multi-prize lottery ticket hypothesis: finding accurate binary neural networks by pruning a randomly weighted network," 2021, https://arxiv.org/abs/2103.09377.

[32] A. Livochka and A. Shekhovtsov, "Initialization and transfer learning of stochastic binary networks from real-valued ones," in *Proceedings of the 2021 IEEE/CVF Conference on Computer*

Vision and Pattern Recognition Workshops (CVPRW), pp. 4655–4663, Nashville, TN, USA, June 2021.

[33] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "The German traffic sign recognition benchmark: a multi-class classification competition," in *Proceedings of the The 2011 International Joint Conference on Neural Networks*, pp. 1453–1460, San Jose, CA, USA, August 2011.

[34] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," 2017, https://arxiv.org/abs/1708.07747.

[35] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[36] T. Wei, X. Chen, and Y. Yin, *Research on Traffic Sign Recognition Method Based on Multi-Scale Convolution Neural Network*, Xibei Gongye Daxue Xuebao/Journal of Northwestern Polytechnical University, Fremont, CA, USA, 2021.

[37] D. C. Ciresan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," 2012, https://arxiv.org/abs/1202.2745.

[38] S. Natarajan, A. K. Annamraju, and C. S. Baradkar, "Traffic Sign Recognition Using Weighted Multi-convolutional Neural Network," *IET Intelligent Transport Systems*, Wiley, Hoboken, NJ, USA, 2018.

[39] I. Freeman, L. Roese-Koerner, and A. Kummert, "Effnet: an efficient structure for convolutional neural networks," in *Proceedings of the 2018 25th IEEE International Conference on Image Processing (ICIP)*, pp. 6–10, Athens, Greece, October 2018.

[40] Z. Sun, X. Qi, and X. Mei, "Dcr: dual compression method for traffic signs recognition and embedded deployment," in *Proceedings of the 2022 7th International Conference on Computational Intelligence and Applications (ICCIA)*, pp. 195–199, Nanjing, China, June 2022.

[41] P. Cheng, W. Liu, Y. Zhang, and H. Ma, "Loco: Local Context Based Faster R-Cnn for Small Traffic Sign Detection," *MultiMedia Modeling*, vol. 10704, 2018.

[42] L. Kai, H. Bing, and Z. Jingtao, "Traffic sign detection and recognition based on conditional random field and multi-scale convolutional neural network," *Computer Applications*, vol. 38, no. 2, pp. 270–275, 2018.

[43] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang, "Random Erasing Data Augmentation," 2020, https://arxiv.org/abs/1708.04896.

[44] A. Nøkland and L. H. Eidnes, "Training Neural Networks with Local Error Signals," 2019, https://arxiv.org/abs/1901.06656.

[45] J. Rajasegaran, V. Jayasundara, S. Jayasekara, H. Jayasekara, S. Seneviratne, and R. Rodrigo, "Deepcaps: going deeper with capsule networks," 2019, https://arxiv.org/abs/1904.09546.

[46] Y. Sun, L. Zhang, and H. Schaeffer, "Neupde: Neural Network Based Ordinary and Partial Differential Equations for Modeling Time-dependent Data," 2020, https://arxiv.org/abs/1908.03190.

[47] H. Liu, K. Simonyan, and Y. Yang, "Darts: differentiable architecture search," 2019, https://arxiv.org/abs/1806.09055.

[48] M. Tanveer, M. U. K. Khan, and C. M. Kyung, "Fine-tuning darts for image classification," 2021, https://arxiv.org/abs/2006.09042.

[49] Y. S. Assiri, "Stochastic Optimization of plain Convolutional Neural Networks with Simple Methods," 2019, https://arxiv.org/abs/2001.08856.

[50] A. Byerly, T. Kalganova, and I. D. Dear, "No routing needed between capsules," *Neurocomputing*, vol. 463, pp. 545–553, 2021.

[51] V. Mazzia, F. Salvetti, and M. Chiaberge, "Efficient-capsnet: capsule network with self-attention routing," *Scientific Reports*, vol. 11, no. 1, p. 14634, 2021.