Received September 12, 2019, accepted September 27, 2019, date of publication October 3, 2019, date of current version November 4, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2945488

BitFlow-Net: Toward Fully Binarized Convolutional Neural Networks

LIJUN WU[1,2], PEIQING JIANG[1,2], ZHICONG CHEN[1,2], XU LIN[1,2], YUNFENG LAI[1,2], PEIJIE LIN[1,2], AND SHUYING CHENG[1,2]

[1]College of Physics and Information Engineering, Fuzhou University, Fuzhou 350116, China
[2]Jiangsu Collaborative Innovation Center of Photovoltaic Science and Engineering, Changzhou 213164, China

Corresponding authors: Zhicong Chen (zhicong.chen@fzu.edu.cn) and Yunfeng Lai (yunfeng.lai@fzu.edu.cn)

This work was supported in part by the Fujian Provincial Department of Science and Technology of China under Grant 2019H0006, and in part by the National Natural Science Foundation of China under Grant 51508105 and Grant 61601127.

ABSTRACT Binarization can greatly compress and accelerate deep convolutional neural networks (CNNs) for real-time industrial applications. However, existing binarized CNNs (BCNNs) rely on scaling factor (SF) and batch normalization (BatchNorm) that still involve resource-consuming floating-point multiplication operations. Addressing the limitation, an improved BCNN named BitFlow-Net is proposed, which replaces floating-point operations with integer addition in middle layers. First, it is derived that the SF is only effective in back-propagation process, whereas it is counteracted by BatchNorm in inference process. Then, in model running phase, the SF and BatchNorm are fused into an integer addition, named BatchShift. Consequently, the data flow in middle layers is fully binarized during modeling running phase. To verify its potential in industrial applications with multiclass and binary classification tasks, the BitFlow-Net is built based on AlexNet and verified on two large image datasets, i.e., ImageNet and 11K Hands. Experimental results show that the BitFlow-Net can remove all floating-point operations in middle layers of BCNNs and greatly reduce the memory for both cases without affecting the accuracy. Particularly, the BitFlow-Net can achieve the accuracy comparable to that of the full-precision AlexNet network in the binary classification task.

INDEX TERMS Binarized convolutional neural networks, model acceleration and compression, BatchShift.

I. INTRODUCTION

Rapid development of machine learning techniques has been greatly improving the level of intelligence and automation in numerous industrial applications [1]–[7]. In comparison to traditional machine learning algorithms, deep learning algorithms usually have automatic feature extraction capability and better performance contributed by deep neural network structure [8]. For a higher accuracy, most researchers tend to construct deeper and more complex networks [9], [10], which are usually at the cost of large network size and high computation. For instance, it required 138M parameters, 553MB of storage and 15B floating-point operations to classify an image with the size of 224×224 through the 19-layer VGGNet [9]. In many vision-based industrial applications (such as industrial robots [11], defect detection [12], [13], on-site monitoring [14] etc.), deep convolutional neural network (CNN) models are required to run in real time on embedded platforms with limited computing and storage resources. As a consequence, various model accelerating and/or compressing methods have been proposed to reduce the resource consumption of deep learning, such as quantization [15]–[17], pruning [18]–[20], distillation [21], low-rank decomposition [22], etc. Among all these methods, quantization is most compatible with embedded devices, as it could not only compress but also accelerate the model simultaneously since it replaces the floating-point operations with fix-point operations. Consequently, quantization has attracted ever-increasing focus from both academics and industry in recent years.

As the most efficient quantization method, binarization quantizes the operand from floating-point to only one-bit, which could greatly reduce model storage space and computing resource [23]. Courbariaux *et al.* [24] proposed the BinaryConnect to quantize weights of CNN to only one bit and then replace the multiply-accumulate operation by simple accumulations. Subsequently, Hubara *et al.* [25] proposed the binarized neural networks (BNNs) which further extend the idea of BinaryConnect by applying binarization on both weights and activations. With binarized weights

The associate editor coordinating the review of this manuscript and approving it for publication was Jingchang Huang.

and activations, it is possible to use bit operation to replace the original floating-point multiplication. This can not only compress but also speed up the network, especially on dedicated deep learning hardware. After all, a 32-bit floating-point multiplier requires approximately 200 Xilinx FPGA slices [26], and the 1-bit xnor gate only need a single slice. The BNNs achieved state-of-the-art results on small datasets (e.g., CIFAR-10, SVHN). However, the accuracy is severely degraded when it is tested on large data sets like ImageNet [27], due to a heavy information loss during binarization. Rastegari *et al.* [28] proposed XNOR-Net, which introduces scaling factor (defined as the channel-wise average of the absolute values of weights) to partly address the problem of accuracy degradation by softening the constraint of binarization. Besides, BatchNorm [29] is also adopted to update AlexNet [30] between adjacent binary convolutional layers (BinConv) to stabilize the training of binarized network. Consequently, XNOR-Net outperforms BNN by about 17%. To bridge the accuracy gap between the binarized networks and the full precision networks, binarized models with multiple bits were explored as well [31]. In [32], the ‘‘WAGE’’ framework was proposed to discretize both training and inference, where weights (W), activations (A), gradients (G) and errors (E) among layers are shifted and linearly constrained to low-bitwidth integers. Besides, batch normalization is further replaced by a constant scaling layer. It was shown that with a few more bits, the binarized model could acquire an accuracy that is comparable to their full-precision counterparts.

The obvious similarity among these works is that they all employ the floating-point scaling factor to relax the constraint of binarization and preserve the BatchNorm operation so as to obtain a higher accuracy. In the reference [25], the authors accelerate BatchNorm via bit operation, which replaces the multiplication operations by left or right binary shift. Such replacement is an inaccurate approximation of the multiplier by its power-of-2. To avoid calculating the scaling factors in each training epoch, Mcdonnell [33], [34] train a binary weight network with a constant factor α_{He} and achieves a good result. Recently, Bi-Real Net proposed by Liu *et al.* [35] achieves the top-1 accuracy that is up to 10% higher than the plain network based XNOR-Net, which is based on the residual network that connects the real activations to consecutive blocks by an identity shortcut. In Bi-Real Net, the network is retrained for an extra epoch after it has converged, so as to absorb the scaling factor. But, our study is mainly for the common plain networks, and thus the XNOR network is selected for comparison. However, the aforementioned solutions still rely on floating-point operations (for example, during BatchNorm or multiplication with scaling factor), which would cause significant extra computation cost especially when implementing binarized networks on customizable hardware such as FPGA and ASIC. Besides, it requires frequent data conversion between fixed-point and floating-point data to multiply fixed-bit output of BinConv layers with floating-point scaling factor. Such conversion will cost extra memory resource and time, which compromises the benefits brought by the binarization, as is illustrated in Fig. 1a. Extra data conversions and floating-point multiplications in BatchNorm and scaling factors can be observed in XNOR-Net.

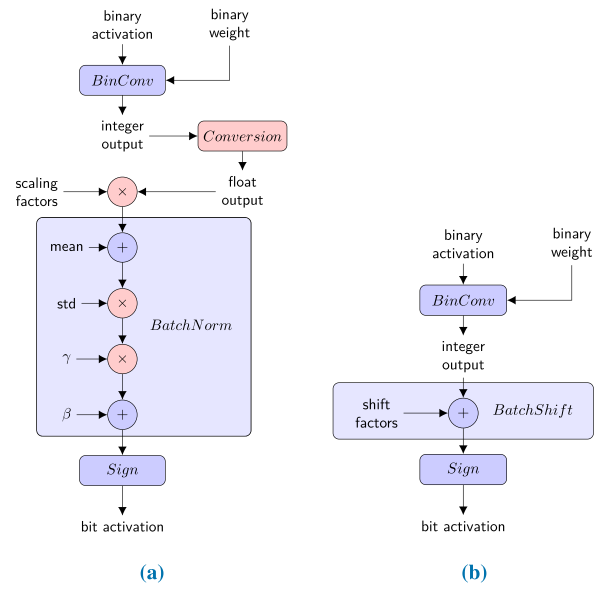

FIGURE 1. Difference between the XNOR-Net and BitFlow-Net. (a) The scaling factor multiplication and BatchNorm after the BinConv in XNOR-Net. The pink steps involve the most time-consuming float point operations, i.e. the integer-float conversion and the float point multiplication. (b) The BatchShift after the BinConv in BitFlow-Net with only a fixed-point accumulation.

To eliminate redundant floating-point operations, the role of the scaling factor in the inference and back-propagation processes is studied separately. Theoretically, it is derived that the floating-point multiplication and addition operations in the inference processes could be greatly simplified by fusing them together into a shift operation, called BatchShift. Since there is only inference during the model running phase, the BatchShift can be employed to realize fully binarized operations in the middle layers when running the model, as shown in Fig. 1b. Based on framework of XNOR-Net, we propose an improved BCNN named as BitFlow-Net, which adopts the BatchShift during the inference process without compromising the network accuracy. BitFlow-Net is firstly tested on the complex multiclass classification problem using the famous ImageNet dataset. Similar to the XNOR-Net, the binarization of parameters in BitFlow-Net will inevitably lead to loss of network accuracy. Moreover, in order to explore the potential of binarized networks for the industrial applications with binary classification problem, BitFlow-Net is also tested on a hand images based gender recognition challenging problem using the image dataset named 11K Hands [36]. That is because resource-limited embedded terminal devices are often adopted in industrial applications to handle binary classification tasks [37], such as detection of defect, damage, fault, etc.

The remainder of this study is structured as follows. In Sec. II, the XNOR-Net is first revisited, and then the role of scaling factor during the inference and back-propagation processes is analyzed. Sec. III conducts theoretical deduction and proposes the new BCNN BitFlow-Net. In Sec. IV, based on the ImageNet benchmark and the 11K Hands, some experiments are carried out to verify the performance of the BitFlow-Net. Conclusions are drawn in Sec. V.

II. REVISITING XNOR-NET

This section revisits the XNOR-Net in terms of binarization and discusses how the accuracy is guaranteed. Then, the mechanisms of scaling factor and BatchNorm are investigated, based on which some unnoticed facts that can potentially further improve the performance are revealed.

A. BASICS OF BINARIZED NEURAL NETWORKS

Convolution operations mainly consist of multiplication-accumulation operations of inputs and kernel weights. But in binarized networks, both inputs and weights are binarized to be either $+1$ or -1, which makes it possible to replace the complex multiplication-accumulation operations in dot product with high efficient xnor-bitcount operations [31], as written in Eq.(1):

$$x \cdot y \approx bitcount(xnor(x^b, y^b)) \tag{1}$$

where x^b and y^b are binarized versions of tensor x and y. This leads to huge reduction of computation. The binarization of input and weight is applied using the deterministic sign function:

$$sign(x) = \begin{cases} +1, & x \geq 0; \\ -1, & otherwise. \end{cases} \tag{2}$$

The core idea of XNOR-Net is to approximate full-precision weight or activation with its sign multiplied by the scaling factor in order to reduce quantization loss. It is also reported that the scaling factor β for inputs is much less efficient than the scaling factor α for weight [28], which suggests that β can be neglected for XNOR-Net in practice. The scaling factor α for weight is formulated in Eq. (3), where the n is the number of elements of weight in one output channel and W is the weight tensor. The product of the sign of weights with its α is formulated as the optimal estimation of the original full-precision weights under the constraint of binarization. According to Eq. (3), it is obvious that α is always greater than 0.

$$\alpha = \frac{1}{n}\sum_{i=1}^{n}|W_i| \tag{3}$$

In XNOR-Net, Rastegari et al. also discussed the suitable block structure for binarization networks. As known, the blocks broadly used in CNN are in the following order: 1-Convolutional, 2-Batch Normalization, 3-Activation and 4-Pooling. To decrease the information loss in binarization, XNOR-Net adopts a new block structure that is in the order of 1-Batch Normalization, 2-Binary Activation, 3-Binary Convolutional, and 4-Pooling [28], which is also adopted in our work. In this way, the output of BinaryConv is then normalized before being binarized.

According to the flowchart illustrated in Fig. 1a, there are three floating-point multiplications, two accumulations and a bit-floating point conversion in every convolution block operation. Such floating-point operations will cost considerable extra memory and computing resource for binarization networks. As the depth of the network increases, the impact will be even greater. Motivated by this, our main goal is to eliminate floating-point computation as much as possible. Therefore, in next subsection, the mechanism of floating-point scaling factor will be first explored.

B. MECHANISM OF SCALING FACTOR

A binary convolution operation is defined as:

$$Y = X^b \circledast (W^b\alpha)^1 \tag{4}$$

where $\circledast$ indicates convolution operation, $X^b, W^b \in \{-1, +1\}$, and $\alpha \in \mathbb{R}^+$. Since the scaling factor α formulated in Eq.(3) is computed channel-wise, Eq.(4) could be reformulated as Eq.(5). As α is extracted, the floating-point convolution operation could be replaced by xnor-bitcount operation formulated in Eq.(1) that can be executed efficiently by resource restricted devices. To multiply with the floating-point scaling factor, the integer output of binary convolution must be converted to floating-point type in advance. Then, the floating-point scaled output is normalized by a BatchNorm layer before being binarized again. Such a procedure brings frequent conversions between integer and floating-point data types, which costs extra computation and memory. Moreover, we find that these extra computations seem to be useless and bring no benefits to the model accuracy during the model running phase.

$$Y = (X^b \circledast W^b)\alpha \tag{5}$$

We first start from the inference process. Usually, BatchNorm is used to channel-wise normalize the input data to zero mean and unit variance across all samples in one input batch. When the scaling factor operation is followed by the BatchNorm layer, the output is normalized to the unit variance regardless of the scaling factor. Therefore, the BatchNorm layer counteracts the effect of scaling factor, as demonstrated in Eq.(6). Since the scaling factor is channel-wise, it can be extracted, i.e., $E[\alpha x] = \alpha E[x]$, and $Var[\alpha x] = \alpha^2\, Var[x]$.

$$\begin{aligned} BN(\alpha x) &= \frac{\alpha x - E[\alpha x]}{\sqrt{Var[\alpha x] + \epsilon}}\gamma + \beta \\ &= \frac{\alpha x - \alpha E[x]}{\sqrt{\alpha^2 Var[x] + \epsilon}}\gamma + \beta \\ &= \frac{\alpha(x - E[x])}{\alpha\sqrt{Var[x] + \epsilon}}\gamma + \beta \\ &= \frac{x - E[x]}{\sqrt{Var[x] + \epsilon}}\gamma + \beta \\ &= BN(x) \end{aligned} \tag{6}$$

As shown in Eq.(6), the BatchNorm output with scaled input is equal to the one without scaled input, which means that the scaling factor does not actually achieve the purpose of improving the binary model by affecting the output. If the scale factor of XNOR-Net is removed during the model running phase, the output will not be different. This is inconsistent with that the scaling factor increases the network accuracy. Therefore, the scaling factor may only take effect in the back-propagation by improving the gradient weight distribution, which will be discussed in detail as belows.

In the XNOR-Net, the scaling factor and $sign(x)$ are combined as a scaled sign function:

$$\widetilde{W} = \alpha \cdot sign(W). \tag{7}$$

For each full-precision weight W_i^C in channel C, its gradient with respect to the loss L is:

$$\begin{aligned}\frac{\partial L}{\partial W_i^C} &= \frac{\partial L}{\partial \widetilde{W}_i^C} \cdot \frac{\partial sign(W_i^C)}{\partial W_i^C} \cdot \alpha \\ &\quad + sign(W_i^C) \cdot \frac{1}{n} \cdot \sum_{j=1}^{n} [\frac{\partial L}{\partial \widetilde{W}_j^C} \cdot sign(W_j^C)]\end{aligned} \tag{8}$$

In order to simplify the expression, firstly let

$$\begin{cases} grad = \dfrac{\partial L}{\partial \widetilde{W}_j^C} \\ sign = sign(W_j^C) \\ mean = \dfrac{1}{n}\displaystyle\sum_{j=1}^{n}[\frac{\partial L}{\partial \widetilde{W}_j^C} sign(W_j^C)] \\ \qquad = \dfrac{1}{n}\displaystyle\sum_{j=1}^{n} sign \cdot grad \end{cases} \tag{9}$$

It is clear that the first item in Eq.(8) is the scaled version of the original gradient of $sign(x)$. Since in this work, the Straight Through Estimator (STE) [25] is adopted, $\frac{\partial L}{\partial \widetilde{W}_i^C} \cdot \frac{\partial sign(W_i^C)}{\partial W_i^C} \cdot \alpha = \frac{\partial L}{\partial \widetilde{W}_i^C} \cdot \alpha$. The second item in Eq.(8) is contributed by α, in which $\frac{1}{n} \cdot \sum_{j=1}^{n}[\frac{\partial L}{\partial \hat{W}_j} \cdot sign(W_j)]$ is a measure of average change of weights after the update of this iteration. The second item serves as a compensation of the gradient of the first item and keeps the gradient steady. We make detailed qualitative analysis as belows.

If the weight in channel C is updated according to $\widetilde{W}_j^{C,update} = \widetilde{W}_j^C - grad$ as the one without scaling factor, it can be found from Eq. (7) and Eq. (8) that when $sign \cdot grad > 0$, $\widetilde{W}_j^C - grad$ will approach or even cross 0; when $sign \cdot grad < 0$, $\widetilde{W}_j^C - grad$ will be away from 0, as summarized in Table 1, where an arrow oriented to '0' indicates the value may approach or even cross 0, an arrow against '0' indicates the value is away from 0, an arrow on the left side of '0' means it's original value is negative, an arrow on the right side of '0' means it's original value is positive. In this case, $mean$ can be regarded as a measurement of the average variation of weight in channel C before and after the updating.

After introducing the scaling factor, the weight in channel C is updated according to $\widetilde{W}_j^{C,update} = \widetilde{W}_j^C - grad \cdot \alpha - sign \cdot mean$. The second item is just a scaled version of the $grad$ that update the weight with the same trend of $grad$ as shown in Table 1, while the third item means an additional updating of the weight in channel C according to $\widetilde{W}_j^{C,update} = \widetilde{W}_j^C - sign \cdot mean$. It can be seen that when $mean > 0$, $\widetilde{W}_j^C - sign \cdot mean$ will approach or even cross 0; when $mean < 0$, $\widetilde{W}_j^C - sign \cdot mean$ will be away from 0, as summarized in Table 1. Thus, the third item is a compensation of the second item, which compensates every weight with the average weight variation of its channel and thus can keep the gradient steady.

TABLE 1. Trend of $\widetilde{W}_j^C - grad$ and $\widetilde{W}_j^C - sign \cdot mean$ under different conditions of $grad$, $sign$ and $mean$.

	$\widetilde{W}_j^C - grad$		$\widetilde{W}_j^C - sign \cdot mean$	
	$grad > 0$	$grad < 0$	$mean > 0$	$mean < 0$
$sign > 0$	$0 \leftarrow$	$0 \rightarrow$	$0 \leftarrow$	$0 \rightarrow$
$sign < 0$	$\leftarrow 0$	$\rightarrow 0$	$\rightarrow 0$	$\leftarrow 0$

C. DISCUSSION OF THE SCALING FACTOR EFFECT IN THE MODEL TRAINING PHASE

Firstly, a very important phenomenon of binarized networks should be noted: the output of a binary convolutional layer for identical input will not change, unless some of the weights' signs are reversed after weight updating. This means that in order to effectively train the model, the magnitude of gradient must be within a certain range during the model training phase. If the amplitudes of gradients are too small, the updated binary weights may be the same as before and the network will be difficult to converge. On the contrary, if the amplitudes of gradients are too large, the updated binary weights may change too much compared with the one before, and the network may be unstable and even may not converge at all. Thus, it is very important to stabilize the amplitude of gradient during training a binarized network.

Therefore, the percentage of weights, whose signs were changed after being updated, can be used to represent the effect of gradient on binarized networks, which is referred as the sign changing rate. It is inspired by [38], in which statistics were made on sign changing rate during training with different settings of learning rate. The results show the sign changing rate in binarized networks with learning rate of 0.01 is about 10^2 larger than that of a full-precision network. This makes the output of the binarized networks change frequently after every training iteration, which is extremely unstable compared to full-precision network and disrupts its training. Therefore, small learning rate is suggested in order to better train binarization model. In Section IV, further investigation on the change of signs over all training epochs will be made.

III. BITFLOW-NET

In binarized networks, the time-consuming floating-point computation in the middle layers is mainly introduced by the scaling factor and the BatchNorm. As discussed in previous section, floating-point scaling factor only takes effect in the back-propagation process that is part of the model training phase. However, during the model running phase, especially when the model is running on resource limited embedded devices, such floating-point operations waste resources and are unnecessary. Based on this observation, we propose a new binarized network named BitFlow-Net, which does not modify the model during the training phase. But, it fuses the scaling factor with the BatchNorm in the model running phase, so as to eliminate float-point operations and let the bit stream flows in the middle layers without being interrupted. This is implemented based on the characteristic of binarized networks, i.e., only the signs of inputs and weights but not the magnitudes are important, when they are binarized.

A. BATCHSHIFT - FUSING SCALING FACTOR WITH BATCHNORM

In binarized networks, the output of binary convolution is discrete, which may cause the loss surface to be extremely rough. Therefore, in binarized networks, BatchNorm is usually necessary to smooth the loss surface and obtain a high accuracy. However, when the BatchNorm exists, the output of the binary convolution must be converted to floating-point data for being normalized. This conflicts with our goal of making a binarized network with all data staying in bit flow. Therefore, we start from the expression of BatchNorm and focus on simplifying it into a floating-point free version.

As known, BatchNorm is defined as:

$$BN_B^{tr}(x) = \frac{x-\mu_B}{\sqrt{\sigma_B^2+\epsilon}}\gamma + \beta \tag{10}$$

where x is the input of a neuron, μ_B and σ_B^2 are the mean and variance of the x in a batch B respectively. γ and β are learnable affine factors. During the inference process, BatchNorm uses the average of μ_B and σ_B^2, i.e., the $E[x]$ and $Var[x]$ calculated and saved during the training phase, to approximate them.[2]

$$BN^{inf}(x) = \frac{x-E[x]}{\sqrt{Var[x]+\epsilon}}\gamma + \beta \tag{11}$$

Since the back-propagation method usually prefers normalized floating-point activation values, it is difficult to remove floating-point multiplications during the model training phase. Therefore, BitFlow-Net aims to eliminate the floating-point multiplications in the model running phase by reformulating the expression of BatchNorm. In the inference process, one can fuse the BatchNorm with input scale α :

$$BN^{inf}(\alpha x) = \frac{\alpha x-E[\alpha x]}{\sqrt{Var[\alpha x]+\epsilon}}\gamma + \beta \tag{12}$$

As shown in Eq.(6), $BN^{inf}(\alpha x) = BN^{inf}(x)$. Therefore, Eq. (12) can be reorganized as Eq.(13), so as to avoid multiplying the input with floating-point α.

$$BN^{inf}(x) = \frac{x-\frac{1}{\alpha}E[\alpha x]}{\sqrt{\frac{1}{\alpha^2}Var[\alpha x]+\epsilon}}\gamma + \beta \tag{13}$$

Since $E[\alpha x]$ and $Var[\alpha x]$ are the statistics that can be calculated in advance, they can be considered as constants. In this way, one can successfully reduce αx into simple x. Since all the parameters (α, $E[\alpha x]$, $Var[\alpha x]$, γ and β) are channel-wise, the Eq.(13) can be reformulated as:

$$BN^{inf}(x) = (x - \frac{1}{\alpha}E[\alpha x] + f\cdot\beta)\frac{1}{f} \tag{14}$$

which can be rewritten as Eq.(15). Obviously, the term $\frac{sign(\gamma)}{f}$ is always positive.

$$BN^{inf}(x) = ((x - \frac{1}{\alpha}E[\alpha x] + f\cdot\beta)sign(\gamma))\frac{sign(\gamma)}{f} \tag{15}$$

As shown in Fig.(1), the output of Eq.(15) is then binarized by BinaryActivation that discards the magnitude information. Therefore, the $\frac{sign(\gamma)}{f}$ can be removed without affecting the sign of the output, and thus Eq.(16) is acquired. Here, we name it as BatchShift, as there is only accumulation operations to shift the input. The multiplication with $sign(\gamma)$ only flips some bits of the output and costs trivial resource. In this way, the resource consuming floating-point multiplications have been successfully removed, and only a shift by constant $\frac{1}{\alpha}E[\alpha x] + f\cdot\beta$ is required.

$$BS^{inf}(x) = (x - \frac{1}{\alpha}E[\alpha x] + f\cdot\beta)sign(\gamma) \tag{16}$$

However, the constant shift is still of floating-point type. As known, the x is an integer, and the $-\frac{1}{\alpha}E[\alpha x] + f\cdot\beta$ is a floating-point constant. The constant can be decomposed into two terms in advance, i.e. an integer A plus a floating-point decimal B that is small than 1. We discuss the sign of $BS^{inf}(x)$ in two situations:

(1) if $|x+A| \geqslant 1$, the sign of $BS^{inf}(x)$ is determined only by the sign of integer item $x+A$.

(2) if $|x+A| = 0$, the sign of $BS^{inf}(x)$ is determined only by the sign of B.

In this way, the BatchShift, i.e. $BS^{inf}(x)$, is simplified without floating point operations, as illustrated in Eq.(17).

$$BS^{inf}(x) = \begin{cases} (x+A)sign(\gamma), & \text{if } |x+A| \geqslant 1 \\ sign(B)\cdot sign(\gamma), & \text{if } |x+A| = 0 \end{cases} \tag{17}$$

[2]Indeed, only one image is input during the testing phase, which cannot provide the calculation of the E[x] and Var[x]. Therefore, Batchnorm approximate the E[x] and Var[x] during the testing phase by the one of the E[x] and Var[x] calculated during the training phase. Similarly, the scaling factor α can also be approximated by the one calculated during the train phase.

B. FIRST LAYER AND LAST LAYER

As the input and output interfaces of deep neural networks, the first layer and last layer are usually not binarized [25], [28], [31]. Therefore, the first layer becomes the most time-consuming part in binarized networks. As a fully-connected layer, the last layer involves most storage cost when the number of output classes is large. This will not be a great problem since the continuous-valued inputs can be converted into a fixed point numbers [25] at the first layer and thus the computation complexity can be reduced. For the last layer, its complexity is directly related to the number of classification that is usually small in industrial embedded application scenarios. Moreover, it can be binarized as well, providing that a scale layer is added before feeding the output of the binarized last layer into the Softmax function [38]. Therefore, the computation complexity of the first layer and the memory required by the last layer can be greatly reduced, which makes it more meaningful to remove the floating-point operations in the middle layers.

C. IMPLEMENTATION OF THE BITFLOW-NET

The proposed BitFlow-Net employs BatchShift to remove the floating-point operations in the middle layers. In this section, the BitFlow-Net is built based on the AlexNet model. Actually, it can be built based on all the plain deep networks. The details of BitFlow-Net are given in Algorithm 1. During the training phase, the weights and the activations are binarized only in forward propagation, and STE is used to obtain the derivative of *sign* function. The weights of BitFlow-Net is still saved as full-precision in the training phase. The adopted BatchNorm is noted as $BS^{tr}(\tilde{x})$, which is shown in Eq.(18). During the model running phase, the BatchShift illustrated in Eq.(17) is used to replace BatchNorm. Even though only a fixed-point accumulation is required, the BatchShift can provide the same result as the full precision BatchNorm for binarized networks.

$$BS^{tr}(\tilde{x}) = \frac{\tilde{x} - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}\gamma + \beta, \tilde{x} = \alpha x \tag{18}$$

Algorithm 1 BitFlow-Net

```
Input: A minibatch of inputs and targets $(I, T)$,
Output: The largest element in the set
1  {1. Forward propagation during training}
2  for $l \leftarrow 1$ to $L$ do
3      // Only binary convolutional layers are
       //considered here.
4      for $k^{th}$ filter in $l^{th}$ layer do
5          $\alpha \leftarrow \frac{1}{n}\|W_{lk}\|_{l1}$
6          $\widetilde{W}_k \leftarrow \alpha sign(W_{lk})$
7      end
8      $\widetilde{a}_{l-1} \leftarrow Sign(BS^{tr}(a_{l-1}))$
9      $z_l \leftarrow Conv(\widetilde{a}_{l-1}, \widetilde{W}_l)$
10     $a_l \leftarrow ReLU(z_l)$
11 end
12 {2. Forward propagation during model running phase}
13 //Compute $\alpha$ for only one time.
14 for $l \leftarrow 1$ to $L$ do
15     for $k^{th}$ filter in $l^{th}$ layer do
16         $\alpha \leftarrow \frac{1}{n}\|W_{lk}\|_{l1}$
17         $\widetilde{W}_k \leftarrow \alpha sign(W_{lk})$
18     end
19 end
20 for $l \leftarrow 1$ to $L$ do
21     $\widetilde{a}_{l-1} \leftarrow Sign(BS^{inf}(a_{l-1}, \alpha_{l-1}))$ //Fuse $\alpha$ with
22     //BatchShift.
23     $\widetilde{W}_l \leftarrow Sign(W_l)$
24     $z_l \leftarrow Conv(a^b_{l-1}, \widetilde{W}_l)$
25     $a_l \leftarrow ReLU(z_l)$
26 end
```

IV. EXPERIMENTS

In this section, the comparison between the BatchShift and the combination of scaling factor and BatchNorm is performed firstly, so as to validate that they have the same effects during the inference process. The performance of the BitFlow-Net is then tested not only on the complex image dataset ImageNet for multiclass object recognition, but also on a hand images dataset 11K Hands for binary gender recognition [36]. Experiments were carried out on a computer with Intel(R) Core(TM) i5-7500 CPU, 32 GB RAM and GeForce GTX1080Ti GPU. PyTorch framework is adopted and the Adam optimizer with default parameter setting in PyTorch is used to update the weights. The model is trained under the same setting as the XNOR-Net. The order of building blocks is arranged as BN-BinActiv-BinConv-pool, following the instruction in [28].

A. EFFECT OF THE SCALING FACTOR IN THE TRAINING PHASE OF THE BITFLOW-NET

The learning rate greatly affects the sign changing rate, which is important for the stability of training in binarized networks [38]. A small learning rate can provide smooth sign changing rate, leading to a better training accuracy. Similarly, the sign changing rate of each layer across all training epochs is monitored in this study, so as to test the effect of scaling factor during the training phase. It is compared with the one without scaling factor, as shown in Fig. 2. Without scaling factor, the sign changing rate curves of the binarized network fluctuate seriously, and therefore it is difficult to separate the curves of different layers. After introducing the scaling factor, the fluctuation of sign changing rate range is greatly reduced. It can be observed that the curves with scaling factor are relatively smooth and steady around 1. In view of the fact that small learning rate that provides smooth sign changing rate can lead to a better training performance [38], Fig. 2 is an evidence of scaling factor's effect on affecting gradient distribution. It can be also found that the sign changing rate curves of binarized networks with scaling factor have a relatively smaller value at the very beginning of the training.

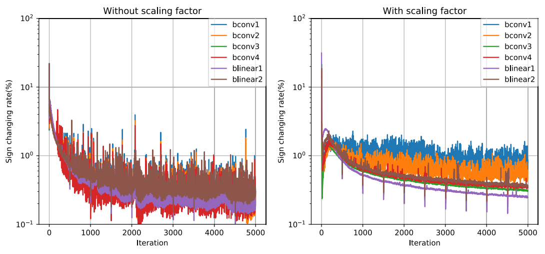

FIGURE 2. Sign changing rate comparison of binarized networks w/ and w/o scaling factor in 5000 iterations. The sign changing rate of binary weights is obviously smoother, when the scaling factor is applied. Left: Without scaling factor; Right: With scaling factor.

This somewhat performs like a warm-up learning rate strategy that has been regarded as an efficient method for accelerating learning [39].

B. VALIDATION OF THE EQUALITY BETWEEN BATCHSHIFT AND THE COMBINATION OF BATCHNORM AND SCALING FACTOR

Once the model has been trained, the BatchShift is used to replace the combination of scaling factor and BatchNorm during the model running phase of the BitFlow-Net. In Fig. 3, we demonstrate an instance of the input and output of both BatchNorm and BatchShift according to Eq. (16). It is worth noting that the result of BatchShift according to Eq. (16) is shown instead of the one of Eq. (17), so as to better illustrate that only the sign instead of the magnitude is important for binarized networks. It can be observed that even though the magnitudes of output are different in BatchNorm and BatchShift, their signs are consistent with each other. It is demonstrated as well that the output is exactly the same after the binarization step. Therefore, it is concluded that the proposed BatchShift can replace the combination of BatchNorm and scaling factor in binarization networks without affecting the output, whilst it can simplify and accelerate the computation.

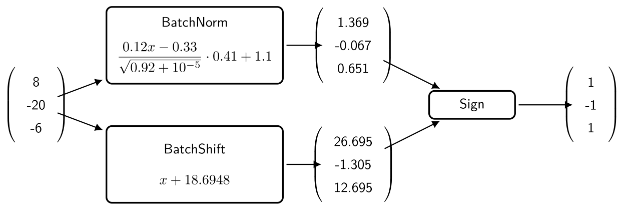

FIGURE 3. Output of the BatchShift. It is different from the BatchNorm but with identical sign.

C. PERFORMANCE OF BITFLOW-NET ON IMAGENET AND 11K HANDS

To verify the accuracy, the proposed BitFlow-Net is first tested and compared with the XNOR-Net on the ImageNet dataset. The epoch, batch size and the learning rate parameters are set to be 50, 128 and 10^{-4} respectively. The results are illustrated in Table 2, from which it can be observed

TABLE 2. Accuracy comparison between XNOR-Net and BitFlow-Net.

Accuracy	ImageNet (Top-1, Top-5)	11K Hands
Alex-Net	(59.30%, 82.80%) [30]	99.80%
XNOR-Net	(44.87%, 69.70%)	98.27%
BitFlow-Net	(44.87%, 69.70%)	98.27%

that even though most of floating-point operations have been removed in binarized network during model running phase, BitFlow-Net performs no accuracy degradation in contrast to XNOR-Net. These results are consistent with expectation, since the introduced BatchShift in BitFlow-Net functions identically to the combination of scaling factor and BatchNorm in XNOR-Net. Through this simplification, the operations are simplified into an integer shift operation in the middle layers, including three floating-point multiplications, two accumulations and a bit-floating-point conversion in each block. In this way, the computation complexity and the memory for storing the feature map of hidden layers are further reduced in the binarization networks. However, such accuracy is still difficult to meet the need of the industrial applications with multiclass classification tasks. But, it may have great potential on the industrial applications with binary classification requirement.

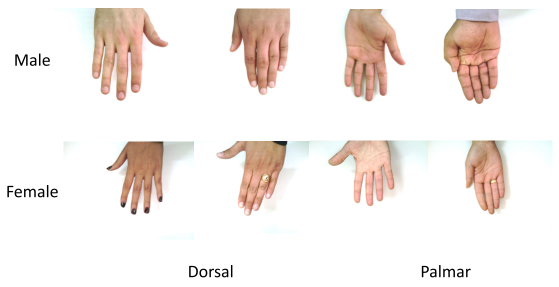

FIGURE 4. Example images of the 11K Hands dataset. Participants were asked to open and hold the fingers of the right and left hands. The hand images were then captured from the back side and the palm side of both hands on the same white background with the same distance from the camera.

Fortunately, many industrial applications just require binary classification, such as online defect detection in production line that is only concerned about whether there are defects [40]. Since we are not involved in any specific industrial applications, we test the proposed BitFlow-Net on a similar gender recognition binary classification problem instead using the public hand images large dataset 11K Hands, so as to demonstrate the potential. The dataset contains 11,076 hand images (1600 × 1200 pixels) of 190 persons, of varying ages from 18 to 75 years old, some samples of which are illustrated in Fig. 4. The gender recognition problem is a challenging task for traditional machine learning classification algorithms [41], since they rely on manual feature extraction and their performance are limited by shallow network structure. Firstly, the full precision deep learning network AlexNet is used to tackle the task as the baseline. Then, the proposed bitFlow-Net is tested on the problem as well,

TABLE 3. Comparison between XNOR-Net and BitFlow-Net. Both memory cost and floating-point computation amount in the running phase are dramatically reduced.

Layer	FMap	KSize	FMapSize	XNOR-Net			BitFlow-Net		
				Size(MB)	Mem.(MB)	MFLOP	Size(MB)	Mem.(MB)	MFLOP
BinConv2	$96 \rightarrow 192$	5	27×27	0.057	2.800	0.699	0.055	0.560	0
BinConv3	$192 \rightarrow 384$	3	13×13	0.085	1.300	0.324	0.079	0.260	0
BinConv4	$384 \rightarrow 256$	3	13×13	0.109	0.865	0.216	0.106	0.173	0
BinConv5	$256 \rightarrow 256$	3	13×13	0.074	0.865	0.216	0.071	0.173	0
BinFC6	$9216 \rightarrow 4096$	-	-	4.563	0.080	0.020	4.502	0.016	0
BinFC7	$4096 \rightarrow 4096$	-	-	2.063	0.080	0.020	2.002	0.016	0
TOTAL	-	-	-	6.951	5.990	1.495	6.815(-2%)	1.198(-80%)	0(-100%)

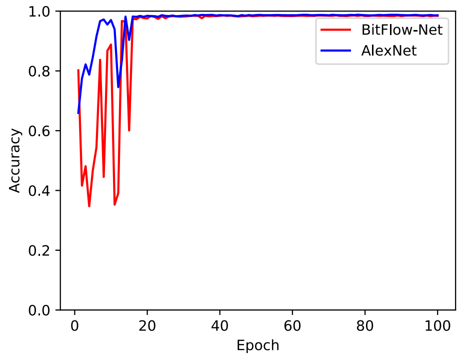

FIGURE 5. Accuracy convergence curves of BitFlow-Net and AlexNet on 11K Hands dataset. After training for 40 epochs, BitFlow-Net model could achieve an accuracy comparable to the AlexNet model.

and its result is compared with the baseline. In this work, all the images in 11K Hands are resized into 224×224. In order to increase the generality of the trained model, the random flipping method is used for the data augmentation of the original dataset. Then, 10-fold cross-validation method is further used for validating the generality performance. In this study, the networks are trained for 100 epochs, while the batch size is set to be 128. In addition, the learning rate is initialized to be 10^{-4} and decays at epoch 5, 15, 35 and 75 by 0.1. The experimental results are shown in Fig. 5 and Table 2. Fig. 5 shows the accuracy curves of BitFlow-Net and its full-precision counterpart during the training on the 11K Hands dataset. It can be seen that the binarized model gradually converges after 40 epochs and its accuracy is very close to the full precision one. Both BitFlow-Net and the full-precision AlexNet perform well on the gender recognition binary classification task. Specifically, the accuracy of BitFlow-Net is just 1.54% lower than the full precision AlexNet, whilst the network is greatly accelerated. Furthermore, the storage and computation resource required in the two binarized networks are analyzed. As shown in Table 3, several parameters of the two network models are compared, including the storage size, required memory and million floating-point operations (MFLOP). It is worth noting that only the resource reduction in the middle layers are demonstrated in Table 3, since this study only focuses on removing the floating point operation in these layers, i.e. the layers of Conv2, Conv3, Conv4, Conv5, FC6 and FC7. Key parameters of these layers are given as well, including the feature map (FMap), kernel size (KSize), and size of feature map (FMapSize). As known, the size of parameters in the middle layers of the AlexNet is 56.8M, which is reduced to be 6.9M by XNOR-Net. In this study, it is further reduced by 2% through removing the floating point parameters. More importantly, the required memory for the feature map is reduced by 80%, since the floating point operations can be fused into one integer operation in the middle layers and consequently the required floating point operations are totally removed. Therefore, the proposed BitFlow-Net method can effectively promote the deployment of binarized networks on resource-constrained embedded hardware.

V. CONCLUSION

Binarized deep neural networks have great potential in resource-limited embedded devices for real-time industrial applications. In this study, based on the AlexNet, we propose an improved binarized convolutional neural network named BitFlow-Net that improves model efficiency by eliminating floating point operations introduced by scaling factor and BatchNorm in the middle layers. Through studying the role of scaling factor and BatchNorm in binarized networks, it is revealed that the scaling factor only takes effect in the back-propagation process, and it is counteracted by the BatchNorm operation in the running phase. Then, in view of the characteristic that only the sign is important instead of the magnitude in the binarization networks, the scaling factor and BatchNorm are fused into a simplified integer operation named BatchShift. Using the BatchShift, floating-point operations in the middle layers of trained models can be completely removed without affecting the model running results. The proposed BitFlow-Net is tested on two large image datasets, i.e. the ImageNet for the complex multiclass classification and the hand images based gender recognition dataset, 11K Hands, for binary classification. Experimental results validate that, in comparison with XNOR-Net, the proposed method can reduce the memory by 80% and remove all floating point operations in the middle layers of BCNNs without affecting the accuracy. Especially, in comparison to the full precision AlexNet, the BitFlow-Net shows competitive performance in the challenging hand images based gender recognition problem. Therefore, it may have great potential in real time embedded devices for many industrial applications especially with binary classification tasks.

REFERENCES

[1] L. Li, K. Ota, and M. Dong, "Deep learning for smart industry: Efficient manufacture inspection system with fog computing," *IEEE Trans. Ind. Informat.*, vol. 14, no. 10, pp. 4665–4673, Oct. 2018.

[2] L. Wen, X. Li, L. Gao, and Y. Zhang, ''A new convolutional neural network-based data-driven fault diagnosis method,'' *IEEE Trans. Ind. Electron.*, vol. 65, no. 7, pp. 5990–5998, Jul. 2018.

[3] B. Ma, Z. Liu, F. Jiang, Y. Yan, J. Yuan, and S. Bu, ''Vehicle detection in aerial images using rotation-invariant cascaded forest,'' *IEEE Access*, vol. 7, pp. 59613–59623, 2019.

[4] J. Huang, X. Zhang, F. Guo, Q. Zhou, H. Liu, B. Li, ''Design of an acoustic target classification system based on small-aperture microphone array,'' *IEEE Trans. Instrum. Meas.*, vol. 64, no. 7, pp. 2035–2043, Jul. 2015.

[5] F. Guo, J. Huang, X. Zhang, Y. Cheng, H. Liu, and B. Li, ''A two-stage detection method for moving targets in the wild based on microphone array,'' *IEEE Sensors J.*, vol. 15, no. 10, pp. 5795–5803, Oct. 2015.

[6] X. Qu, Y. Huang, H. Lu, T. Qiu, D. Guo, T. Agback, V. Orekhov, and Z. Chen, ''Accelerated nuclear magnetic resonance spectroscopy with deep learning,'' *Angewandte Chem. Int.*, to be published.

[7] Z. C. Chen, Y. Chen, L. Wu, S. Cheng, and P. Lin, ''Deep residual network based fault detection and diagnosis of photovoltaic arrays using current-voltage curves and ambient conditions,'' *Energy Convers. Manage.*, vol. 198, Oct. 2019 Art. no. 111793.

[8] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, and F. E. Alsaadi, ''A survey of deep neural network architectures and their applications,'' *Neurocomputing*, vol. 234, pp. 11–26, Apr. 2017.

[9] K. Simonyan and A. Zisserman, ''Very deep convolutional networks for large-scale image recognition,'' 2014, *arXiv:1409.1556*. [Online]. Available: https://arxiv.org/abs/1409.1556

[10] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, ''Densely connected convolutional networks,'' in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 4700–4708.

[11] A. Munawar, P. Vinayavekhin, and G. De Magistris, ''Spatio-temporal anomaly detection for industrial robots through prediction in unsupervised feature space,'' in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Mar. 2017, pp. 1017–1025. doi: 10.1109/WACV.2017.118.

[12] Y.-J. Cha, W. Choi, and O. Büyüköztürk, ''Deep learning-based crack damage detection using convolutional neural networks,'' *Comput.-Aided Civil Infrastruct. Eng.*, vol. 32, no. 5, pp. 361–378, May 2017.

[13] J.-K. Park, B.-K. Kwon, J.-H. Park, and D.-J. Kang, ''Machine learning-based imaging system for surface defect inspection,'' *Int. J. Precision Eng. Manuf.-Green Technol.*, vol. 3, no. 3, pp. 303–310, 2016.

[14] J. Kang, Y.-J. Park, J. Lee, S.-H. Wang, and D.-S. Eom, ''Novel leakage detection by ensemble CNN-SVM and graph-based localization in water distribution systems,'' *IEEE Trans. Ind. Electron*, vol. 65, no. 5, pp. 4279–4289, May 2018.

[15] A. Zhou, A. Yao, Y. Guo, L. Xu, and Y. Chen, ''Incremental network quantization: Towards lossless cnns with low-precision weights,'' 2017, *arXiv:1702.03044*. [Online]. Available: https://arxiv.org/abs/1702.03044

[16] Y. Gong, L. Liu, M. Yang, and L. Bourdev, ''Compressing deep convolutional networks using vector quantization,'' 2014, *arXiv:1412.6115*. [Online]. Available: https://arxiv.org/abs/1412.6115

[17] S. Han, H. Mao, and W. J. Dally, ''Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding,'' 2015, *arXiv:1510.00149*. [Online]. Available: https://arxiv.org/abs/1510.00149

[18] X. Dong, L. Liu, G. Li, P. Zhao, and X. Feng, ''Fast CNN pruning via redundancy-aware training,'' in *Proc. Int. Conf. Artif. Neural Netw.*, vol. 11139, 2018, pp. 3–13. doi: 10.1007/978-3-030-01418-6_1.

[19] X. Dai, H. Yin, and N. K. Jha, ''Grow and prune compact, fast, and accurate LSTMs,'' 2018, *arXiv:1805.11797*. [Online]. Available: https://arxiv.org/abs/1805.11797

[20] J. Zhong, G. Ding, Y. C. Guo, J. Han, and B. Wang, ''Where to prune: Using LSTM to guide end-to-end pruning,'' in *Proc. IJCAI*, 2018, pp. 3205–3211.

[21] M. Shi, F. Qin, Q. Ye, Z. Han, and J. Jiao, ''A scalable convolutional neural network for task-specified scenarios via knowledge distillation,'' in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, Mar. 2017, pp. 2467–2471. doi: 10.1109/ICASSP.2017.7952600.

[22] S. Lin, R. Ji, C. Chen, D. Tao, and J. Luo, ''Holistic CNN compression via low-rank decomposition with knowledge transfer,'' *IEEE Trans. Pattern Anal. Mach. Intell.*, to be published. doi: 10.1109/TPAMI.2018.2873305.

[23] Y. Li, Z. Liu, W. Liu, Y. Jiang, Y. Wang, W. L. Goh, H. Yu, and F. Ren, ''A 34-FPS 698-GOP/s/W binarized deep neural network-based natural scene text interpretation accelerator for mobile edge computing,'' *IEEE Trans. Ind. Electron.*, vol. 66, no. 9, pp. 7407–7416, Sep. 2019. doi: 10.1109/TIE.2018.2875643.

[24] M. Courbariaux, Y. Bengio, and J.-P. David, ''BinaryConnect: Training deep neural networks with binary weights during propagations,'' in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 3123–3131.

[25] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, ''Binarized neural networks,'' in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 4107–4115.

[26] G. Govindu, L. Zhuo, S. Choi, and V. Prasanna, ''Analysis of high-performance floating-point arithmetic on FPGAs,'' in *Proc. 18th Int. Parallel Distrib. Process. Symp.*, Apr. 2004, p. 149. doi: 10.1109/IPDPS.2004.1303135.

[27] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, ''ImageNet: A large-scale hierarchical image database,'' in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255. doi: 10.1109/CVPR.2009.5206848.

[28] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, ''Xnor-net: Imagenet classification using binary convolutional neural networks,'' in *Proc. Eur. Conf. Comput. Vis.*, Springer, 2016, pp. 525–542.

[29] S. Ioffe and C. Szegedy, ''Batch normalization: Accelerating deep network training by reducing internal covariate shift,'' 2015, *arXiv:1502.03167*. [Online]. Available: https://arxiv.org/abs/1502.03167

[30] A. Krizhevsky, I. Sutskever, and G. E. Hinton, ''Magenet classification with deep convolutional neural networks,'' in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[31] S. Zhou, Y. Wu, Z. Ni, X. Zhou, H. Wen, and Y. Zou, ''Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients,'' 2016, *arXiv:1606.06160*. [Online]. Available: https://arxiv.org/abs/1606.06160

[32] S. Wu, G. Li, F. Chen, and L. Shi, ''Training and inference with integers in deep neural networks,'' 2018, *arXiv:1802.04680*. [Online]. Available: https://arxiv.org/abs/1802.04680

[33] M. D. McDonnell, ''Training wide residual networks for deployment using a single bit for each weight,'' 2018, *arXiv:1802.08530*. [Online]. Available: https://arxiv.org/abs/1802.08530

[34] K. He, X. Zhang, S. Ren, and J. Sun, ''Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification,'' in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1026–1034.

[35] Z. Liu, B. Wu, W. Luo, X. Yang, W. Liu, and K.-T. Cheng, ''Bi-Real Net: Enhancing the performance of 1-bit CNNs with improved representational capability and advanced training algorithm,'' in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 722–737.

[36] M. Afifi, ''11K Hands: Gender recognition and biometric identification using a large dataset of hand images,'' *Multimedia Tools Appl.*, vol. 78, pp. 20835–20854, Aug. 2019. doi: 10.1007/s11042-019-7424-8.

[37] S. Khan and T. Yairi, ''A review on the application of deep learning in system health management,'' *Mech. Syst. Signal Process.*, vol. 107, pp. 241–265, Jul. 2018.

[38] W. Tang, G. Hua, and L. Wang, ''How to train a compact binary neural network with high accuracy?'' in *Proc. AAAI*, 2017, pp. 2625–2631.

[39] I. Loshchilov and F. Hutter, ''Sgdr: Stochastic gradient descent with warm restarts,'' 2016, *arXiv:1608.03983*. [Online]. Available: https://arxiv.org/abs/1608.03983

[40] P. Sassi, P. Tripicchio, and C. A. Avizzano, ''A smart monitoring system for automatic welding defect detection,'' *IEEE Trans. Ind. Electron.*, vol. 66, no. 12, pp. 9641–9650, Dec. 2019. doi: 10.1109/TIE.2019.2896165.

[41] J. Mansanet, A. Albiol, and R. Paredes, ''Local deep neural networks for gender recognition,'' *Pattern Recognit. Lett.*, vol. 70, pp. 80–86, Jan. 2016.

LIJUN WU received the B.S. and M.S. degrees in communication engineering from Xiamen University, China, in 2005 and 2009, respectively, and the Ph.D. degree in civil engineering from the University of Pavia, Italy, in 2013.

From 2013 to 2014, she was a Research Assistant with the Hong Kong Polytechnic University. Since 2014, she has been an Associated Professor with the College of Physics and Information Engineering, Fuzhou University, China. Her research interests include deep learning, machine learning, computer vision, image processing, and their application in industrial sensing, detection, diagnosis, and prognosis.

PEIQING JIANG received the B.S. and M.S. degrees in electronic communication engineering from College of Physics and Information Engineering, Fuzhou University, China, in 2016 and 2019, respectively.

ZHICONG CHEN received the B.S. and M.S. degrees in electronic engineering from Xiamen University, China, in 2005 and 2008, respectively, and the Ph.D. degree in civil engineering from the University of Pavia, Italy, in 2011, where he was a Postdoctoral Researcher, from 2011 to 2013.

Since 2014, he has been an Associate Professor with the College of Physics and Information Engineering, Fuzhou University, China. His current research interests include wireless sensors networks, signal processing, machine learning, and their application in industrial detection, diagnosis, and prognosis.

XU LIN, photograph and biography not available at the time of publication.

YUNFENG LAI, photograph and biography not available at the time of publication.

PEIJIE LIN, photograph and biography not available at the time of publication.

SHUYING CHENG, photograph and biography not available at the time of publication.