Accurate and Compact Convolutional Neural Networks with Trained Binarization

Zhe Xu
zhexu22-c@my.cityu.edu.hk
Ray C.C. Cheung
r.cheung@cityu.edu.hk

Department of Electrical Engineering
City University of Hong Kong
Hong Kong SAR, China

Abstract

Although convolutional neural networks (CNNs) are now widely used in various computer vision applications, its huge resource demanding on parameter storage and computation makes the deployment on mobile and embedded devices difficult. Recently, binary convolutional neural networks are explored to help alleviate this issue by quantizing both weights and activations with only 1 single bit. However, there may exist a noticeable accuracy degradation when compared with full-precision models. In this paper, we propose an improved training approach towards compact binary CNNs with higher accuracy. Trainable scaling factors for both weights and activations are introduced to increase the value range. These scaling factors will be trained jointly with other parameters via backpropagation. Besides, a specific training algorithm is developed including tight approximation for derivative of discontinuous binarization function and L_2 regularization acting on weight scaling factors. With these improvements, the binary CNN achieves 92.3% accuracy on CIFAR-10 with VGG-Small network. On ImageNet, our method also obtains 46.1% top-1 accuracy with AlexNet and 54.2% with Resnet-18 surpassing previous works.

1 Introduction

Convolutional neural networks (CNNs) have achieved great success in a wide range of real-world applications such as image classification [10, 17, 27], object detection [8, 21, 25], visual relation detection [31, 32] and image style transfer [3, 7] in recent years. However, the powerful CNNs are also accompanied by the large model size and high computational complexity. For example, VGG-16 [27] requires over 500MB memory for parameter storage and 31GFLOP for a single image inference. This high resource demanding makes it difficult to deploy on mobile and embedded devices.

To alleviate this issue, several kinds of approaches have been developed to compress network and reduce computational cost. The first approach is to design compact network architectures with similar performance. For example, SqueezeNet [15] utilized 1x1 convolution layer to squeeze the output channel number before computationally expensive 3x3 convolution operations. More recently, MobileNet [12] and ShuffleNet [33] introduced efficient depth-wise and group convolution to replace the normal complex convolution operation. The second approach is to reduce or compress network parameters. [18] and [23] achieved to compress network weights using tensor decomposition. Connection pruning was employed in [9] to reduce parameters by up to 13 times for AlexNet and VGG-16. The third category is to quantize network parameters presented in [5, 11, 20, 34, 36, 37]. Network quantization can reduce memory requirement efficiently because parameter values are represented with less bits. At the same time, it can alleviate the computational cost issue since floating-point calculations are transferred into fixed-point calculations with less computation resources.

Furthermore, as an extreme case of parameter quantization, binary convolutional neural networks quantize both weights and activations with 1 bit [14, 24]. It has attracted large research interests because binary CNNs can reduce network size by 32 times compared with full precision and replace the multiplication with bitwise logic operations. As a result, binary CNNs are suitable for accelerator implementations on hardware platforms such as FPGA [29]. However, network binarization may decrease accuracy noticeably due to extremely limited precision. It is still a challenge and needs further exploration towards better inference performance.

In this paper, an improved training approach for binary CNNs is proposed which is easy to be implemented on hardware platforms. Our approach includes three novel ideas:

1. Trainable scaling factors are introduced for weight and activation binarization. Previous works such as XNOR-Net [24] set the mean value of weights as the scaling factor, however it results in minimum quantization error but cannot ensure the best inference accuracy. Instead, we employ the trainable scaling factors for both weights and activations and update them jointly with other network parameters.

2. Derivative approximation is discussed for binary network training. Since the derivative of binarization function is like an impulse function, it is not suitable for backpropagation. We propose to use a higher order approximation for weight binarization and a long-tailed approximation for activation binarization as a trade-off between tight approximation and smooth backpropagation.

3. The L_2 regularization term is now acting on the weight scaling factors directly. In our approach, weight scaling factors represent the actual binary filters, the L_2 regularization should be modified accordingly for better generalization capability.

The proposed binary network approach achieves better inference accuracy and faster convergence speed. The rest of this paper is organized as follows. Section 2 reviews previous related works. The proposed binary network training approach is introduced in detail in Section 3. Experimental results are provided in Section 4. Finally, Section 5 gives the conclusion.

2 Related Work

Previous works [5, 11, 20, 34, 36, 37] already demonstrated that quantization can reduce much memory resources and computational complexity for various CNN structures. One extreme case of quantization is to constrain real value with only 1 bit, *i.e*. binarization. It can be further divided into two subcategories: one is only binarizing weights and leaving activations full-precision or quantized, another is binarizing both weight and activation values.

Weight-only binarization methods: Courbariaux *et al*. [4] firstly proposed to train networks constraining weights to only two possible values, -1 and +1. They introduced a training method, called BinaryConnect, to deal with binary weights in both forward and backward propagations and obtained promising results on small datasets. In [2], Cai *et al*. binarized weights and proposed a half-wave Gaussian quantizer for activations. The proposed quantizer exploited the statistics of activations and was efficient for low-bit quantization. Later Wang *et al*. [30] further extended [2]'s idea with sparse quantization. Besides, they proposed a two-step quantization framework: code learning and transformation function learning. In code learning step, weights were of full-precision and activations were quantized based on Gaussian distribution and sparse constraint. Then the weights binarization was solved as a non-linear least regression problem in the second step. In [13], Hu *et al*. proposed to transfer binary weight networks training problem into a hashing problem. This hashing problem was solved with alternating optimization algorithm and the binary weight network was then fine-tuned to improve accuracy.

Complete binarization methods: Although weight-only binarization methods already save much memory resources and reduce multiplications, further binarization on activation can transfer arithmetic to bit-wise logic operations enabling fast inference on embedded devices. As far as our knowledge goes, [14] was the first work binarizing both weights and activations to -1 and +1. The work obtained 32 times compression ratio on weights and 7 times faster inference speed with comparative results to BinaryConnect on small datasets like CIFAR-10 and SVHN. However, later results showed that this training method was not suitable for large datasets with obvious accuracy degradation. Later Rastegari *et al*. [24] proposed XNOR-Net to improve the inference performance on large-scale datasets. It achieved better trade-off between compression ratio and accuracy in which scaling factors for both weights and activations were used to minimize the quantization errors. DoReFa-Net [35] proposed by Zhou *et al*. inherited the idea of XNOR-Net and provided a complete solution for low-bit quantization and binarization of weights, activations and gradients.

In [28], Tang *et al*. explored several strategies to train binary neural networks. The strategies included setting small learning rate for binary neural network training, introducing scaling factors for weights using PReLU activation function and utilizing regularizer to constraint weights close to +1 or -1. They showed the binary neural networks achieved similar accuracy to XNOR-Net with simpler training procedure. In [19], Lin *et al*. proposed ABC-Net towards accurate binary convolutional neural network. Unlike other approaches, they proposed to use the linear combination of multiple binary weights and binary activations for better approximation of full-precision weights and activations. With adequate bases, the structure could finally achieve close results to full-precision networks. In recent work [22], Liu *et al*. proposed Bi-Real Net in which the real activations are added to the binary activations through a shortcut connection to increase the representational capability. Moreover, they provided a tight approximation of sign function and proposed to pre-train full-precision neural networks as the initialization for binary network training.

3 Proposed Binary Network Training

Training an accurate binary CNN has two major challenges: one is the extremely limited value range because of binary data, another is the difficult backpropagation in training procedure caused by the derivative of binarization function. In this section, the proposed binary CNN training approach is introduced in order to address the above two issues.

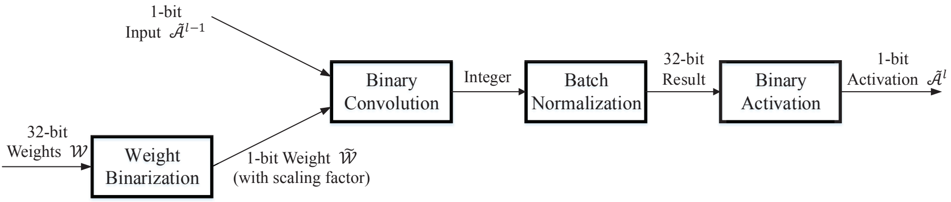


Figure 1: Forward process in binary convolution layer.

3.1 Binarization with Trainable Scaling Factors

We first briefly review the binarization operation in one convolution layer, which is shown in Figure 1. As binary weighs are difficult to update with gradient-based optimization methods due to the huge gap between different values, it is common to reserve full-precision weights during training process. The binary weights are then obtained from real values via the binarization function. The input of the convolution operation is actually the activation output $\tilde{\mathcal{A}}$ of the previous layer, the binary convolution is represented as

$$z = \tilde{\mathcal{A}} * \tilde{\mathcal{W}} \tag{1}$$

Since the input $\tilde{\mathcal{A}}$ and weights $\tilde{\mathcal{W}}$ are both binary, the convolution operation can be implemented with bitwise logic and *popcnt* operations to get integer results z similar as [24]. After batch normalization, the integer results z become real values within a certain range and they are binarized in the activation step to generate binary output feature map for the next convolution layer.

We use a sign function $sgn(x)$ and a unit step function $H(x)$ for weight and activation binarization, respectively. $sgn(x)$ and $H(x)$ are

$$sgn(x) = \begin{cases} 1, & x \geq 0 \\ -1, & x < 0 \end{cases} \qquad H(x) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases} \tag{2}$$

However, $sgn(x)$ and $H(x)$ highly restrict the results to be only three possible values $\{-1, 0, +1\}$. To increase the value range, [24] and [35] proposed to use the average of absolute values of each output channel as scaling factors for weights. This minimizes the binarization errors but does not ensure the optimal inference performance. Instead we propose to set trainable scaling factors directly and these scaling factors are updated through backpropagation as part of the network parameters.

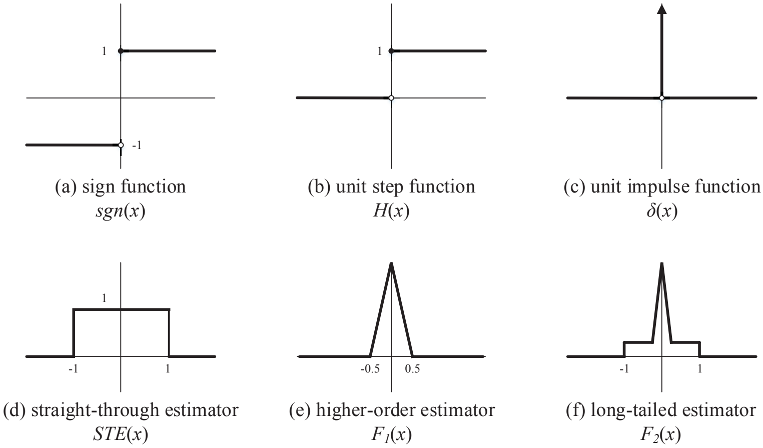
Given the real value weight filter $\mathcal{W} \in \mathbb{R}^{c_o \times c_i \times k \times k}$ where c_o and c_i stand for output and input channels respectively, we set a scaling factor $\alpha \in \mathbb{R}^{c_o}$. Then the weight binarization is represented as

$$\tilde{\mathcal{W}}_i = \alpha_i \cdot sgn(\mathcal{W}_i) \tag{3}$$

$\tilde{\mathcal{W}}_i$ stands for the binary weight at output channel i. Importing the scaling factor α enables binary weights $\tilde{\mathcal{W}}$ to have different value magnitudes in each output channel.

Similarly, we set the scaling factor β for binary activation $\tilde{\mathcal{A}} \in \mathbb{R}^{c_o \times w \times h}$

$$\tilde{\mathcal{A}}_i = \beta \cdot H(\mathcal{A}_i - \tau_i) \tag{4}$$



(a) sign function $sgn(x)$

(b) unit step function $H(x)$

(c) unit impulse function $\delta(x)$

(d) straight-through estimator $STE(x)$

(e) higher-order estimator $F_1(x)$

(f) long-tailed estimator $F_2(x)$

Figure 2: Different approximations for derivatives of $sgn(x)$ and $H(x)$.

We set a threshold $\tau \in \mathbb{R}^{c_o}$ as a shift parameter trying to extract more information from the original full-precision $\mathcal{A}$. By importing parameter τ, only important values will be activated and other small values are set to be zero. It should be noted that the shift parameter τ will also be updated during network training. β is identical for a single activation layer to make it compatible with hardware implementation which will be discussed in Section 3.4.

3.2 Training Algorithm

As we discussed before, $sgn(x)$ and $H(x)$ are used to binarize weights and activations, respectively. They are illustrated in Figure 2(a) and 2(b). To update network parameters, their derivatives are required. However as we can see in Figure 2(c), their derivatives are like an impulse function whose value is zero almost everywhere and infinite at zero. Thus it cannot be applied for backpropagation and parameter updating directly. In [14, 24, 35], a clip function was employed to approximate the binarization functions. The derivate of $sgn(x)$ is represented in Eq. 5 based on straight through estimator (STE) [1] and it is illustrated in Figure 2(d).

$$\frac{d}{dw} sgn(x) \approx \mathbf{1}_{\{|w| \leq 1\}} \tag{5}$$

Instead of Eq. 5, [22] utilized a piecewise polynomial function as a tighter approximation. In this paper, we further develop this higher-order estimator approach. For $sgn(x)$ used in weight binarization, we give derivative approximation $F_1(x)$ whose active range is between $[-0.5, 0.5]$ illustrated in Figure 2(e). It is a piecewise linear function.

$$\frac{d}{dw} sgn(x) \approx F_1(x) = \begin{cases} 4 - 8|x|, & -0.5 \leq x \leq 0.5 \\ 0, & otherwise \end{cases} \tag{6}$$

For the derivative of $H(x)$, an overtight approximation may not be an optimal choice because it will affect the backpropagation to shallow layers far away from the network output. For an overtight approximation, the gradient value will be near zero in a large scope resulting in gradient vanishing. To alleviate this problem, we use a long-tailed higher-order estimator $F_2(x)$ whose active range is between $[-1,1]$, shown in Figure 2(f). It is a piecewise function with tight approximation near zero and small constant value in a wider range.

$$\frac{d}{dw}H(x) \approx F_2(x) = \begin{cases} 2-4|x|, & -0.4 \le x \le 0.4 \\ 0.4, & 0.4 < |x| \le 1 \\ 0, & otherwise \end{cases} \tag{7}$$

Based on Eq. 6 and 7,the backpropagation can be performed smoothly during binary network training. It should be noted that there certainly exist other estimators as a trade-off between tight approximation and smooth backpropagation, such as Swish-like function [6]. In this section, we provide a simple yet efficient approximation and leave other options as our future work.

3.3 Regularization on Scaling Factors

Since deep CNNs usually have a tremendous parameter set, a good regularization term is necessary during training process for robust generalization capability. L_2 regularization, also called ridge regression, is widely used in full-precision networks. It uses squared magnitude of weights as penalty term to help avoid over-fitting issue.

In our binary network training approach, weight scaling factors, α, stand for the actual binary filters for feature extraction. Thus, the L_2 regularization term is modified to restrict the scaling factors accordingly. The total loss function is then represented as

$$J(\alpha^l,\gamma) = L(\alpha^l,\gamma) + \frac{\lambda}{2}\sum_l ||\alpha^l||_2^2 \tag{8}$$

where $L(\alpha^l,\gamma)$ is the task-related loss such as cross entropy loss. α^l is the weight scaling factors in which the superscript l stands for different layers. γ stands for other parameters adopted in CNN structure. The second term $\frac{\lambda}{2}\sum ||\alpha^l||_2^2$ is the new L_2 regularization with weight decay parameter λ. The regularization term tends to decrease the magnitude of scaling factor α^l.

3.4 Compatibility with Hardware Implementation

In this section we show that our binary network can be easily implemented on hardware platforms via bitwise operations. To simplify the discussion, we take 3x3 convolution as an example and let the input channel to be 1. $\tilde{\mathcal{W}}_i \in \mathbb{R}^{3\times3}$ is the convolution kernel where i indicates the output channel, the input of convolution is actually the binary activation $\tilde{\mathcal{A}}$ of the previous layer. The 3×3 convolution operation is

$$\begin{aligned} z &= \tilde{\mathcal{A}} * \tilde{\mathcal{W}}_i = \alpha_i\beta(\mathcal{B}^a * \mathcal{B}^w_i) \\ \mathcal{B}^a &= H(\mathcal{A}-\tau) \quad \in \{0,1\} \\ \mathcal{B}^w_i &= sgn(\mathcal{W}_i) \quad \in \{-1,1\} \end{aligned} \tag{9}$$

where α_i and β are trained scaling factors for weights and activations respectively. For the same output channel, $\alpha_i\beta$ is a constant so it can be integrated into the following batch normalization.

Table 1: Truth Table of Binary Multiplication $b^a \times b_i^w$.

b^a	b_i^w	*pos*	*neg*
0	0	0	0
0	1	0	0
1	0	0	1
1	1	1	0

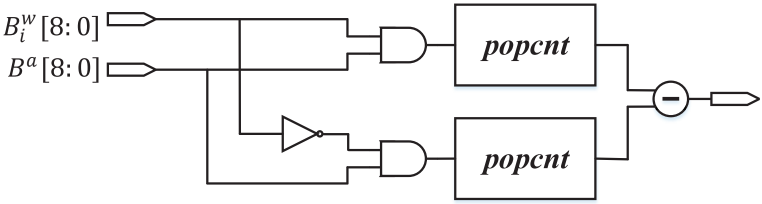


Figure 3: Hardware architecture of the binary convolution.

$\mathcal{B}^a$ and $\mathcal{B}_i^w$ are both binary values thus $\mathcal{B}^a * \mathcal{B}_i^w$ can be implemented with bitwise operations. In each bit, the result of basic multiplication $b^a \times b_i^w$ is one of three values $\{-1, 0, 1\}$. If we let binary "0" stands for the value -1 in b_i^w, the truth table of binary multiplication is then shown in Table 1, in which *pos* means the result is $+1$ and *neg* means the result is -1. It is easy to see that

$$pos = \mathcal{B}^a \& \mathcal{B}_i^w \qquad\qquad neg = \mathcal{B}^a \& (\overline{\mathcal{B}_i^w}) \tag{10}$$

With *popcnt* operation counting number of "1" in a binary sequence, we can get the binary convolution

$$\begin{aligned} \mathcal{B}^a * \mathcal{B}_i^w &= popcnt(pos) - popcnt(neg) \\ &= popcnt(\mathcal{B}^a \& \mathcal{B}_i^w) - popcnt(\mathcal{B}^a \& (\overline{\mathcal{B}_i^w})) \end{aligned} \tag{11}$$

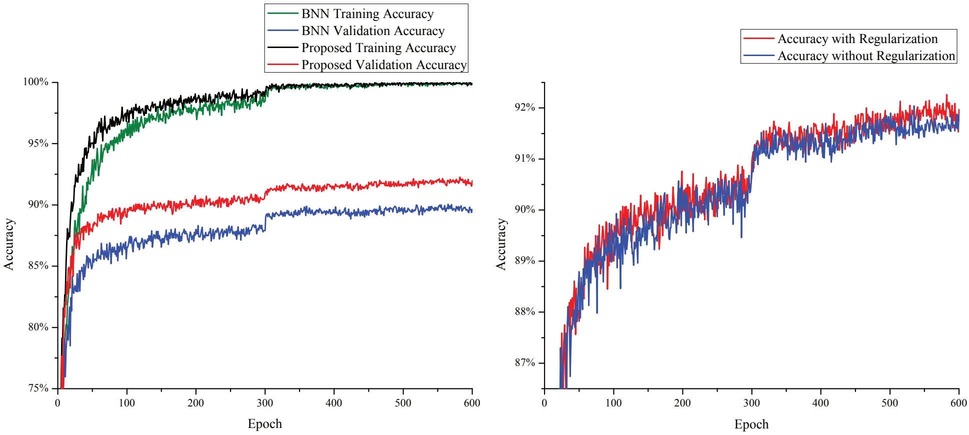
Figure 3 shows the hardware architecture of binary convolution based on Eq. 11. As we can see, for a 3×3 convolution with 9 multiplications and 8 additions, the binary calculation only requires 2 *popcnt* operations and 1 subtraction.

4 Experimental Results

In this section, the performance of our proposed method is evaluated on image classification task. The performance of other tasks such as object detection and pose estimation will be included in the future work. Two image classification datasets: CIFAR-10 [16] and ImageNet ILSVRC12 [26] are used for evaluation. The CIFAR-10 dataset consists of 50,000 training images with size 32×32 belonging to 10 classes and 10,000 testing images. The large ImageNet ILSVRC12 consists of 1000 classes with about 1.2 million images for training and 50,000 images for validation. We build the same VGG-Small network as [4, 14, 24] for evaluation on CIFAR-10 dataset. For ImageNet, we implement AlexNet [17] and Resnet-18 [10] networks. All networks are trained from random initialization without pre-training. Following previous works [14, 24, 35], we binarize all the convolution and fully-connected layers

Table 2: Results Comparison of VGG-Small on CIFAR-10.

Method	Bit-width (W/A)	Accuracy
Ours	1/1	**92.3%**
BNN [14]	1/1	89.9%
XNOR-Net [24]	1/1	89.8%
HWGQ [2]	1/2	92.5%
Full-Precision	32/32	93.6%



(a) Results of our approach versus BNN [14].

(b) Results with and without regularization.

Figure 4: Training curves comparison of VGG-Small on CIFAR-10.

except the first and the last layer. For regularization, we set the weight decay parameter λ to be 10^{-6}. [14] and [24] pointed out that ADAM converges faster and usually performs better for binary inputs. Thus we use ADAM optimization for parameter updating with an initial learning rate 10^{-3} and 2×10^{-4} for CIFAR-10 and ImageNet, respectively.

4.1 Performance on CIFAR-10

Table 2 presents the results of the VGG-Small network on CIFAR-10 dataset. The second column *Bit-width* denotes quantization bits for weights and activations. The third column shows the accuracy results. Two binary network approaches, BNN [14] and XNOR-Net [24], and one low-precision quantization approach, HWGQ [2], are selected for comparison. The result of full-precision network model is also presented as a reference. The proposed training approach achieves 92.3% accuracy on CIFAR-10, exceeding BNN and XNOR-Net by 2.4% and 2.5%, respectively. Moreover, our result on binary network is very close to HWGQ [2] with 2-bit activation quantization.

To further evaluate the proposed method, we show the training curves in Figure 4. First in Figure 4(a), training curves of BNN [14] and our approach are compared. Although they have similar training accuracy, our approach is faster to train and improves the accuracy a lot on validation set. Moreover, it can be observed that our validation accuracy curve is more stable than BNN due to the regularization. The effectiveness of regularization is then validated in Figure 4(b). The red curve is the result with our new L_2 regularization. It has

Table 3: Results Comparisons of AlexNet and Resnet-18 on ImageNet.

Method	Bit-width (W/A)	AlexNet Accuracy		Resnet-18 Accuracy	
		Top-1	Top-5	Top-1	Top-5
Ours	1/1	**46.1%**	**70.9%**	**54.2%**	**77.9%**
XNOR-Net [24]	1/1	44.2%	69.2%	51.2%	73.2%
BinaryNet [28]	1/1	41.2%	65.6%	42.2%	67.1%
ABC-Net [19]	1/1	-	-	42.7%	67.6%
DoReFa-Net [35]	1/1	43.6%	-	-	-
Full-Precision	32/32	56.6%	80.2%	69.6%	89.2%

Table 4: Network Model Size Comparison.

Network	Full Model Size	Binary Model Size	Compression Ratio
VGG-Small	53.52MB	1.75MB	30.6×
AlexNet	237.99MB	22.77MB	10.5×
Resnet-18	49.83MB	3.51MB	14.2×

less fluctuation relatively and the regularization term does help to improve the accuracy from 92.0% to 92.3% indicating better generalization.

4.2 Evaluations and Comparisons on ImageNet

On large ImageNet ILSVRC12 dataset, we test the accuracy performance of the proposed approach for AlexNet and Resnet-18 networks. The experimental results are presented in Table 3. We select four existing methods for comparison including XNOR-Net [24], BinaryNet [28], ABC-Net [19] and DoReFa-Net [35]. It should be noted that the Resnet-18 accuracy results of BinaryNet [28] are quoted from [19]. Some results of ABC-Net [19] and DoReFa-Net [35] are not provided so we leave them blank. Similarly, the results of full-precision network are provided in Table 3 as a reference.

For AlexNet, our approach achieves 46.1% top-1 accuracy and 70.9% top-5 accuracy. It is the best result among five binary network solutions and surpasses other works by up to 4.9% and 5.3%, respectively. For Resnet-18, our method obtains 54.2% top-1 accuracy and 77.9% top-5 accuracy, improving the performance by up to 12.0% and 10.8% compared with other works. It is also shown that the proposed approach succeeds to reduce the accuracy gap between full-precision and binary networks to about 10%. These indicates our approach can improve the inference performance of binary convolutional neural networks effectively.

4.3 Analysis of Network Model Size

Ideally, binary neural network should achieve 32× compression ratio compared with full-precision model. But in our approach the first and the last layer are excluded from binarization operation following [14, 24, 35]. Besides, batch normalization parameters and scaling factors should be in full precision for better network representation capability. These will affect the actual network compression ratio. Table 4 shows the actual parameter size comparison of three network structures. For a typical network, binarization can reduce model size by over 10 times. Besides, it can also be observed that with a better network structure, the binary network can achieve better performance in terms of model compression ratio.

5 Conclusion

This paper proposes an approach to train binary CNNs with higher inference accuracy including three ideas. First, trainable scaling factors for both weights and activations are employed to provide different value ranges of binary number. Then, higher-order estimator and long-tailed estimator for derivative of binarization function are proposed to balance the tight approximation and efficient backpropagation. At last, the L_2 regularization is performed directly on weight scaling factors for better generalization capability. The approach is effective to improve network accuracy and it is suitable for hardware implementation.

References

[1] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

[2] Zhaowei Cai, Xiaodong He, Jian Sun, and Nuno Vasconcelos. Deep learning with low precision by half-wave gaussian quantization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5918–5926, 2017.

[3] Dongdong Chen, Lu Yuan, Jing Liao, Nenghai Yu, and Gang Hua. Stylebank: An explicit representation for neural image style transfer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1897–1906, 2017.

[4] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. In *Advances in neural information processing systems*, pages 3123–3131. 2015.

[5] Yinpeng Dong, Renkun Ni, Jianguo Li, Yurong Chen, Jun Zhu, and Hang Su. Learning accurate low-bit deep neural networks with stochastic quantization. In *British machine vision conference (BMVC)*, 2017.

[6] Stefan Elfwing, Eiji Uchibe, and Kenji Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural Networks*, 107:3–11, 2018.

[7] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2414–2423, 2016.

[8] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 580–587, 2014.

[9] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*, pages 1135–1143. 2015.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 770–778, 2016.

[11] Lu Hou and James T Kwok. Loss-aware weight quantization of deep networks. In *International conference on learning representations (ICLR)*, 2018.

[12] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[13] Qinghao Hu, Peisong Wang, and Jian Cheng. From hashing to cnns: Training binary weight networks via hashing. In *Thirty-Second AAAI Conference on Artificial Intelligence (AAAI)*, 2018.

[14] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks. In *Advances in neural information processing systems*, pages 4107–4115. 2016.

[15] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

[16] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105. 2012.

[18] Vadim Lebedev, Yaroslav Ganin, Maksim Rakhuba, Ivan Oseledets, and Victor Lempitsky. Speeding-up convolutional neural networks using fine-tuned cp-decomposition. 2015.

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

[20] Zhouhan Lin, Matthieu Courbariaux, Roland Memisevic, and Yoshua Bengio. Neural networks with few multiplications. In *International conference on learning representations (ICLR)*, 2016.

[21] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision (ECCV)*, pages 21–37. Springer, 2016.

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

[23] Alexander Novikov, Dmitrii Podoprikhin, Anton Osokin, and Dmitry P Vetrov. Tensorizing neural networks. In *Advances in neural information processing systems*, pages 442–450. 2015.

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

[25] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 779–788, 2016.

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

[27] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International conference on learning representations (ICLR)*, 2015.

[28] Wei Tang, Gang Hua, and Liang Wang. How to train a compact binary neural network with high accuracy? In *Thirty-First AAAI Conference on Artificial Intelligence (AAAI)*, 2017.

[29] Yaman Umuroglu, Nicholas J Fraser, Giulio Gambardella, Michaela Blott, Philip Leong, Magnus Jahre, and Kees Vissers. Finn: A framework for fast, scalable binarized neural network inference. In *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA)*, pages 65–74. ACM, 2017.

[30] Peisong Wang, Qinghao Hu, Yifan Zhang, Chunjie Zhang, Yang Liu, and Jian Cheng. Two-step quantization for low-bit neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4376–4384, 2018.

[31] Hanwang Zhang, Zawlin Kyaw, Shih-Fu Chang, and Tat-Seng Chua. Visual translation embedding network for visual relation detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 5532–5540, 2017.

[32] Hanwang Zhang, Zawlin Kyaw, Jinyang Yu, and Shih-Fu Chang. Ppr-fcn: weakly supervised visual relation detection via parallel pairwise r-fcn. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 4233–4241, 2017.

[33] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6848–6856, 2018.

[34] Aojun Zhou, Anbang Yao, Yiwen Guo, Lin Xu, and Yurong Chen. Incremental network quantization: Towards lossless cnns with low-precision weights. In *International Conference on Learning Representations (ICLR)*, 2017.

[35] Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160*, 2016.

[36] Chenzhuo Zhu, Song Han, Huizi Mao, and William J Dally. Trained ternary quantization. In *International conference on learning representations (ICLR)*, 2017.

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