Supplementary Material for INSTA-BNN: Binary Neural Network with INSTAnce-aware Threshold

Changhun Lee[1] Hyungjun Kim[2] Eunhyeok Park[1] Jae-Joon Kim[3]

[1] Pohang University of Science and Technology (POSTECH), Pohang, Korea

[2] SqueezeBits Inc., Seoul, Korea [3] Seoul National University, Seoul, Korea

A Experimental setups

1 Experimental setup for CIFAR-10 dataset

Extensive experiments for Fig. 3 in the main paper were conducted using ResNet-20 [3] with CIFAR-10 dataset. More precisely, we used ResNet-20 with additional skip connections from [8] and attached PReLU after the element-wise addition of residual path and identity shortcut. Therefore, the convolution block structure is similar to the block used in ReActNet [7] (Fig. 2a of the main paper). We used AdamW optimizer [9] with a weight decay value of 1e-4, and the networks were trained up to 400 epochs. The initial learning rate was 0.003, and the cosine annealing schedule was used. The batch size was set to 256. We used the weight scaling factor from [11].

2 Experimental setup for the ablation study

We used a simplified training process for the ablation studies, in which we trained a BNN model directly from scratch for 90 epochs without help of the pre-trained or teacher models. Adam optimizer [6] was used with an initial learning rate of 0.001, which is multiplied by 0.1 at epochs 40, 60, 80. Weight decay was not used, and learning rate warmup was used for the first 5 epochs. Precision other than 1-bit and 32-bit was not used for the ablation study. Experiments that produced the results in Table 1 of the main paper and the rest of the supplementary material experiments (Sec. F, G, H) also followed this setup.

B Details and training methods for the quantization of SE-like modules

As mentioned in Sec. 5.1 of the main paper, we quantized the weights of the SE-like modules. We followed the fine-tuning scheme of learned step size quantization (LSQ) [2]. We used channel-wise learnable step size for 8-bit weight quantization and symmetric quantization ($Q_N = 2^{b-1}$ and $Q_P = 2^{b-1}-1$, encoding with b bits) was applied.

After the two-stage training, we obtain the conventional BNN with binary convolution blocks and all other real-valued components. When the SE module is used, additional 10 epochs of fine-tuning were performed with 8-bit SE weights. The initial learning rate was 1e-4 and we used a linear learning rate decay scheduler. We did not use the weight decay for fine-tuning. This 8-bit SE weight quantization did not cause accuracy degradation in our experiments.

C Detailed rules for the computational cost analysis

In INSTA-BNN, the additional operations are mainly from normalizing and cubic operations. We calculated total number of opertations (OPs) as OPs = FLOPs + (BOPs / 64), following [7, 8]. In case of parameters, binary weights are 1-bit, weights of SE-like modules are 8-bit, and other real-valued parameters and weights are considered as 32-bit.

We manually calculated the computation cost and parameter size of the previous methods and our proposed INSTA-BNN. In this section, we introduce additional detailed rules that we used to calculate the computation cost of the network. First, we only counted the number of floating point (FP) multiplication for the FLOPs calculation following the method used in [7, 8]. That is, we count one floating point multiply-accumulate (MAC) as one FLOP, so our FLOPs calculation includes a pair of multiplication and addition. The same rule applies to the calculation of BOPs.

For the Batch Normalization layer, we assumed that the layer has $2{\times}C$ sets of parameters because four BN parameters can be merged into the scale and shift terms offline. FP multiplication cost of BN layer is $H{\times}W{\times}C$. However, when the Sign function directly comes after BN layer, FP multiplication cost is removed and we only need $1{\times}C$ set of parameters, following [5]. In case of using weight scaling factor [11], the element-wise FP multiplication cost of $H_{out}{\times}W_{out}{\times}C_{out}$ is required, where the subscript out means the output of the convolution. However, when a method has conv-BN layer ordering, instead of doing element-wise multiplication for each scaling factor and

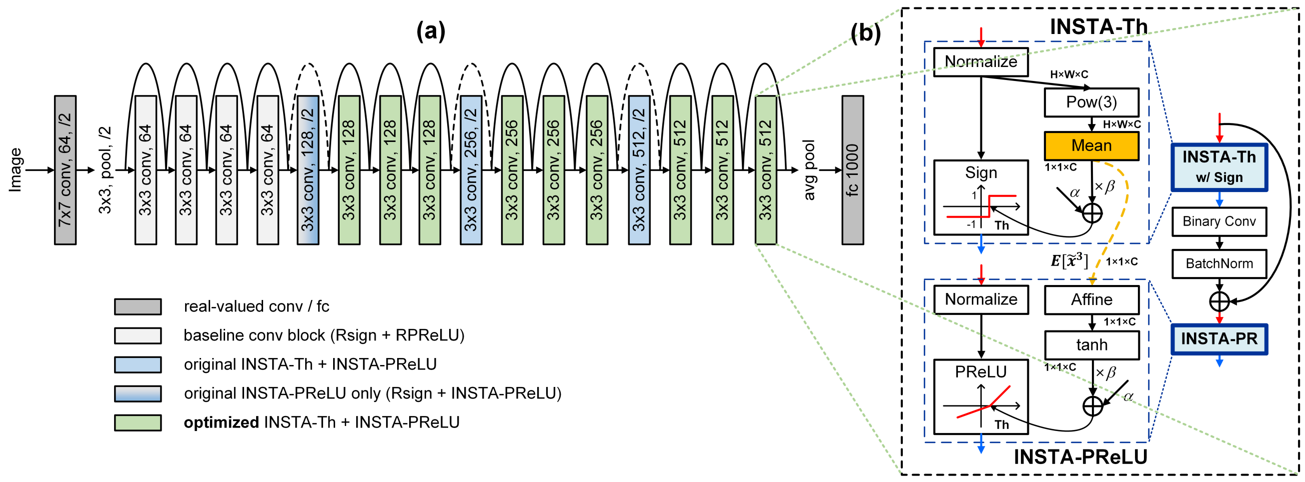

Figure 1. (a) Block diagram of latency-optimized INSTA-BNN-ResNet18. (b) Binary convolution block that contains the optimized INSTA-Th and INSTA-PReLU modules. Orange block indicates the calculated $E\left[\tilde{x}^3\right]$ value in the INSTA-Th module. We can see that $E\left[\tilde{x}^3\right]$ value is reused for the INSTA-PReLU (indicated by the yellow dotted line).

BN layer, we can first multiply analytically calculated scaling factor and the merged BN scale parameter. In this way, we can also remove one set of $H{\times}W{\times}C$ element-wise multiplications. We ignored the operation cost for PReLU layer because the cost may vary depending on the implementation, and its effect is very small compared to the real-valued convolution cost.

Finally, we assumed that all the ResNet-based methods use the shortcut option B reported in [3]. In addition, we assumed that average pooling, real-valued 1x1 convolution, and BN layer are used for the downsampling path, except the BNN [4] case, which uses binary 1x1 convolution.

Our proposed INSTA-Th needs $4{\times}C$ set of parameters ($\hat{\mu}$, $\hat{\sigma}$, α, β). For INSTA-PReLU, total five parameters per channel (including the learned slope of PReLU) are required. Both modules need $H{\times}W{\times}C$ floating point operations for normalization and $2{\times}H{\times}W{\times}C$ floating point operations for cubic operation. For INSTA-Th, additional $2{\times}C$ operations are needed for spatial averaging and multiplying channel-wise parameters. In addition, for INSTA-PReLU, $3{\times}C$ operations are needed because we additionally use 3*tanh($x/3$) for INSTA-PReLU. Note that the computational cost of these operations is proportional to the channel size, which is negligible compared to the computational cost of the element-wise operations.

For the optimized INSTA-BNN structure, the feature statistics reuse (Sec. 4.2 of the main paper) eliminates a cubic operation for the corresponding INSTA-PReLU module. It also removes spatial averaging thereby reducing the number of required extra operations from $3{\times}C$ to $2{\times}C$. Instead, two channel-wise parameters are used for the affine transform of the reused statistics (Fig. 1b).

INSTA-Th$^+$ and INSTA-PReLU$^+$ further require $2C^2/r$ 8-bit parameters for fully-connected layers of each SE-like module (r: reduction ratio) instead of one channel-wise parameter (α). We also included the additional computational cost caused by the SE weight quantization in total OPs calculation. Due to the channel-wise step size of 8-bit weight quantization, we need $C + C/r$ real-valued parameters and multiplications for each SE-like module.

D Detailed model structure of latency-optimized INSTA-BNN

For latency-optimized INSTA-BNN models, we used the INSTA modules from the point where the feature size is halved for the third time in the model structure ($H = 28$ when input image size is 224). Fig. 1 shows the optimized model structure of INSTA-BNN-ResNet18. In Fig. 1a, although the RPReLU (INSTA-PReLU) of each conv block comes after a shortcut is added, the diagram is presented differently for simplicity. These latency reduction schemes are similarly applied to the MobileNet-based INSTA-BNN.

E Analysis of feature statistics reuse in INSTA module

Fig. 2 shows the relationships between statistics ($E\left[\tilde{x}^3\right]$) of INSTA-Th and the subsequent INSTA-PReLU (Fig. 2a-d) introduced in the main text, as well as the correlation between $E\left[\tilde{x}^3\right]$ value of INSTA-PReLU and next conv block's INSTA-Th (Fig. 2e-h). Each subgraph shows the correlation between a pair of internal INSTA modules for each ResNet block. We compared the $E\left[x^3\right]$ value calculated immediately after normalization for 256 training set images. Figures are drawn using channel 0 data in this example. We can observe that statistics ($E\left[\tilde{x}^3\right]$) calculated from INSTA-Th and those from the subsequent INSTA-PReLU have high correlation (Fig. 2a-d) so that we can reuse the

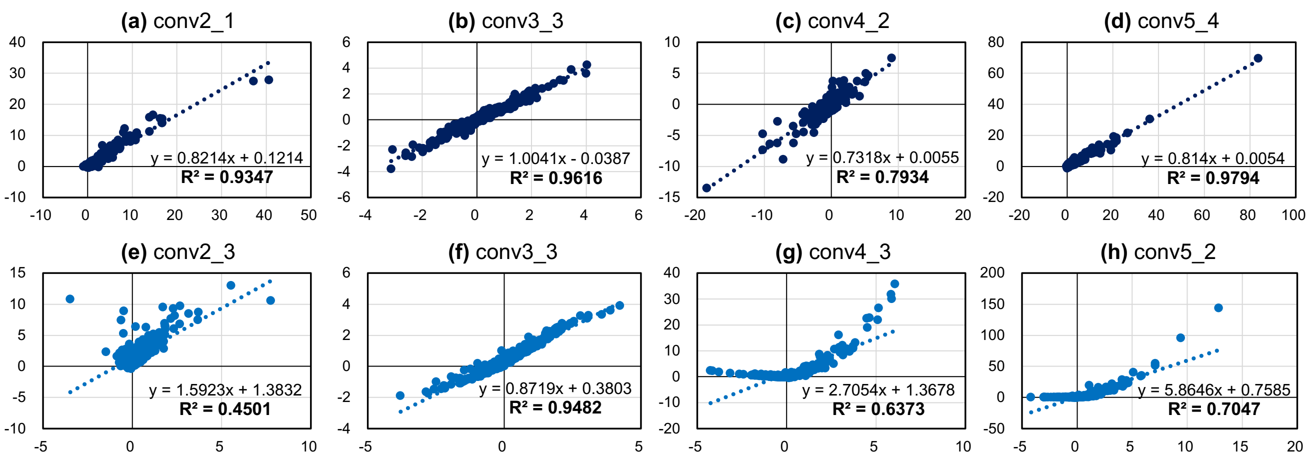

Figure 2. (a)-(d) Correlation between $E[\tilde{x}^3]$ values from INSTA-Th and subsequent INSTA-PReLU. (e)-(h) Correlation between $E[\tilde{x}^3]$ values from INSTA-PReLU and next conv block's INSTA-Th.

	Range	INSTA-PReLU	SE (INSTA-Th$^+$)
$sigmoid(x)$	$0 \sim 1$	61.8	61.7
$tanh(x)$	$-1 \sim 1$	61.9	61.4
$3tanh(x/3)$	$-3 \sim 3$	**62.1**	**61.7**

Table 1. ImageNet top-1 valid accuracy (%) according to the non-linear component of the each INSTA-PReLU and SE of INSTA-Th$^+$ module. Range means possible output range of each function.

	Top-1 Acc. (%)
BN - sign - conv - PReLU	59.5
BN (w/o γ, β) - sign - conv - PReLU	59.0
INSTA-Th - conv - PReLU	60.5

Table 2. ImageNet top-1 accuracy comparison between the BN-sign-conv layer ordered network with and without INSTA-Th module. When the INSTA-Th module is used, its normalization layer replaces a BN layer of the network.

statistics obtained from INSTA-Th for INSTA-PReLU. In contrast, Fig. 2g, h show some problematic cases, in which significant non-linearity is observed due to the intervening PReLU. Note that there is a large difference between the x-axis and y-axis value scales. For this reason, we do not reuse the statistics obtained from INSTA-PReLU for INSTA-Th of the next conv block.

F Effect of the threshold value range in INSTA-PReLU/INSTA-TH$^+$

We use the bounded non-linear functions in the INSTA-PReLU, INSTA-Th$^+$, and INSTA-PReLU$^+$ modules to map the threshold values in the constrained range (Fig. 4a, b in the main paper). We evaluated three options; sigmoid(x), tanh(x), and 3*tanh($x/3$). Sigmoid is a widely used function and was used in the original SE work. However, the output of the sigmoid is always greater than zero, and hence it can only move the threshold to one direction from zero. On the other hand, tanh is bidirectional. We observed that the max value of α in Eq. (8) for the trained networks was 2.45, so that we included 3*tanh($x/3$) in the options. From the Table 1, 3*tanh($x/3$) was found to be the best non-linear function for the INSTA-PReLU, and it showed similar accuracy to that of sigmoid for SE module. Based on the results of Table 1 and the observed α ranges, we used 3*tanh($x/3$) for the experiments in this paper.

G INSTA-Th for different block structure

In recent studies [1, 10], a different convolution block layer order from the one used in the Bi-real-net [8] has been used. Its layer order is BatchNorm (BN)-Sign-conv-PReLU-$\oplus$, where $\oplus$ means element-wise addition of residual path and identity shortcut starting just before BN. To verify the versatility of the proposed method, we also tested INSTA-Th to this block structure. One thing to note is that the normalization layer inside the newly added INSTA-Th comes immediately after the BN of the original structure (BN - Normalization - Sign with modified TH). In this case, the scale and shift effect of the original BN layer may be inhibited, and the meaning of the added normalization layer is also faded. To handle this, we tried to merge BN and the newly added normalization layer (replacing the original BN with the proposed INSTA-Th), and its result is in Table 2. Removing the scale and shift from the original BN brought a slight decrease in accuracy (row 1 and 2), but merging BN and normalization layer of INSTA-Th shows the improved threshold controlling ability (row 1 and 3). This suggests that the proposed INSTA-Th plays a similar role to the shift of BN, but it works instance-wise and improves the performance further. This result supports the effectiveness of the proposed INSTA-Th in various block structures.

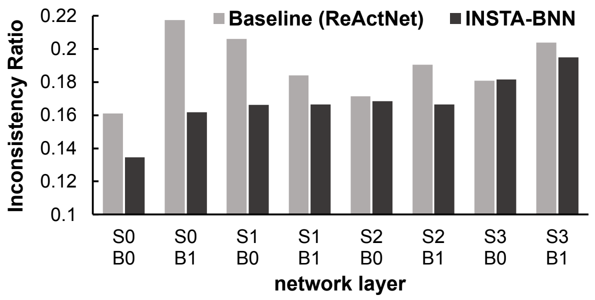

Figure 3. Comparison of the inconsistent sign ratio between the baseline and the proposed INSTA-BNN. S and B represent the stage and the block of ResNet structure. Note that ResNet-18 consists of four stages with two blocks each, and therefore S0 B0 contains conv2 and conv3 layers.

H Discussion: Reducing the inconsistent sign problem of binary convolution

CI-BCNN [12] discussed that one of the key reasons for the quantization errors in BNN is the sign mismatch between the binary convolution results and the counterpart full-precision results $(sign(W_r \otimes A_r) \neq sign(W_b \oplus A_b))$. Then they tried to solve the inconsistency by imposing the channel-wise priors on the feature maps. Meanwhile, tuning the threshold is a direct way of changing the sign of binary activation, which consequently affects the sign of the binary convolution output. Hence, we checked the inconsistent ratio (ratio of pixels with inconsistent signs) for our baseline ReActNet and the proposed INSTA-BNN, and observed that the ratio was consistently lower in the INSTA-BNN (Fig. 3). The results demonstrate that our INSTA-BNN has the ability to reduce the quantization error by controlling the binary threshold without using channel-wise priors.

I Visualization results of t-SNE

We compared three models based on ResNet-20 having different threshold determination methods: Sign function with a constant threshold of zero, RSign [7] function, and INSTA-Th. Fig. 4 shows t-SNE visualization of features extracted before the classifier of each trained model. For the INSTA-Th case (Fig. 4c), cat (red) and dog (brown) classes, which are relatively difficult to distinguish in other models, are better distinguished. Also, deer (purple) and bird (green) are relatively well clustered in the model with INSTA-Th.

J Comparison with DyBNN

DyBNN [13] and INSTA-BNN have an apparent difference in that DyBNN relies solely on the channel-wise mean, while we employ higher-order statistics as well. Fig. 1 of the main paper shows the motivation for this: the higher-order statistics clearly describe the characteristics of the pre-activation distribution. Based on this, INSTA-BNN determines a more appropriate threshold for each instance.

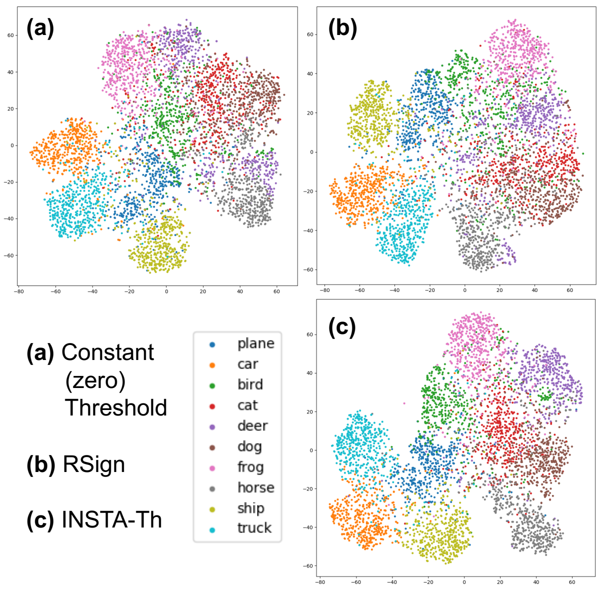

Figure 4. t-SNE visualization of features extracted before the classifier (fully-connected layer) of each model. Half of the CIFAR-10 test set was used (500 images per class). Best viewed in color.

Please note that basic INSTA-BNN outperforms DyBNN (71.7% vs. 71.2%) without extra parameters of Fully-Connected (FC) modules. Moreover, INSTA-BNN+ (72.2%) further improves accuracy by combining FC modules, illustrating the orthogonality of higher-order statistics and FCs. Therefore, INSTA-BNN offers an additional choice to balance between accuracy and operations/parameters, and this extension capability is superior to DyBNN. Additionally, we provide hardware deployment results and optimization methods, which are critical directions for the practical use of BNN works.

References

[1] Adrian Bulat, Brais Martinez, and Georgios Tzimiropoulos. High-capacity expert binary networks. In *International Conference on Learning Representations (ICLR)*, 2021. 3

[2] Steven K Esser, Jeffrey L McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S Modha. Learned step size quantization. In *International Conference on Learning Representations*, 2020. 1

[3] 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*, pages 770–778, 2016. 1, 2

[4] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks. *Advances in neural information processing systems*, 29, 2016. 2

[5] Hyungjun Kim, Yulhwa Kim, and Jae-Joon Kim. In-memory batch-normalization for resistive memory based binary neural network hardware. In *Proceedings of the 24th Asia and South Pacific Design Automation Conference*, pages 645–650, 2019. 1

[6] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 1

[7] Zechun Liu, Zhiqiang Shen, Marios Savvides, and Kwang-Ting Cheng. Reactnet: Towards precise binary neural network with generalized activation functions. In *European Conference on Computer Vision*, pages 143–159. Springer, 2020. 1, 4

[8] 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. 1, 3

[9] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 1

[10] Brais Martinez, Jing Yang, Adrian Bulat, and Georgios Tzimiropoulos. Training binary neural networks with real-to-binary convolutions. In *ICLR*. 2020. 3

[11] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European conference on computer vision*, pages 525–542. Springer, 2016. 1

[12] Ziwei Wang, Jiwen Lu, Chenxin Tao, Jie Zhou, and Qi Tian. Learning channel-wise interactions for binary convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 568–577, 2019. 4

[13] Jiehua Zhang, Zhuo Su, Yanghe Feng, Xin Lu, Matti Pietikäinen, and Li Liu. Dynamic binary neural network by learning channel-wise thresholds. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1885–1889. IEEE, 2022. 4