Input Layer Binarization with Bit-Plane Encoding

Input Layer Binarization with Bit-Plane Encoding

Lorenzo Vorabbi[1,2][0000−0002−4634−2044], Davide Maltoni[2][0000−0002−6329−6756], and Stefano Santi[1]

[1] Datalogic Labs, Bologna 40012, IT
[2] University of Bologna, DISI, Cesena Campus, Cesena 47521, IT
{lorenzo.vorabbi2,davide.maltoni}@unibo.it
{lorenzo.vorabbi,stefano.santi}@datalogic.com

Abstract. Binary Neural Networks (BNNs) use 1-bit weights and activations to efficiently execute deep convolutional neural networks on edge devices. Nevertheless, the binarization of the first layer is conventionally excluded, as it leads to a large accuracy loss. The few works addressing the first layer binarization, typically increase the number of input channels to enhance data representation; such data expansion raises the amount of operations needed and it is feasible only on systems with enough computational resources. In this work, we present a new method to binarize the first layer using directly the 8-bit representation of input data; we exploit the standard bit-planes encoding to extract features bit-wise (using depth-wise convolutions); after a re-weighting stage, features are fused again. The resulting model is fully binarized and our first layer binarization approach is model independent. The concept is evaluated on three classification datasets (CIFAR10, SVHN and CIFAR100) for different model architectures (VGG and ResNet) and, the proposed technique outperforms state of the art methods both in accuracy and BMACs reduction.

Keywords: Binary Neural Networks · Input Layer Binarization · Deep Learning

1 Introduction

Deep Neural Networks showed in the last years impressive results, sometimes reaching accuracy better than human level, with applications in a wide variety of domains. These improvements have been achieved by increasing the depth and complexity of the network; such huge models can run smoothly on expensive GPU-based machines but cannot be easily deployed to edge devices (i.e., small mobile or IoT systems), which are typically resource-constrained. Various techniques have been introduced to mitigate this problem, including network quantization [1, 2, 3, 4, 5], network pruning [6, 7] and efficient architecture design [8, 9, 10, 11, 12].

In 2016, Courbariaux and Bengio [13] first showed the potential of the extreme quantization level that uses only 1-bit to represent both weights and activations. By representing $+1$ with an unset bit and -1 with a set bit, the multiplications of weights and activations can be executed with xnor gates, saving

hardware resources and greatly reducing power consumption. Such BNN model was able to achieve comparable accuracy results on small datasets like CIFAR10 [14] and SVHN [15] but on wider dataset like Imagenet [16] a relevant accuracy drop was reported. Recent works [4, 17, 18, 19, 20, 21, 22, 23, 24, 25] on BNNs have significantly improved the accuracy on large datasets like Imagenet filling the gap with real-valued networks.

Most of the BNNs do not fully exploit the benefits of 1-bit quantization, since they exclude from binarization the first and last layers that normally work with fixed-point numbers. In general, the number of parameters and the computational effort of the first layer are relatively low compared to intermediate deep convolutional layers employed in VGG [26] or ResNet [27] models, since input data has typically fewer channels (e.g. color images have three channels). This usually leads to deploy the first layer of BNN models using floating-point or quantizing it using 8-bit; the consequence is that two different type of multipliers (8-bit for first layer, binary for the remaining), with different bit widths, are needed to execute the computations leading to a solution which increases the power consumption (8-bit multiplier requires more power than xnor) and consumes more hardware resources (e.g. an FPGA design) than xnor gates. Conversely, the challenge of binarizing both weights and activations in the input layer is due to the small number of input channels [28]. Therefore, almost all the works addressing the binarization of the first layer tried to increase the number of input channels to enrich data representation.

FBNA [28] proposes a two-step optimization scheme that consists of binarization and pruning; during binarization phase the number of input channels is increased by a factor $256\times$ and then, during pruning, lowest bits of input data are dropped away. The constraint of FBNA is that the encoded vector must be a power of two. BIL [29] attempts to directly unpack the 8-bit fixed-point input data, called *DBID*, and adding an additional binary pointwise convolutional layer between the unpacked input data and the first layer to increase the number of channels, dubbed as *BIL*. The authors of FracBNN [30] propose to use thermometer encoding to transform a pixel to a thermometer vector (expanding each input channel to 32 binary channels) that then is transformed to the $\{-1, +1\}$ bipolar representation.

In contrast with previous works where the number of input channels has been increased, our method directly uses the fixed-point representation of a pixel. The results show that the proposed technique is competitive both in term of efficiency and accuracy. Our contributions can be summarized as follows:

- we propose a general approach to binarize the first layer of a CNN using the native 8-bit fixed-point inputs. We rearrange the 8-bit input data into 8 bit planes, each bit plane is consumed by a binary depth-wise convolutional layer which gives more importance (using a multiplier, actually a shift operation) to the most significant bit planes. Finally, all feature maps are fused together through an addition operator. The entire process, depicted in Fig. 3, does not rely on floating-point computation, resulting more suitable to be deployed on ASIC or FPGA systems.

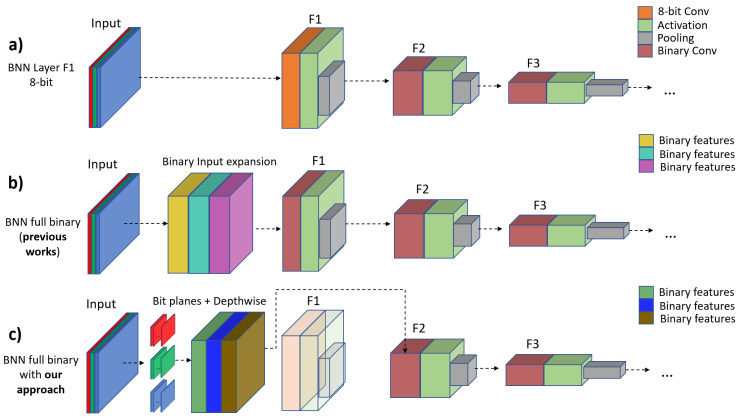


Fig. 1: *(a)* Standard scenario of BNNs where the first convolutional layer is not binarized; weights and inputs are used in 8-bit/floating-point representation. *(b)* Typical approach of the works that binarized the first layer F_1 incrementing the number of input channels; in this case the input expansion is actually an additional layer. *(c)* Our approach, where depth-wise convolutions are applied to input bit-planes and the resulting maps can replace the F_1 layer, producing a more compact model.

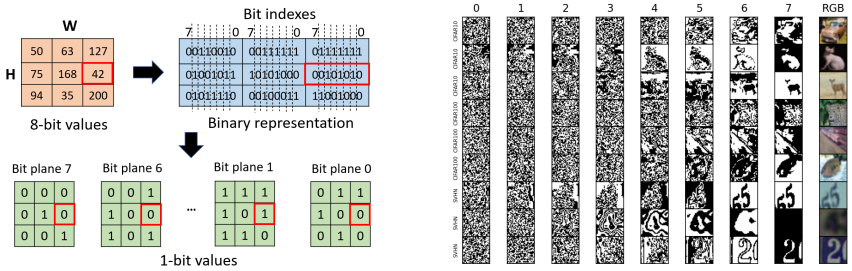
- we show that the feature maps resulting from our bit-plane manipulations allow to skip the original[3] F_1 first network layer (see Fig. 1c) with a minimal accuracy loss, leading to a model which uses less BMACs.
- we evaluate our concept on three classification datasets (SVHN, CIFAR10 and CIFAR100 [14]) showing that our solution outperforms all previous methods introduced to binarize the input layer.

2 Method

A common CNN model employed for computer vision problems works with RGB input images; it takes an input volume with three channels ($H \times W \times C$, where C is the number of channels) and extracts the features using convolutional blocks. To increase the receptive field of the network, a sequence of *pooling* operations is used. Each input pixel p is usually a fixed-point integer with 8 bit precision, namely $p = \sum_{m=0}^{7} x_m \cdot 2^m$.

In BNNs, typically the first layer (usually a convolutional one) is not binarized, all the input pixels are processed using 8-bit weights, producing F_1 output 8-bit feature maps (Fig. 1a). The previous works in literature that addressed the problem to generate F_1 binary feature maps, adopted different techniques to increase the number of input channels C (generating a more sparse representation) in order to use binary weights and inputs for layer F_1; usually a

[3] Before the addition of our depth-wise convolutions.



(a) Bit Planes example

(b) Bit Planes on CIFAR10, CIFAR100 and SVHN

Fig. 2: (a) Example of bit plane representation for a 3×3 8-bit image. (b) Image representation in bit planes. Each column refers to a bit index extracted from image; for representation purposes, bit 1 is converted to 255 while bit 0 remains 0. In this example all bit planes refer to channel G of RGB images.

good tradeoff between accuracy and increment of first layer MACs is to wide the number of channels C by $32\times$ [29, 30]. This process is depicted in Fig. 1b, where the increment of input channels leads to a bigger model footprint; in fact, a linear increment of the number of input channels, linearly increases also the kernel parameters of a 2D convolutional layer.

The intuition behind our approach is that, extracting a different bit plane for each bit position, the semantic spatial information is preserved for most of the high index bits (4 to 7), as shown in Fig. 2b. Lower bit indexes $(0-3)$ contain less correlated spatial information of image pixels and, depending on the dataset, they can be selectively omitted to further reduce the computational effort. The overall diagram of our method is reported in Fig. 3 and it is composed by the following steps:

1. **Bit Rearrangement:** An input image $\mathcal{I}$ $(W, H, C$, where C is the number of channels), having M bits for each pixel (usually 8), is rearranged into bit planes (as shown in Fig. 3a); each 8-bit input channel is decomposed into eight 1-bit planes. A bit plane x is a 1-bit map containing only the bit of index x for all pixels (see Fig. 2a). The bit-plane image bp corresponding to channel c can be indicated as $\mathcal{I}(c, bp)$.
2. **Feature Extraction:** Each binary bit-plane is consumed by a binary depth-wise convolution layer that generates $\mathbf{N}$ feature maps for each bit plane, as reported in Fig. 3b. The output of feature extraction $(\mathcal{FE})$ step can be formulated as:

$$\mathcal{FE}(c, bp) = \gamma(c, bp) \frac{(\mathcal{I}(c, bp) * W(c, bp) + b(c, bp)) - \mu(c, bp)}{\sigma(c, bp)} + \beta(c, bp) \tag{1}$$

where $*$ is the convolution operator, $W(c, bp)$ and $b(c, bp)$ represent the weights of the depth-wise convolution while γ, μ, σ and β are the Batch Normalization (BN) [31] parameters; Eq. 1 refers to a single feature map of depth-wise convolution, which is dependent on channel c and bit plane bp. In Eq. 1 the non-linear activation function can be omitted because binarization of activation and weights already introduce non-linearity. The use of Batch Normalization after each binary layer plays a key role in BNNs because it promotes a smoother optimization process allowing a stable behavior of the gradients. BN layer is usually executed in floating-point precision when mixed with binary layers, but the authors of [32] proved that it can be executed with 8-bit fixed point without accuracy loss.

3. **Features Re-Weight:** Following the intuition based on Fig. 2b, where high index bit planes preserve the spatial information of the image, this stage re-weights the feature maps based on the bit plane index. Higher bit planes are multiplied by higher scalar values. In order to simplify this stage, the multiplication can be replaced by a shift operation. The N feature maps of each bit plane are shifted by the same quantity (namely a power of two).
4. **Features Fusion:** The re-weighted feature maps, corresponding to a different 8-bit input channel, are summed to combine the information encoded by different bit indexes and can be expressed as:

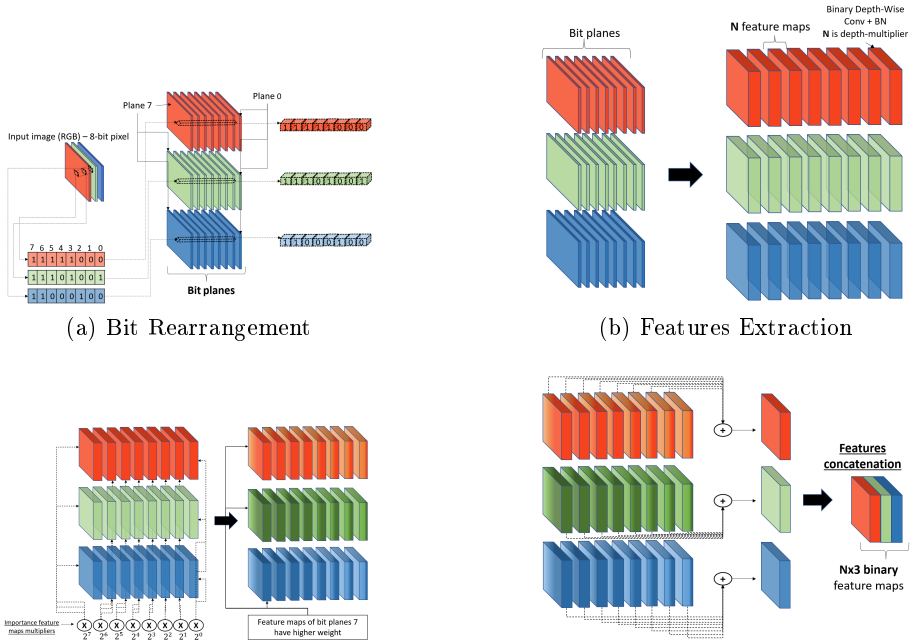
$$\mathcal{FWF}(c, bp) = \sum_{i=0}^{M} \mathcal{FE}(c, bp) \cdot 2^i \tag{2}$$

In Eq. 2 the multiplication by 2^i represents the re-weight of feature maps that can be implemented with a shift operation. The sum instead can be implemented accumulating features over 32-bit register; if the subsequent layer extracts *sign* from inputs, then the 32-bit output maps can be reduced to 1-bit saving memory overhead. The N feature maps corresponding to a different channel are concatenated to create a volume of $N \times 3$ maps that is used to feed the network, Fig. 1c. Such volume of N maps can replace the first layer of the CNN with almost no accuracy loss, as showed in Sec. 4 and, thus reducing the complexity of the overall model. F_1 requires a topology change of the first network layer when its weights and inputs are binarized and the expansion of depth-wise convolutions (Fig. 3b) can be set up in order to keep a number of feature maps equivalent to the layer F_1.

Table 1 reports the MACs of different approaches used to binarize input layer of a CNN, reporting the theoretical speedup of the methods; our solution is clearly competitive, in terms of MACs, with respect to the baseline (input not binarized) and other existing approaches.

3 Datasets and Implementation Details

We evaluate our method on three classification datasets: CIFAR10, CIFAR100 and SVHN with different BNN architectures. For each model architecture we



(a) Bit Rearrangement (b) Features Extraction

(c) Features Re-Weight (d) Features Fusion

Fig. 3: Binarization process of input layer. (a) shows the rearrangement phase that extracts, for each bit position of the encoded pixel a bit plane. (b) shows the binary depth-convolution block applied to each bit plane; the depth multiplier (N) is an hyperparameter and it is dataset and model dependent. (c) shows how to weight differently feature maps extracted from different bit planes; maps related to most significant bits receive a higher multiplication factor. In (d), the feature maps related to the same 8-bit input channel are fused together through an addition.

tested different state-of-the-art binarization techniques of input layer; input binarization does not modify the other layers of the network, which remain unaltered. For each dataset, we conducted our experiments with the same training procedure (same number of epochs, optimizer, learning rate scheduling, loss function) for all topologies without adding distillation losses or special regularization to the overall loss function. The binarization of weights and activations always happens at training time using an approximation of the gradient (STE [34] or derived solution that are model dependent) for *sign* function. The augmentation procedure for all datasets is performed with floating-point arithmetic but, before feeding data to the network, input image is quantized using 8-bit fixed precision. We adopted the following datasets:

Method	Type	# MACs	# weights	$\frac{MACs\ method}{MACs\ Baseline}$ [1]	Speedup[2]
Baseline	8-bit	$HWCF^2F_1$	CF^2F_1	1×	1×
DBID[29]	1-bit	$HWCMF^2F_1$	$CMK + F^2F_1K$	8×	1.12×
BIL[29]	1-bit	$HWK\left(CM + F^2F_1\right)$	CKF^2F_1	10.8×	0.81×
Thermometer[30]	1-bit	$HWCKF^2F_1$	CF^2N_1M	32×	0.27×
ours $(P = 8, N_1)$	1-bit	$HWCPF^2N_1$	CPF^2N_1	2.6×	**3.42**×
ours $(P = 4, N_1)$	1-bit	$HWCPF^2N_1$	CMF^2N_1	1.3×	**6.84**×
ours $(P = 4, N_2)$	1-bit	$HWCPF^2N_2$	CPF^2N_2	1×	**9**×

[1] A lower ratio means a higher reduction of MACs.
[2] According to [33] (Fig. 2), the worst case speedup of binary convolution compared to 8-bit is 9×.

Table 1: Comparison of the first layer MACs required by our method with respect to the state of the art solutions. Input data has a shape $H \times W \times C\,(32 \times 32 \times 3)$ and a precision of M bits; in this example the first convolutional layer has $F_1 =$ (128) filters with size $F \times F\,(3)$. The expansion channels is $K = 32$ for methods [29, 30]. The depthwise multiplier of our method can be chosen as $N_1 = \left\lfloor \frac{F_1}{C} \right\rfloor = 42$. We conducted our experiments using also a lower value, $N_2 = 32$ instead of N_1 and only 4 bits of input pixels. P represents the number of bit planes extracted by step 3a.

CIFAR10 and CIFAR100 The RGB images are scaled to the interval $[-1.0\,;+1.0]$ and the following data augmentation was used: zero padding of 4 pixels for each size, a random 32×32 crop and a random horizontal flip. No augmentation is used at test time. The models have been trained for 140 epochs.

SVHN The RGB input images are scaled to the interval $[-1.0\,;+1.0]$ and the following data augmentation procedure is used: random rotation (±8 degrees), zoom ($[0.95, 1.05]$), random shift ($[0; 10]$) and random shear ($[0; 0.15]$). The models have been trained for 70 epochs.

We evaluated the following networks:

VGG-Small[35] Network structure is the following: $2 \times (128 - C3) + MP2 + 2 \times (256 - C3) + MP2 + 2 \times (512 - C3) + MP2 + FC1024 + FC1024 + Softmax$[4]. The VGG-Small model adopted uses the straight-through-estimator (STE) to approximate the gradient on non-differentiable layers[2, 34].

VGG-11[19] Network structure is the following: $64 - C3 + MP2 + 128 - C3 + MP2 + 2 \times (256 - C3) + MP2 + 2 \times (512 - C3) + MP2 + 2 \times (512 - C3) + MP2 + Softmax$ [4]. Even VGG-11 uses the STE estimator for binarization operation during back-propagation.

[4] $m \times (n - CK)$ stands for m consecutive convolutional layers, each one with n output channels and K kernel size. $MP2$ is the max pooling layer with subsample 2 while FCx is a fully-connected layer having x neurons. $Softmax$ represents the last dense classification layer using softmax as activation.

BiRealNet[17] It is a modified version of classical ResNet that proposes to preserve the real activations before the sign function to increase the representational capability of the 1-bit CNN, through a simple shortcut. Bi-RealNet adopts a tight approximation to the derivative of the non-differentiable sign function with respect to activation and a magnitude-aware gradient to update weight parameters. We used two instances of the network, an *18-layer* and a *34-layer* Bi-Real net [5].

ReactNet[24] To further compress compact networks, this model constructs a baseline based on MobileNetV1 [8] and add shortcut to bypass every 1-bit convolutional layer that has the same number of input and output channels. The 3×3 depth-wise and the 1×1 point-wise convolutional blocks of MobileNet are replaced by the 3×3 and 1×1 vanilla convolutions in parallel with shortcuts in React Net[6]. As for Bi-Real Net, we tested two different versions of React Net: a *18-layer* and a *34-layer*.

4 Results and Conclusions

	VGG-Small	VGG-11	BiReal-18	BiReal-34	React-18	React-34	
$DBID$[29]$(P=8)$	84.5	77.6	81.8	85.0	86.0	86.7	$1st$
BIL[29]$(P=8)$	82.0	78.9	81.3	82.2	84.0	83.5	
$Therm$[30]$(K=32)$	84.2	80.1	85.2	85.3	86.6	86.6	
ours$(P=8, N_1)$	**85.9**	79.1	**87.7**	**88.5**	**89.9**	**90.2**	
baseline	89.2	84.7	89.1	89.3	90.6	90.6	
$DBID(P=4)$	83.6	76.8	74.9	83.7	83.7	85.3	$2nd$
$BIL(P=4)$	80.9	82.4	80.8	82.3	82.7	83.4	
$Therm(K=16)$	83.8	79.6	84.7	85.9	86.5	86.8	
ours$(P=4, N_2)$	**85.0**	78.3	**86.9**	**87.7**	**88.5**	**89.0**	
baseline	88.3	83.7	87.4	88.3	88.8	89.1	

Table 2: Top1 accuracy (%) results of test set on CIFAR10. In first part we report the result of first test scenario (standard conditions); in second half, the results achieved in the second scenario (reducing the MACs of binarization of input layer).

The validation of our solution has been accomplished through two different test scenarios; in the first one, we compared the accuracy (measured on test set) of our binarization method w.r.t. the state-of-the-arts input layer binarization

[5] Refer to the following `https://github.com/liuzechun/Bi-Real-net` repository for all the details.

[6] Refer to the following `https://github.com/liuzechun/ReActNet` repository for all the details.

	VGG-Small	VGG-11	BiReal-18	BiReal-34	React-18	React-34	
DBID[29]($P=8$)	94.5	92.2	94.3	95.1	94.9	95.1	
BIL[29]($P=8$)	93.5	92.1	94.3	93.4	94.1	94.7	
Therm[30]($K=32$)	89.7	88.9	89.2	89.8	89.8	90.2	$1st$
ours($P=8, N_1$)	**94.8**	**93.4**	94.3	95.0	**95.1**	**95.7**	
baseline	95.7	95.5	94.3	95.1	95.5	95.9	
DBID($P=4$)	94.3	92.1	94.3	94.7	94.8	95.0	
BIL($P=4$)	93.4	92.1	94.4	93.5	93.8	94.5	
Therm($K=16$)	89.5	88.6	89.8	89.7	89.8	90.1	$2nd$
ours($P=4, N_2$)	**94.8**	**93.3**	94.3	**95.0**	**95.1**	**95.7**	
baseline	95.6	95.0	94.4	95.1	95.5	96.0	

Table 3: Top1 accuracy (%) results of test set on SVHN.

	VGG-Small	VGG-11	BiReal-18	BiReal-34	React-18	React-34	
DBID[29]($P=8$)	53.6	43.1	51.8	58.5	56.3	58.0	
BIL[29]($P=8$)	50.0	42.9	52.7	56.0	55.4	55.5	
Therm[30]($K=32$)	53.0	43.5	57.2	57.1	57.4	57.9	$1st$
ours($P=8, N_1$)	**56.5**	**46.0**	**58.7**	**60.6**	**61.7**	**62.9**	
baseline	60.6	52.3	63.4	65.0	64.9	65.3	
DBID($P=4$)	52.3	41.8	50.5	56.5	55.2	56.7	
BIL($P=4$)	49.5	42.0	52.1	54.5	52.1	53.6	
Therm($K=16$)	52.1	42.6	56.7	54.5	56.8	58.6	$2nd$
ours($P=4, N_2$)	**54.8**	**44.5**	**57.7**	**59.6**	**60.2**	**62.0**	
baseline	60.3	50.3	60.0	61.7	62.0	63.4	

Table 4: Top1 accuracy (%) results of test set on CIFAR100.

approaches, keeping unaltered the structure of the network except for the input data binarization layer (first half of Tables 2, 3 and 4). In this first scenario all the 8-bits planes are exploited, layer F1 (Fig. 1) is executed and our proposed solution is able to reach a better accuracy compared to other input binarization methods, closing the accuracy gap with the baseline.

In the second scenario, to further reduce the MACs of our solution, we propose an optimization of our method that uses only the 4 most significant bits and reduces the depth-wise multiplier from N_1 to N_2 (the reduction to 4 bits is based on Fig. 2b, that shows how the bit planes corresponding to less significant bits convey less information). In the second half of tables 2, 3 and 4, we report the results of the optimized version compared with other solutions properly modified in order to compute an equivalent number of channels [7]. As reported, our solution is able to preserve the baseline accuracy using less input bits while

[7] For *DBID*, *thermometer* and *baseline* methods, we reduced to 32 the number of output channels of layer $F1$; for *BIL* and *ours*, we skipped the layer $F1$ because

the other methods get a consistent accuracy drop when reducing input bits and binary channels.
Differently from other works, our solution re-weights the feature extracted by bit-planes giving more importance to the features corresponding to the most significant bit-planes; this stage contributes to scale down the footprint of our binarization approach simplifying the deployment on resource constrained devices (low-power embedded CPUs). Furthermore, the accuracy of our method is higher than *thermometer encoding*[30], which preserves the feature similarity after binarizing the input layer, as pointed out by Anderson et al. [28].

In conclusion, this paper introduced a novel input layer binarization method that reaches higher accuracy when compared to state-of-the-art solutions reducing the gap to the baseline on average by 2.2 percentage points. Our solution was able to preserve model accuracy when only 4 bits of input pixels are used in the input binarization layer, proving to be more resource-constrained device friendly than existing ones. In the future, we intend to further investigate the latency speedup of our method on real hardware devices like Raspberry Pi Model 3B/4B exploiting the computation capabilities of NEON ARM[8] SIMD engine.

References

[1] Jungwook Choi et al. "Pact: Parameterized clipping activation for quantized neural networks". In: *arXiv preprint arXiv:1805.06085* (2018).
[2] Itay Hubara et al. "Binarized neural networks". In: *Advances in neural information processing systems* 29 (2016).
[3] Xiaofan Lin, Cong Zhao, and Wei Pan. "Towards accurate binary convolutional neural network". In: *Advances in neural information processing systems* 30 (2017).
[4] Mohammad Rastegari et al. "Xnor-net: Imagenet classification using binary convolutional neural networks". In: *European conference on computer vision*. Springer. 2016, pp. 525–542.
[5] Shuchang Zhou et al. "Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients". In: *arXiv preprint arXiv:1606.06160* (2016).
[6] Song Han, Huizi Mao, and William J Dally. "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding". In: *arXiv preprint arXiv:1510.00149* (2015).
[7] Wei Wen et al. "Learning structured sparsity in deep neural networks". In: *Advances in neural information processing systems* 29 (2016).
[8] Andrew G Howard et al. "Mobilenets: Efficient convolutional neural networks for mobile vision applications". In: *arXiv preprint arXiv:1704.04861* (2017).

the convolution operation is already exploited within the input layer binarization process. For *DBID*, *BIL* and *ours* we used only the 4 most significant bits of input data. For *thermometer* we applied also a reduced expansion factor of $K = 16$.

[8] `https://www.arm.com/technologies/neon`

[9] Ningning Ma et al. "Shufflenet v2: Practical guidelines for efficient cnn architecture design". In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 116–131.

[10] Mingxing Tan and Quoc Le. "Efficientnet: Rethinking model scaling for convolutional neural networks". In: *International conference on machine learning*. PMLR. 2019, pp. 6105–6114.

[11] Mingxing Tan and Quoc Le. "Efficientnetv2: Smaller models and faster training". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 10096–10106.

[12] Qibin Hou, Daquan Zhou, and Jiashi Feng. "Coordinate attention for efficient mobile network design". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2021, pp. 13713–13722.

[13] Matthieu Courbariaux et al. "Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1". In: *arXiv preprint arXiv:1602.02830* (2016).

[14] Alex Krizhevsky, Geoffrey Hinton, et al. "Learning multiple layers of features from tiny images". In: (2009).

[15] Yuval Netzer et al. "Reading digits in natural images with unsupervised feature learning". In: (2011).

[16] Olga Russakovsky et al. "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision (IJCV)* 115.3 (2015), pp. 211–252. DOI: 10.1007/s11263-015-0816-y.

[17] Zechun Liu et al. "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)*. 2018, pp. 722–737.

[18] Jiaxin Gu et al. "Projection convolutional neural networks for 1-bit cnns via discrete back propagation". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 33. 01. 2019, pp. 8344–8351.

[19] Yinghao Xu et al. "A main/subsidiary network framework for simplifying binary neural networks". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 7154–7162.

[20] Haotong Qin et al. "Forward and backward information retention for accurate binary neural networks". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 2250–2259.

[21] Joseph Bethge et al. "MeliusNet: Can binary neural networks achieve mobilenet-level accuracy?" In: *arXiv preprint arXiv:2001.05936* (2020).

[22] Adrian Bulat, Brais Martinez, and Georgios Tzimiropoulos. "High-capacity expert binary networks". In: *arXiv preprint arXiv:2010.03558* (2020).

[23] Brais Martinez et al. "Training binary neural networks with real-to-binary convolutions". In: *arXiv preprint arXiv:2003.11535* (2020).

[24] Zechun Liu et al. "Reactnet: Towards precise binary neural network with generalized activation functions". In: *European conference on computer vision*. Springer. 2020, pp. 143–159.

[25] Xulong Shi et al. "RepBNN: towards a precise Binary Neural Network with Enhanced Feature Map via Repeating". In: *arXiv preprint arXiv:2207.09049* (2022).

[26] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

[27] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[28] Alexander G Anderson and Cory P Berg. "The high-dimensional geometry of binary neural networks". In: *arXiv preprint arXiv:1705.07199* (2017).

[29] Robert Dürichen et al. "Binary Input Layer: Training of CNN models with binary input data". In: *arXiv preprint arXiv:1812.03410* (2018).

[30] Yichi Zhang et al. "FracBNN: Accurate and FPGA-efficient binary neural networks with fractional activations". In: *The 2021 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*. 2021, pp. 171–182.

[31] Sergey Ioffe and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: *International conference on machine learning*. PMLR. 2015, pp. 448–456.

[32] Lorenzo Vorabbi, Davide Maltoni, and Stefano Santi. "Optimizing dataflow in Binary Neural Networks". In: *arXiv preprint arXiv:2304.00952* (2023).

[33] Tom Bannink et al. "Larq compute engine: Design, benchmark and deploy state-of-the-art binarized neural networks". In: *Proceedings of Machine Learning and Systems* 3 (2021), pp. 680–695.

[34] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. "Estimating or propagating gradients through stochastic neurons for conditional computation". In: *arXiv preprint arXiv:1308.3432* (2013).

[35] Dongqing Zhang et al. "Lq-nets: Learned quantization for highly accurate and compact deep neural networks". In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, pp. 365–382.