LETTER

Low-Complexity Training for Binary Convolutional Neural Networks Based on Clipping-Aware Weight Update

Changho RYU[†], *Nonmember and* **Tae-Hwan KIM**[†a)], *Member*

SUMMARY This letter presents an efficient technique to reduce the computational complexity involved in training binary convolutional neural networks (BCNN). The BCNN training shall be conducted focusing on the optimization of the sign of each weight element rather than the exact value itself in convention; in which, the sign of an element is not likely to be flipped anymore after it has been updated to have such a large magnitude to be clipped out. The proposed technique does not update such elements that have been clipped out and eliminates the computations involved in their optimization accordingly. The complexity reduction by the proposed technique is as high as 25.52% in training the BCNN model for the CIFAR-10 classification task, while the accuracy is maintained without severe degradation.

key words: *binarized neural networks, low complexity, quantization-aware training, convolutional neural networks*

1. Introduction

Binary convolutional neural networks (BCNN) are convolutional neural networks in which each element in the feature and weight is binarized so that can be represented with a single bit [1]. BCNN can achieve an efficient inference owing to the reduced memory bandwidth for the binarized data. The accuracy degradation caused by the binarization can be relieved to great extent if the training is performed in a *quantization-aware* manner [1], [2], exceptionally for the small or medium-scale classification tasks such as CIFAR-10 [3] and SVHN [4]. The quantization-aware training however entails intractably high complexity because it still needs to be performed with real-valued data.

There are several previous researches regarding BCNN; some of them developed novel BCNN models to alleviate the accuracy degradation caused by the binarization [1], [2], [5] and others tried to achieve efficient training of them [6]. BinaryConnect [5] is a pioneering work that showed a potential to achieve a comparable accuracy even with the binarized weight. Binarized Neural Network [1] binarizes the feature as well as weight without degrading the accuracy severely. XNOR-Net [2] employs the scaling factors to compensate for the loss due to the binarization thereby improving the accuracy. The empirical study in [6] provides several practical techniques to achieve an efficient BCNN training.

Manuscript received November 3, 2020.
Manuscript revised January 29, 2021.
Manuscript publicized March 17, 2021.

[†]The authors are with School of Electronics & Information Engineering, Korea Aerospace University, 76, Hanggongdaehak-ro, Deogyang-gu, Goyang-si, Gyeonggi-do, Republic of Korea.
a) E-mail: taehwan.kim@kau.kr

DOI: 10.1587/transinf.2020EDL8143

Table 1 Mathematical notations.

Notation	Description
$\|\|\mathbf{T}\|\|$	Average magnitude of the elements in a tensor or set $\mathbf{T}$.
$\text{sign}(\mathbf{T})$	Bipolarizing each element in a tensor $\mathbf{T}$ by picking its sign.
$\|\mathbf{T}\|$	Number of the elements in a tensor or set $\mathbf{T}$.
$\text{clip}(\mathbf{T}, \delta)$	Clipping each element in a tensor $\mathbf{T}$ to be in the range from $-\delta$ to δ, where $\delta \geq 0$.

This letter presents a novel technique to achieve a low-complexity BCNN training. In the proposed technique, the weight elements, which have been clipped out and thus their signs are not likely to be flipped any more, are not updated. The computations involved in updating them are thus eliminated effectively to reduce the complexity. In training the BCNN model for the CIFAR-10 classification task, the complexity reduction is as high as 25.52%, while the accuracy is maintained without severe degradation. Table 1 summarizes the mathematical notations that will be used throughout this letter.

2. BCNN Training

BCNN are formed by stacking several blocks of identical processing structure. Figure 1 shows the structure of the BCNN block with the dimension of each tensor annotated. In the figure, $\mathcal{L}$ denotes the overall loss. $\mathbb{R}$ is the set of the real numbers and $\mathbb{B}$ is the set of two numbers, i.e., the cardinality of $\mathbb{B}$ is 2. **Activation** binarizes the feature. **ScaledSign** binarizes the (real-valued) elements in the weight $\mathbf{W}$ and scales them by $\|\mathbf{W}\|$ to calculate the approximate weight elements in $\hat{\mathbf{W}}$ [2]; that is, $\hat{\mathbf{W}} = \|\mathbf{W}\| \,\text{sign}(\mathbf{W})$. It should be noted that **ScaledSign** has only to be carried out in the training because the inference can be performed directly with $\hat{\mathbf{W}}$ without necessitating $\mathbf{W}$. The mathematical delineation of each procedure in the block processing has been omitted for brevity as it can be found from other previous literatures [1], [2].

The BCNN training optimizes the parameters ($\mathbf{W}$, β, and γ in every block) iteratively. Each iteration is performed in the order of the forward propagation, backward propagation, and update. Here, β and γ denote the biasing and scaling parameters of **BatchNormalization**. Table 2 summarizes the computational complexity involved in the forward and backward propagations of the tensors whose dimensions are annotated in Fig. 1. In analyzing the computational complexity, the real-valued operation has been considered

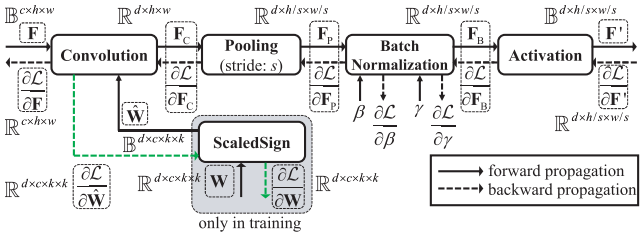


Fig. 1 Processing structure of the basic building block of BCNN [2], where the input feature **F** is processed to produce the output feature **F′**.

Table 2 Analysis of the computational complexity involved in processing the block illustrated in Fig. 1.

Procedure	Forward	Backward	
Convolution	$2dchwk^2$	$\frac{\partial \mathcal{L}}{\partial \mathbf{F}}$	$2dchwk^2 + dhw$
		$\frac{\partial \mathcal{L}}{\partial \tilde{\mathbf{W}}}$	$2dchwk^2 - dck^2$
Pooling	dhw	$\frac{\partial \mathcal{L}}{\partial \mathbf{F}_\mathrm{C}}$	dhw
BatchNormalization	$8dhw/s^2 + 2d$	$\frac{\partial \mathcal{L}}{\partial \mathbf{F}_\mathrm{P}}$	$19dhw/s^2 + 8d$
		$\frac{\partial \mathcal{L}}{\partial \beta}$	$dhw/s^2 - d$
		$\frac{\partial \mathcal{L}}{\partial \gamma}$	$2dhw/s^2 - d$
Activation	dhw/s^2	$\frac{\partial \mathcal{L}}{\partial \mathbf{F}_\mathrm{B}}$	dhw/s^2
ScaledSign	$2dck^2$	$\frac{\partial \mathcal{L}}{\partial \tilde{\mathbf{W}}}$	$8dck^2 - d$

even for the forward propagation through **Convolution**, so that conforms to the practical implementations [7], [8] of the quantization-aware training. The one multiplication has been counted by 1.

The complexity directly involved in the weight update is considerably high. The gradient of $\mathbf{W}$ is calculated through the branch, which has been highlighted with a different color in Fig. 1, from the main backward propagation path. This involves calculating $\frac{\partial \mathcal{L}}{\partial \tilde{\mathbf{W}}}$ as well as $\frac{\partial \mathcal{L}}{\partial \mathbf{W}}$, and its complexity is relatively high as analyzed in Table 2. $\mathbf{W}$ can be updated by $\mathrm{clip}\left(\mathbf{W} - \eta \cdot \frac{\partial \mathcal{L}}{\partial \tilde{\mathbf{W}}}, \delta\right)$, where η represents the learning rate and δ is a clipping value that needs set to a positive value so as to avoid exploding the magnitude of each element [1], [6].

3. Clipping-Aware Weight Update

It can be expected that the sign of a weight element will be flipped with a low probability after its magnitude has been updated to such a large value to be clipped out in the BCNN training. This may be convinced by the fact that the update amount is gradually decreased according as the overall loss is decreased with decaying learning rate as the training goes. To observe the realization for the expectation above, experiment has been conducted as follows: a BCNN model for the CIFAR-10 classification task has been designed based on the structure presented in [1], [5] and trained conventionally as presented in the previous section with the clipping value of 0.1. Figure 2 shows the statistical results obtained by the experiment, where each iteration has been executed with each batch of 500 data. In the figure, $P^{(i)}$ is the sign-flipping probability for the elements in $\bar{\mathbf{W}}^{(i)}$ from the $(i+1)$-th iteration through the end of the training, $i \geq 1$, where $\bar{\mathbf{W}}^{(i)}$ is the set of the elements that have become clipped while in the i-th iteration. $|\mathbf{C}^{(i)}|/|\mathbf{W}|$ is the cumulative proportion of the clipped elements, where $\mathbf{C}^{(i+1)} \triangleq \mathbf{C}^{(i)} \cup \bar{\mathbf{W}}^{(i)}$ is the cumulative set of the clipped elements by the i-th iteration, where $\mathbf{C}^{(1)}$ is initialized to null.

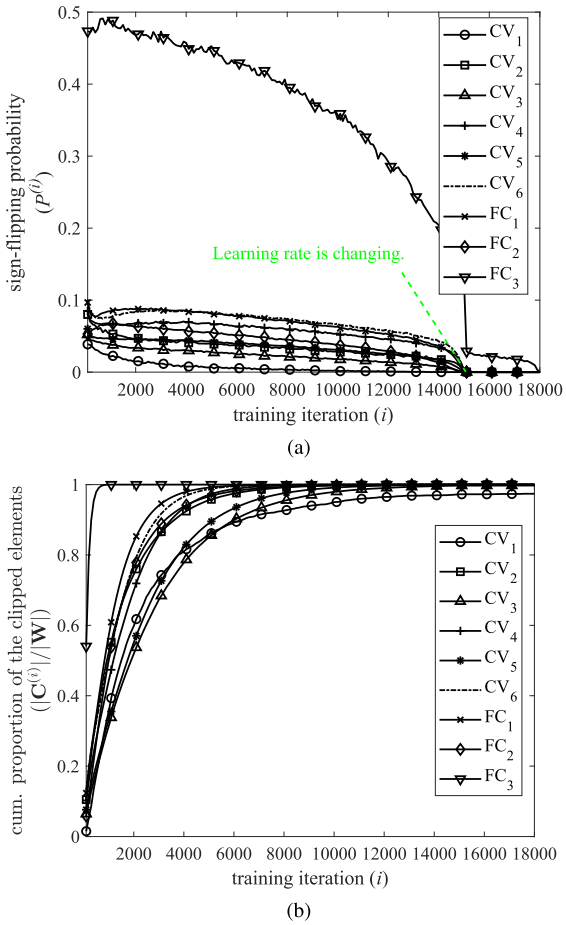


Fig. 2 Statistical results in training the BCNN model for the CIFAR-10 classification task, where CV_n and FC_n represent the n-th convolutional and fully-connected blocks, respectively: (a) sign-flipping probability of the clipped elements and (b) cumulative proportion of the clipped elements.

The results shown in Fig. 2 correspond well to our expectation. The probability of the sign-flipping event for the clipped elements is very low and diminishes further as the training goes. The last block (FC_3) does not perform **Activation** to produce the real-valued results and thus the gradient is unlimitedly propagated from the loss, so the update amount of each weight is relatively large; nonetheless, the probability of the sign-flipping event for the clipped elements is not so high even in the last block. In Fig. 2b, the proportion of the clipped elements grows rapidly to a high value. It is interesting to find that most of the elements becomes clipped after some iterations and their signs are rarely flipped until the end of the training.

It is less meaningful to update the weight elements whose signs will not be flipped any more, noting that we do not need their exact values but their signs after the training ends. Considering that the clipped weight elements will be flipped with a low probability as observed in Fig. 2a, the proposed technique does not update them any more until the

training ends and forces current values (which have been clipped) to their optimization results. This can eliminate the computations involved in their update, where the related complexity is considerably high as analyzed in the previous section.

The proposed technique can be applied on a per-block basis. It does not update entire weight elements associated with a block only after the proportion of the cumulative number of the clipped elements in the block has become larger than a threshold. Listing 1 describes the backward propagation and update within the block illustrated in Fig. 1 based on the proposed technique with the threshold denoted by τ. As applied on the per-block basis, the proposed technique can be carried out in such a structured way that can eliminate entire computations regarding the update of $\mathbf{W}$ within a block, which is useful for the vectorized implementation.

Some additional remarks that are worth noting are followed.

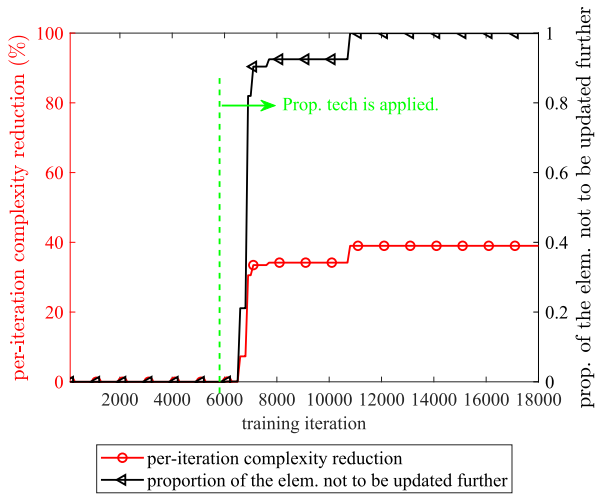
- The complexity reduction by the proposed technique is affected by the proportion of the elements that have been determined so as not to be updated as well as the amount of the eliminated computations with respect to them.
- The accuracy can be affected by the proposed technique because the training results may become different. This is because it is not absolutely impossible that the clipped elements are going to be flipped.
- The clipping-aware update cannot be applied for the **BatchNormalization** parameters (β and γ) because those parameters are not binarized. Therefore, even in the case that the entire weight elements have been determined so as not to be updated, we cannot eliminate the computations of other gradients except for those related with the weight elements.

Listing 1 Backward propagation and update within the block illustrated in Fig. 1 for the i-th iteration based on the proposed technique.

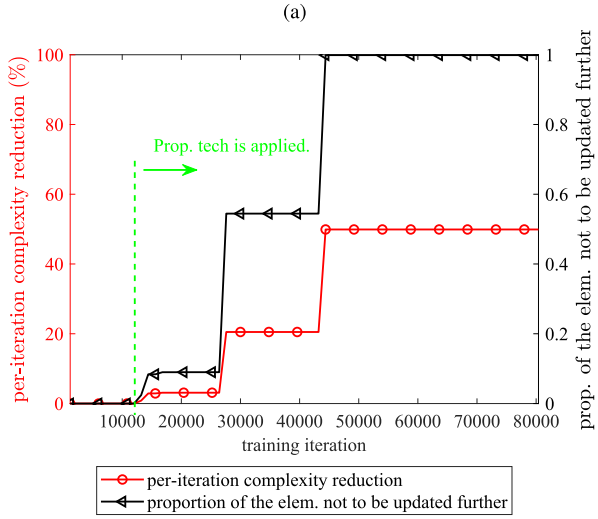
Input: Gradient of $\mathbf{F}'$: $\frac{\partial \mathcal{L}}{\partial \mathbf{F}'}$
Output: Gradient of $\mathbf{F}$: $\frac{\partial \mathcal{L}}{\partial \mathbf{F}}$
1: Calculate $\frac{\partial \mathcal{L}}{\partial \mathbf{F}_\mathrm{B}}$ using $\frac{\partial \mathcal{L}}{\partial \mathbf{F}'}$.
2: Calculate $\frac{\partial \mathcal{L}}{\partial \mathbf{F}_\mathrm{P}}$ using $\frac{\partial \mathcal{L}}{\partial \mathbf{F}_\mathrm{B}}$.
3: Calculate $\frac{\partial \mathcal{L}}{\partial \mathbf{F}_\mathrm{C}}$ using $\frac{\partial \mathcal{L}}{\partial \mathbf{F}_\mathrm{P}}$.
4: Calculate $\frac{\partial \mathcal{L}}{\partial \gamma}$ using $\frac{\partial \mathcal{L}}{\partial \mathbf{F}_\mathrm{B}}$.
5: Update γ using $\frac{\partial \mathcal{L}}{\partial \gamma}$.
6: Calculate $\frac{\partial \mathcal{L}}{\partial \beta}$ using $\frac{\partial \mathcal{L}}{\partial \mathbf{F}_\mathrm{B}}$.
7: Update β using $\frac{\partial \mathcal{L}}{\partial \beta}$.
8: **if** $\frac{|\mathbf{C}^{(i)}|}{|\mathbf{W}|} < \tau$ **then** ▷ Do not update $\mathbf{W}$ unless this condition is met.
9: Calculate $\frac{\partial \mathcal{L}}{\partial \tilde{\mathbf{W}}}$ using $\frac{\partial \mathcal{L}}{\partial \mathbf{F}_\mathrm{C}}$.
10: Calculate $\frac{\partial \mathcal{L}}{\partial \mathbf{W}}$ using $\frac{\partial \mathcal{L}}{\partial \tilde{\mathbf{W}}}$.
11: Update $\mathbf{W}$ using $\frac{\partial \mathcal{L}}{\partial \mathbf{W}}$.
12: $\mathbf{C}^{(i+1)} \leftarrow \mathbf{C}^{(i)} \cup \bar{\mathbf{W}}^{(i)}$, where $\mathbf{C}^{(1)}$ is initialized to null and $\bar{\mathbf{W}}^{(i)}$ is the set of the elements that have become clipped while executing the 11-th line.
13: **end if**
14: Calculate $\frac{\partial \mathcal{L}}{\partial \mathbf{F}}$ using $\frac{\partial \mathcal{L}}{\partial \mathbf{F}_\mathrm{C}}$ and return it.

4. Evaluation

The effectiveness of the proposed technique has been evaluated for the BCNN training of the models for the CIFAR-10 and SVHN classification tasks. The BCNN model for the CIFAR-10 classification task has been designed with the structure described in [1], [5], and that for the SVHN classification task has been designed based on AlexNet [9], by adapting the dimensions of its first and last blocks appropriately. They have been trained with 60000 and 604388 images, respectively. The proposed technique has been applied since the 6000-th and the 12000-th iterations in training the models for the two classification tasks, respectively, and the



(a)



(b)

Fig. 3 Complexity reduction and proportion of the elements that have been determined so as not to be updated further by the proposed technique, in training the BCNN models for (a) the CIFAR-10 classification and (b) SVHN classification tasks, respectively.

clipping value has been set to 0.1 consistently. The proposed technique does not update the entire weight elements associated with a block only after the cumulative proportion of the clipped ones is larger than 0.9 and 0.8 in training the two models, respectively. The stochastic-gradient-descent optimizer has been employed with the momentum of 0.9.

The proposed technique reduces the overall complexity involved in the BCNN training significantly. Figure 3 shows the complexity reduction in each iteration for the BCNN training along with the proportion of the elements that have been determined so as not to be updated by the proposed technique. The complexity has been estimated by counting the number of the arithmetic operations involved in the forward and backward propagations as described in the previous section. The complexity reduction becomes more significant as the proportion of the elements that have been determined so as not to be updated increases. The proportion of the elements that are determined so as not to be updated increases stepwise as the proposed technique is applied per block. The proposed technique reduces the overall complexity by 25.52% and 28.83% for the two models designed for the CIFAR-10 and SVHN classification tasks, respectively.

The proposed technique maintains the accuracy without severe degradation. Figure 4 shows that some difference exists between the training curves obtained with and without the proposed technique but is not noticeable in terms of the final loss. Table 3 compares the test accuracy achieved by the BCNN models trained with and without the proposed technique. The accuracy degradation is not higher than 2.75% and 1.31% for the models designed for the CIFAR-10 and SVHN classification tasks, respectively.

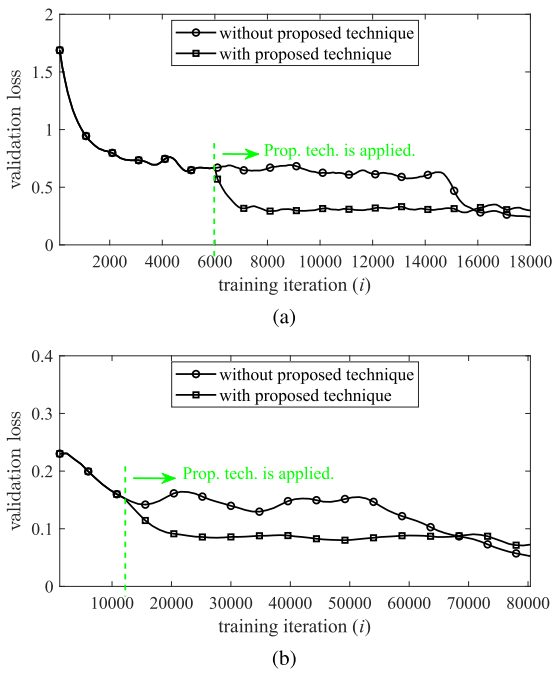


Fig. 4 Validation loss in training the BCNN models for (a) the CIFAR-10 classification and (b) SVHN classification tasks, respectively.

Table 3 Test accuracy.

Classification task	without proposed tech.	with proposed tech.
CIFAR-10	87.73%	84.98%
SVHN	95.37%	94.06%

5. Conclusion

A novel technique has been proposed to reduce the complexity involved in the BCNN training. The proposed technique does not update the weight elements, whose magnitudes have been updated to such large values that can be clipped out, thus eliminating the related computations effectively. Evaluated for the BCNN model designed for the CIFAR-10 classification task, the proposed technique can reduce the overall complexity involved in the training by 25.52% without degrading the accuracy severely.

Acknowledgments

This work was supported by the GRRC program of Gyeonggi province (2017-B02, Study on 3D Point Cloud Processing and Application Technology) and Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (2018R1D1A1A09082763). The tools were supported by IDEC, Korea.

References

[1] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks," Proc. Adv. Neur. Inf. Proc. Syst., pp.4107–4115, NeurIPS Foundation, Dec. 2016.
[2] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-net: Imagenet classification using binary convolutional neural networks," Proc. Euro. Conf. Comput. Vis., pp.525–542, Springer, Oct. 2016.
[3] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Tech. Rep., Citeseer, April 2009.
[4] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A.Y. Ng, "Reading digits in natural images with unsupervised feature learning," Proc. NeurIPS Work. Deep Learning & Unsupervised Feature Learning, pp.1–9, NeurIPS Foundation, Dec. 2011.
[5] M. Courbariaux, Y. Bengio, and J.P. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," Proc. Adv. Neur. Inf. Proc. Syst., pp.3123–3131, Dec. 2015.
[6] M. Alizadeh, J. Fernández-Marqués, N.D. Lane, and Y. Gal, "An empirical study of binary neural networks' optimisation," Proc. Int'l Conf. Learning & Representation, pp.1–11, 2018.
[7] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," Proc. Adv. Neur. Inf. Proc. Syst., pp.8026–8037, 2019.
[8] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D.G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, "Tensorflow: A system for large-scale machine learning," USENIX Symp. Op. Sys. Design & Impl., pp.265–283, Nov. 2016.
[9] A. Krizhevsky, I. Sutskever, and G.E. Hinton, "ImageNet classification with deep convolutional neural networks," Comm. ACM, vol.60, no.6, pp.84–90, May 2017.