IEICE TRANS. INF. & SYST., VOL.E104–D, NO.10 OCTOBER 2021

LETTER

Gradient Corrected Approximation for Binary Neural Networks

Song CHENG[†,††a)], ***Student Member***, **Zixuan LI**[†,††], **Yongsen WANG**[†,††], **Wanbing ZOU**[†,††], **Yumei ZHOU**[†,††], **Delong SHANG**[††,†††], ***and*** **Shushan QIAO**[†,††b)], ***Nonmembers***

SUMMARY Binary neural networks (BNNs), where both activations and weights are radically quantized to be $\{-1, +1\}$, can massively accelerate the run-time performance of convolution neural networks (CNNs) for edge devices, by computation complexity reduction and memory footprint saving. However, the non-differentiable binarizing function used in BNNs, makes the binarized models hard to be optimized, and introduces significant performance degradation than the full-precision models. Many previous works managed to correct the backward gradient of binarizing function with various improved versions of straight-through estimation (STE), or in a gradual approximate approach, but the gradient suppression problem was not analyzed and handled. Thus, we propose a novel gradient corrected approximation (GCA) method to match the discrepancy between binarizing function and backward gradient in a gradual and stable way. Our work has two primary contributions: The first is to approximate the backward gradient of binarizing function using a simple leaky-steep function with variable window size. The second is to correct the gradient approximation by standardizing the backward gradient propagated through binarizing function. Experiment results show that the proposed method outperforms the baseline by 1.5% Top-1 accuracy on ImageNet dataset without introducing extra computation cost.

key words: *binary neural network, deep learning, gradient approximation, fine-tuning*

1. Introduction

In decades, Deep Convolutional Neural Networks (CNNs) has privileged in domain of computer vision and revolutionized massive complicated applications including image classification, face detection and recognition, object segmentation. For higher accuracy, researchers tend to design CNNs more complex with intensive computational cost and memory footprint requirement. While cloud servers equipped with GPUs can effectively handle the training and inference of such models, deploying them to resource-constrained ultra-low-power edge devices [1] is still challenging. To address the challenge, numerous research efforts including light-weight architecture design, compression, quantization [2], have been proposed to reduce complexity of CNNs, seeking for an efficient tradeoff between computation cost and prediction accuracy. As a radical case of quantization, binary neural networks (BNNs) [3] have attracted increasing attention due to its beneficial properties, of which both activations and weights are quantized to $\{-1, +1\}$. Further, significant run-time performance improvement can be achieved by replacing expensive convolutional operations with efficient bit operations such as *xnor* and *popcount*.

However, in backward propagation, the discontinuous and non-differentiable binarizing function used in BNNs, makes the binarized models hard to be optimized, and introduces significant performance degradation than the full-precision models. Many previous works [4] managed to correct the backward gradient mismatch of binarizing function with various improved static binarizing function originating from straight-through estimation (STE), or in a gradual approximate approach. The static approaches diminish the gradient of outliers and exist inevitable gradient mismatch. The gradual approximate [5], [6] approach suppresses the gradient during the process of approximation and makes the information capacity of backward gradient collapsed.

To handle the above problems, we propose a gradient corrected approximation method to match the discrepancy between binarizing function and backward gradient in a gradual and stable way. Our work has two primary contributions:

(1) A simple leaky-steep function with a variable window size is proposed to approximate the backward gradient of binarizing function, which is efficient to approximate binarizing function in a gradual approach, and prevents the gradient diminish of outliers.
(2) Backward gradients are standardized to correct the gradient approximation, which guarantees the information capacity of gradients is stable during the process of gradual approximation.

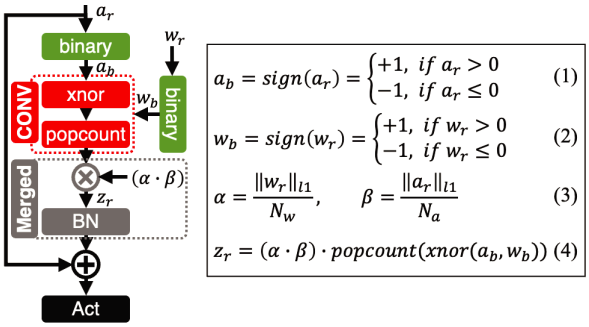
Our approximation method is verified with Bi-Real [7] as baseline. Experiment results show that the proposed method outperforms the baseline by 1.5% accuracy on ImageNet2012 dataset and 2.23% accuracy on CIFAR-10 dataset without introducing extra computation cost.

2. Proposed Method

2.1 Revising BNNs and Gradient Approximation Functions

BNNs is CNNs of which activations and weights are in

Manuscript received March 16, 2021.
Manuscript publicized July 5, 2021.
[†]The authors are with University of Chinese Academy of Sciences, Beijing, China.
[††]The authors are with Institute of Microelectronics of Chinese Academy of Sciences, Beijing, China.
[†††]The author is with Nanjing Institute of Intelligent Technology, IMECAS, Nanjing, China.
a) E-mail: chengsong@ime.ac.cn
b) E-mail: qiaoshushan@ime.ac.cn
DOI: 10.1587/transinf.2021EDL8026



$$a_b = sign(a_r) = \begin{cases} +1, & if\ a_r > 0 \\ -1, & if\ a_r \le 0 \end{cases} \quad (1)$$

$$w_b = sign(w_r) = \begin{cases} +1, & if\ w_r > 0 \\ -1, & if\ w_r \le 0 \end{cases} \quad (2)$$

$$\alpha = \frac{\|w_r\|_{l1}}{N_w}, \qquad \beta = \frac{\|a_r\|_{l1}}{N_a} \quad (3)$$

$$z_r = (\alpha \cdot \beta) \cdot popcount(xnor(a_b, w_b)) \quad (4)$$

Fig. 1 The binarization process of the basic block in our baseline.

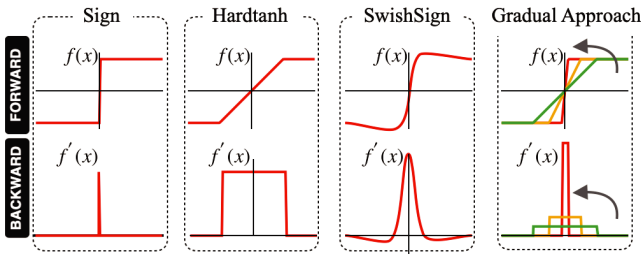


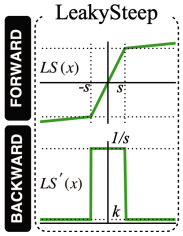
Fig. 2 Typical STE-modified static gradient approximation functions and schematic process of the gradual approximate approach.

binary presentations. The basic block of our baseline is presented in Fig. 1, where indicates the input activations $a_r \in R^{c_i \cdot w_i \cdot h_i}$, weights $w_r \in R^{c_o \cdot c_i \cdot k_i \cdot k_i}$, output activations $z_r \in R^{c_o \cdot w_o \cdot h_o}$. Note that a_r, w_r, z_r are full-precision tensors, a_b, w_b are binarized tensors, $\alpha \in R^{c_o}$ and $\beta \in R^{c_i}$ are scaling factors of weights and input activations. As proved by previous works [8], the optimal solution of binarized estimation for full-precision activations or weights are obtained through a *sign* function. Thus, the binarization process of a_r and w_r expressed as (1) and (2), and scale factors are extracted as (3).

While batch normalization (BN) layer succeeds convolutional layer in the baseline, $(\alpha \cdot \beta)$ item of (4) can be discarded. The remain hamper to optimize BNNs lies on the gradient approximation of sign function, of which the actual derivative function is an impulse function that only has a non-zero value at 0 and is hard to be optimized. Traditionally, the backward gradient of sign function simply employs a *hardtanh* function as the straight-through estimation (STE), which introducing large discrepancy. Thus, multiple differentiable gradient approximation functions [4] have been proposed to correct this gradient mismatch in a static way. However, previous works [9] demonstrate that higher reverse ratio is favourable at the start of training, and lower reverse ratio in sequence. Gradual approximate approaches [5], [6] emerge to be superior to static approaches.

2.2 Leaky-Steep Gradual Approximation

As illustrated in Fig. 2, the typical STE-modified approximation functions perform worse, because of both the gradient vanish of outliers from the blocked region (BR) on the edge, and inevitable gradient mismatch in the central straight-through region (SR). To handle the issue, a leaky-steep gradual approximation method is proposed. The *leaksteep* function and its derivative function can be expressed as (5) (6) and be illustrated in Fig. 3.



$$LS_{k,s}(x) = \begin{cases} k \cdot (x - s) + 1, & if\ x > s \\ x/s, & otherwise \\ k \cdot (x + s) - 1, & if\ x < -s \end{cases} \quad (5)$$

$$LS'_{k,s}(x) = \begin{cases} k, & if\ x > s \\ 1/s, & otherwise \\ k, & if\ x < -s \end{cases} \quad (6)$$

Fig. 3 The illustrations and formulas of leaky-steep function and its derivative function.

Where *LS* represents the *leaksteep* function, k represents the leaky gradient of BR, and s represents the halved window size of SR. k is a fixed variable which has been searched in a collection of {0.0, 0.005, 0.01, 0.02, 0.03}.

In order to gradually approximate the gradient of *sign* function, s is a control variable varying during the training process, which is decreasing in a cosine manner. The runtime value of s can be expressed as:

$$s = max\left\{\frac{S_{start}}{2}\left(cos\left(\pi \cdot \frac{curren\_epoch}{total\_epoches}\right) + 1\right), \varepsilon\right\} \quad (7)$$

where S_{start} and ε represents the halved window size initially and finally, which is set to be 5.0 and 0.1 in our experiment. This setting is efficient to straight through all the gradients including outliers in the beginning and effectively minimize the gradient mismatch at last.

2.3 Backward Gradient Correction

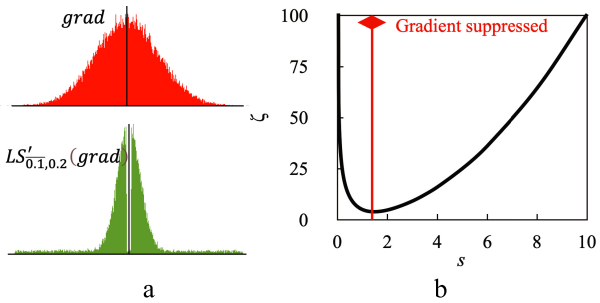
Despite leaky-steep gradual approximate benefits gradient reservation of outliers, eminent gradient suppression along decreasing of s, impedes optimization process, which also exits in the typical gradual approximation approaches [5], [6].

The forward and backward process of binarization and convolution can be formulated as:

$$y_i = \sum_{j=1}^{C} w_{ij} \cdot f(a_j) \quad (8)$$

$$a'_j = \sum_{i=1}^{C} f'(w_{ij}^T \cdot y'_i) \quad (9)$$

where $a \in R^{c_i \cdot w_i \cdot h_i}$ are input float-point activations, $w \in R^{c_o \cdot c_i \cdot k_i \cdot k_i}$ are binary weights fitting *Bernoulli distribution* with candidate set $\{-\frac{1}{\sqrt{C}}, +\frac{1}{\sqrt{C}}\}$ and $p = 0.5$, $y \in R^{c_o \cdot w_o \cdot h_o}$ are output float-point activations, f is the binarizing function adopting simplified *leaky-steep* function with $k = 0.0$. We assume w and $f(a)$ conform to an identical and independent distribution. As the convolutional operations of feature



a b

Fig. 4 (a) Illustration of gradients' distribution after propagated through LS′. (b) Suppression ratio of standard deviation σ of gradients after back-propagated through a leaky-steep function.

maps and weights fitting *Bernoulli distribution* generate output feature maps fitting *Normal distribution*, the variance of y_i during forward process and the variance of a'_i during backward process can be deduced as:

$$var(y_i) = \sum_{j=1}^{C} \{var\left(w_{ij}\right) \cdot var(f(a_j))\} = var(a_j) \tag{10}$$

$$var(a'_i) = \sum_{j=1}^{C} var\left(f'\left(w_{ij}^T y'_j\right)\right) < \sum_{j=1}^{C} var\left(w_{ij}^T y'_j\right) = var(y'_j) \tag{11}$$

They demonstrate that the activations' variance during forward process retained, while their gradients during backward process suppressed. We verify this issue by back-propagating gradients which consistent with normal distributed through a window-size-variable *leaky-steep* function. We define the suppression ratio (SR) of variance to indicate the information capacity decrease of back-propagated gradient:

$$\zeta(s) = var(grad)/var(LS'_{k,s}(grad)) \tag{12}$$

As illustrated in Fig. 4, significant gradient suppression exits when window size s is large or small. Large s slows down the convergence of training while retains outliers' gradient, and small s introduces speedy gradient vanish while better approximates the sign function. This huge fluctuation of gradients margin also impedes tune of hyper-parameters.

In order to handle gradient suppression problem of gradual approaches, we manage to retain the variance of gradients backpropagated through *leaky-steep* function. Thus, gradients in the SR are scaled to compensate the gradient suppression, while gradients in the BR multiply k as back-propagation algorithm required. This process can be formulated as follows:

$$\frac{|A_{i\in SR}|_{l2} + |A_{i\in BR}|_{l2}}{N} = \frac{\left|\frac{\gamma}{s} \cdot A_{i\in SR}\right|_{l2} + |k \cdot A_{i\in BR}|_{l2}}{N} \tag{13}$$

where γ represents scaling factor of weights/activations in SR, N is the total number of items. Then, γ can be deduced:

$$\gamma = s \cdot \sqrt{1 + \frac{(1-k^2)\,|A_{i\in BR}|_{l2}}{|A_{i\in SR}|_{l2}}} \tag{14}$$

Further, moving average per-channel is applied to smooth out γ, to avoid the fluctuation introduced by outliers. The final formula of γ is expressed as follows:

$$\gamma = \delta * \gamma + (1-\delta) * \gamma_{old} \tag{15}$$

where δ represents the average factor and adopts 0.9 in our experiments. After correcting the gradients, it's possible for *leaky-step* function to search the optimal decay trend of window size stably in a wider range.

3. Experimental Details

In this section, we first raise an ablation study with Bi-Real20 on CIFAR-10 dataset in order to search for the optimal hyperparameters including initial halved windows size S_{start} and leaky gradient of blocked region k. Furthermore, we evaluate the accuracy improvement by the proposed gradient corrected approximation method with *Bi-Real18* on *ImageNet 2012* dataset.

Algorithm 1: Training Flow of BNNs using the Gradient Corrected Approximation.

Input: A minibatch of inputs and targets.

Output: The largest element in the set.

Forward propagation:

1 Compute binary weights and input activations:
$A_{l,b}$=sign($A_{l,r}$), $W_{l,b}$=sign(($W_{l,r}$- $\mu(W_{l,r})$)/ $\sigma(W_{l,r})$)/sqrt(C)

2 Compute output activations:
$Z_{l,r}$=popcount(xnor($A_{l,b}$,$W_{l,b}$)), $A_{l+1,r}$=Act(BN($Z_{l,r}$+ $A_{l,r}$))

Backward propagation:

3 Calculate halved window size s using (7)

4 Calculate gradients of weights/activations per-layers in sequential using (6)

5 Correct gradients using (15)

6 Update gradients

The training flow of BNNs using gradient corrected approximation is illustrated in Algorithm 1. The forward propagation process first binarizes weights and input activations, then computes output activations. Minor differences are introduced in this stage. First, $W_{l,r}$ is normalized before binarization and divided with square root of input/output channels C after binarization forcing binarized weight approximate Bernoulli distribution with variance of layers unified to 1/C, which is convenient for the hyperparameters search, without introducing extra computation cost during the inference. Second, scale factors of W_b and A_b are discarded, because BN layer can effectively merge them. The backward propagation adopts our gradient corrected approximation method, where the halved windows size per-epoch is extracted firstly, then the corrected gradients layer-by-layer are calculated. Finally, gradients are updated.

4. Ablation Studies

Ablation studies are evaluated with *Bi-Real20* on *CIFAR-10* dataset. The model adopts *kaiming* initialization and

Table 1 Top-1 accuracy results of Bi-Real20 with variable leaky gradient k of BR. The best result is shown in bold face. All the accuracy is averaged over 4 experiments with random weight initialization.

Bi-Real20 / CIFAR-10, with S_{start} =5.0					
k	0.0	0.005	0.01	0.02	0.03
Accuracy	86.17	**86.48**	86.24	86.01	85.55
(mean+std)%	±0.39	**±0.27**	±0.32	±0.28	±0.19

Table 2 Top-1 accuracy results of Bi-Real20 baseline and improved versions with variable initial halved window size S_{start}. The best result is shown in bold face.

Bi-Real20 / CIFAR-10, with k=0.005								
S_{start}	2	3	4	5	6	7	8	baseline
Top-1 Acc %	86.13	86.45	86.36	**86.48**	86.32	85.87	85.42	84.25

trains from scratch. The training flow runs for 400 epochs with 128 batch-size. Optimization process applies SGD optimizer with momentum=0.9, weight decay=10^{-4}, initial learning rate=0.1, and cosine learning rate decay.

Firstly, we conduct a series of preliminary experiments aimed at determining the optimal value of the leaky gradient of blocked region k varies a collection of {0.0, 0.005, 0.01, 0.02, 0.03}, while the value of initial halved windows size S_{start} is fixed to be 5.0, which is effective to straight-through the gradients of all weights and activations initially. Each experiment repeats four times with random weight initialization (seed=0), and the accuracy results are averaged. The accuracy results represent in Table 1, which indicate that larger k introduces smaller variance of the accuracy, because of reservation of gradients in BR. However, oversized k makes the mismatch of gradient considerable at the end stage, which introduces significant gradient error backpropagated through basic blocks of the baseline. Thus, the ablation study demonstrates that 0.05 is a favorable value of k.

Secondly, the decay trend of the halved window size s is investigated. On account of the cosine decay function applied to schedule the decay trend of s, initial halved window size S_{start} impacts the convergence. Undersized S_{start} will make block region enlarged earlier and impede the convergence at start stage. Oversized S_{start} will delay the matching of gradients at end stage. Thus S_{start} is thoroughly searched in collection of {8, 7, 6, 5, 4, 3, 2}. While W_r mostly are distributed in range (−4, +4), and A_r vary in range (−1, +1), this searching scope is effective to cover.

As Fig. 5 and Table 2 demonstrated, the improved version with GCA outperforms the Bi-Real20 baseline without GCA over 2.23% accuracy. Furthermore, the accuracy of Bi-Real20 with GCA is not sensitive to the value of initial halved window size S_{start}.

Then, we extends the evaluation to a larger image classification dataset *ImageNet* containing 1.2 M training samples and 50,000 validation samples. The training configurations are the same as *Bi-Real20/CIFAR-10*, except *Bi-Real18* is selected as the baseline, training epoch sets 160, and input images are cropped to be 224 as references required. S_{start} equals 5.0 and k adopts 0.005. The model trains on 4 Nvidia

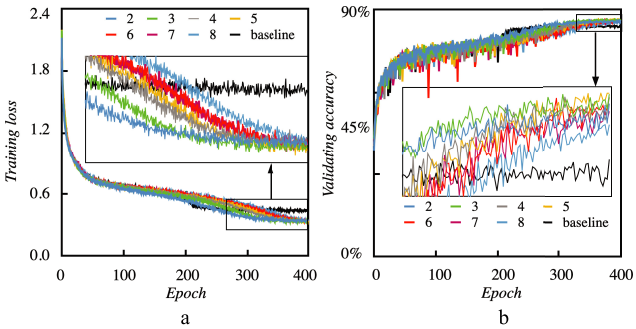


Fig. 5 (a) Training loss curve and (b) validating accuracy curve during training. Bi-Real20 baseline without GCA and improved versions with GCA varying S_{start} in collection of {8, 7, 6, 5, 4, 3, 2} are evaluated.

Table 3 Accuracy comparison with SOTA BNNs on ImageNet. The best result is shown in bold face.

model	*Top-1 Acc %*	*Top-5 Acc %*
XNOR [8]	51.2	73.0
BNN Plus [4]	53.0	72.6
Impro. Bi-Real [10]	57.1	80.2
Bi-Real18 [7]	56.4	79.5
Ours w/o GCA	56.8	79.9
Ours w/ GCA	**57.9**	**80.6**

RTX2080Ti GPUs with a total batch size of 128. Table 3 shows a number of SOTA BNNs. We can observe that, *Bi-Real18* without correcting gradients, which gradual approximate the *sign* function with the *leakysteep* function, outperforms the Bi-Real18 by 0.4%. Furthermore, *Bi-Real18* with GCA boost accuracy improvement to 1.5%.

5. Conclusions

In this letter, we propose a novel gradient corrected approximation method to match the discrepancy between binarizing function and backward gradient in a gradual and stable way, which contains a leaky-steep function to prevent the gradient diminish of outliers and gradients correcting to stabilize the gradual approximation. After applying GCA methods, *Bi-Real20* network gains 2.23% accuracy on *CIFAR-10* dataset, and *Bi-Real18* network gains 1.5% accuracy on *Imagenet2012* dataset. Our work provides a method to narrow the performance gap between BNNs and its full-precision counterparts, which is beneficial for CNNs' deployment on low-power edge devices.

References

[1] J. Zhang, Y. Guo, X. Hu, and R. Li, "Design and Implementation of Deep Neural Network for Edge Computing," IEICE Trans. Inf. & Syst., vol.E101-D, no.8, pp.1982–1996, 2018.

[2] J. Wu, H. Qin, Y. Hua, L. Shao, J. Hu, and S. Yang, "Vector Quantization of High-Dimensional Speech Spectra Using Deep Neural Network," IEICE Trans. Inf. & Syst., vol.E102-D, no.10, pp.2047–2050, 2019.

[3] M. Courbariaux, I. Hubara, and D. Soudry, "Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or −1," arXiv preprint arXiv:1602.02830, 2016.

[4] S. Darabi and B. Mouloud, "Bnn+: Improved binary network training," arXiv preprint arXiv:1812.11800, 2018.
[5] H. Kim, K. Kim, and J. Kim, "Binaryduo: Reducing gradient mismatch in binary activation network by coupling binary activations," International Conference on Learning Representations (ICLR), Addis Ababa, Ethiopia, April 2020.
[6] S. Liu and H. Zhu, "Binary Convolutional Neural Network with High Accuracy and Compression Rate," Proc. 2019 2nd International Conference on Algorithms, Computing and Artificial Intelligence, pp.43–48, Sanya, China, Dec. 2019.
[7] Z. Liu, B. Wu, W. Luo, X. Yang, W. Liu, and K.-T. Cheng, "Bi-Real Net: Enhancing the Performance of 1-Bit CNNs with Improved Representational Capability and Advanced Training Algorithm," Computer Vision – ECCV 2018, Lecture Notes in Computer Science, vol.11219, pp.747–763, Springer International Publishing, Cham, 2018.
[8] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks," Computer Vision – ECCV 2016, Lecture Notes in Computer Science, vol.9908, pp.525–542, Springer International Publishing, Cham, 2016.
[9] T. Wei, G. Hua, and L. Wang, "How to train a compact binary neural network with high accuracy?," AAAI Conference on Artificial Intelligence, San Francisco, USA, Feb. 2017.
[10] D.H. Le and T.V. Pham, "Improving Bi-Real Net with block-wise quantization and multiple-steps binarization on activation," 2020 RIVF International Conference on Computing and Communication Technologies (RIVF), pp.1–6, Ho Chi Minh, Vietnam, April 2020.