Binary Forward-Only Algorithms

Baichuan Huang, *Student Member, IEEE,* and Amir Aminifar, *Senior Member, IEEE*

***Abstract*—Today, the overwhelming majority of Internet of Things (IoT) and mobile edge devices have extreme resource limitations, e.g., in terms of computing, memory, and energy. As a result, training Deep Neural Networks (DNNs) using the complex Backpropagation (BP) algorithm on such edge devices presents a major challenge. Forward-only algorithms have emerged as more computation- and memory-efficient alternatives without the requirement for backward passes. In this paper, we investigate binarizing state-of-the-art forward-only algorithms, which are applied to the forward passes of PEPITA, FF, and CwComp. We evaluate these forward-only algorithms with binarization and demonstrate that weight-only binarization may be up to $\sim 31\times$ more efficient in terms of memory, with minor degradation in classification performance. Furthermore, we investigate and compare BP and forward-only algorithms in terms of binarization, finding that PEPITA and FF are more vulnerable to binary activations. The code is available at https://github.com/whubaichuan/BinaryFO.**

***Index Terms*—TinyML, IoT, forward-only, binary neural network, memory efficient, computational efficient.**

I. Introduction

Today, trillions of Internet of Things (IoT) devices are deployed and distributed for intelligent perception and decision-making at the very edge of the cloud [1]. Nevertheless, state-of-the-art Deep Neural Networks (DNNs) based on Backpropagation (BP) [2] consume massive amounts of computation and memory, posing a significant challenge for training and deployment on devices with limited storage, battery power, and compute capabilities. To address this challenge, the evolution of deep learning tailored to resource-constrained devices is rapidly advancing [3].

Forward-only algorithms [4]–[6] have emerged as a powerful supplement to resource-efficient machine learning algorithms. These forward-only algorithms offer computational and memory efficiency by avoiding backpropagation of errors across the entire networks. For instance, PEPITA [4] and Forward-Forward (FF) algorithms [5] enable only forward passes with layer-wise updating, greatly reducing computation and memory overheads in the forward pass. As shown in Fig. 1 (a), the memory overheads of PEPITA [4] and FF [5] in the forward pass are 15 Megabyte (MB) and 16 MB, respectively, while the memory overhead of BP is 54 MB. Despite these advantages of forward-only algorithms, the memory requirements for PEPITA [4] and FF [5] in forward passes still exceed the capacity of typical Microcontrollers (MCUs) used in the state-of-the-art IoT and wearable devices, e.g., e-Glass [7] with STM32L476 ultra-low-power ARM Cortex-M4, which has only 1 MB of Flash Memory (Flash).

This research has been partially supported by the Swedish Wallenberg AI, Autonomous Systems and Software Program (WASP), the Swedish Research Council (VR), Swedish Foundation for Strategic Research (SSF), and the ELLIIT Strategic Research Environment. The computations were enabled by resources provided by the National Academic Infrastructure for Supercomputing in Sweden (NAISS) at C3SE partially funded by the Swedish Research Council (VR) through grant agreement no. 2022-06725.

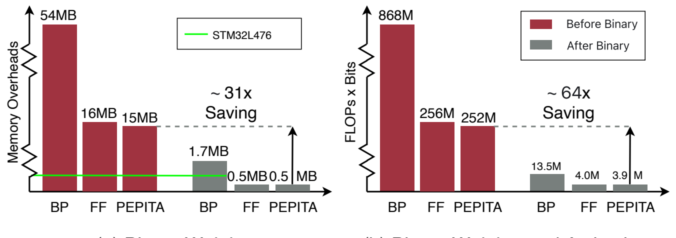


Fig. 1: The memory and computation saving for BP [2], PEPITA [4], and FF [5].

Although several methods have been proposed in the evolution of deep learning, such as pruning, quantization, tensor decomposition, distillation, compact architecture design, and neural architecture search, these methods mainly focus on reducing the memory and computation during inference [8]. In addition, efficient training techniques, such as mixed-precision training, activation data recomputation, and low-rank gradient descent, are proposed to reduce memory and computation during training. However, these techniques are predominantly designed for BP-based scheme and have not been explored and verified for forward-only algorithms [1], [3]. The training scheme for BP and forward-only algorithms differ significantly. The gradients of weights in BP rely on the chain rule, while the gradients of weights in PEPITA and FF are based on hidden activations.

In this paper, we investigate binarizing state-of-the-art forward-only algorithms, such as PEPITA and FF. Considering the forward passes of these forward-only algorithms and Fully Connected Network (FC)-based networks, the majority of memory overheads stem from the weights. To address this, we binarize the weights during the forward passes, effectively reducing memory overheads. This binarization replaces many multiply-accumulate operations with additions and subtractions, thereby significantly decreasing computation overheads. Moreover, much of the computation involves multiplying full-precision weights by full-precision activations, with multipliers being the most computation-intensive components of digital neural network implementations. Therefore, to further alleviate computational overheads in the forward passes, we also binarize the activations, substantially reducing computation through 1-bit XNOR with pop-count operations. Our main contributions are summarized below:

- We investigate binarization in state-of-the-art forward-only algorithms, applied to the forward passes of these forward-only algorithms, including PEPITA and FF.
- We show that PEPITA and FF with weight-only bina-

rization reduce the memory overheads by $\sim$31$\times$ times, as shown in Fig. 1 (a), and computational overheads by $\sim$2$\times$ times, with minor degradation in classification performance. We also show that PEPITA and FF with both binary weights and binary activations reduce the computational overheads by $\sim$64$\times$ times, as shown in Fig. 1 (b), and memory overheads by $\sim$31$\times$ times with significantly worse classification performance. Our extensive evaluation and analysis of binary activations indicate that PEPITA and FF exhibit more degradation in classification performance compared to BP.

II. Binary Forward-Only Algorithms

Let us consider an L-layer DNN with a binary-weight forward pass. The input $\boldsymbol{x}$ and the target $\boldsymbol{y}$ are utilized for training the DNN. The activations of the hidden layer l of the DNN are denoted as $\boldsymbol{h}_l$, where $\boldsymbol{h}_0 = \boldsymbol{x}$. Let us define the function composition $f_l : \boldsymbol{h}_{l-1} \to \boldsymbol{h}_l$ for the hidden layer l. The output activations $\boldsymbol{h}_l$ based on the input activation $\boldsymbol{h}_{l-1}$, are calculated as follows:

$$\begin{aligned} \boldsymbol{h}_l &= f_l(\boldsymbol{h}_{l-1}) \qquad (1)\\ &= \sigma_l(\boldsymbol{W}_l^b \boldsymbol{h}_{l-1} + \boldsymbol{b}_l), \end{aligned}$$

where $\boldsymbol{W}_l^b$ is the binary weights between hidden layers $l-1$ and l. σ_l is the activation function for the hidden layer l. All binarization is based on the Sign function with the average of absolute parameter values as a scaling factor [9], that is, $\boldsymbol{W}_l^b = \frac{1}{n}\sum_{i=1}^{n} |\boldsymbol{W}_{l,i}| \cdot \text{Sign}(\boldsymbol{W}_l)$.

A. PEPITA with Binary Weights

PEPITA with binary weights is based on two binary forward passes, namely the standard (denoted by superscript s) forward pass, i.e., Equation (2), and the modulated (denoted by superscript m) forward pass, i.e., Equation (3), as shown in Fig. 2 (b). For the hidden layer l, the gradient of the binary weights is denoted as Equation (4):

$$\begin{aligned} \boldsymbol{h}_l^{(s)} &= \sigma_l(\boldsymbol{W}_l^b \boldsymbol{h}_{l-1}^{(s)} + \boldsymbol{b}_l), && (2)\\ \boldsymbol{h}_l^{(m)} &= \sigma_l(\boldsymbol{W}_l^b \boldsymbol{h}_{l-1}^{(m)} + \boldsymbol{b}_l), && (3)\\ \delta \boldsymbol{W}_l^b &= (\boldsymbol{h}_l^{(s)} - \boldsymbol{h}_l^{(m)}) \otimes (\boldsymbol{h}_{l-1}^{(m)})^T, && (4) \end{aligned}$$

where $\boldsymbol{h}_0^{(s)} = x^{(s)}$ and $\boldsymbol{h}_0^{(m)} = x^{(m)}$. In the second forward pass, $\boldsymbol{x}^{(m)} = \boldsymbol{x}^{(s)} + \boldsymbol{F}(\boldsymbol{h}_L^{(s)} - \boldsymbol{y})$ and $\boldsymbol{F}$ is a fixed random matrix. The operator $\otimes$ is the Kronecker Product. Finally, to streamline the notation for the gradient of the binary weights, i.e., $\delta(\boldsymbol{W}_l^b)$, we simply denote it as $\delta \boldsymbol{W}_l^b$.

B. FF with Binary Weights

FF with binary weights is also based on two binary forward passes, namely the positive (denoted by superscript p) forward pass, i.e., Equation (5), and the negative (denoted by superscript n) forward pass, i.e, Equation (6), as shown in Fig. 2 (c), where the positive forward pass consists of the samples with the correct labels and the negative forward pass consists of the samples with the incorrect labels [5]. For the hidden layer l, the gradient of the binary weights is denoted as Equation (7):

$$\begin{aligned} \boldsymbol{h}_l^{(p)} &= \sigma_l(\boldsymbol{W}_l^b \boldsymbol{h}_{l-1}^{(p)} + \boldsymbol{b}_l), && (5)\\ \boldsymbol{h}_l^{(n)} &= \sigma_l(\boldsymbol{W}_l^b \boldsymbol{h}_{l-1}^{(n)} + \boldsymbol{b}_l), && (6)\\ g_l^{(p)} &= (\boldsymbol{h}_l^{(p)})^T \boldsymbol{h}_l^{(p)}, &&\\ g_l^{(n)} &= (\boldsymbol{h}_l^{(n)})^T \boldsymbol{h}_l^{(n)}, &&\\ \delta \boldsymbol{W}_l^b &= \boldsymbol{h}_l^{(p)} \cdot \delta g_l^{(p)} \cdot \sigma_l' \otimes (\boldsymbol{h}_{l-1}^{(p)})^T &&\\ &\quad + \boldsymbol{h}_l^{(n)} \cdot \delta g_l^{(n)} \cdot \sigma_l' \otimes (\boldsymbol{h}_{l-1}^{(n)})^T, && (7) \end{aligned}$$

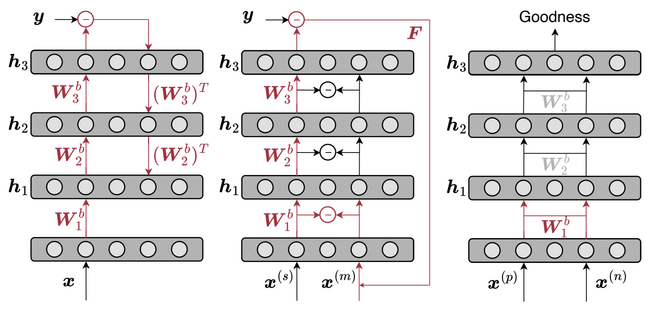
The g_l is the Goodness in FF [5]. The gradient of the Goodness g_l for hidden layer l is denoted as $\delta g_l^{(p)} = \text{Sigmoid}(g_l^{(p)} - \theta) - 1$ for samples in positive forward pass and $\delta g_l^{(n)} = \text{Sigmoid}(g_l^{(n)} - \theta) - 0$ for samples in negative forward pass, where θ is the threshold.

In PEPITA with binary weights and FF with binary weights, we further modify the gradient of the binary weights, as shown below:

$$\delta \boldsymbol{W}_l = \delta \boldsymbol{W}_l^b \odot \boldsymbol{1}_{|\boldsymbol{W}_l| \le 1},$$

where $\odot$ is the element-wise product. This operation preserves the gradient information and cancels the gradient when the weights are too large to avoid degradation in classification performance [10].

In the first forward pass, binary weights are utilized in the calculation from the bottom to top layers, reducing memory by approximately 32$\times$ times and computation by approximately 2$\times$ times, as illustrated in Table I. PEPITA [4] needs to store all the weights and activations (shown with "All" in Table I) in the first forward pass until the modulated input arrives, as shown in Fig. 2 (b); whereas FF [5] only requires to store the maximum memory overheads among every two



(a) BP+$\boldsymbol{W}^b$ (b) PEPITA+$\boldsymbol{W}^b$ (c) FF+$\boldsymbol{W}^b$

Fig. 2: An overview of BP+$\boldsymbol{W}^b$, PEPITA+$\boldsymbol{W}^b$, and FF+$\boldsymbol{W}^b$. The paths for updating weights are illustrated by red arrows.

TABLE I: Analysis of memory and computation overheads (F: FLOPs, B: Bits).

Algorithms	PEPITA	PEPITA+$\boldsymbol{W}^b$	FF	FF+$\boldsymbol{W}^b$
Memory	All	All/32	Max	Max/32
Computation	2F×32B	1F×32B	2F×32B	1F×32B

consecutive layers for weights and activations (shown with "Max" in Table I), as shown in Fig. 2 (c). In the second forward pass, the activations are used to calculate gradients of binary weights locally. After the two forward passes, the gradients and the full-precision weights are exploited during the update step because maintaining precision weights during the updates is essential for the Stochastic Gradient Descent (SGD) algorithm to function effectively [11]. Additionally, the updated weights are clipped within the range $[-1, 1]$ to prevent full-precision weights from growing excessively large without affecting binary weights [10]. Once the update is completed, there is no need to retain the full-precision weights.

III. Experimental & Results

A. Experimental Setup

1) Datasets: To evaluate binary forward-only algorithms, we consider two standard datasets: the MNIST dataset[1] of handwritten digits and the CIFAR-10 dataset[2] of object recognition. Moreover, we extend our evaluation to a real-world application on wearable IoT devices with stringent computational and memory constraints, specifically, cardiac arrhythmia classification based on the MIT-BIH Arrhythmia Electrocardiogram (ECG) Dataset[3].

2) Implementation and Evaluation: Our analysis is performed in PyTorch[4]. For the DNNs, we exploit the default architectures in FF [5] and PEPITA [4], i.e., 4 hidden layers and 2000 neurons for each layer in FF [5], 3 hidden layers and 1024 neurons for each layer in PEPITA [4]. For the memory overheads in the forward pass, we primarily consider layer activations and DNN parameters (weights and biases), excluding other scratch buffers to ensure generalizability, as their impact depends on low-level implementation details. For measuring computation, we consider the metric #Floating Point Operations (FLOPs)$\times$Bits in the inference phase. The default full-precision of tensors in Pytorch is Float32, i.e., 32 bits.

Our experiments utilize balanced datasets, with classification performance assessed using error, i.e., the total number of incorrectly classified inputs divided by the total number of inputs. All algorithms are trained on a server equipped with 2×16-core Intel (R) Xeon (R) Gold 6226R (Skylake) Central Processing Units (CPUs) and 1 NVIDIA Tesla T4 Graphics Processing Card (GPU).

B. PEPITA with Binary Weights and FF with Binary Weights

In this section, we extensively evaluate PEPITA+$\boldsymbol{W}^b$ (PEPITA [4] with weight-only binarization) and FF+$\boldsymbol{W}^b$ (FF [5] with weight-only binarization), in terms of error versus memory and computational overheads. Specifically, binary weights between hidden layer $l-1$ and l are denoted as $\boldsymbol{W}_l^b$. The errors for variants of BP [2], PEPITA [4], and FF [5] are presented in Table II, where a lower error means higher classification accuracy. For the MNIST, CIFAR-10, and MIT-BIH datasets, the error of PEPITA+$\boldsymbol{W}^b$ is close to the error of PEPITA [4]. For instance, on CIFAR-10, the error of PEPITA+$\boldsymbol{W}^b$ is 56.40%, while the original PEPITA achieves 54.43% [4]. The error of FF+$\boldsymbol{W}^b$ also exhibits a close error to FF [5] across these three datasets. In this scenario, the computational overheads of PEPITA+$\boldsymbol{W}^b$ and FF+$\boldsymbol{W}^b$ are reduced by $\sim$2$\times$ times, and the memory overheads of PEPITA+$\boldsymbol{W}^b$ and FF+$\boldsymbol{W}^b$ are reduced by $\sim$31$\times$ times compared to the original forward-only algorithms, as shown in Fig. 3.

TABLE II: Errors (%) for variants of BP, PEPITA and FF.

Algorithms	MNIST	CIFAR-10	MIT-BIH
BP [2]	1.32	42.56	7.92
BP+$\boldsymbol{W}^b$	1.48	44.77	8.37
BP+$\boldsymbol{W}_l^b$+$\boldsymbol{h}_{l-1}^b$	2.81	50.06	10.88
BP+$\boldsymbol{W}_l^b$+$\boldsymbol{h}_l^b$	3.09	53.14	14.20
BP+Binary Gradient	62.28	79.95	68.36
PEPITA [4]	2.51	54.43	13.33
PEPITA+$\boldsymbol{W}^b$	**3.76**	**56.40**	**18.32**
PEPITA+$\boldsymbol{W}_l^b$+$\boldsymbol{h}_{l-1}^b$	20.68	67.21	33.28
PEPITA+$\boldsymbol{W}_l^b$+$\boldsymbol{h}_l^b$	89.68	90.00	79.77
FF [5]	1.53	46.53	10.57
FF+$\boldsymbol{W}^b$	**3.14**	**53.44**	**11.66**
FF+$\boldsymbol{W}_l^b$+$\boldsymbol{h}_{l-1}^b$	6.60	55.08	21.89
FF+$\boldsymbol{W}_l^b$+$\boldsymbol{h}_l^b$	90.42	90.00	71.42

BP+$\boldsymbol{W}^b$ (BP with weight-only binarization) shows a considerably smaller error gap with BP across these datasets. Additionally, BP+$\boldsymbol{W}^b$ outperforms PEPITA+$\boldsymbol{W}^b$ and FF+$\boldsymbol{W}^b$ in terms of classification performance. For instance, the error of BP+$\boldsymbol{W}^b$ on MNIST is 1.48%, while the error of PEPITA+$\boldsymbol{W}^b$ on MNIST is 3.76% and the error of FF+$\boldsymbol{W}^b$ on MNIST is 3.14%.

C. Ablation Study

In the ablation study, we continue to binarize the activations for PEPITA [4] and FF [5]. Binary activations can be further categorized into binary input activations $\boldsymbol{h}_{l-1}^b$ in Equation (1), where $\boldsymbol{h}_{l-1}^b = \frac{1}{n}\sum_{i=1}^n |\boldsymbol{h}_{l-1,i}| \cdot \text{Sign}(\boldsymbol{h}_{l-1})$, and binary output activations $\boldsymbol{h}_l^b$ in Equation (1), where $\boldsymbol{h}_l^b = \frac{1}{n}\sum_{i=1}^n |\boldsymbol{h}_{l,i}| \cdot \text{Sign}(\boldsymbol{h}_l)$.

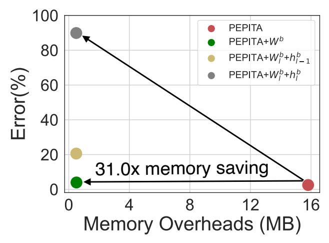
For PEPITA+$\boldsymbol{W}^b$, the gradients for input and output activations, based on Equation (4), are denoted as follows:

$$\delta\boldsymbol{W}_l^b(\text{input}) = (\boldsymbol{h}_l^{(s)} - \boldsymbol{h}_l^{(m)}) \otimes (\boldsymbol{h}_{l-1}^{(m),b})^T, \tag{8}$$

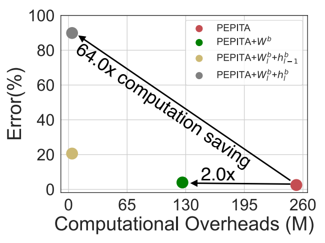
$$\delta\boldsymbol{W}_l^b(\text{output}) = (\boldsymbol{h}_l^{(s),b} - \boldsymbol{h}_l^{(m),b}) \otimes (\boldsymbol{h}_{l-1}^{(m)})^T, \tag{9}$$

where PEPITA+$\boldsymbol{W}^b$ with binary input activations is denoted as PEPITA+$\boldsymbol{W}_l^b$+$\boldsymbol{h}_{l-1}^b$ (Equation (8)); PEPITA+$\boldsymbol{W}^b$ with binary output activations is denoted as PEPITA+$\boldsymbol{W}_l^b$+$\boldsymbol{h}_l^b$ (Equation (9)).

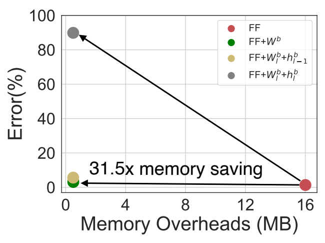
[1]http://yann.lecun.com/exdb/mnist/
[2]https://www.cs.toronto.edu/~kriz/cifar.html
[3]https://physionet.org/content/mitdb/1.0.0/
[4]https://github.com/whubaichuan/BinaryFO



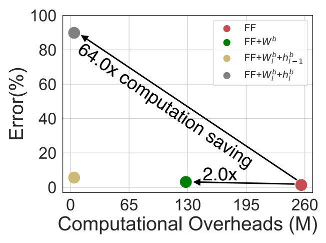
(a) MNIST-PEPITA-Memo



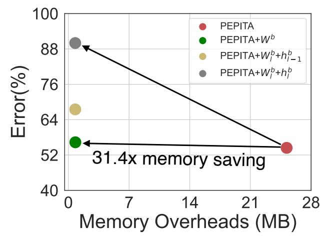
(b) MNIST-PEPITA-Comp



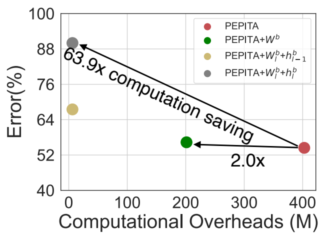
(c) MNIST-FF-Memo



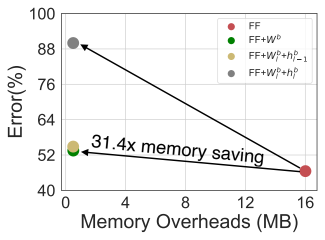
(d) MNIST-FF-Comp



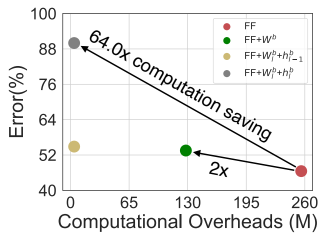
(e) CIFAR-10-PEPITA-Memo



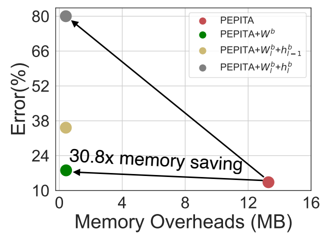
(f) CIFAR-10-PEPITA-Comp



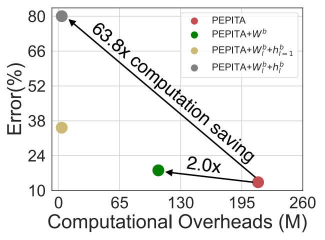
(g) CIFAR-10-FF-Memo



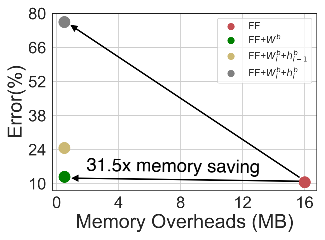
(h) CIFAR-10-FF-Comp



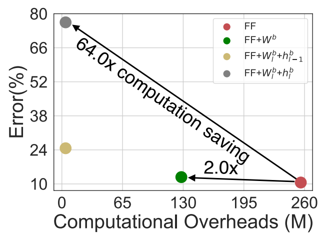
(i) MIT-BIH-PEPITA-Memo



(j) MIT-BIH-PEPITA-Comp



(k) MIT-BIH-FF-Memo



(l) MIT-BIH-FF-Comp

Fig. 3: Error (%) versus Memo (memory overheads) and Comp (computational overheads) for variants of PEPITA and FF.

For FF+$\boldsymbol{W}^b$, the gradients for input and output activations, based on Equation (7), are denoted as follows:

$$\delta \boldsymbol{W}_l^b(\text{input}) = \boldsymbol{h}_l^{(p)} \cdot \delta g_l^{(p)} \cdot \sigma_l' \otimes (\boldsymbol{h}_{l-1}^{(p),b})^T + \boldsymbol{h}_l^{(n)} \cdot \delta g_l^{(n)} \cdot \sigma_l' \otimes (\boldsymbol{h}_{l-1}^{(n),b})^T, \quad (10)$$

$$\delta \boldsymbol{W}_l^b(\text{output}) = \boldsymbol{h}_l^{(p),b} \cdot \delta g_l^{(p)} \cdot \sigma_l' \otimes (\boldsymbol{h}_{l-1}^{(p)})^T + \boldsymbol{h}_l^{(n),b} \cdot \delta g_l^{(n)} \cdot \sigma_l' \otimes (\boldsymbol{h}_{l-1}^{(n)})^T, \quad (11)$$

where FF+$\boldsymbol{W}^b$ with binary input activations is denoted as FF+$\boldsymbol{W}_l^b$+$\boldsymbol{h}_{l-1}^b$ (Equation (10)); FF+$\boldsymbol{W}^b$ with binary output activations is denoted as FF+$\boldsymbol{W}_l^b$+$\boldsymbol{h}_l^b$ (Equation (11)).

1) Binary Input Activations: As shown in Table II, the error of PEPITA+$\boldsymbol{W}_l^b$+$\boldsymbol{h}_{l-1}^b$ increases when compared with PEPITA [4]. For example, on CIFAR-10, the error of PEPITA+$\boldsymbol{W}_l^b$+$\boldsymbol{h}_{l-1}^b$ increases from 54.43% to 67.21% compared to PEPITA [4]. The error of FF+$\boldsymbol{W}_l^b$+$\boldsymbol{h}_{l-1}^b$ also increases when compared with FF [5] across these three datasets. Despite the degradation in classification performance of forward-only algorithms with binary weights and binary input activations, the computational overheads are reduced by $\sim$64$\times$ times, and the memory overheads are reduced by $\sim$31$\times$ times compared to the original forward-only algorithms, as shown in Fig. 3.

We also evaluate BP with binary weights and binary input activations (BP+$\boldsymbol{W}_l^b$+$\boldsymbol{h}_{l-1}^b$), which significantly outperforms PEPITA+$\boldsymbol{W}_l^b$+$\boldsymbol{h}_{l-1}^b$ and FF+$\boldsymbol{W}_l^b$+$\boldsymbol{h}_{l-1}^b$ in terms of classification performance. For instance, the error of BP+$\boldsymbol{W}_l^b$+$\boldsymbol{h}_{l-1}^b$ on MNIST is 2.81%, while the error of PEPITA+$\boldsymbol{W}_l^b$+$\boldsymbol{h}_{l-1}^b$ on MNIST is 20.68% and the error of FF+$\boldsymbol{W}_l^b$+$\boldsymbol{h}_{l-1}^b$ on MNIST is 6.60%. The results presented in Table II also demonstrate that BP+$\boldsymbol{W}_l^b$+$\boldsymbol{h}_{l-1}^b$ achieves a close error compared with BP for MNIST, CIFAR-10, and MIT-BIH datasets. However, the comparable error with binary input activations observed in BP [2] is not reflected in PEPITA [4] and FF [5]. We will discuss this in Section III-D.

2) Binary Output Activations: As shown in Table II, the error of PEPITA+$\boldsymbol{W}_l^b$ + $\boldsymbol{h}_l^b$ increases significantly when compared with PEPITA [4]. For example, on CIFAR-10, the error of PEPITA with binary weights and output activations increases from 54.43% to 90.00% compared to PEPITA [4]. The error of FF+$\boldsymbol{W}_l^b$ + $\boldsymbol{h}_l^b$ also increases significantly when compared with FF [5] across these three datasets. In this case, the computational overheads are reduced by $\sim$64$\times$ times, and the memory overheads are reduced by $\sim$31$\times$ times compared to the original forward-only algorithms, as shown in Fig. 3.

We also evaluate BP with binary weights and binary output activations (BP+$\boldsymbol{W}_l^b$ + $\boldsymbol{h}_l^b$), which also significantly outperforms PEPITA+$\boldsymbol{W}_l^b$+$\boldsymbol{h}_l^b$ and FF+$\boldsymbol{W}_l^b$+$\boldsymbol{h}_l^b$ in terms of classification performance. For instance, the error of BP+$\boldsymbol{W}_l^b$+$\boldsymbol{h}_l^b$ on MNIST is 3.09%, while the error of PEPITA+$\boldsymbol{W}_l^b$+$\boldsymbol{h}_l^b$ on

MNIST is 89.68% and the error of FF+$\boldsymbol{W}_l^b$+$\boldsymbol{h}_l^b$ on MNIST is 90.42%. The error between BP and BP+$\boldsymbol{W}_l^b + \boldsymbol{h}_l^b$ is only slightly worse, a trend not reflected in PEPITA [4] and FF [5]. We will discuss this in Section III-D.

D. Discussion

The error gap between BP+$\boldsymbol{W}^b$ and PEPITA+$\boldsymbol{W}^b$ as well as FF+$\boldsymbol{W}^b$, when binarizing the activations, warrants further investigation and discussion.

In BP [2], the gradient $\delta\boldsymbol{W}_l$ based on the chain rule is denoted as follows (where σ_l is the ReLU activation function, and its derivative σ_l' is 1 when the input is positive and 0 otherwise):

$$\delta\boldsymbol{W}_l = \left[(\boldsymbol{W}_{l+1}^T(\boldsymbol{W}_{l+2}^T...(\boldsymbol{W}_L^T\delta\boldsymbol{h}_L \odot \sigma_L')... \odot \sigma_{l+2}') \odot \sigma_{l+1}')\right] \odot \sigma_l' \otimes \boldsymbol{h}_{l-1}^T. \quad (12)$$

In BP+$\boldsymbol{W}^b$ with binary activations, to alleviate significant information loss, the first and the last layers are not binarized to avoid performance degradation [12]. Consequently, $\delta\boldsymbol{h}_L$ is kept in full precision, i.e., Float32. During its backward pass, the binary weights and binary activations construct the linear combination of $\delta\boldsymbol{h}_L$ by -1 and 1 based on Equation (12). Therefore, gradients $\delta\boldsymbol{W}_l^b$ can have the high precision of Float32 despite the binarization of weights and activations in the hidden layers. The gradient $\delta\boldsymbol{W}_l^b$ in BP+$\boldsymbol{W}^b$ with binary activations based on Equation (12) when all the σ_l' values are 1, is simplified as follows:

$$\delta\boldsymbol{W}_l^b \propto \left[(\boldsymbol{W}_{l+1}^b)^T(\boldsymbol{W}_{l+2}^b)^T...(\boldsymbol{W}_L^b)^T\delta\boldsymbol{h}_L\right] \otimes (\boldsymbol{h}_{l-1}^b)^T.$$

The high-precision of Float32 gradient $\delta\boldsymbol{W}_l^b$ preserves sufficient amounts of information, thereby preventing training failure. In contrast, Table II demonstrates that BP with binary gradient achieves a considerable degradation in classification performance compared to the original BP.

However, in PEPITA [4] and FF [5], the gradient is directly derived from the activations, according to Equation (4) and (7). There are two cases of binary activations: binary input activations ($\boldsymbol{h}_{l-1}^b$) or binary output activations ($\boldsymbol{h}_l^b$). For PEPITA+$\boldsymbol{W}^b$ with binary input activations (i.e., PEPITA+$\boldsymbol{W}_l^b$+$\boldsymbol{h}_{l-1}^b$ in Table II), according to Equation (8), the gradient $\delta\boldsymbol{W}_l^b$ is based on $\boldsymbol{h}_l^{(s)}$, $\boldsymbol{h}_l^{(m)}$, and $\boldsymbol{h}_{l-1}^{(m),b}$, where $\boldsymbol{h}_l^{(s)}$ and $\boldsymbol{h}_l^{(m)}$ are denoted as follows:

$$\boldsymbol{h}_l^{(s)} = \sigma_l(\boldsymbol{W}_l^b\boldsymbol{h}_{l-1}^{(s),b} + \boldsymbol{b}_l), \quad (13)$$

$$\boldsymbol{h}_l^{(m)} = \sigma_l(\boldsymbol{W}_l^b\boldsymbol{h}_{l-1}^{(m),b} + \boldsymbol{b}_l), \quad (14)$$

where $\boldsymbol{h}_l^{(s)}$ and $\boldsymbol{h}_l^{(m)}$ are mainly involved in binary matrix multiplication between $\boldsymbol{W}_l^b \in \{-1,1\}^{D_l \times D_{l-1}}$ and $\boldsymbol{h}_{l-1}^b \in \{-1,1\}^{D_{l-1}\times 1}$ (D_l is the size of output of layer l and D_{l-1} is the size of input to layer l). The binary matrix multiplication results in the value range $[-D_{l-1}, D_{l-1}]$ for each element in $\boldsymbol{h}_l^{(s)}$ and $\boldsymbol{h}_l^{(m)}$, with a range significantly smaller than the range of Float32. Moreover, $\boldsymbol{h}_l^{(s)} - \boldsymbol{h}_l^{(m)}$ obtains a range $[-2\cdot D_{l-1}, 2\cdot D_{l-1}]$. For $\delta\boldsymbol{W}_l^b$, $(\boldsymbol{h}_l^{(s)} - \boldsymbol{h}_l^{(m)}) \otimes (\boldsymbol{h}_{l-1}^{(m),b})^T$ obtains a range $[-2\cdot D_{l-1}, 2\cdot D_{l-1}]$ due to Kronecker product, which has a range significantly smaller than the range of Float32. As shown in Table II, PEPITA+$\boldsymbol{W}_l^b$+$\boldsymbol{h}_{l-1}^b$ has a significantly worse classification performance compared to PEPITA [4].

For PEPITA+$\boldsymbol{W}^b$ with binary output activations (i.e., PEPITA+$\boldsymbol{W}_l^b$+$\boldsymbol{h}_l^b$ in Table II), according to Equation (9), $(\boldsymbol{h}_l^{(s),b} - \boldsymbol{h}_l^{(m),b}) \in \{-2,0,2\}$. At the same time, each binary output activation is the input activation to the next layer, for instance, the output activation $\boldsymbol{h}_{l-1}^{(s),b}$ from layer $l-1$ is the input activation to layer l. We assume the input activation $\boldsymbol{h}_{l-1}^{(m)} \in \{-1,1\}$. Therefore, the $\delta\boldsymbol{W}_l^b \in \{-2,0,2\}$, which is significantly smaller than the range of $[-2\cdot D_{l-1}, 2\cdot D_{l-1}]$, indicating that PEPITA+$\boldsymbol{W}_l^b$+$\boldsymbol{h}_l^b$ has significantly lower classification performance compared to PEPITA+$\boldsymbol{W}_l^b$+$\boldsymbol{h}_{l-1}^b$ in Table II.

For FF+$\boldsymbol{W}^b$ with binary input activations (i.e., FF+$\boldsymbol{W}_l^b$+$\boldsymbol{h}_{l-1}^b$ in Table II), according to Equation (10), taking the positive forward pass as an example, its gradient is based on $\boldsymbol{h}_l^{(p)}$, $\boldsymbol{h}_{l-1}^{(p),b}$, and $\delta g_l^{(p)} = \mathrm{Sigmoid}(g_l^{(p)} - \theta) - 1$, where $\boldsymbol{h}_l^{(p)} = \sigma_l(\boldsymbol{W}_l^b\boldsymbol{h}_{l-1}^{(p),b} + \boldsymbol{b}_l)$. $\boldsymbol{h}_l^{(p)}$ has a range $[-D_{l-1}, D_{l-1}]$ and $\boldsymbol{h}_{l-1}^{(p),b} \in \{-1,1\}$. The Kronecker product between $\boldsymbol{h}_l^{(p)}$ and $\boldsymbol{h}_{l-1}^{(p),b}$ results in a range $[-D_{l-1}, D_{l-1}]$. $\delta g_l^{(p)}$ is a scalar, leading to loss of information when $\boldsymbol{h}_l^{(p)}$ is in range $[-D_{l-1}, D_{l-1}]$ instead of Float32, because $g_l^{(p)} = \sum_i(\boldsymbol{h}_{l,i}^{(p)})^2$ and $\delta g_l^{(p)} = \mathrm{Sigmoid}(g_l^{(p)} - \theta) - 1$. Therefore, the range of the gradient is significantly smaller than the range of Float32, incurring a worse classification performance compared to FF [5], as shown in Table II.

For FF+$\boldsymbol{W}^b$ with binary output activations (i.e., FF+$\boldsymbol{W}_l^b$+$\boldsymbol{h}_l^b$ in Table II), according to Equation (11), $\boldsymbol{h}_l^{(p),b} \in \{-1,1\}$. At the same time, each binary output is the input activation to the next layer. We assume the input activation $\boldsymbol{h}_{l-1}^{(p)} \in \{-1,1\}$. Thus, the $\boldsymbol{h}_l^{(p),b} \otimes (\boldsymbol{h}_{l-1}^{(p)})^T \in \{-1,1\}$, which is significantly smaller than the range of binary input activations, i.e., $[-D_{l-1}, D_{l-1}]$. Moreover, $\delta g_l^{(p)}$ also loses information when $\boldsymbol{h}_l^{(p)} \in \{-1,1\}$ instead of range $[-D_{l-1}, D_{l-1}]$. This analysis demonstrates that FF+$\boldsymbol{W}_l^b$+$\boldsymbol{h}_l^b$ has a significantly worse classification performance compared to FF+$\boldsymbol{W}_l^b$+$\boldsymbol{h}_{l-1}^b$ in Table II.

The discussion above supports the results presented in Table II, indicating that the error of PEPITA+$\boldsymbol{W}^b$ (FF+$\boldsymbol{W}^b$) is lower than PEPITA+$\boldsymbol{W}_l^b$+$\boldsymbol{h}_{l-1}^b$ (FF+$\boldsymbol{W}_l^b$+$\boldsymbol{h}_{l-1}^b$); the error of PEPITA+$\boldsymbol{W}_l^b$+$\boldsymbol{h}_{l-1}^b$ (FF+$\boldsymbol{W}_l^b$+$\boldsymbol{h}_{l-1}^b$) is significantly lower than PEPITA+$\boldsymbol{W}_l^b$+$\boldsymbol{h}_l^b$ (FF+$\boldsymbol{W}_l^b$+$\boldsymbol{h}_l^b$).

In summary, our findings indicate that in PEPITA+$\boldsymbol{W}^b$ and FF+$\boldsymbol{W}^b$, binarizing only the weights reduces memory overheads by $\sim$31$\times$ times and computational overheads by $\sim$2$\times$ times, with minor degradation in classification performance compared to the original forward-only algorithms. When both the weights and input activations are binarized, PEPITA and FF achieve a reduction in computational overheads by $\sim$64$\times$ times in total. However, binarizing input activations results in a loss of information from full-precision gradients, leading to decreased classification performance. Additionally, binarizing output activations causes an even more pronounced degradation in classification performance.

TABLE III: Errors (%) for variants of CwComp.

Predictor	Algorithms	MNIST	CIFAR-10
Global Averaging Predictor	CwComp [6]	1.23	26.08
	CwComp+$\boldsymbol{W}^b$	**2.69**	**31.24**
	CwComp+$\boldsymbol{W}_l^b$+$\boldsymbol{h}_{l-1}^b$	31.11	59.79
	CwComp+$\boldsymbol{W}_l^b$+$\boldsymbol{h}_l^b$	69.61	85.39
Softmax Predictor	CwComp [6]	0.66	22.93
	CwComp+$\boldsymbol{W}^b$	**0.70**	**25.35**
	CwComp+$\boldsymbol{W}_l^b$+$\boldsymbol{h}_{l-1}^b$	11.63	34.33
	CwComp+$\boldsymbol{W}_l^b$+$\boldsymbol{h}_l^b$	11.36	53.73

E. Extension to CwComp

We extend the binarization to the most recent state-of-the-art forward-only algorithms, i.e., CwComp [6], due to its significantly better performance on benchmarks. We report these results with two kinds of predictors for CwComp [6] in Table III, namely, the *global averaging predictor* and the *softmax predictor*. CwComp+$\boldsymbol{W}^b$ (weight-only binarization) achieves a lower error on MNIST and a significantly lower error on CIFAR-10 when compared to PEPITA+$\boldsymbol{W}^b$ and FF+$\boldsymbol{W}^b$. To qualitatively analyze the gradient information loss, we binarize the forward pass in training.

For the global averaging predictor, the channel-wise competition (CwC) loss function, with the convolutional positive goodness, is exploited. The goodness makes the gradient similar to FF [5], as we presented in Section III-D. As presented in Table III, with binary activation, the error of CwComp+$\boldsymbol{W}^b$ is significantly increased.

For the softmax predictor, the standard Softmax layer was trained with Cross-Entropy loss using the last block. The last layer is not binarized as BP [12]; therefore, the gradient of this classifier is kept in high-precision of Float32, similar to BP [2], as we presented in Section III-D. Therefore, with binary activation, the error of CwComp+$\boldsymbol{W}^b$ with softmax predictor is also increased, but far lower than the global averaging predictor, as presented in Table III.

IV. Conclusion

Forward-only algorithms avoid error backpropagation across the entire network, hence may potentially save substantial amount of computation and memory. In this paper, we investigate binarizing several state-of-the-art forward-only algorithms, including PEPITA and FF. Our evaluation demonstrates that weight-only binarization reduces the memory overheads by $\sim 31\times$ times and the computational overheads by $\sim 2\times$ times, with minor degradation in classification performance. Binary activations reduce the computational overheads by $\sim 64\times$ times and memory overheads by $\sim 31\times$ times, albeit with a significant decrease in classification performance. Additionally, we investigate the gradient derivation differences among BP, PEPITA, and FF. Our findings indicate that binarizing activations degrade the classification performance of PEPITA and FF more than BP.

References

[1] M. Shafique, T. Theocharides, V. J. Reddy, and B. Murmann, "Tinyml: Current progress, research challenges, and future roadmap," in *2021 58th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2021, pp. 1303–1306.

[2] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[3] C. R. Banbury, V. J. Reddi, M. Lam, W. Fu, A. Fazel, J. Holleman, X. Huang, R. Hurtado, D. Kanter, A. Lokhmotov *et al.*, "Benchmarking tinyml systems: Challenges and direction," in *SysML 2020, Proceedings*, 2020.

[4] G. Dellaferrera *et al.*, "Error-driven input modulation: solving the credit assignment problem without a backward pass," in *International Conference on Machine Learning*. PMLR, 2022, pp. 4937–4955.

[5] G. Hinton, "The forward-forward algorithm: Some preliminary investigations," *arXiv preprint arXiv:2212.13345*, 2022.

[6] A. Papachristodoulou, C. Kyrkou, S. Timotheou, and T. Theocharides, "Convolutional channel-wise competitive learning for the forward-forward algorithm," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 13, 2024, pp. 14 536–14 544.

[7] R. Zanetti, A. Arza, A. Aminifar, and D. Atienza, "Real-time eeg-based cognitive workload monitoring on wearable devices," *IEEE transactions on biomedical engineering*, vol. 69, no. 1, pp. 265–277, 2021.

[8] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding," in *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

[9] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *European conference on computer vision*. Springer, 2016, pp. 525–542.

[10] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks," *Advances in neural information processing systems*, vol. 29, 2016.

[11] M. Courbariaux, Y. Bengio, and J.-P. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," *Advances in neural information processing systems*, vol. 28, 2015.

[12] C. Yuan and S. S. Agaian, "A comprehensive review of binary neural network," *Artificial Intelligence Review*, vol. 56, no. 11, pp. 12 949–13 013, 2023.

Baichuan Huang is a Ph.D. student, affiliated with the Wallenberg AI, Autonomous Systems and Software Program, in the Department of Electrical and Information Technology, LTH, at Lund University, Sweden. His research interests are bio-inspired deep learning, efficient training and inference, The Internet of Things (IoT), and edge AI.

Amir Aminifar is an Assistant Professor in the Department of Electrical and Information Technology at Lund University, Sweden. His research interests are centered around edge and embedded machine learning for Internet of Things (IoT) systems, intelligent mobile-health, and wearable systems.