Improving Accuracy of Binary Neural Networks using Unbalanced Activation Distribution

Hyungjun Kim[1]   Jihoon Park[1]   Changhun Lee[1]   Jae-Joon Kim[1,2]

[1]Department of Convergence IT Engineering, [2]Graduate School of Artificial Intelligence

Pohang University of Science and Technology (POSTECH), Korea

{hyungjun.kim, jihoon.park, changhun.lee, jaejoon}@postech.ac.kr

Abstract

Binarization of neural network models is considered as one of the promising methods to deploy deep neural network models on resource-constrained environments such as mobile devices. However, Binary Neural Networks (BNNs) tend to suffer from severe accuracy degradation compared to the full-precision counterpart model. Several techniques were proposed to improve the accuracy of BNNs. One of the approaches is to balance the distribution of binary activations so that the amount of information in the binary activations becomes maximum. Based on extensive analysis, in stark contrast to previous work, we argue that unbalanced activation distribution can actually improve the accuracy of BNNs. We also show that adjusting the threshold values of binary activation functions results in the unbalanced distribution of the binary activation, which increases the accuracy of BNN models. Experimental results show that the accuracy of previous BNN models (e.g. XNOR-Net and Bi-Real-Net) can be improved by simply shifting the threshold values of binary activation functions without requiring any other modification.

1. Introduction

Deep Neural Networks (DNNs) have achieved human-level performance in many computer vision tasks such as image classification, object detection, and segmentation. However, the increased compute cost and memory requirement of large DNN models pose a burden on resource-constrained environments such as mobile devices. To mitigate this problem, various techniques including network quantization [7, 17, 23, 29, 37], network pruning [15, 34], and efficient architecture design [16, 31, 35] were introduced to reduce the compute cost and memory requirement of DNN models. Among them, the network quantization technique is being actively studied and recent works have shown that a DNN model can even be quantized to a 1-bit

model [17, 25, 26, 29]. When a DNN model is binarized to a Binary Neural Network (BNN) model, the memory requirement of the model is reduced by 32x since 32-bit floating-point weights can be represented by 1-bit weights. In addition, high precision multiply-and-accumulate operations can be replaced by XNOR-and-popcount logics in BNNs since both activation and weight have 1-bit precision. Due to the lightweight nature, BNNs are garnering interests as a promising solution for DNN computing on edge devices.

However, BNNs still suffer from the accuracy degradation caused by the aggressive quantization (32-bit to 1-bit). While recent researches have shown that a DNN model can be quantized to 2-bit precision with marginal accuracy loss [11], a severe performance gap still exists between a BNN model and its full-precision counterpart DNN model. In general, it is known that weight quantization is much easier than activation quantization [5, 20, 37]. In addition, when quantizing the activation, the accuracy loss is marginal until 2-bit quantization, but a significant accuracy drop occurs when quantizing the activation to 1-bit precision [20, 23, 27]. Previous works tried to explain the sharp accuracy drop from 2-bit activation to 1-bit case based on the gradient mismatch problem caused by the non-differentiable binary activation function [9, 25]. Since the quantization functions are non-differentiable, gradients cannot propagate through the quantization layer in the back-propagation process. Therefore, previous works used straight-through-estimator (STE) to compute the approximate gradient on non-differentiable layers [1, 17]. While STE enables back-propagation through quantization layers, the discrepancy between the actual function and the approximated function causes gradient mismatch problem. Especially in BNNs, sign function is used to binarize activations and is usually approximated as hardtanh function in back-propagation. Compared to other multi-bit quantization functions, sign function shows more severe gradient mismatch which leads to sharp accuracy degradation. Hence, several works tried to design better approximation function [9, 25] or to reduce the gradient mismatch using

neuron coupling [20].

In this work, we argue that there is another reason for the poor performance of BNN in addition to the gradient mismatch. We speculate that the symmetry of the sign function is also partly responsible for the degradation of BNN performance. Most DNN models use ReLU activation function instead of sigmoid or Tanh functions. While the output distributions of sigmoid and Tanh functions are symmetric with respect to zero, ReLU function replaces all the negative values to zero so that the distribution of ReLU output is highly skewed. When quantizing activations to 2-bit or a higher precision, ReLU-based quantization functions are usually used [6, 7, 11, 37]. In other words, multi-bit quantization functions also output unbalanced activation distributions similar to ReLU function. However, when binarizing activations, sign function is used, and hence the distribution of the binary activation becomes symmetric (or balanced). We show that a model with unbalanced activation distribution performs better than that with balanced activation distribution. We first show that the claim is valid even in the full precision activation case by comparing hardtanh and ReLU6 activation functions. We then show that the performance of BNN can be improved by simply shifting the threshold of the sign function. We also analyze the effect of training the threshold of the sign function and show that the thresholds cannot be trained efficiently through back-propagation. Our contributions can be summarized as follows:

- To the best of our knowledge, we are the first to report that the unbalanced distribution of binary activation actually helps improve the accuracy of BNNs.

- We propose to shift the thresholds of binary activation functions to make the distribution of binary activation unbalanced.

- Experimental results show that the accuracy of previous BNN models can be improved at almost no cost by simply shifting the threshold of the binary activation function.

2. Related Work

2.1. Network binarization

There have been several approaches to quantize weights and/or activations into 1-bit precision. Courbariaux et al. [8] demonstrated binary weight networks which can be successfully trained on small datasets such as CIFAR-10 and SVHN. Hubara et al. [17] further proposed BNN in which both weights and activations are binarized. To apply back-propagation through a sign function which is non-differentiable, the straight-through-estimator (STE) concept was used [1]. Rastegari et al. [29] proposed XNOR-

Net which uses real-valued scaling factors when binarizing weights and activations and demonstrated acceptable accuracy on large scale ImageNet dataset. When binarizing ResNet models, real-valued shortcut connections play a critical role in propagating high resolution information. Liu et al. [25] proposed to use additional shortcut connections so that a shortcut connection exists for every binary convolution layer.

2.2. Training quantization parameters

Recent studies suggested to train the quantization intervals and ranges using back-propagation to improve the accuracy of quantized neural networks. Choi et al. [6, 7] proposed the parameterized clipping activation function (PACT) in which the clipping range is trained using back-propagation. While only the clipping range was trained in PACT, several following works proposed to train both quantization interval and range [11, 19, 36]. Note that these works focused on training multi-bit networks and hence did not report results on BNNs. For BNNs, several recent works proposed to train the threshold of binary activation function. Liu et al. [24] proposed to use trainable activation functions so that the distribution of the activation can be balanced. Wang et al. [33] proposed the trainable binarization which learns the threshold as well as the gradient clipping range used in back-propagation. We, however, observed results contrary to the findings from previous works on trainable threshold. Our experimental results show that the bias term in Batch Normalization (BN) layer is already learning the threshold and therefore the effect of training the threshold of binary activation is limited. Details will be described in Sec. 3.4.

2.3. Managing activation distribution

Since there are only two values available for activation in BNNs, the distribution of binary activation plays a critical role in BNNs. There have been few works that tried to manipulate the distribution of binary activation to improve the accuracy of BNNs. Ding et al. [10] proposed to regularize the distribution of pre-activation values to tackle the difficulties that occurred during training BNNs. The work mostly focused on avoiding extreme cases such as the case when all the pre-activation values have the same sign. Liu et al. [24] proposed to reshape the distribution of binary activation using trainable thresholds. Using the trainable activation functions, they made the distribution of binary activation more balanced.

2.4. Additional activation function

Another simple yet effective technique to improve the accuracy of BNNs is to use an additional activation function (e.g. PReLU) between the binary convolution layer and the following BN layer. Rastegari et al. [29] men-

tioned that inserting a ReLU function after the binary convolution helps training BNNs. Tang et al. [32] proposed to use PReLU instead of ReLU to absorb the weight scale factors. Bulat et al. [2, 4] explained that increasing the non-linearity in BNNs helps training, and compared the effects of ReLU and PReLU. Based on these findings, many recent works utilized additional activation layers in their models [24, 26, 28].

3. Method

In this section, we introduce how to improve the accuracy of BNNs using unbalanced activation distribution. We first discuss the activation functions in conventional full precision models to describe the motivation for using unbalanced activation distribution (Sec. 3.1). Then we show how to make the distribution of the binary activation unbalanced using threshold shifting in BNNs and how much the proposed technique can improve the accuracy on various benchmarks (Sec. 3.2). Experimental results on ImageNet dataset will be given to show that the accuracy of various existing BNN models can be improved by simply shifting the threshold of binary activation functions (Sec. 3.3). After that, we discuss the ineffectiveness of the methodology to train the threshold of binary activation in BNNs (Sec. 3.4). We also show that the additional activation functions (i.e. ReLU or PReLU) used in recent BNN models make the activation distribution unbalanced thereby helping to improve the accuracy (Sec. 3.5).

3.1. Accuracy gap between hardtanh and ReLU6

Before ReLU activation function was proposed, sigmoid or Tanh functions had been used as an activation function. It is widely known that ReLU works better because it solves the gradient vanishing problem that occurs in sigmoid or Tanh functions [14]. However, the gradient vanishing problem is largely diminished by using BN layers in recent models [18], so there might be other reasons for the success of the ReLU function. We suspect that another main reason for the good performance of ReLU is that the output distribution of the ReLU function is highly skewed. While the output distributions of sigmoid and Tanh functions are symmetric with respect to each mean value, ReLU makes all the negative inputs to zero and passes the positive inputs. Therefore, the output distribution of the ReLU function is positively skewed (mean value is larger than the median value). We conducted a few experiments to monitor whether the unbalanced activation distribution due to the nature of ReLU function helps to improve accuracy. To make the problem simple, we compare the performance of hard hyper-tangent (hardtanh) and ReLU6 functions instead of ReLU. The ReLU6 is a slight variant of the ReLU function where its positive outputs are clipped to 6. The hardtanh function is shown in Fig. 1a. It has been reported that
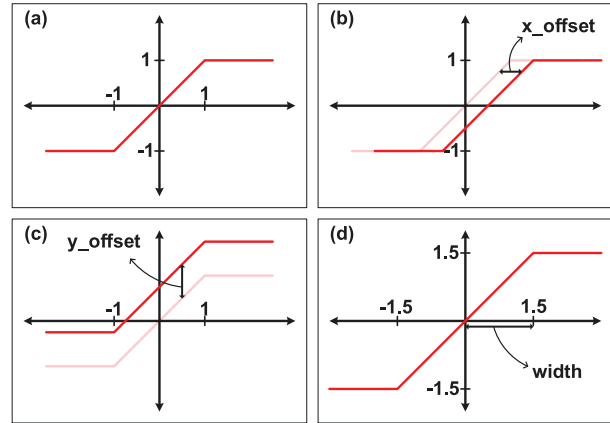

Figure 1. (a) Original hardtanh function. Hardtanh function shifted along (b) the x-axis by x_offset and (c) the y-axis by y_offset. (d) Hardtanh function with the increased range.

the performance of ReLU6 is as good as that of ReLU or sometimes even better [22]. Also, ReLU6 and hardtanh have very similar shape and ReLU6 can be thought as a shifted-and-scaled form of hardtanh. As shown in Fig. 1, we modified the hardtanh function in three different ways; (b) shifting along the x-axis, (c) shifting along the y-axis and (d) increasing the range. Note that the hardtanh function is identical to the ReLU6 function when it is shifted along the x- and y-axis by 3 and its range is increased by 3 times. We trained vgg-small model [30] on CIFAR-10 dataset with different activation functions. The vgg-small model has 4 convolution layers with 64, 64, 128, 128 output channels in sequence followed by 3 fully-connected layers with 512 neurons. We trained the model in the same condition with 10 different seeds and report the mean and the standard deviation of the test accuracy. Detailed setup for the training is described in the supplementary material. When ReLU6 function is used as the activation function, 89.21% of test accuracy was achieved while 88.55% was achieved when hardtanh function is used without any modification. When the hardtanh activation function is modified as described in Fig. 1, the test accuracy of the model also changes. Fig. 2 shows the change in the test accuracy depending on the amount of shift along (a) the x-axis or (b) y-axis or (c) increase in the range of the hardtanh activation function. We observe that shifting the hardtanh activation function along the x-axis increases the accuracy substantially (Fig. 2a). The highest test accuracy was obtained when shifting the hardtanh function in a positive direction along the x-axis by 1.2. On the other hand, shifting the hardtanh activation along the y-axis or increasing the range does not improve the accuracy as much as shifting the hardtanh along the x-axis (Fig. 2(b and c)). When the hardtanh function is shifted to the right along the x-axis, the number of negative outputs increases and the output distribution
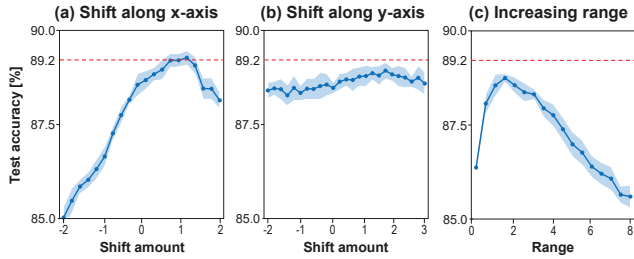
Figure 2. Test accuracy of vgg-small model with modified hard-tanh activation functions. Red dotted line represents the test accuracy of the model with ReLU6 activation function. The mean (line) and standard deviation (shade) of 10 runs are plotted.
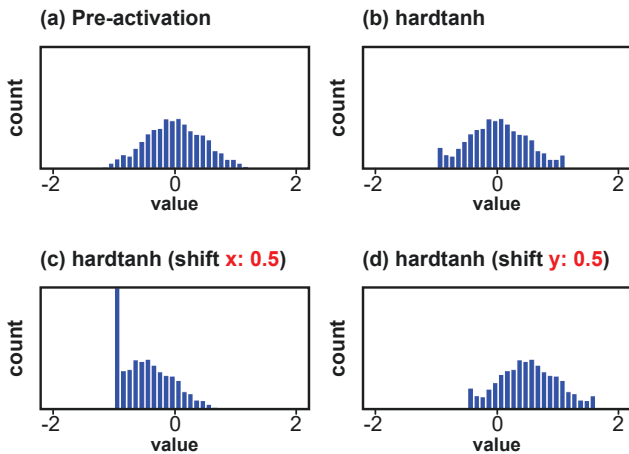

Figure 3. Distributions of (a) pre-activation, (b) output of hard-tanh activation function, (c) output of hardtanh activation function shifted to the right along the x-axis by 0.5, and (d) output of hard-tanh activation function shifted up along the y-axis by 0.5.

becomes positively skewed (Fig. 3c). Note that shifting the hardtanh function along the y-axis also makes the output activation distribution to have a non-zero mean. However, it only shifts the activation distribution and the distribution is not skewed in such a case (Fig. 3d). Therefore, we think that the higher performance of ReLU activation function compared to the hardtanh partly comes from the unbalanced distribution of activation outputs. Interestingly, the effect of the activation distribution becomes even more noticeable in BNNs when the hardtanh function is replaced by the sign function.

3.2. BNNs with unbalanced activation distribution

As described in the previous section, breaking the balance of the activation distribution by shifting the activation function along the x-axis helps improve the accuracy of a model. When quantizing activations, previous multi-bit quantization methods [5, 6, 7, 11, 12, 19, 27, 37] used ReLU-based quantization which outputs unbalanced activation distributions. However, in BNNs, previous works used sign function for binary activation function thus the distri-

bution of binary activation is balanced [3, 9, 10, 17, 25, 29]. Note that in BNNs, there are only two output values (+1 or -1) and the balanced distribution means that the ratio of +1 to -1 is close to 1:1. We believe that the poor performance of BNNs observed in previous literature [23, 27] is partly due to the use of the sign function. Since shifting the hardtanh function along the x-axis improved the test accuracy in the full precision model, we also tried shifting the sign function along the x-axis. Note that shifting the sign function is the same as just changing the threshold value of the activation function. As expected, shifting the activation function along the x-axis increases the accuracy of BNNs also (Fig. 4a). To check the effect of the shape of the distribution of activation outputs, we observed the distributions of the binary activation in different layers. Fig. 4b shows the distribution of the binary activations of the first, third, and fifth activation layer when the original sign function is used. For the first and the fifth activation layer, the distribution of pre-activation is Gaussian-like with zero mean and hence the ratio of +1 to -1 in binary activation is close to 1:1. In the case of the third activation layer, the distribution of pre-activation values does not have zero mean due to the preceding Max Pooling layer even when the original sign function is used. Note that the convolution results go through the Max Pooling layer first and then a BN layer before the binary activation layer. The Max Pooling layer makes the distribution positively skewed, thus the mean of the distribution becomes larger than the median of the distribution. Since the BN layer centers the distribution based on the mean value, there exist more negative values than positive values in pre-activation values. As a result, the number of -1 is larger than that of +1 in the binary activation after the third activation layer (Fig. 4b). However, the output distributions of all other layers which do not come after Max Pooling layer are balanced. When the threshold of binary activation function is shifted by 1.2, which results in the best accuracy, the distribution of binary activation changes. Fig. 4c shows the distribution of the binary activations when the threshold is shifted by 1.2. As expected, the output distributions of all the binary activation layers become unbalanced with the shifted threshold. Note that even though the distribution of binary activation in a layer is unbalanced, the imbalance does not propagate through layers because the binary weights in the following binary convolution layer are zero-centered.

3.2.1 Effect of Max Pooling

Another interesting observation from the results of shifting the activation function along the x-axis was that the accuracy improved only when the activation function was shifted in a positive direction. Since the Max Pooling layer is the only layer that gives asymmetry in the model, we suspect
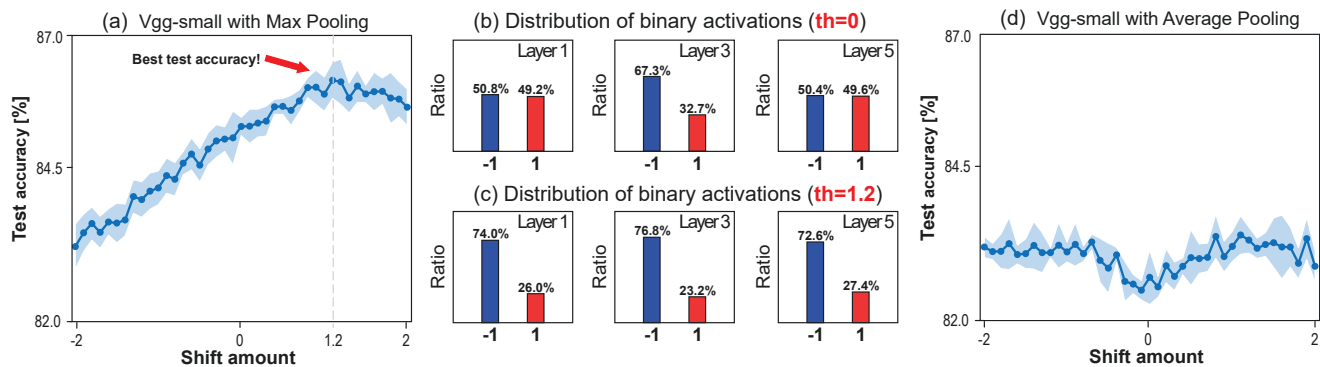
Figure 4. (a) Test accuracy vs. threshold shift for vgg-small model. (b) Distribution of the binary activation of the first, third, and fifth activation layer when threshold shift = 0. (c) Distribution of the same data with threshold shift = 1.2. (d) Test accuracy vs. threshold shift when Max Pooling layers are replaced by Average Pooling layers.
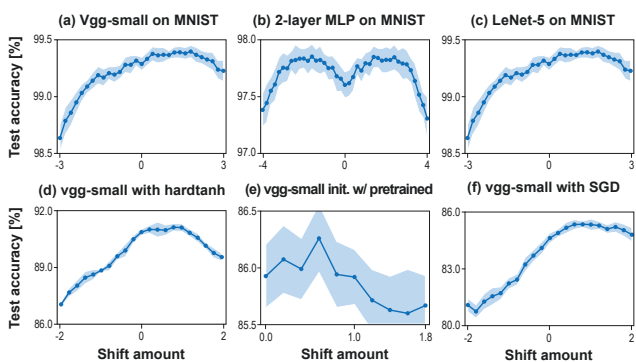

Figure 5. Test accuracy vs. threshold shift for (a) binary vgg-small model on MNIST dataset, (b) binary 2-layer MLP model on MNIST dataset, (c) binary LeNet-5 on MNIST dataset, (d) full-precision vgg-small model on CIFAR-10 dataset, (e) binary vgg-small model initialized with full-precision pretrained model, and (f) binary vgg-small model optimized using SGD.

that Max Pooling causes the accuracy difference between shifting of the threshold to the positive direction and the negative direction. To gauge the effect of the Max Pooling, we replaced the Max Pooling layers with Average Pooling layers which do not have asymmetric effects. Fig. 4d shows the training results. Since the model with Average Pooling is symmetric, the effect of shifting the activation function is also symmetric and we can observe that the accuracy is improved when the threshold is shifted away from zero in either direction.

3.2.2 Effect of other conditions

In this section, we investigate the effect of threshold shifting with various (1) datasets, (2) models, (3) initialization methods, and (4) optimizers to show that the proposed technique is not limited to the specific condition or benchmark.
Dataset. We first changed the dataset to MNIST. To solely investigate the dependency on dataset, we used the same

vgg-small model which we used for CIFAR-10 in Sec. 3.2. For MNIST dataset, we trained each model with 30 different random seeds and average results are presented. As shown in Fig. 5a, the accuracy of the vgg-small model can also be improved by shifting the threshold when trained on MNIST dataset. We also evaluate the ImageNet dataset and the results will be described in the Sec. 3.3.2.

Model architecture. We also evaluated two additional model architectures (2-layer MLP and LeNet-5) to verify the effect of threshold shifting. Fig. 5b and c show the result of MLP and LeNet-5, respectively. Although the accuracy improvement with threshold shifting is relatively small (~0.2%), a clear trend is observed in the test accuracy as the activation functions are shifted along the x-axis in both models. Note that the trend in the MLP model is symmetric because the model do not have the Max Pooling layer in it. We also conducted experiments on much larger and complex models (AlexNet and ResNet), and the results will be described in the Sec. 3.3.2.

Initialization method. In all the previous experiments, we trained the models from scratch using Xavier normal initialization [13]. In recent literature [25, 26], pretrained full-precision models are often used to initialize BNN models. We also demonstrate that the proposed technique is effective with such an initialization method. As shown in Sec. 3.1, the accuracy of a model with hardtanh function can also be improved by shifting the activation function to the right along the x-axis. Since pretrained full-precision models typically use hardtanh function instead of ReLU function to minimize the mismatch with the sign activation function in the initialization stage, we shifted the hardtanh function in the full-precision model and used the results as the initialization points. Fig. 5d shows the training results of vgg-small model with full-precision weight and activation. We initialized the BNN version of vgg-small model with the pretrained model in which the hardtanh function is shifted by 1. While using the same shift amount for binary ac-
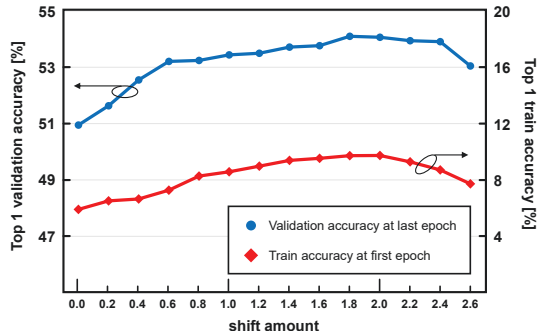
Figure 6. Top-1 train accuracy at the first epoch and Top-1 validation accuracy at the end of training with various threshold shift amount. The two graphs show similar trends.

tivation and hardtanh function seems natural, there might be a better shift amount for the binary activation other than the shift amount for the pretrained model. Therefore, we searched for the optimal shift amounts for the pretrained model and BNN model separately. Fig. 5e shows the trend of test accuracy versus the threshold shifting amount. While the pretrained model used 1.0 as the shift amount, the best shift amount for the BNN model was 0.6 so using different amounts of shift for the pretrained model and the BNN produces higher accuracy.

Optimizer. Although most previous BNN training methods used Adam optimizer [21], we also tested SGD with momentum to see the sensitivity to the optimizer. Fig. 5f shows that a similar trend is maintained with SGD while the accuracy is slightly degraded compared to the Adam case.

3.3. Experiments on ImageNet

In this section, we apply the threshold shifting technique to various previous BNN models on ImageNet dataset and show that the technique can be added to existing BNN training techniques without requiring any modification other than the simple threshold shift. We also introduce a simple method to find the appropriate shifting amount for each model to avoid time-consuming search for optimal threshold shift for large neural networks.

3.3.1 Finding shift amount

As shown in previous results, the best shift amount differs across models and datasets. To find the optimal shift amount, we propose to use the training accuracy in earlier training steps. For example, we choose the best shift amount based on the train accuracy at the end of the first epoch for ImageNet dataset. This approach is valid because the trend in the train accuracy at the first epoch and that in the validation accuracy after training is finished are similar enough. Fig. 6 shows the two trends when training XNOR-Net (ResNet-18) [29] on ImageNet dataset. The shapes of the two trends resemble each other, and both trends show

the peak performance when their shift amounts are around 2.0. Using the proposed method, we can find the optimal value for the threshold shift easily without training the model to the end for every shift amount.

3.3.2 Results on ImageNet

Using the proposed method to find optimal threshold shift, we trained several previous BNN models for ImageNet dataset with the threshold shifting technique. Table 1 shows how much accuracy improvement can be achieved by the proposed threshold shifting technique on different BNN models. We first reproduced the baseline accuracy of previous works with less than 0.3% accuracy difference. With the same hyper-parameter setting, we only changed the threshold of binary activation functions in each model. The shift amount for each model is shown in the Table 1. The results indicate that the proposed threshold shifting technique also works well for large models on ImageNet dataset. In addition, the proposed threshold shifting technique can be easily combined with other BNN training methods (i.e. weight scaling factor and double skip connection).

3.4. Effect of training threshold

In several recent works, it has been proposed to train the interval and range of quantization functions via back-propagation [7, 11, 19, 36]. Training the threshold of binary activation function in BNNs has been also proposed recently [24, 33]. The threshold training approaches are based on the belief that the best threshold value for each binary activation function can be found using back-propagation. However, here we show that training the threshold has a limited effect on the performance of BNNs.

3.4.1 Batch normalization bias

The most important reason why the threshold training is not effective is that the role of trainable threshold is already covered by the bias values in the BN layer which comes right before the binary activation layer. The compute process of a BN layer and the following binary activation layer can be represented as

$$Y = \begin{cases} -1 & \text{if } \gamma\frac{X-\mu}{\sigma} + \beta \leq th \\ +1 & \text{if } \gamma\frac{X-\mu}{\sigma} + \beta > th \end{cases}. \quad (1)$$

Here, X and Y are inputs to the BN layer and output from the binary activation layer, respectively. μ, σ, γ, and β are mean, standard deviation, weight, and bias of the BN layer, and th is the trainable threshold. As shown in the Eq. 1, the bias term in BN serves exactly the same role as the trainable threshold. Both of them are initialized to zero in the beginning and their gradients are values of the same magnitude and the opposite sign. We trained the vgg-small model

Table 1. Accuracy of previous BNN models trained on ImageNet dataset with and without the threshold shifting technique. [†]For Bi-Real-Net, the shift amounts for the pretrained model and the BNN model are shown together.

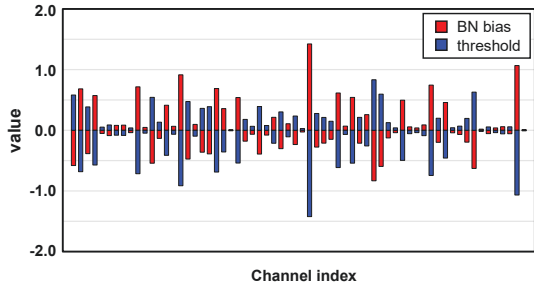| Model	Baseline accuracy [%]		Shift amount	Accuracy w/ shift [%]		Accuracy gain [%]	
	Top-1	Top-5		Top-1	Top-5	Top-1	Top-5
BNN (AlexNet)	41.5	66.1	0.6	42.1	66.6	0.6	0.5
XNOR-Net (AlexNet)	44.4	68.6	0.8	45.6	69.6	1.2	1.0
XNOR-Net (ResNet-18)	51.2	74.9	2.0	54.2	77.6	3.0	2.7
Bi-Real-Net (ResNet-18)	56.1	79.1	(2.0, 1.6)[†]	57.2	80.2	1.1	1.1
Bi-Real-Net (ResNet-34)	61.9	83.9	(2.0, 1.8)[†]	62.8	84.5	0.9	0.6

Figure 7. Comparison of the BN bias values in the second BN layer and the threshold values in the following activation layer of the vgg-small model. The thresholds are trained following the approach suggested in [24].
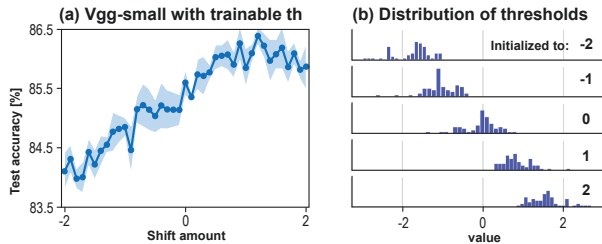

Figure 8. (a) Test accuracy of vgg-small model with trainable thresholds with various initial threshold values. (b) Distribution of effective thresholds $(th - \beta)$ of the second binary activation layer after training is finished. Each distribution from top to bottom represents the case when the thresholds are initialized to -2, -1, 0, 1, and 2, respectively.

with trainable thresholds following [24] and observed the bias values of the second BN layer and the threshold values of the following binary activation layer. Fig. 7 shows the values of BN biases and the trained thresholds in the vgg-small model. For each channel, the BN bias and the threshold have the same values with an opposite sign. Therefore, training the thresholds of binary activation functions means nothing more than doubling the learning rate of BN biases. Please refer to section S2 in the supplementary material for more discussion.

3.4.2 Limited learning capability

While the shifted threshold values in the proposed method can also be absorbed by the bias values in BN after a train-

ing is finished, it differs from the trainable threshold method in that the fixed shift amount of the binary activation function serves as an initialization point as well. If the BN bias and the threshold can be trained to an optimal point via back-propagation from an arbitrary initial point, the threshold shifting technique might not be as effective. We, however, found that the final BN bias and the trainable threshold values heavily depend on the initial values, and hence initializing them with the proposed threshold shifting technique strongly affects the training performance. Fig. 8a shows the training result of vgg-small model with trainable threshold. To see the effect of initialization values, we varied the amount of the threshold shift and applied the trainable threshold methods. Similar to the result with a fixed threshold, the test accuracy strongly depends on the initialization point (or threshold shifting amount). We further analyzed the distribution of effective threshold $(th - \beta)$ after training is finished for five different initialization values. As shown in Fig. 8b, the final distribution of the effective threshold strongly depends on the initial values and do not change much from the initial points. For example, the first case (top) in Fig. 8b is when the threshold of the binary activation function is set to -2 and the BN bias is initialized to 0. Even though the thresholds of the binary activation functions, which are initially -2, are trainable, back-propagation does not train them to the optimal value which is close to 1.

3.5. Effect of additional activation function

Recently, several works proposed to use additional activation function (i.e. PReLU) after convolution layer in BNNs [2, 4, 24, 26, 28, 32]. While the performance improvement by the additional activation function was significant, the reason for the accuracy improvement is not clearly understood yet. Previous works have described that BNNs usually lack nonlinearity in the model due to the simple activation functions and the additional activation functions give more nonlinearity to the model [2, 29]. In addition to the increased nonlinearity, we observed that the accuracy improvement by additional activation function is also related to the unbalanced activation distribution. We first evaluate the effect of additional PReLU layers after binary convolution layers in the ResNet-20 model. For experiments, we
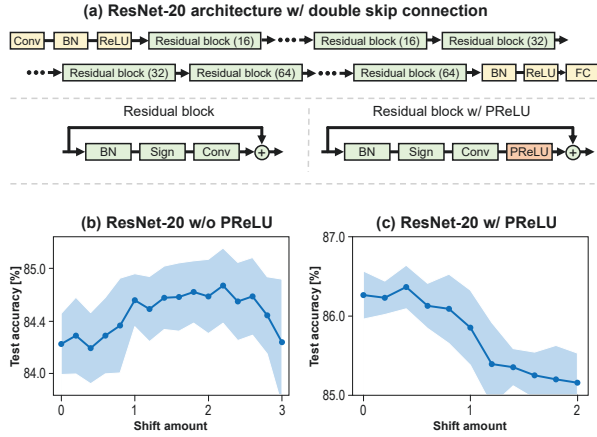
Figure 9. (a) Model description of ResNet-20 with double skip connection. ResNet-20 with PReLU model has an additional PReLU layer at the end of every residual block. Test accuracy of ResNet-20 (b) without and (c) with an additional PReLU layer after every binary convolution layer.
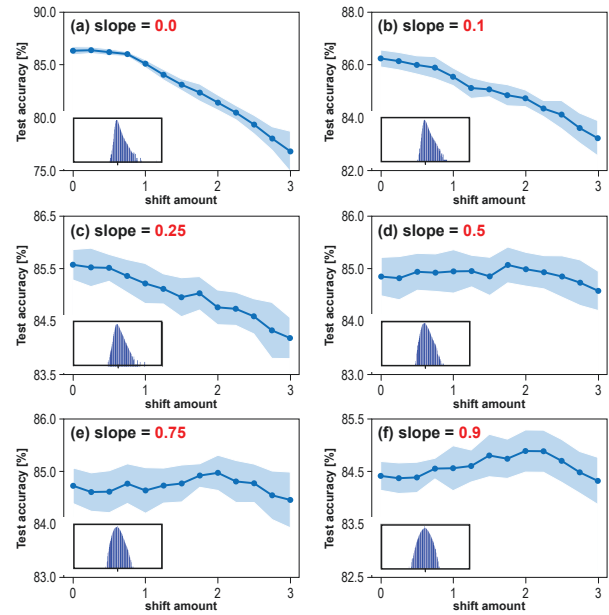

Figure 10. Effect of the slope of the negative range of LeakyReLU layers on the dependence of the test accuracy on the threshold shift amount. The slope of the negative range of LeakyReLU is (a) 0.0, (b) 0.1, (c) 0.25, (d) 0.5, (e) 0.75, and (f) 0.9. Inset: distribution of pre-activation when shift amount is 0 in each case.

used ResNet-20 model with double skip connections [25] as shown in Fig. 9a. The model consists of the first convolution layer, 18 residual blocks, and the last fully-connected layer. In the ResNet-20 model with additional PReLU layers, a PReLU layer is inserted at the end of every residual block as shown in Fig. 9a. The effect of threshold shifting on ResNet-20 model with and without the additional PReLU layers is shown in Fig. 9b and c, respectively. While the accuracy is improved by shifting the threshold of binary activation function in the ResNet-20 model, the threshold shifting degrades the accuracy when additional PReLU layers are used jointly. The additional PReLU layers already play a role in distorting the activation distribution, and hence further shifting the threshold of binary activation function is excessive. To validate the claim, we slightly modified the additional activation function and analyzed the effect of threshold shifting technique. We replaced the additional PReLU layers with LeakyReLU layers and varied the slope of the negative range of the LeakyReLU layers from 0 to 1. Note that when the slope is 1, the LeakyReLU layer becomes an identity function and therefore the model becomes the same as the ResNet-20 model without PReLU layers. Fig. 10 shows the training results of ResNet-20 with LeakyReLU layers with different slopes. As the slope of the negative range of LeakyReLU decreases, the distortion of the activation distribution becomes more severe. When the slope is close to 1, the LeakyReLU layer does not skew the distribution of pre-activation much. Hence, the distribution of pre-activation values is very close to a Gaussian distribution with zero mean. As a result, the accuracy improvement by shifting the threshold of binary activation function was clearly observed as in the ResNet-20 model without PReLU layers. However, as the slope decreases, the LeakyReLU layer makes the activation distribution unbalanced as shown

in Fig. 10. Note that LeakyReLU layer scales down the negative values only so that the mean value of the distribution increases while the median value of the distribution does not change. Therefore, it has a similar effect to shifting the threshold of binary activation function in that the activation distribution becomes unbalanced.

4. Conclusion

In this paper, we analyzed the impact of activation distribution on the accuracy of BNN models. While previous BNNs used sign function as binary activation function which balances the distribution of binary activation, we claim that the accuracy of BNN models can be improved when the distribution of binary activation is unbalanced. By simply shifting the thresholds of binary activation functions, we demonstrated that the accuracy of previous BNN models could be further improved. We also identified that the unbalanced activation distribution partly accounts for the improved accuracy of BNN models which used additional activation functions.

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] 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.

[3] Adrian Bulat and Georgios Tzimiropoulos. Xnor-net++: Improved binary neural networks. In *In British Machine Vision Conference*, 2019.

[4] Adrian Bulat, Georgios Tzimiropoulos, Jean Kossaifi, and Maja Pantic. Improved training of binary networks for human pose estimation and image recognition. *arXiv preprint arXiv:1904.05868*, 2019.

[5] Zhaowei Cai, Xiaodong He, Jian Sun, and Nuno Vasconcelos. Deep learning with low precision by half-wave gaussian quantization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5918–5926, 2017.

[6] Jungwook Choi, Swagath Venkataramani, Vijayalakshmi Viji Srinivasan, Kailash Gopalakrishnan, Zhuo Wang, and Pierce Chuang. Accurate and efficient 2-bit quantized neural networks. *Proceedings of Machine Learning and Systems*, 1:348–359, 2019.

[7] Jungwook Choi, Zhuo Wang, Swagath Venkataramani, Pierce I-Jen Chuang, Vijayalakshmi Srinivasan, and Kailash Gopalakrishnan. Pact: Parameterized clipping activation for quantized neural networks. *arXiv preprint arXiv:1805.06085*, 2018.

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

[9] Sajad Darabi, Mouloud Belbahri, Matthieu Courbariaux, and Vahid Partovi Nia. Bnn+: Improved binary network training. *arXiv preprint arXiv:1812.11800*, 2018.

[10] Ruizhou Ding, Ting-Wu Chin, Zeye Liu, and Diana Marculescu. Regularizing activation distribution for training binarized deep networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11408–11417, 2019.

[11] Steven K Esser, Jeffrey L McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S Modha. Learned step size quantization. In *International Conference on Learning Representations*, 2019.

[12] Julian Faraone, Nicholas Fraser, Michaela Blott, and Philip HW Leong. Syq: Learning symmetric quantization for efficient deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4300–4309, 2018.

[13] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.

[14] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 315–323, 2011.

[15] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

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

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

[18] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[19] Sangil Jung, Changyong Son, Seohyung Lee, Jinwoo Son, Jae-Joon Han, Youngjun Kwak, Sung Ju Hwang, and Changkyu Choi. Learning to quantize deep networks by optimizing quantization intervals with task loss. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4350–4359, 2019.

[20] 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.

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

[22] Alex Krizhevsky and Geoff Hinton. Convolutional deep belief networks on cifar-10. *technical report*, 2010.

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

[24] 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 (ECCV)*, 2020.

[25] 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.

[26] Brais Martinez, Jing Yang, Adrian Bulat, and Georgios Tzimiropoulos. Training binary neural networks with real-to-binary convolutions. In *International Conference on Learning Representations*, 2019.

[27] Asit Mishra, Eriko Nurvitadhi, Jeffrey J Cook, and Debbie Marr. Wrpn: Wide reduced-precision networks. In *International Conference on Learning Representations*, 2018.

[28] Hadi Pouransari, Zhucheng Tu, and Oncel Tuzel. Least squares binary quantization of neural networks. In *Proceed-*

ings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, pages 698–699, 2020.

[29] 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.

[30] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[31] Mingxing Tan and Quoc V Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*, 2019.

[32] Wei Tang, Gang Hua, and Liang Wang. How to train a compact binary neural network with high accuracy? In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pages 2625–2631, 2017.

[33] Peisong Wang, Xiangyu He, Gang Li, Tianli Zhao, and Jian Cheng. Sparsity-inducing binarized neural networks. In *AAAI*, pages 12192–12199, 2020.

[34] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. In *Advances in neural information processing systems*, pages 2074–2082, 2016.

[35] Bichen Wu, Alvin Wan, Xiangyu Yue, Peter Jin, Sicheng Zhao, Noah Golmant, Amir Gholaminejad, Joseph Gonzalez, and Kurt Keutzer. Shift: A zero flop, zero parameter alternative to spatial convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9127–9135, 2018.

[36] Dongqing Zhang, Jiaolong Yang, Dongqiangzi Ye, and Gang Hua. Lq-nets: Learned quantization for highly accurate and compact deep neural networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 365–382, 2018.

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