Energy-Efficient Image Processing using Binary Neural Networks with Hadamard Transform

Jaeyoon Park[0000−0001−8382−0474] and Sunggu Lee[0000−0003−3858−0779]

Pohang University of Science and Technology (POSTECH), South Korea
{jaeyoonpark,slee}@postech.ac.kr

Abstract. Binary neural networks have recently begun to be used as a highly energy- and computation-efficient image processing technique for computer vision tasks. This paper proposes a novel extension of existing binary neural network technology based on the use of a *Hadamard transform* in the input layer of a binary neural network. Previous state-of-the-art binary neural networks require floating-point arithmetic at several parts of the neural network model computation in order to maintain a sufficient level of accuracy. The Hadamard transform is similar to a Discrete Cosine Transform (used in the popular JPEG image compression method) except that it does not include expensive multiplication operations. In this paper, it is shown that the Hadamard transform can be used to replace the most expensive floating-point arithmetic portion of a binary neural network. In order to test the efficacy of this proposed method, three types of experiments were conducted: application of the proposed method to several state-of-the-art neural network models, verification of its effectiveness in a large image dataset (ImageNet), and experiments to verify the effectiveness of the Hadamard transform by comparing the performance of binary neural networks with and without the Hadamard transform. The results show that the Hadamard transform can be used to implement a highly energy-efficient binary neural network with only a miniscule loss of accuracy.

Keywords: Binary neural network · Hadamard transformation · DCT.

1 Introduction

Although deep neural networks have resulted in highly accurate image classification, object recognition, and other computer vision tasks, such networks typically involve excessive amounts of numerical computation with excessive memory storage requirements, making them difficult to use in energy or computation capability constrained environments. A popular neural network compression method that can be used in such cases is binarization, in which 32-bit floating point parameters are approximated using single-bit numbers.

Binary Neural Networks (BNNs) are neural networks that use extensive levels of binarization throughout the network to achieve extreme network compression with a concomitant but relatively small loss of accuracy. In order to maintain

acceptable accuracy levels, such BNNs typically use a mixture of highly accurate numbers (such as 32-bit floating point) and highly inaccurate binary numbers for different types of parameters and/or different layers of the neural network. For example, binarization of AlexNet [14] through the method proposed by Hubara *et al.* [10], which is one of the early BNNs, can reduce the model size by 32 times at the cost of a 28.7% reduction in accuracy [25] on the ImageNet dataset [2]. Later research works on BNNs attempted to reduce this extremely high accuracy gap. The current state-of-the-art (SOTA) BNN [16] has approximately the same model size as [10] with only a 1.9% reduction in accuracy on the ImageNet dataset when compared to the equivalent non-binarized neural network model. A standard method for measuring the inference cost of a neural network has been proposed by Zhang *et al.* [31]. Referred to as *arithmetic computation effort (ACE)*, it counts the number of multiply-accumulate (MAC) operations, which are the most computationally expensive operations used in a neural network, weighted by the bit-widths of the operands used in those MAC operations.

In almost all previous state-of-the-art (SOTA) BNN models, the input layer uses floating-point arithmetic. This is because binarization of the input layer severely degrades the accuracy of a BNN [16–19, 25]. However, due to its use of floating-point arithmetic, the input layer has been found to be *the major* contributor to the computation cost of a SOTA BNN. For example, when using the popular SOTA BNN referred to as ReActNet [16], the input layer contributes to approximately 65% out of the entire network ACE.

Previous studies on CNN have found that input layer extracts abstract features such as colors and various edge directions in images [7, 29]. The filters of input layer resemble the Gabor filter [20] which analyzes specific frequency components in each local area, so that can detect edge of image. The discrete cosine transform (DCT), which is the encoding method used in the popular JPEG compression format, also computes in a similar manner. Using this fact, Guegeun *et al.* proposed to feed discrete cosine transformed data directly into a CNN, without first decoding that JPEG compressed image into a raw image [7]. This enabled the first few layers of the neural network to be pruned without any accuracy loss.

The Hadamard transform, which is also known as the Walsh-Hadamard transform, is similar transformation to the DCT. The main difference is that the Hadamard transform is multiplication-free, and it only requires add/subtract operations [23].

In this paper, we propose a new input layer using the Hadamard transform for an energy-efficient BNN. The proposed layer is fed with raw images, and it can replace conventional expensive floating-point MAC operations with light 8-bit add/subtract and logical operations. The input layer is expected to reduce energy consumption of BNN, which can be measured by ACE metric, and to achieve acceptable level of accuracy degradation. Experiments were conducted to reveal a possibility of proposed input layer for BNN. First of all, a generality of the layer was tested by applying it on binarized versions of two widely used CNN architectures, MobileNetV1 [9] and ResNet-18 [8], on a small image

dataset, *i.e.*, CIFAR-10. Secondly, accuracy drop evaluation was performed on ReActNet [16], which is the SOTA BNN in terms of the accuracy gap from its real-valued counterpart. The test was conducted using the ImageNet dataset, which consists of large-scale real images, so that the practicality of new layer for real-world problem can be demonstrated. Lastly, the proposed input layer structure with trainable weight filters, instead of the Hadamard Transformation, was investigated on ReActNet to validate the efficacy of the Hadamard transform for BNNs.

2 Related Work

2.1 Binary Neural Networks

Parameter quantization is one of the methods of compressing convolutional neural networks. It is a method representing weight parameters and activations of neural network, which are normally 32-bit floating point numbers, with fewer number of N-bit width, such as 8, 4, and 2-bit. A size of the compressed network can be reduced by $32/N$ times, and an improvement of inference speed can be obtained [24].

A binary neural network is a special case of quantization with a single-bit precision, which is the smallest bit width in computer system. The process of quantization is binarization and it can be simply implemented using signum function, which outputs the sign bit of input values. An exceptional advantage of BNN over other quantized neural networks is in a convolution operation. The convolution operation with the 1-bit operands requires bit-wise logical operator, which is fast and energy-efficient, instead of expensive and relatively slow floating-point MAC unit [25]. Binary convolution refers to the convolution with operands of single-bit precision. Therefore, BNN can save massive energy consumption and reduce the size of deep neural network. Although early studies on BNN achieved comparable level of accuracy on tiny image dataset [10], such as MNIST [3] and SVHN [21], training results of BNNs on large-scale image showed poor image classification accuracy [25]. So until recently, most BNN studies have tried to mitigate the accuracy degradation.

Authors in [25], proposed that binarization error from real-valued operands to its binarized version can be reduced by introducing scaling factor. [15] designed a convolution layer with multiple binary convolution bases. The multiple outputs from the multiple bases are accumulated to enhance representability of BNN. [18] suggested adding high-precision values before binarization to the output of binary convolution via short-cut and it improved model capacity. [17] proposed that activations with high-precision should be binarized not simply by their signs but by a threshold, which determines a value to be +1 or -1. The author implemented signum function with trainable parameters to learn the appropriate threshold during training time. The network with the method achieved the smallest accuracy loss, which is caused by binarization of original full-precision network.

In addition to increasing the accuracy of BNNs, there is a study to reduce inference cost of BNN. [31] presented an energy-efficient convolution block for BNN. They also proposed arithmetical computing efficiency(ACE), which is a metric to measure efficiency of neural network. It calculates energy consumption on neural network inference by counting the number of MAC operations and weighting the bit width of operands. Table 4 shows a summary of the accuracy of this method as well as the previous methods described in this section.

2.2 Input Layer of Convolutional Neural Networks

CNN consists of convolutional layers which extract features of spatial data. Each layer receives an data in the form of feature maps, extracts specific features, and passes them to subsequent layer. What features to be extracted are determined through training process. More specifically, filters of convolutional layer are shaped differently by training dataset. Interestingly, input layer of the neural network captures general features, such as color and texture. It is relatively independent of the dataset used for training [29]. Subsequent layers are learned to extract more detailed features based on the general features.

It is known that the general filters in the input layer resemble the Gabor filter in image processing [7, 29]. The Gabor filter is mainly used to extract edge and texture of image [20]. Parameters in the filter, such as angle, width, and repetition period of edge to be extracted from an image, can be selected by engineer. The general filters of input layer, which are acquired from network training, are similar to a set of several the Gabor filters with various combinations of parameters.

Based on this fact, there are studies that apply the image processing technique to CNN's input layer. For example, discrete cosine transform(DCT) is proved its usefulness as an input layer by [7]. The paper showed that using DCT as input layer, instead of conventional trainable input layer, can achieve better accuracy for ResNet-50 architecture. In addition, the authors attempted to train input layer with a regularizer, whose role is guiding the filters of input layer to resemble DCT. But they concluded that training DCT-like filter is hard and inefficient.

In case of BNN, all of the aforementioned methods for increasing the accuracy of the BNN are not applied to input layer. This is because binarization of input layer directly can degrades model's performance severely [30], while the improvement of execution time is small [25]. Recently, [30] suggested to transform input image with thermometer encoding, which contains division, ceiling and rounding operations, so it can avoid direct binarization of input layer. Alternatively, [31] proposed 8-bit quantization for input layer rather than binarization.

2.3 Hadamard Transform

The Hadamard transform is used as a feature extractor in the field of image and video processing. In [5], specific basis vectors of the Hadamard transform is selected to detect shot boundary of videos. The transform can be used for

image compression [28, 6, 4]. [4] suggested image compression method by taking advantage of simple and efficient property of the Hadamard transform. A recent study on CNN proposed a layer with Hadamard transform, which is designed to replace 1 x 1 convolution layer of neural networks, to achieve faster network inference [22].

3 Hadamard Transform as an Input Layer

3.1 Hadamard Matrix

Hadamard transform is one of the linear image transforms [26]. The transform is an operator capable of processing 2D images and has averaging property [23]. And the transformed image can be inversely transformed into the original spatial domain. Hadamard transform can perform the same function as DCT more efficiently. This is because they are both orthogonal transforms [26] but Hadamard transform uses Hadamard matrix, whose entries are +1 and -1, making the operation simpler.

Hadamard matrix(H) is in square array form, and the matrix of N=2^n order can be obtained using the Kronecker product(1). When N=1, the entry is one with 1, and for N=$2^n \geq 2$, the Hadamard matrix can be derived by recursively utilizing the matrix of N=$2^{(n-1)}$ order. For example, when N=4, the Hadamard matrix H_4 consists of four H_2, and the H_2 holds four H_1 which is one 1 with appropriate sign of entries according to equation(1).

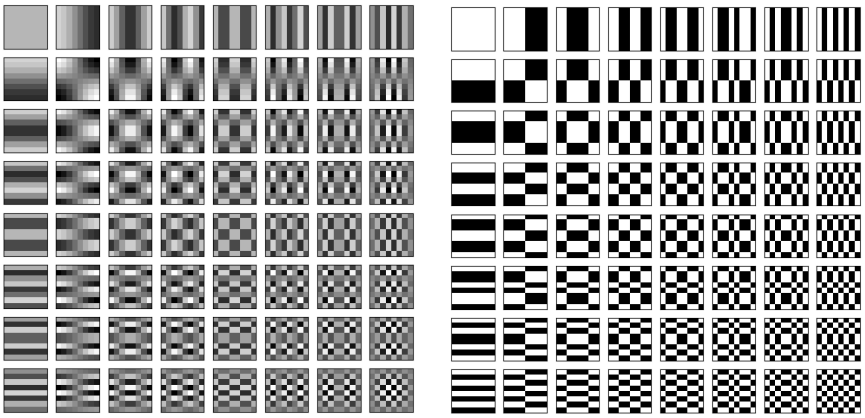
$$H_{2^n} = H_2 \otimes H_{2^{n-1}} = \begin{bmatrix} H_{2^n-1} & H_{2^n-1} \\ H_{2^n-1} & -H_{2^n-1} \end{bmatrix} \tag{1}$$

$$H_1 = \begin{bmatrix} 1 \end{bmatrix} \quad H_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad H_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \underset{\text{ordered}}{\longrightarrow} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & -1 \end{bmatrix} \tag{2}$$

The matrix has several properties. The first property is that the matrix is symmetric. Second, each row is orthogonal to each other. Third, how many times the sign of the entries of the row changes is called sequency, and the Hadamard matrix of N order consists of rows with sequency from 0 to $N-1$. If the rows are ordered in ascending, it is exactly same as the Walsh matrix [23]. In this paper, we refers to Hadamard matrix as the matrix with the rows of the ascending ordered.

3.2 Hadamard Transform

Two-dimensional image can be processed with Hadamard transform using Eq.(4). In the equation, the original image in spatial domain is denoted by $s(x, y)$ and transformed image is represented by $G(u, v)$. The size of processed image is the



(a) Discrete cosine transform (b) Hadamard transform

Fig. 1: Visualized 2D kernels of (a) DCT and (b) Hadamard transform for block size of 8. Ordered Hadamard matrix is used to obtain (b).

number of 2D kernels of Hadamard transform. The 2D kernels can be obtained by outer product of rows and columns of Hadamard matrix. For example, from the ordered H_4, the first two kernels are 4×4 as they are in (5). In a same way, DCT on 2D image can be done using Eq.(3) and the kernels from the DCT can be obtained. Fig. 1 shows that the two kernel sets from DCT and Hadamard transform are similar to each other. Additionally, transformed images using the transforms are illustrated in Fig. 2 to provide qualitative comparison.

$$F(u,v) = \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} s(x,y) \exp\left(-\frac{2\pi i}{N}(ux+vy)\right) \tag{3}$$

$$G(u,v) = \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} s(x,y) g_{(u,v)}(x,y) \tag{4}$$

$$g(1,1) = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \quad g(1,2) = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 \end{bmatrix} \tag{5}$$

The characteristic of Hadamard transform is that no multiplication is required, which results in efficient computation. Secondly, the energy before and after transformation is preserved (6). And the computational result using the zero sequency kernel($g(1,1)$) means the average brightness of the spatial domain image (7). It is the same operation with average pooling layer in neural networks. Moreover, the energy of most images is concentrated in this area, and for the

(a) Original image (b) Discrete cosine transform (c) Hadamard transform

Fig. 2: (a) Original image from ImageNet dataset and its transformed images using (b) Hadamard transform and (c) DCT.

higher sequency kernels, relatively small amount of energy is held [23], which enables image compression [6, 28, 4].

$$\sum_{x=0}^{N-1}\sum_{y=0}^{N-1}|s(x,y)|^2 = \frac{1}{N^2}\sum_{u=0}^{N-1}\sum_{v=0}^{N-1}|G(u,v)|^2 \tag{6}$$

$$G(0,0) = \frac{1}{N^2}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1}s(x,y) \tag{7}$$

3.3 Proposed Input Layer

As DCT can be used as input layer of CNN [7], and the DCT and Hadamard transform are functionally same, it is possible to use kernels of Hadamard transform input layer of the BNN. However, there are several considerations to materialize it.

Assume that there is a single-channel 2D image. Hadamard transform processes on N by N size blocks of image in the spatial domain, where the blocks are non-overlapped. In terms of convolution operation, this is same as windowing weight filters with stride step of N and producing N^2 output channels. However, to utilize Hadamard transform in input layer, the transform must be implemented to overlap the N by N block size. This is to provide feature maps of particular dimension, which can be different from structures of existing BNNs but can not be covered with stride N, for subsequent layer. At the same time, the overlapping should properly extracts features without hurting network's performance. It has been proved that overlapping 2D kernels on the spatial domain image, which is called modified DCT(MDCT) [27], can extract features well in CNN [11, 27]. Considering the same functionality of DCT and Hadamard, it is possible to adopting the MDCT manner on Hadamard transform.

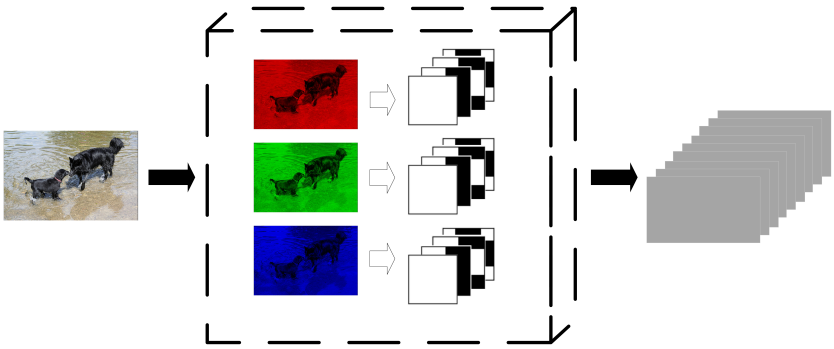


Fig. 3: Hadamard transform in proposed input layer. The operation can be done for each channels with same N^2 kernels. The number of kernels can be vary depending on order of Hadamard matrix. The transformed images are concatenated in channel-wise.

In addition, an input image with 3-channels is normally fed into neural network. Therefore, the kernels should be applied to each channel, which can be regarded as grouped convolution [14], and the transform will eventually output $3\text{x}N^2$ channels. The process is depicted in Fig.3. However, the aforementioned particular dimension for subsequent layer also includes the number of channels(or depth). Therefore, the channel of $3\text{x}N^2$ size need to flexibly modified depending on possible BNN structures. To address this issue, pointwise binary convolution with shortcut [17], which operates in bit-wise operators, is followed by the Hadamard transform. When the dimension of shortcut and output of pointwise convolution is not matched, channel-wise zero padding can used for the shortcut. This proposed input layer is illustrated in Fig.4.

Moreover, it is not necessary to have N^2 kernels for transformation in the input layer. Hadamard transform preserves the energy of the pre-transformed spatial domain, while the high-sequency kernels could result fewer energy portions. Even if these high-sequency kernels are discarded, the energy of the spatial domain does not change significantly. This concept is used in one of the lossy compression method, JPEG. Thus, the number of operations can be reduced by ignoring insignificant energy loss. For example, when $N = 4$, the number of 2D kernels to preserve entire energy is 16. However, our experiments witnessed that no accuracy drop occurred using 10 kernels of low frequencies.

4 Evaluation

Experiments on proposed input layer have three parts. First of all, a generality of proposed layer is validated. We took two representative used BNNs, whose full-precision networks are based on ResNet and MobileNetV1 respectively, and tested the proposed input layer on them. Also, the experiment includes comparison between Hadamard transform to DCT as input layer. Next, in order to test

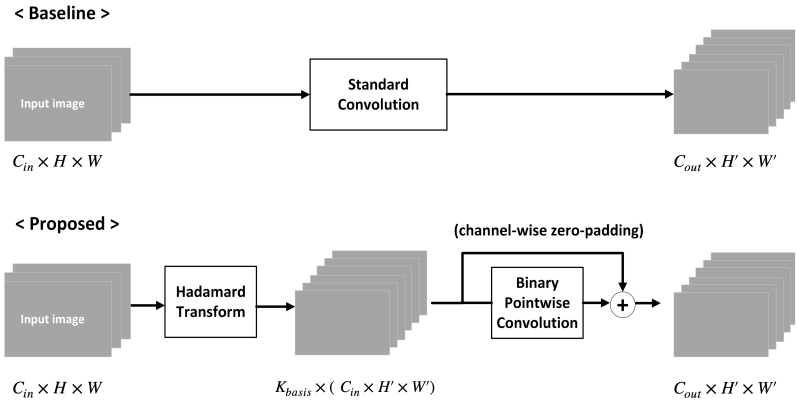


Fig. 4: Structures of conventional and proposed input layer. The Hadamard transform in proposed structure is performed by grouped convolution with 8-bit integer. Output dimension of the operation depends on $K_{basis} \leq N^2$, which is the number of 2-d kernels of Hadamard transform. In this paper, we used N=4 and K_{basis}=10. The binary pointwise convolution can be operated by bitwise XNOR and bit count. And shortcuts may requires channel-wise zero padding to match the dimension of the binary pointwise convolution.

practicality of proposed layer, we replaced input layer of ReActNet [16], which showed the best performance regarding accuracy degradation in this field, with the proposed layer. The network with proposed layer was tested on real-world large scale images, the ImageNet [2]. Lastly, we replaced the kernels of Hadamard from the proposed input layer with binarized filters through training. Datasets used in the experiments are CIFAR-10 [13] and ImageNet [2]. CIFAR-10 is a representative small image(32x32) dataset and has 10 categories in it. The dataset consists 50K training images and 10K test images. ImageNet contains 1.2M training images and 50K validation images each of which can be categorized in 1K classes. Unlike CIFAR-10, the image sizes are different from one another, so they are normally resized to fit a particular size(e.g. 224x224) for training and validation. Image classification accuracy and energy consumption of the MAC operation are considered to compare the efficacy of proposed input layer. To measure the energy consumption, we used ACE which is proposed by [31]. The metric counts the number of MAC operations and each operation are weighted by bit width of the operands. ACE for different precision is summarized in the Table 1.

4.1 Generality of the Hadamard Transform as an Input Layer

Implementation Details. We implemented two BNNs based on ResNet and MobileNet, which are widely used in BNN studies so far [15, 19, 17, 30, 31], and trained them with CIFAR10 [13]. The binarization techniques, which are used

Table 1: ACE metric [31]

Precision	**float**			**int**				
	32	**16**	**bfloat 16**	**32**	**8**	**4**	**4**	**1**
ACE	1024	256	256	1024	64	16	4	1

in this experiment, follow ReActNet [17]. On top of them, minor modifications of BNN models were processed. Specifically, when experimenting with a ResNet model, the ResNet-18 structure was used instead of ResNet-20, which has 3x3 kernel size at input layer. The same structure was used in [1]. Afterward we refer to this network as ReActNet-18. And when testing MobileNetV1 based BNN, we took the structure proposed by ReActNet [16] and reduced the stride step of input layer from 2 to 1. The MobileNet-based BNN will be referred to ReActNet-A, regardless of stride size at input layer.

There are three differences between baseline input layer and proposed input layer. For the baseline, kernel size is 3x3 and standard 2d convolution is used. In other words, group size is 1. And operands are high-precision with 32-bit floating point. On the other hand, proposed input layer has a kernel size of 4x4, and the grouped convolution with group size of the input channel(RGB channels for conventional input image). Each groups take 10 2D kernels of Hadamard transform. And binary point wise convolution is followed by the transform, to flexibly control the number of output channels.

Two-stage training strategy, which is widely used in BNN training [19, 18, 17, 31], is adopted for training the BNNs. In the first step, only activations are binarized, and weights remain 32-bit floating point number. In the second step, the previously trained model becomes the initial state, and then additional binarization function for weights are added in the network. Thus in this step, both activations and weights are binarized. Adam [12] optimizer was used, and hyperparameters were set as follows. Training 100K steps for each stage with 256 epochs, batch size of 128 and learning rate of 5e-4. Weight decay is used in the first stage of learning, but not in the second stage[16].

Results. ReActNet-18 with conventional input layer, shows accuracy of 93.49% on CIFAR10. ACE for the network is $2.36G$ and the input layer accounts for 76.63% out of the entire network ACE. On the other hand, in the case of using the proposed layer, the accuracy dropped by 1.17% resulting in accuracy level of 92.51%. The network saved 75.13% of ACE compared to baseline. In ReActNet-A, the trend were same as the ResNet-18. The accuracy of the baseline is 90.74%, and the ACE is $1.31G$. The baseline input layer occupies 69.18% of entire ACE. However, when the input layer is replaced with the proposed layer, accuracy level is 89.60% which is loss of 1.39% point, and the ACE is decreased by 66.70% compared to the baseline. Additionally to compare Hadamard transform with DCT,

we implemented proposed input layer with DCT instead of Hadamard transform. As the two transformations are same in terms of functionality, accuracy levels achieved with DCT are similar to with Hadamard transform. However, DCT consumes more energy than Hadamard transform because the latter is multiplication-free. The results are summarized in Table 2.

Table 2: Results on CIFAR10

Network	Input layer	Accuracy	ΔAcc.(%p)	ACE(1e9)	Δ ACE
ReActNet-18	Baseline	93.94%	-	2.36	-
	DCT	92.97%	-0.97	1.06	-55.18%
	Proposed	92.51%	-1.43	0.59	-75.13%
ReActNet-A	Baseline	90.74%	-	1.31	-
	DCT	89.41%	-1.33	0.91	-30.67%
	Proposed	89.35%	-1.39	0.44	-66.70%

4.2 BNN with Proposed Input Layer on ImageNet

Implementation Details. The baseline for this experiment is exactly same as proposed in [16]. Stride step is 2 for both baseline and proposed input layer. The two-stage strategy was applied on this experiment. The proposed network was trained with 256 epochs, batch size of 256 and learning rate of 5e-6 for each stage as the authors in [16] suggested. Weight decay was set 5e-6 and used only for the first step.

Results. The baseline has a validation accuracy of 70.5% and ACE of 16.96G, where input layer alone accounts for 65.42%. However, with the proposed input layer, the ACE decrease by 63.08% at the cost of 1.38% of accuracy loss which is summarized in Table 3. Compared to ReActNet's real-valued counter part, it was finally reduced by 3.28% point. The gap is superior to FracBNN [30], which showed the second best result in the accuracy gap. As a result, the SOTA BNN with the proposed input layer still showed the smallest accuracy gap from real-valued counterpart and achieved better energy-efficiency. This result is summarized in Table 4.

4.3 Hadamard Transform vs. Trained Binary Weights

As mentioned in Section 3, using the Hadamard transform as an input layer means using the transform's 2D kernels as weight filters. Since the filters consist of only +1 and -1, the convolution operation consists of only add/sub without multiplication. Thanks to this, we were able to implement energy-efficient BNNs.

Table 3: Results on ImageNet

Network	Input layer	Accuracy	⊿Acc.(%p)	ACE(1e9)	⊿ ACE
ReActNet-A	Baseline	70.5%	-	16.9	-
	Proposed	69.12%	-1.39	6.26	-63.08%

Table 4: Top-1 accuracy of BNNs on ImageNet.

Network	Method	Top-1 accuracy(%)	Gap(%)
AlexNet	Full-precision	56.6	-
	BinaryNet [10]	27.9	-28.7
	XNOR-Net [25]	44.2	-12.4
ResNet-18	Full-precision	69.3	-
	ABC-Net (5 bases) [15]	65.0	-4.3
	ABC-Net (1 base)	42.7	-28.6
	Bi-RealNet [18]	56.4	-12.9
	ReActNet [17]	65.5	-3.8
PokeBNN	Full-precision	79.2	-
	PokeBNN-1.0 [31]	73.4	-5.8
FracBNN	Full-precision	75.6	-
	FracBNN [30]	71.8	-3.8
ReActNet-A	Full-precision	72.4	-
	ReActNet(Adam) [16]	70.5	-1.9
	Ours	**69.12**	**-3.28**

However, this result can be attributed not to the kernels of Hadamard, but to filters composed of +1 and -1. To make it clear, we created the same structure as the proposed input layer and trained binarized weights from the scratch.

The experiment was conducted on ReActNet-A with CIFAR10 dataset. The result showed that training filter was less accurate than proposed input layer and the training curve was largely fluctuated as illustrated in Fig.5. If the learning process is unstable, the final accuracy can be deteriorated [16]. In general, it is a phenomenon that occurs when the sign of binarized weights changes only at a few steps, because the real-valued weights before binarization are close to zero during the training time[16]. Empirically, this case can be solved by lowering the learning rate, but BNN already uses a much lower learning rate than full-precision network learning, so overall learning time can be increased to achieve the same result.

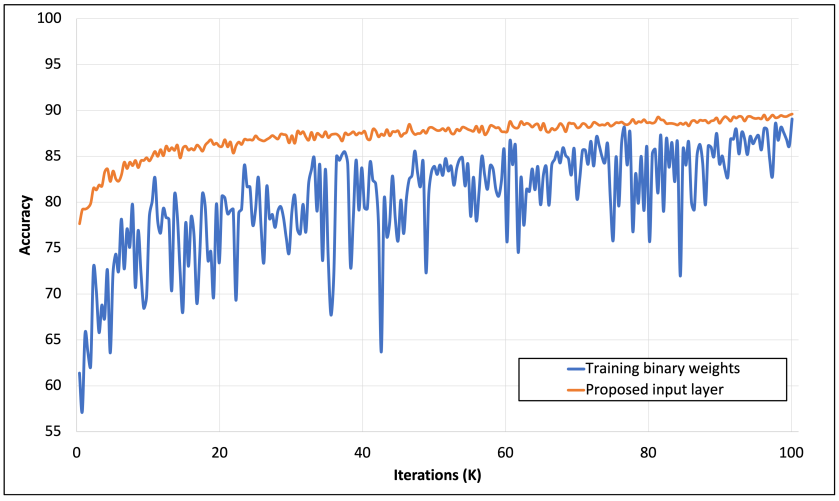


Fig. 5: Training curves of ReActNet-A on CIFAR10. The blue line is the network with training input layer with binary weights, which is unstable. The orange line is the network with the proposed input layer. Unlike the blue line, the accuracy of proposed network increases without fluctuation even the filters are binary values.

5 Conclusion

The input layer of in state-of-the-art binary neural networks (BNNs) typically use floating-point arithmetic because of the resulting steep drop in accuracy when quantized and its negligible effect on inference speed. However, from an energy consumption perspective, the layer consumes an abnormal amount of energy. To address this issue, we proposed an energy-efficient input layer for binary neural networks using a Hadamard transform. The proposed input layer has been tested on ReActNet-A and ReActNet-18, which are MobileNetV1 and ResNet-18 based BNN respectively. The energy consumption of BNN was measured by ACE, and with the proposed input layer, the networks' ACE value was reduced by up to 75%. In addition, the accuracy degradation caused by this input layer was less than 1.5%.

Acknowledgements This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No.2019-0-01906, Artificial Intelligence Graduate School Program(POSTECH))

References

1. Chen, T., Zhang, Z., Ouyang, X., Liu, Z., Shen, Z., Wang, Z.: " bnn-bn=?": Training binary neural networks without batch normalization. In: Proceedings of the

IEEE/CVF conference on computer vision and pattern recognition. pp. 4619–4629 (2021)
2. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition. pp. 248–255. Ieee (2009)
3. Deng, L.: The mnist database of handwritten digit images for machine learning research. IEEE Signal Processing Magazine **29**(6), 141–142 (2012)
4. Diana Andrushia, A., Thangarjan, R.: Saliency-based image compression using walsh–hadamard transform (wht). In: Biologically rationalized computing techniques for image processing applications, pp. 21–42. Springer (2018)
5. GG, L.P., Domnic, S.: Walsh–hadamard transform kernel-based feature vector for shot boundary detection. IEEE Transactions on Image Processing **23**(12), 5187–5197 (2014)
6. Ghrare, S.E., Khobaiz, A.R.: Digital image compression using block truncation coding and walsh hadamard transform hybrid technique. In: 2014 International Conference on Computer, Communications, and Control Technology (I4CT). pp. 477–480. IEEE (2014)
7. Gueguen, L., Sergeev, A., Kadlec, B., Liu, R., Yosinski, J.: Faster neural networks straight from jpeg. Advances in Neural Information Processing Systems **31** (2018)
8. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
9. Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861 (2017)
10. Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R., Bengio, Y.: Binarized neural networks. Advances in neural information processing systems **29** (2016)
11. Ju, S., Lee, Y., Lee, S.: Convolutional neural networks with discrete cosine transform features. IEEE Transactions on Computers (2022)
12. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
13. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images. Technical Report (2009)
14. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems **25** (2012)
15. Lin, X., Zhao, C., Pan, W.: Towards accurate binary convolutional neural network. Advances in neural information processing systems **30** (2017)
16. Liu, Z., Shen, Z., Li, S., Helwegen, K., Huang, D., Cheng, K.T.: How do adam and training strategies help bnns optimization. In: International Conference on Machine Learning. pp. 6936–6946. PMLR (2021)
17. Liu, Z., Shen, Z., Savvides, M., Cheng, K.T.: Reactnet: Towards precise binary neural network with generalized activation functions. In: European Conference on Computer Vision. pp. 143–159. Springer (2020)
18. Liu, Z., Wu, B., Luo, W., Yang, X., Liu, W., Cheng, K.T.: 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). pp. 722–737 (2018)
19. Martinez, B., Yang, J., Bulat, A., Tzimiropoulos, G.: Training binary neural networks with real-to-binary convolutions. arXiv preprint arXiv:2003.11535 (2020)

20. Mehrotra, R., Namuduri, K.R., Ranganathan, N.: Gabor filter-based edge detection. Pattern recognition **25**(12), 1479–1494 (1992)
21. Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., Ng, A.Y.: Reading digits in natural images with unsupervised feature learning. In: NIPS Workshop on Deep Learning and Unsupervised Feature Learning (2011)
22. Pan, H., Badawi, D., Cetin, A.E.: Fast walsh-hadamard transform and smooth-thresholding based binary layers in deep neural networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4650–4659 (2021)
23. Pratt, W.K., Kane, J., Andrews, H.C.: Hadamard transform image coding. Proceedings of the IEEE **57**(1), 58–68 (1969)
24. Qin, H., Gong, R., Liu, X., Bai, X., Song, J., Sebe, N.: Binary neural networks: A survey. Pattern Recognition **105**, 107281 (2020)
25. Rastegari, M., Ordonez, V., Redmon, J., Farhadi, A.: Xnor-net: Imagenet classification using binary convolutional neural networks. In: European conference on computer vision. pp. 525–542. Springer (2016)
26. Salomon, D.: Data compression: the complete reference. Springer Science & Business Media (2004)
27. Ulicny, M., Krylov, V.A., Dahyot, R.: Harmonic convolutional networks based on discrete cosine transform. Pattern Recognition **129**, 108707 (2022)
28. Valova, I., Kosugi, Y.: Hadamard-based image decomposition and compression. IEEE Transactions on Information Technology in Biomedicine **4**(4), 306–319 (2000)
29. Yosinski, J., Clune, J., Bengio, Y., Lipson, H.: How transferable are features in deep neural networks? Advances in neural information processing systems **27** (2014)
30. Zhang, Y., Pan, J., Liu, X., Chen, H., Chen, D., Zhang, Z.: Fracbnn: Accurate and fpga-efficient binary neural networks with fractional activations. In: The 2021 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays. pp. 171–182 (2021)
31. Zhang, Y., Zhang, Z., Lew, L.: Pokebnn: A binary pursuit of lightweight accuracy. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 12475–12485 (2022)