

# INSTA-BNN: Binary Neural Network with INSTance-aware Threshold

Changhun Lee<sup>1</sup> Hyungjun Kim<sup>2</sup> Eunhyeok Park<sup>1</sup> Jae-Joon Kim<sup>3</sup>

<sup>1</sup> Pohang University of Science and Technology (POSTECH), Pohang, Korea

<sup>2</sup> SqueezeBits Inc., Seoul, Korea <sup>3</sup> Seoul National University, Seoul, Korea

## Abstract

Binary Neural Networks (BNNs) have emerged as a promising solution for reducing the memory footprint and compute costs of deep neural networks, but they suffer from quality degradation due to the lack of freedom as activations and weights are constrained to the binary values. To compensate for the accuracy drop, we propose a novel BNN design called Binary Neural Network with INSTance-aware threshold (INSTA-BNN), which controls the quantization threshold dynamically in an input-dependent or instance-aware manner. According to our observation, higher-order statistics can be a representative metric to estimate the characteristics of the input distribution. INSTA-BNN is designed to adjust the threshold dynamically considering various information, including higher-order statistics, but it is also optimized judiciously to realize minimal overhead on a real device. Our extensive study shows that INSTA-BNN outperforms the baseline by 3.0% and 2.8% on the ImageNet classification task with comparable computing cost, achieving 68.5% and 72.2% top-1 accuracy on ResNet-18 and MobileNetV1 based models, respectively.

## 1. Introduction

Deep neural networks (DNNs) are well-known for their abilities across diverse vision tasks, *e.g.*, image classification [7, 13, 26, 27], object detection [14, 15, 24, 25], and semantic segmentation [19, 21]. These DNNs usually achieve excellent accuracy by using a large model, but the large model having massive memory usage and computing cost prevents us from deploying it on mobile devices having insufficient resources. In order to minimize the computation and memory overhead, network binarization is an appealing optimization because weights and activations are quantized into 1-bit precision domain. Binary Neural Networks (BNNs) can achieve  $32\times$  reduction in memory requirement compared to their 32-bit floating point counterparts, and most modern CPUs or GPUs can serve the binary logic operations much faster than 32-bit floating point operations.

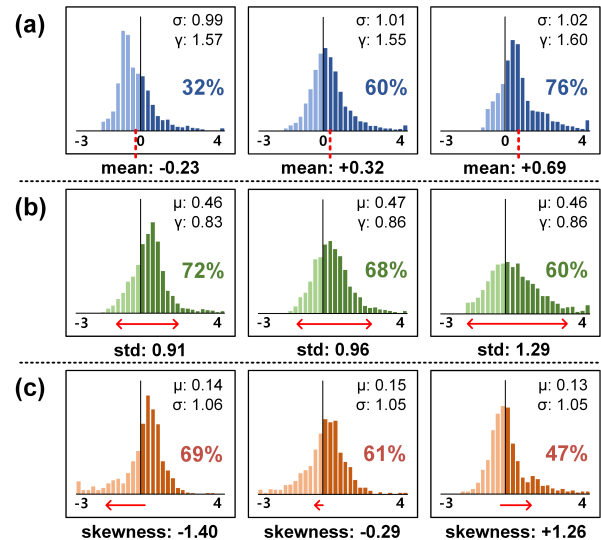


Figure 1. Instance-wise (a) mean, (b) standard deviation, and (c) skewness of pre-activation from three different instances. The percentage in each distribution represents the ratio of +1's in binary activation when sign function is used as activation function (threshold is 0). When observing one component (*e.g.* mean ( $\mu$ )) for the (a) case), the other two (*e.g.* std ( $\sigma$ ), skewness ( $\gamma$ )) are matched to be as close as possible. The other two values are displayed on the top right of each subgraph.

However, BNNs usually suffer from accuracy degradation due to the aggressive data quantization. In spite of the great efficiency of BNN, the accuracy degradation limits the deployment of BNNs in real-world applications. Therefore, a large number of techniques have been introduced to minimize the accuracy degradation of BNNs focusing on weight binarization [22, 23], shortcut connections [17], and threshold optimization [12, 16, 29].

In BNNs, only two values (+1 and -1) are available for activations. Therefore, the threshold of quantization that decides the mapping to either +1 or -1 plays a critical role in BNNs. Some of the previous works tried to train the thresholds of the binary activation function to control the activation distributions via back-propagation, where the threshold is fixed statically after the training [16, 29]. However, one

can easily observe that the per-instance activation is heavily distorted depending on the input data (Fig. 1), even while the batch-wise statistics are stabilized via batch normalization. As shown in the figure, the per-instance statistics numerically indicate the large distortion of pre-activation, and the corresponding outputs have a heavy fluctuation of the ratio of +1’s. In BNNs, static thresholds may not provide the adequate threshold for each instance, thereby resulting in sub-optimal results in terms of overall accuracy.

In this work, we propose a novel INSTA-BNN that calculates the thresholds dynamically using the input-dependent or instance-wise information (*e.g.* mean, variance, and skewness). The instance-aware threshold enriches the quality of binary features significantly, resulting in higher accuracy of BNNs. In addition, we provide a variant of Squeeze-and-Excitation (SE) [8] with instance-wise adaptation as an additional option that one can exploit for even higher accuracy with extra parameters. The proposed modules are extensively evaluated on a real device with a large-scale dataset [6], and we provide a practical guideline for network design that maximizes the benefit of INSTA-BNN while minimizing the increased overhead of it. With the aid of the guidelines, INSTA-BNN achieves higher accuracy with a similar number of parameters/operations and latency compared to previous works. We claim that the proposed INSTA-BNN can be an attractive option for BNN design.

## 2. Related works

**Network Binarization** Network binarization can significantly reduce the size of a model and the compute cost by reducing the precision of weights and activations to 1-bit. Hubara *et al.* [10] proposed to binarize both the weights and activations of a model, and suggested to use the straight-through-estimator (STE) [1] for binarization function. XNOR-Net [23] introduced the real-valued scaling factor for binarized weights and activations to minimize the quantization error. To increase the representational capability of a model, a real-valued shortcut was suggested in Bi-real-net [17]. There are several more approaches to improve the accuracy of binarized networks, such as exploiting extra component [20, 30], new model structures [3, 17], and advanced training techniques [11, 20, 22]. In most previous BNNs, binarization of weights and activations were conducted with the Sign function:

$$x_b = \text{Sign}(x_r) = \begin{cases} +1, & \text{if } x_r \geq 0 \\ -1, & \text{if } x_r < 0 \end{cases} . \quad (1)$$

In case of activations, Eq. (1) is also called the binary activation function. The  $x_r$  indicates a real-valued pre-activation value, and  $x_b$  is a binarized (+1 and -1) activation value. The value  $x_r$  is compared with the threshold of the binary activation function.

**Threshold Optimization** Several works proposed to modify the thresholds of binary activation functions to improve the accuracy of BNNs. ReActNet [16] proposed the activation function with learnable threshold for which channel-wise parameters were trained via back-propagation. Although the proposed technique improved the accuracy of BNNs, the learned thresholds were fixed after training so that the same thresholds were always used for every input. Kim *et al.* [12] pointed out that back propagation alone cannot train the thresholds to have an optimal value. They demonstrated that the distribution of learned thresholds strongly depended on the initial values and did not deviate from the initial point much. While Kim *et al.* showed that initializing the thresholds with small positive values improved the accuracy of BNNs, they did not provide how to find optimal threshold values analytically. In summary, previous methods which tried to find optimal thresholds are sub-optimal, and all of them used the fixed threshold values during inference, failing to consider instance-wise statistical information of activation distributions.

## 3. Instance-Aware BNN

### 3.1. Importance of instance-wise threshold

Compared to full precision DNNs, BNNs are more sensitive to the distribution of pre-activation values since the binarized activations are constrained to either +1 or -1. Fig. 1a shows the distribution of pre-activations for three different images. Note that all three distributions are from the same layer and channel of a model, and they are the outputs after the batch normalization layer. As shown in the figure, the mean of each distribution has a non-zero value because, even with batch normalization, the mean value of activation depends on the distribution of the corresponding input instance. However, the small mean drift in the pre-activation results in a large difference of binarized activation. In Fig. 1a, while the two distributions (left and center) have mean values that are relatively close to 0 (-0.23 and +0.32), the ratio of +1’s shows a large difference (32% and 60%). The static threshold of BNNs, combined with the insufficient expression capability, is not enough to generate high-quality binary outputs adequately aware of the delicate change of input data.

To address the aforementioned issue, we introduce a novel module that determines the quantization threshold considering the instance-wise information. Fig. 2a shows the conventional block structure used in ReActNet [16]. To use the channel-wise learnable thresholds, ReActNet modified the Sign activation function as follows:

$$x_b = \begin{cases} +1, & \text{if } x_r \geq \alpha \\ -1, & \text{if } x_r < \alpha \end{cases} . \quad (2)$$

Note that all the learnable variables used in this section are

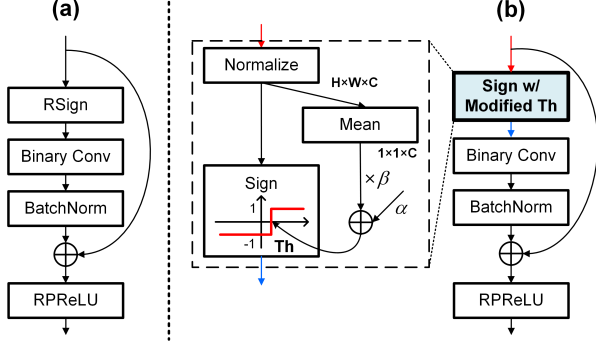


Figure 2. Convolution blocks and the proposed module structure that uses the instance-wise mean information. (a) Binary convolution block structure without instance-aware modules. (b) Binary convolution block that contains the proposed module.

channel-wise variables.

While the parameter  $\alpha$  was trained using back-propagation in ReActNet, we propose to replace it with an instance-aware threshold ( $TH$ ) as follows:

$$x_b = \begin{cases} +1, & \text{if } x_r \geq TH \\ -1, & \text{if } x_r < TH \end{cases} \quad (3)$$

The instance-aware threshold is updated using per-instance statistical information, and we need to design an appropriate module for it. The module should be simplified as much as possible to minimize the computation/storage overhead, but it should be able to reflect the key information of input activation.

The most straightforward implementation is extracting input data information and linearly combining it to update  $TH$ . We validate the effect of instance-wise  $TH$  based on the mean of activation, which has a critical impact on the ratio of +1's and -1's, as shown in Fig. 1a. To maximize the benefit of  $TH$  generation process, we utilize the difference between the batch-wise statistics and instance-wise statistics to see how much each instance deviates from the batch statistics. To calculate the drift between the per-instance mean and the mean over the entire dataset, we first normalize the pre-activation values using the Batch Normalization (BN) layer without applying the affine transformation. After the normalization layer, the instance-wise mean is calculated as follows:

$$\tilde{x}_r = \frac{x_r - \hat{\mu}}{\hat{\sigma}} \quad E[\tilde{x}_r] = \frac{1}{H \times W} \sum_{i=1}^H \sum_{j=1}^W \tilde{x}_r(i, j), \quad (4)$$

where  $H$  and  $W$  are the height and width of the feature map. Using the instance-wise mean information, we formulate the instance-aware threshold as follows:

$$TH = \alpha + \beta E[\tilde{x}_r]. \quad (5)$$

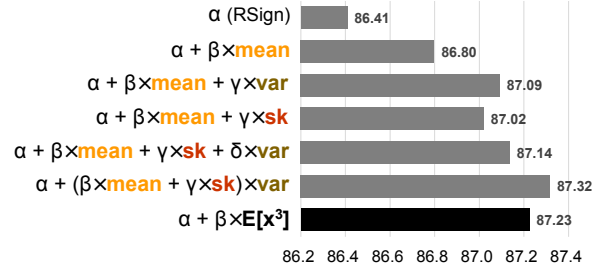


Figure 3. Test accuracy of binary ResNet-20 based model on CIFAR-10 dataset when different instance-wise statistic information are used for thresholds. Various combinations of mean, variance (var), and skewness (sk) are evaluated.

The parameters  $\alpha$  and  $\beta$  are learned via back-propagation and jointly trained with model weights. In addition to the learnable parameter  $\alpha$  that is used in RSign (Eq. (2)) of ReActNet, we use an additional scaling factor  $\beta$  which is related to instance-wise statistics. The proposed module is shown in Fig. 2b.

We evaluate the effect of instance mean-aware thresholds on the ResNet-20 based model trained on the CIFAR-10 dataset. We compare the average accuracy of 6 runs with different random seeds. As shown in the top two rows in Fig. 3, the proposed instance mean-aware threshold improves the accuracy by 0.39% compared to the input-agnostic learnable threshold of RSign.

### 3.2. Importance of higher-order statistics information

In the previous section, the simple instance-aware threshold works successfully with the mean of activation. However, we can easily identify that higher-order statistics, e.g. variance and skewness of distribution, also well describe the important characteristics of activation (Fig. 1b and c). In this work, the skewness of data is estimated based on Fisher's moment coefficient of skewness [31]:

$$\begin{aligned} \text{Skewness}(X) &= E \left[ \left( \frac{X - \mu}{\sigma} \right)^3 \right] \\ &= \frac{E[X^3] - 3\mu E[X^2] + 3\mu^2 E[X] - \mu^3}{\sigma^3} \quad (6) \\ &= \frac{E[X^3] - 3\mu\sigma^2 - \mu^3}{\sigma^3}. \end{aligned}$$

As shown in the Fig. 1b and c, even when the mean values are comparable, the characteristics of the binary activation could be highly varied depending on the variance or skewness of data distribution. Especially the skewness of distribution has been neglected so far, but it contains extra information that is not captured by mean and variance values (Fig. 1c). Therefore, instance-wise  $TH$  module should consider higher-order statistics jointly.

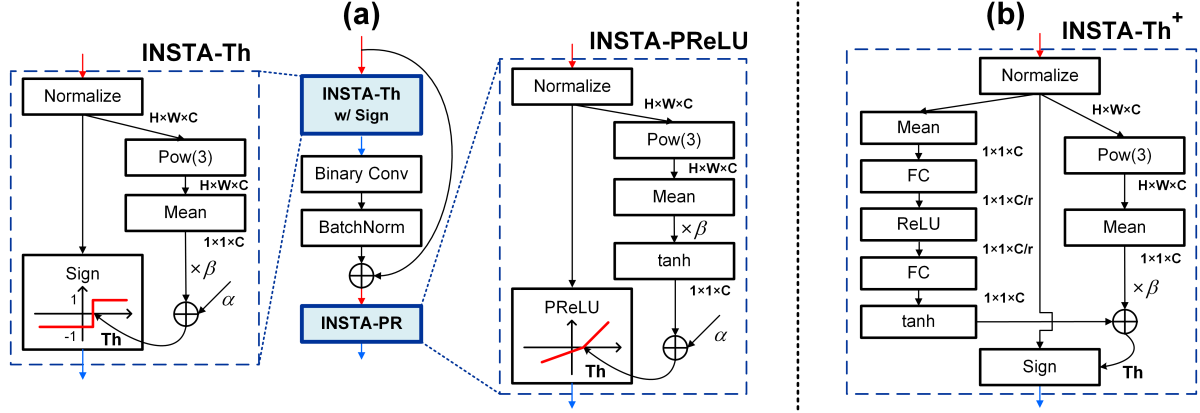


Figure 4. Convolution blocks and the proposed module structures of INSTA-BNN. (a) Binary convolution block that contains the proposed INSTA-Th and INSTA-PReLU modules. (b) INSTA-Th<sup>+</sup> module structure with SE-like block. INSTA-Th<sup>+</sup> can replace the INSTA-Th module in the structure of (a). Pow(3) indicates cubic operation,  $x^3$ .

In order to validate the importance of higher-order information, we modify  $TH$  (Eq. (5)) in a diverse way to add the influence of variance and skewness information on thresholds (Fig. 3). As expected, adding more statistical information for the instance-aware threshold improves the accuracy even further. When  $TH = \alpha + (\beta \times \text{mean} + \gamma \times \text{skewness}) \times \text{var}$ , we could achieve 87.32% accuracy which is about 0.9% higher than the baseline that uses Re-ActNet’s learnable threshold.

While using more per-instance statistical information improves accuracy, computing such information for each instance requires a high computing cost. For example, calculating all of mean, variance, and skewness of pre-activation for each instance and each channel is highly costly, incurring long inference latency. Here we introduce a simple alternative way to consider instance-wise mean, variance, and skewness at the same time. First, we can re-write the Eq. (6) as follows:

$$E[X^3] = (\text{Skewness}(X))\sigma^3 + 3\mu\sigma^2 + \mu^3. \quad (7)$$

We can see that the equation has information of mean, variance, and skewness and produces similar accuracy (87.23%) to the case that achieves the highest accuracy, 87.32% (Fig. 3). Therefore, we propose to use higher-order  $E[X^3]$  term instead of calculating all the mean, variance, and skewness of the pre-activation distribution separately. The proposed threshold can be expressed as:

$$TH = \alpha + \beta E[\tilde{x}_r^3] \quad (8)$$

$$x_b = \begin{cases} +1, & \text{if } \tilde{x}_r \geq TH \\ -1, & \text{if } \tilde{x}_r < TH \end{cases}$$

We name this threshold as **INSTA**n**ce**-**A**ware **T**h**re**sh**o**ld (**INSTA-Th**). The INSTA-Th module is shown in Fig. 4a.

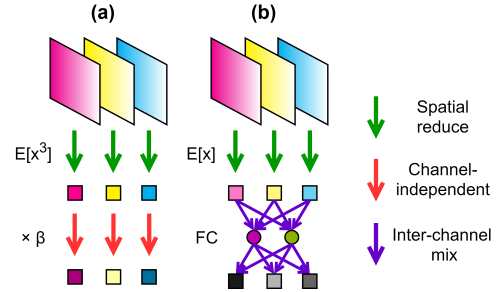


Figure 5. Comparison of proposed INSTA-Th and SE module. (a)  $\beta E[\tilde{x}_r^3]$  component of INSTA-Th. (b) SE module. Non-linear functions are skipped for simplicity.

### 3.3. Squeeze-and-Excitation Module

Some of recent BNN works [20] used the Squeeze-and-Excitation (SE) module [8] to recalibrate the channel-wise magnitude based on a kind of instance-wise information. Although the SE module and INSTA-Th may look similar in that both of them use instance-wise information, there is a notable difference. As shown in Fig. 5, the SE module uses two fully-connected layers to assess the relative importance among different channels. On the other hand, the proposed INSTA-Th computes  $E[X^3]$  value channel-wise and then propagates the information to each channel independently. Note that the INSTA-Th focuses on the difference between batch-wise and instance-wise statistical information rather than the difference between channels.

Since the proposed INSTA-Th module and the SE module have different roles, the SE module can be combined with our INSTA modules for additional accuracy improvement. In Eq. (8), we modify the parameter  $\alpha$  to be the output of the SE-like module instead of the channel-wise learnable parameter using back-propagation, as shown in Fig. 4b. We replace the Sigmoid function in the original SE module with

tanh function. To be more specific, we use the empirically found  $3*\tanh(x/3)$  to control the output range (Refer to Sec. F of the supplementary material for more detail). In such a way, we intend  $\alpha$  to focus on inter-channel relationship and  $\beta E[X^3]$  to focus on instance-wise information. Note that  $\alpha$  is also an instance-aware variable in this case while it was a fixed value in previous works as in Eq. (2). We name the new threshold **INSTA-Th<sup>+</sup>**. While INSTA-Th<sup>+</sup> requires a larger number of parameters than INSTA-Th, it can be clearly seen that using both INSTA-Th and SE-like modules improves the accuracy by a large margin (see Sec. 5.2). Therefore, we leave it as an option to select either INSTA-Th or INSTA-Th<sup>+</sup> depending on the requirement for the accuracy and parameter size.

### 3.4. Instance-aware PReLU

In recent works, additional activation functions such as ReLU and PReLU were used in BNNs for intermediate real-valued activations. ReActNet used additional learnable parameters for these PReLU layers (RPReLU) as well as the Sign layers (RSign) as shown in Fig. 2a. In addition to the INSTA-Th, we also propose replacing the learnable parameter (learnable shifts for x-axis) for RPReLU layers in ReActNet with instance-aware alternatives. We left the y-axis shift of RPReLU as it is. Fig. 4a shows the proposed **INSTance-aware PReLU (INSTA-PReLU)** which replaces the PReLU. In the INSTA-PReLU, we constrain the output range of  $\beta E[\tilde{x}^3]$ . In case of the INSTA-Th, the output range was not important since the end layer of INSTA-Th was the Sign function. However, as the PReLU layer does not constrain its output, extremely large values from the cubic operation may cause an unrecoverable problem in the network. Therefore, we used the additional tanh function to constrain the output of the  $\beta E[\tilde{x}^3]$  term. The computation process of the modified PReLU layer is as follows:

$$\begin{aligned} \tilde{x} &= \frac{x - \hat{\mu}}{\hat{\sigma}} & TH_{PR} &= \alpha + \tanh(\beta E[\tilde{x}^3]) \\ y &= \begin{cases} \tilde{x} - TH_{PR}, & \text{if } \tilde{x} \geq TH_{PR} \\ \gamma(\tilde{x} - TH_{PR}), & \text{if } \tilde{x} < TH_{PR} \end{cases} \end{aligned} \quad (9)$$

Here,  $\gamma$  is a learnable slope parameter of the PReLU layer. We can further control the output range of  $\beta E[\tilde{x}^3]$  term by replacing the  $\tanh(x)$  function with the  $c*\tanh(x/c)$ . We empirically found that  $c=3$  worked well, so we chose to use  $3*\tanh(x/3)$  for all other experiments. The effect of INSTA-PReLU will be discussed in Sec. 5.4.1. Meanwhile, the parameter  $\alpha$  in INSTA-PReLU (Eq. (9)) can also be replaced by the SE-like module similar to the case in the previous section. We call the module as **INSTA-PReLU<sup>+</sup>**. In the remainder of the paper, we refer to the proposed BNN model as **INSTance-aware BNN (INSTA-BNN)**, which employs both INSTA-Th and INSTA-PReLU modules.

layer name	Layers using INSTA modules				Top-1 Acc.(%)
	conv2_x	conv3_x	conv4_x	conv5_x	
width	64	128	256	512	
original					62.1
INSTA-BNN	✓	✓	✓	✓	62.1
INSTA-BNN variants	✓	✓	✓	✓	61.8
	✓	✓	✓	✓	61.7
	✓	✓	✓		60.9
				✓	61.1
			✓		60.6
	✓				60.2
Baseline					60.0

Table 1. Variants of INSTA-BNN that selectively use INSTA modules and their ImageNet validation accuracy comparison.

## 4. Practical guidelines for INSTA-BNN

Although INSTA-Th and INSTA-PReLU modules show noticeable performance improvement (Sec. 5.2), the cubic operations inside the modules cause non-negligible computational overhead. In this section, we introduce some techniques that we used to reduce the latency overhead of the proposed INSTA-BNN model while maintaining its benefit.

### 4.1. Selective use of the INSTA module

In most convolutional neural networks (CNNs), the feature size decreases and the number of channels (width) increases in deeper layers. Meanwhile, the cubic operation within the INSTA module is an element-wise multiplication and its cost is proportional to the layer’s feature size:  $H \times W \times C$ . As a result, the shallower layers have a larger cost for cubic operations, and the cost becomes smaller in deeper layers. Therefore, in terms of the amount of computation, it is more desirable to eliminate the cubic operation in earlier layers as far as accuracy is maintained. Table 1 shows the accuracy comparison results for ResNet-18 based INSTA-BNN models in which the INSTA modules are applied to selective layers. Interestingly, the use of INSTA modules at the deeper part of the model mainly accounts for the accuracy improvement. When we apply INSTA modules to each block one by one, the results show a clear correlation and lead to a similar conclusion. Therefore, we can improve the accuracy of the model with relatively small computational overhead by skipping the INSTA modules for earlier layers.

### 4.2. Reuse of activation statistics

Since Bi-real net [17], most of BNNs have been adopting shortcuts for convolution blocks. In the proposed INSTA-BNN, the feature information used for INSTA-Th is directly transferred to INSTA-PReLU via an identity shortcut path



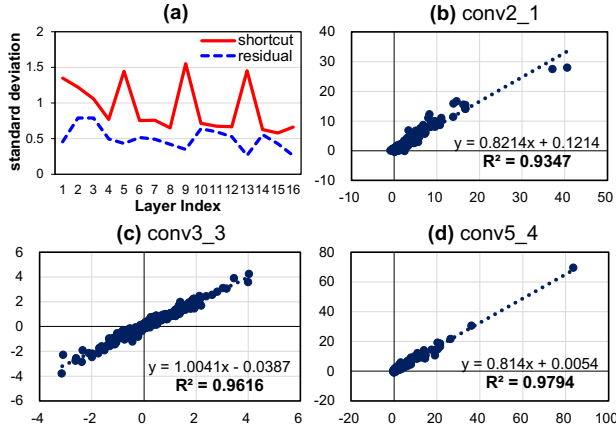


Figure 6. (a) The standard deviation of the values from the shortcut and residual right before they are added. (b)-(d) Correlation between  $E[\tilde{x}^3]$  values from INSTA-Th and subsequent INSTA-PReLU of 256 training set instances. x axis:  $E[\tilde{x}^3]$  values of INSTA-Th. y axis:  $E[\tilde{x}^3]$  values of subsequent INSTA-PReLU. Charts show the correlation at (b) the front, (c) the middle, and (d) the rear of the ResNet-18 based INSTA-BNN.

while the binary convolution output is fed to INSTA-PReLU via the residual path at the same time (Fig. 4a). We compared the strength of the values from residual path with the ones from shortcut path by calculating their standard deviations (Fig. 6a). We can observe that the response strength of the shortcut path is larger than that of the residual path, which is consistent with the argument of the ResNet [7] paper that the residual functions might have generally smaller values than the non-residual functions.

Based on the observation, we can expect that statistics ( $E[\tilde{x}^3]$ ) calculated from INSTA-PReLU module are similar to the statistics from the preceding INSTA-Th module, although they are not identical. We check their correlations for trained models. Fig. 6b-d shows that calculated  $E[\tilde{x}^3]$  values inside INSTA-PReLU are highly correlated with previous  $E[\tilde{x}^3]$  of INSTA-Th module, whereas non-linear relations are observed between INSTA-PReLU and subsequent INSTA-Th because of the intervening PReLU (Sec. E in the supplementary material). Thus, we reuse the  $E[\tilde{x}^3]$  obtained from INSTA-Th for the subsequent INSTA-PReLU module to reduce the computational burden. In order to adjust the difference caused by the residual, we apply the channel-wise affine transform to the reused  $E[\tilde{x}^3]$  value. When the shortcut contains 1x1 convolution, we calculate the feature statistics for INSTA-PReLU separately. The experimental results show that applying the two schemes, the selective use of INSTA module and the reuse of feature statistics, do not degrade the accuracy while reducing the latency significantly. Detailed results are shown in Sec. 5.2 and Sec. 5.3. Please refer to the supplementary material Sec. D for the detailed model structure, which employs the two schemes.

## 5. Experiments

We applied the proposed schemes to state-of-the-art BNN models and evaluated the accuracy improvement on ImageNet (ILSVRC12) dataset [6].

### 5.1. Experimental Setup

We used the PyTorch framework to implement state-of-the-art BNN models and the proposed INSTA-BNNs. We chose the ReActNet [16] as the baseline model. We replaced the RSign layer in ReActNet with INSTA-Th (or INSTA-Th<sup>+</sup>) module and the PReLU layer with INSTA-PReLU (or INSTA-PReLU<sup>+</sup>) module. The reduction ratio of 16 is used for INSTA-Th<sup>+</sup> and INSTA-PReLU<sup>+</sup>, if not specified. We followed the two-stage training strategy in [20]. In the first stage, we trained a model with binary activation and full-precision weights from scratch. In the second stage, we trained a BNN model by initializing it with pre-trained weights from the first stage. We used Adam optimizer with a linear learning rate decay scheduler. In both stages, we trained the model for 256 epochs with batch size of 512 and initial learning rate of 0.001. We only applied random resized crop and random horizontal flip for the training data augmentation and used default STE [1] rather than a piecewise polynomial function from [17]. We also used the distributional loss [16] using ResNet-34 as a teacher model. Note that our proposed INSTA-BNN does not rely on any specific BNN training methods. To reduce the additional parameter cost for the proposed scheme, we applied the 8-bit quantization to the weights of the SE-like module. Please refer to the supplementary materials for more details on weight quantization and cost analysis.

### 5.2. Comparison on ImageNet Classification

In this section, we compare the accuracy and cost of previous BNN models and the proposed INSTA-BNN models. In Table 2, we grouped the models with similar model parameter sizes. For the proposed models, INSTA-BNN represents a model with INSTA-Th and INSTA-PReLU modules, and INSTA-BNN<sup>+</sup> represents a model with INSTA-Th<sup>+</sup> and INSTA-PReLU<sup>+</sup> modules.

INSTA-BNN-ResNet18 / INSTA-BNN<sup>+</sup>-ResNet18 use the ReActNet-ResNet18 model as a backbone network while INSTA-BNN / INSTA-BNN<sup>+</sup> use the ReActNet-A (based on MobileNetV1) model as a backbone network. For experimental results, INSTA-BNN<sup>+</sup> (bottom row of Table 2) used the INSTA-Th<sup>+</sup> and INSTA-PReLU (without SE modules) to avoid large increase in the parameter size.

Compared to BNN models with less than 50 Mbit parameters, the proposed INSTA-BNNs with ResNet-18 backbone achieved much higher accuracy. With a small overhead, INSTA-BNN with ResNet-18 backbone achieved 67.6% top-1 accuracy which is already 2.1% higher than

Method	Bit-width (W/A)	BOPs ( $\times 10^9$ )	FLOPs ( $\times 10^8$ )	OPs ( $\times 10^8$ )	Params (Mbit)	Top-1 Acc. (%)	Top-5 Acc. (%)
ResNet18 FP [5]	32/32	0	18.2	18.2	374	69.8	89.1
BNN [10]	1/1	1.70	1.20	1.47	28.1	42.2	67.1
XNOR-Net [23]	1/1	1.68	1.40	1.66	33.4	51.2	73.2
Bi-real Net 18 [17]	1/1	1.68	1.40	1.67	33.5	56.4	79.5
XNOR-Net++ [4]	1/1	1.68	1.42	1.68	33.5	57.1	79.9
IR-Net [22]	1/1	1.68	1.40	1.67	33.5	58.1	80.0
Bi-real Net 34 [17]	1/1	3.53	1.42	1.97	43.8	62.2	83.9
Real-to-Binary [20]	1/1	1.68	1.40	1.66	42.6	65.4	86.2
ReActNet-ResNet18 [16] ‡	1/1	1.68	1.40	1.67	34.0	65.5	-
ReCU [32]	1/1	1.68	1.40	1.67	34.0	66.4	86.5
AdaBin [28]	1/1	1.68	1.43	1.70	34.0	66.4	86.5
<b>INSTA-BNN-ResNet18</b>	1/1	1.68	1.43	1.70	34.8	<b>67.6</b>	<b>87.5</b>
<b>INSTA-BNN<sup>+</sup>-ResNet18</b>	1/1	1.68	1.44	1.70	37.3	<b>68.5</b>	<b>88.2</b>
ReActNet-A [16] ‡	1/1	4.82	0.12	0.87	63.1	69.4	-
ReActNet-B [16]	1/1	4.69	0.44	1.17	68.0	70.1	-
AdaBin [28]	1/1	4.82	0.21	0.96	63.1	70.4	-
<b>INSTA-BNN</b>	1/1	4.82	0.20	0.95	65.6	<b>71.7</b>	<b>90.3</b>
DyBNN [33]	1/1	4.82	0.14	0.90	137.4	71.2	89.8
High-Capacity Expert [2]	1/1	1.70*	1.10*	1.37	83.0	71.2	90.1
ReActNet-C [16]	1/1	4.69	1.40	2.14	84.7	71.4	-
<b>INSTA-BNN<sup>+</sup></b>	1/1	4.82	0.20	0.96	71.2	<b>72.2</b>	<b>90.5</b>

Table 2. Comparison of the number of operations, parameter size, and ImageNet top-1 and top-5 validation accuracy of different BNN models. BOPs and FLOPs mean the number of binary operations and floating point operations, respectively. \* indicates that those numbers are from their original paper [2]. ‡ denotes the baseline model of the proposed INSTA-BNN(-ResNet18) models.

ResNet-18 based ReActNet. In addition, with additional parameter cost, INSTA-BNN<sup>+</sup> with ResNet-18 backbone achieved even higher accuracy (68.5% top-1 accuracy) which is 3.0% higher than baseline, and only 1.3% lower than Full-Precision ResNet-18 baseline. Similar improvement was achieved with larger BNN models. INSTA-BNN achieved 2.3% higher accuracy than ReActNet-A with marginal overhead. In case of INSTA-BNN<sup>+</sup>, it achieved much higher top-1 accuracy than the state-of-the-art models, ReActNet-C [16] and High-Capacity Expert [2], with much smaller parameter size and computing cost.

We trained the full-precision (fp) network with ReActNet structure using the official ReActNet implementation [18] and got 73.1% top-1 accuracy. Applying weight decaying not only to the convolution layers but also to the linear weights prevented the overfitting of fp model training, resulting in higher accuracy than originally reported. As a result, there is a reasonable gap between ReActNet fp model and INSTA-BNN<sup>+</sup> accuracy. The superiority of the proposed models is more clearly demonstrated in Fig. 7. The proposed INSTA-BNN models are placed in the upper-left side of previous state-of-the-art models in both accuracy vs. parameter size graph and accuracy vs. operations graph.

### 5.3. Inference latency evaluation

In the previous section, we showed that INSTA-BNNs produce higher accuracy with small operation overhead.

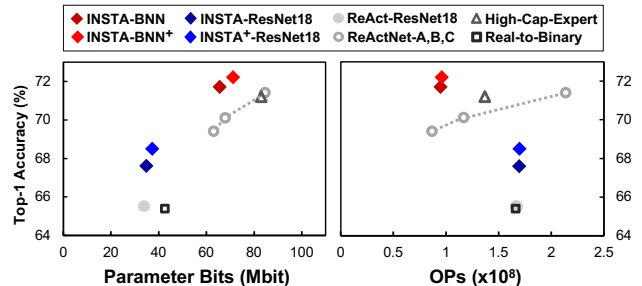


Figure 7. (a) Parameter size versus ImageNet top-1 accuracy. (b) The number of operations (OPs) versus ImageNet top-1 accuracy.

Similar to our analysis in previous section, previous works mostly reported the number of operations (OPs) to assess the computational overhead, but simple counting of OPs may not directly translate into real hardware latency characteristics. So, we also evaluated the latency of BNNs (Table 3) using the Huawei Noah’s Ark Lab’s Bolt [9] library on Google pixel 3, which has 2.80 GHz Qualcomm Kryo 385 Gold and 1.77 GHz Silver (customized ARM Cortex-A75 and A55, respectively) CPUs inside. We customized the bolt kernel to make cubic operations and element-wise addition more computationally efficient. As shown in Table 3, INSTA-BNNs have marginal overhead compared to their baselines. We report the average latency of 10 inference loops in the table. Using a single thread / Cortex-A75 core, INSTA-BNN runs with just 2 ms (6%) and 3 ms (4%)

Model	Top-1 Acc. (%)	.bolt file size (MB)	A75 latency (ms)	A55 latency (ms)
Bi-real Net 18	56.4	2.68	30	83
Bi-real Net 34	62.2	3.91	52	135
Real-to-Binary	65.4	3.34	33	93
ReActNet-RN18	65.5	2.74	32	91
<b>INSTA-BNN-RN18</b>	67.6	2.85	34	98
<b>INSTA-BNN<sup>+</sup>-RN18</b>	68.5	3.55	35	102
ReActNet-A	69.4	5.55	70	178
<b>INSTA-BNN</b>	71.7	5.79	73	192
<b>INSTA-BNN<sup>+</sup></b>	72.2	7.19	75	197

Table 3. Inference time comparison of BNNs on mobile device. Measured using a single thread. .bolt is a converted file that can be used for inference. RN18 = ResNet18.

latency overhead compared to ResNet-18 and MobileNetV1 baselines, respectively, while maintaining the higher accuracy. This shows that INSTA-BNN can operate efficiently on real hardware while achieving higher accuracy.

## 5.4. Ablation Study

### 5.4.1 Effect of individual component in INSTA-BNN

Table 4 shows the effect of each component proposed in Sec. 3 on the ImageNet top-1 accuracy. Experimental setup for the ablation study is described in Sec. A.2 of the supplementary material. First, using the instance-wise mean information on the threshold (Eq. (5)) improves the top-1 accuracy by 0.6% when compared to the baseline ReActNet model. On top of that, when variance and skewness information are added on the threshold (INSTA-Th), another 0.5% improvement is achieved. Replacing the learnable parameter  $\alpha$  in Eq. (8) with the SE-like module with reduction ratio ( $r$ ) of 8 achieves 61.7% top-1 accuracy that is 1.7% higher than baseline result. We also evaluate different combinations of INSTA-Th and INSTA-PReLU with and without the SE-like modules. When both modules use the SE-like module (INSTA-Th<sup>+</sup> and INSTA-PReLU<sup>+</sup>), we increase the  $r$  to 16 to match the additional parameter cost. Regardless of where the SE-like module is combined with INSTA modules, it improves the accuracy by a large margin. However, the cases in which only SE module is used in the proposed architecture (2nd and 3rd row of Table 4) show lower accuracy than the cases when INSTA modules are used.

### 5.4.2 Effect of the normalization layer

In Sec. 3.1, we proposed using the normalization layer before calculating instance-wise mean information to capture the difference between instance-wise and batch-wise statistical data. Although the proposed modules can still make instance-aware thresholds without the normalization layer,

Network	Top-1 Acc. (%)
Baseline (RSign + RReLU)	60.0
Baseline + SE (RSign <sup>+</sup> + RReLU) (r=8)	60.8
Baseline + SE (RSign <sup>+</sup> + RReLU <sup>+</sup> ) (r=16)	61.1
Normalization and mean (Eq. (5))	60.6
INSTA-Th	61.1
INSTA-Th <b>-norm</b>	60.8
INSTA-Th <sup>+</sup> (r=8)	61.7
INSTA-Th + INSTA-PReLU	62.1
INSTA-Th + (INSTA-PReLU <b>-norm</b> )	61.5
INSTA-Th <sup>+</sup> + INSTA-PReLU (r=8)	63.0
INSTA-Th + INSTA-PReLU <sup>+</sup> (r=8)	62.8
INSTA-Th <sup>+</sup> + INSTA-PReLU <sup>+</sup> (r=16)	62.9

Table 4. The effect of each component of INSTA-BNN on the ImageNet top-1 validation accuracy. <sup>+</sup> and  $r$  indicate the usage of SE module and the reduction ratio, respectively.

we observed that removing the normalization from either INSTA-Th or INSTA-PReLU modules limits the performance of the modules. When normalization layers are not used (**-norm** indication in the Table 4), the modules generate the threshold using the instance-wise statistical data themselves. In contrast, when the normalization layers are used, the difference between the instance-wise and overall training dataset statistics are used to generate the threshold (5th row in Table 4), resulting in better accuracy than using pure instance-wise statistics (6th row).

Ablation study for the effect of the threshold value ranges in INSTA modules, the effect of INSTA-Th for different block structure, discussion of reducing the inconsistent sign problem of binary convolution, and visualization results of t-SNE are discussed in detail in the supplementary materials.

## 6. Conclusion

In this paper, we argue that the traditional BNNs with the input-agnostic threshold are sub-optimal. Instead, we demonstrate that the instance-wise statistics, including mean, variance, and skewness, must be considered to determine the better threshold values dynamically. Based on the idea, we propose the BNN with instance-aware threshold control (INSTA-Th) and demonstrate that the proposed BNN outperforms the previous BNNs by a large margin. We further improve the performance of the BNN with INSTA-Th by adding the instance-aware PReLU (INSTA-PReLU) and a variant of the Squeeze-and-Excitation module (INSTA-Th<sup>+</sup>). Experimental results show that our INSTA-BNN<sup>+</sup> achieves the top-1 accuracy up to 72.2% on the ImageNet dataset.



## Acknowledgement

This work was supported in part by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2021-0-00105, Development of model compression framework for scalable on-device AI computing on Edge applications, and No. 2021-0-02068, Artificial Intelligence Innovation Hub).

## 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, 6**
- [2] Adrian Bulat, Brais Martinez, and Georgios Tzimiropoulos. High-capacity expert binary networks. In *International Conference on Learning Representations (ICLR)*, 2021. **7**
- [3] Adrian Bulat and Georgios Tzimiropoulos. Binarized convolutional landmark localizers for human pose estimation and face alignment with limited resources. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3706–3714, 2017. **2**
- [4] Adrian Bulat and Georgios Tzimiropoulos. Xnor-net++: Improved binary neural networks. *arXiv preprint arXiv:1909.13863*, 2019. **7**
- [5] Torch Contributors. Torchvision.models. <https://pytorch.org/vision/stable/models.html#classification>, 2017. Accessed: 2023-03-08. **7**
- [6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. **2, 6**
- [7] 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, 6**
- [8] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018. **2, 4**
- [9] Huawei-noah. Bolt. <https://github.com/huawei-noah/bolt>, 2020. **7**
- [10] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks. *Advances in neural information processing systems*, 29, 2016. **2, 7**
- [11] Hyungjun Kim, Kyungsu Kim, Jinseok Kim, and Jae-Joon Kim. Binaryduo: Reducing gradient mismatch in binary activation network by coupling binary activations. In *International Conference on Learning Representations*, 2019. **2**
- [12] Hyungjun Kim, Jihoon Park, Changhun Lee, and Jae-Joon Kim. Improving accuracy of binary neural networks using unbalanced activation distribution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7862–7871, 2021. **1, 2**
- [13] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012. **1**
- [14] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017. **1**
- [15] 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*, pages 21–37. Springer, 2016. **1**
- [16] 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, 2, 6, 7**
- [17] 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, 2, 5, 6, 7**
- [18] liuzechun. Reactnet. <https://github.com/liuzechun/ReActNet>, 2020. **7**
- [19] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015. **1**
- [20] Brais Martinez, Jing Yang, Adrian Bulat, and Georgios Tzimiropoulos. Training binary neural networks with real-to-binary convolutions. In *ICLR*. 2020. **2, 4, 6, 7**
- [21] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE international conference on computer vision*, pages 1520–1528, 2015. **1**
- [22] Haotong Qin, Ruihao Gong, Xianglong Liu, Mingzhu Shen, Ziran Wei, Fengwei Yu, and Jingkuan Song. Forward and backward information retention for accurate binary neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2250–2259, 2020. **1, 2, 7**
- [23] 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, 2, 7**
- [24] 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*, pages 779–788, 2016. **1**
- [25] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28:91–99, 2015. **1**
- [26] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. **1**

- [27] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015. [1](#)
- [28] Zhijun Tu, Xinghao Chen, Pengju Ren, and Yunhe Wang. Adabin: Improving binary neural networks with adaptive binary sets. In *European conference on computer vision*, pages 379–395. Springer, 2022. [7](#)
- [29] Peisong Wang, Xiangyu He, Gang Li, Tianli Zhao, and Jian Cheng. Sparsity-inducing binarized neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 12192–12199, 2020. [1](#)
- [30] 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. [2](#)
- [31] Wikipedia contributors. Skewness — Wikipedia, the free encyclopedia. <https://en.wikipedia.org/w/index.php?title=Skewness&oldid=1140693810>, 2023. [Online; accessed 8-March-2023]. [3](#)
- [32] Zihan Xu, Mingbao Lin, Jianzhuang Liu, Jie Chen, Ling Shao, Yue Gao, Yonghong Tian, and Rongrong Ji. Recu: Reviving the dead weights in binary neural networks. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5198–5208, 2021. [7](#)
- [33] 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. [7](#)