IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 17 October 2023, accepted 9 November 2023, date of publication 13 November 2023, date of current version 20 November 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3332472

RESEARCH ARTICLE

SFAO: Sign-Flipping-Aware Optimization for Early-Stopping of Binarized Neural Networks

JU YEON KANG[1], (Student Member, IEEE),
CHANG HO RYU[2], (Graduate Student Member, IEEE),
SUK BONG KANG[1], (Student Member, IEEE),
AND TAE HEE HAN[2,3], (Senior Member, IEEE)
[1]Department of Electrical and Computer Engineering, Sungkyunkwan University, Suwon 16419, South Korea
[2]Department of Artificial Intelligence, Sungkyunkwan University, Suwon 16419, South Korea
[3]Department of Semiconductor Systems Engineering, Sungkyunkwan University, Suwon 16419, South Korea

Corresponding author: Tae Hee Han (than@skku.edu)

This work was supported in part by the Technology Innovation Program (or Industrial Strategic Technology Development Program-Public-Private Joint Investment Advanced Semiconductor Talent Development Project), Development of CXL/DDR5-Based Memory Subsystem for AI Accelerators, funded by the Ministry of Trade, Industry and Energy (MOTIE), South Korea, under Grant RS-2023-00237136 and Grant 1415187686; in part by the National Research and Development Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science and ICT (MSIT) under Grant 2020M3H2A1076786; in part by the Institute of Information and Communications Technology Planning and Evaluation (IITP) funded by the Korean Government (MSIT) through the Artificial Intelligence Graduate School Support Program, Sungkyunkwan University, under Grant 2019-0-00421; in part by Samsung Electronics Company Ltd., under Grant IO201209-07877-01; and in part by the BK21 FOUR Project.

ABSTRACT One of the vital challenges for the binary neural networks (BNNs) is improving their inference performance by expanding their data representation capabilities for figuring out delicate patterns and nuances in the data. Addressing the explosive computational demands on neural network training is essential to guarantee sustainable development and scalable deployment. However, mitigating the increase in the computational cost during the training phase is critical for ensuring sustainability and scalability during deployment. In this study, an advanced sign-flipping-aware optimizer (SFAO) that focuses on BNNs was introduced to diminish the computational burden. SFAO balanced the model performance and computational cost through sign-flipping-aware updating rules throughout the training of BNNs. SFAO optimizer, tailored for BNNs with binary weight-specific updating rules, considerably reduced the computing resources needed for training on the CIFAR-10 dataset. Specifically, it surpassed the conventional full-precision updating rule by reducing the total instruction count by 21.89%. In contrast, SFAO showed a marginal 0.44% decline in the image classification accuracy relative to the updating rules for the full-precision parameters. Furthermore, the implementation of early stopping using the sign flip rate led to a notable reduction of 9.37% in the average computation time per network for the ImageNet dataset.

INDEX TERMS Artificial intelligence, model compression, optimizer, efficient machine learning, binarized neural networks, layer freezing.

I. INTRODUCTION

Binarized neural networks (BNNs) have emerged as promising alternative to lightweight super-sized artificial intelligence (AI) models to ensure sustainability and scalability in computationally demanding applications [1], [2], [3]. With binary weights and activations, BNNs can cope with computational complexities and storage requirements, making them suitable for resource-constrained devices. However, aggressive binarization, which is helpful for reducing hardware demands, causes quantization errors and severely restrains the representation of the network that the parameters can express, leading to a decrease in network performance.

Two dominant approaches have been proposed to compensate for the accuracy loss originating from the binarization in BNNs. The first approach involves amplifying the

The associate editor coordinating the review of this manuscript and approving it for publication was Juan A. Lara.

representational capability of each layer by expanding the diversity of cases that the parameters can represent [4], [5], [6], [7], [8], [9], [10], [11]. The second method focuses on refining the gradient mismatch in the backward path [1], [12], [13]. By employing straight-through estimation (STE) [14] and scaling factors, Rastegari et al. [2] demonstrated a notable expansion in the network representation and more accurate parameter updates. In addition, Liu et al. [1] introduced the STE transformation of a piecewise polynomial function from the derivative of the sign function, which has impulsive features that cause a gradient mismatch in backpropagation.

Advancements in the inference performance of BNNs have reached a stage where they can be applied to real-world, close-to-human image classification tasks, such as the ImageNet dataset [15]. However, it is crucial to acknowledge the considerable escalation in incidental computations associated with training these networks to achieve the desired performance level. Despite notable improvements in reasoning capabilities within the 1-bit domain for inference, training BNNs requires compensation for representational capabilities compared with neural networks with single-precision 32-bit floating-point (FP32) parameters [1], [2], [16], [17]. Therefore, efforts must be directed toward developing sustainable training methodologies that retain inference performance and reduce computational costs.

The complexity of the learning methods utilized in BNNs, such as elastic binarization and STE, necessitates the development of new optimization techniques for lightweight training [18], [19], [20], [21]. Conventional optimizers applied to neural networks composed of FP32 must recognize the discrete properties of binary weights, which requires tailored approaches to address the unique characteristics of BNNs. Novel optimization strategies are needed to improve weight the update rules, balance performance and efficiency to overcome the complex training processes of BNNs. BNN training with full-precision weights requires more work to update the weights that remain unchanged because of the high threshold for the sign function. Therefore, the optimizers and weight-update methods for full-precision parameters must be changed to make them more efficient for BNNs.

One promising methodology involves stabilizing the layers of the BNN according to the sign flip rate (SFR) [22]. Along these lines, it directly addresses the fundamental cause of the computational burden: backpropagation and weight updating processes and training becomes faster and more efficient by reducing computational demands. Moreover, this approach ensures that any decrease in accuracy, which is a common trade-off in model simplification, is minimized. Hence, we balanced computational efficiency and model performance, demonstrating a practical method for overcoming hardware resource limitations.

The primary contributions of this study are the acceleration of BNN training by implementing enhancements to the weight-updating rule.

- We propose a method for calculating the percentage of unproductive weight updates derived from sign functions during BNN training with latent weights.
- Computational redundancy was demonstrated by quantifying the change in the binary weight with respect to the gradient magnitude.
- Enhancing the learning speed of BNNs by minimizing unnecessary latent weight updates.
- Experiments were conducted on system-level architectural simulator gem5 [23] to corroborate the computational reduction and quantify the consequent gains in energy efficiency of the proposed method.

The remainder of this paper is organized as follows: Section II provides a comprehensive review of the relevant literature and highlights the knowledge gaps that this study seeks to address. Section III describes the motivation behind the weight-freezing updating rule and presents a detailed description of the training algorithms used to test our hypotheses. Section IV presents the experimental results and ablation studies that provide detailed insights into the effectiveness and limitations of the proposed approach. Section V discusses the implications of our experimental results for the sign-flipping-aware optimizer (SFAO) and future research directions, given the limitations. Finally, Section VI summarizes the findings.

II. RELATED WORK

Additional training techniques were introduced to enhance the inference performance of BNNs, with a specific focus on the backward process. However, these operations come at the cost of a substantial increase in computational requirements, thereby limiting the scalability of deployment for training.

A. BNN TRAINING WITH LATENT WEIGHT

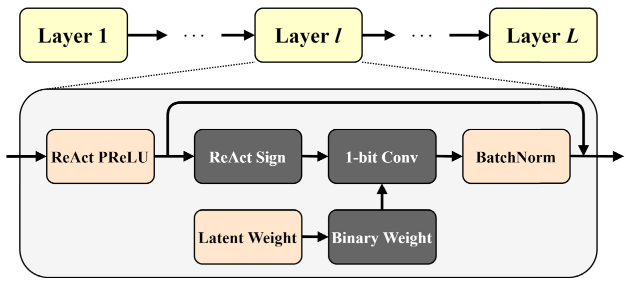

FIGURE 1. BNN training methods with latent weights.

Since the pioneering investigation by Courbariaux et al. [24], [25], BNNs have emerged as a promising approach that offers benefits, such as reduced memory usage and computational cost compared to 32-bit full-precision neural networks. The BNN training methodology, which employs latent weights to establish mappings from continuous values to binary representations, facilitates rapid computation and the efficient utilization of computing resources, making it a compelling option for many applications. In particular, Liu et al. [3] employed techniques involving the reshaping and shifting of weight distributions within parameterized sign and activation functions, as depicted in Fig. 1. The ReAct Sign function is currently learning the optimal channel-wise threshold to binarize the input feature map properly. The input distribution can be easily shifted using ReAct PReLU, and the best place to use coefficients to fold the distribution can be determined.

B. BNN PERFORMANCE IMPROVEMENTS: GRADIENTS REVIVAL

To enhance the accuracy of BNNs, various attempts have been made to lessen the gradient mismatch of the sign function and decreasing the prevalence of "dead weight," referred to as weights that do not update due to zero gradient. Xu et al. [26] successfully reduced the quantization error and improved weight utilization during updates by reviving the dead weight through the implementation of a rectified clamp unit. Liu et al. [22] quantified the stability of a BNN for optimization by employing the flip-flop (FF) rate, which represents the percentage of weights that change signs during training. They observed that a high FF ratio indicates frequent weight flipping, which adversely affects performance. First, a real-valued network is trained with weight decay to establish a solid initialization for the binary network, followed by the application of a weight decay of zero to the latent weights.

C. WEIGHT TRANSFER AND FREEZING

Owing to the sheer scale and complexity of convolutional neural networks (CNNs), lightweight methods (quantization [27], [28], [29], [30], pruning [31], [32], and low-rank approximation [33]) are required. Xiao et al. [34] proposed layer freezing training based on the gradient magnitude, which aims to shorten the training time of CNNs. By normalizing the freezing rate of the frozen layer and calculating the layer to start freezing at a fixed epoch period, the computational reduction in backpropagation was maximized.

Moreover, Isikogan et al. [35] introduced a novel approach for freezing specific parameters within each layer, substituting multipliers with fixed scalers and replacing the network with optimized full-pipeline hardware blocks. The proposed network organization offers a balance between flexibility and cost because the weights can be configured at various scales and levels of abstraction, distinguishing it from conventional layer-by-layer freezing technique.

Another approach to alleviating performance drops caused by quantization errors is to exploit weight freezing during fine-tuning using binary or ternary weights. Cavigelli et al. [36] achieved a top-1 accuracy improvement of approximately 3% over XNOR-Net [2] in image classification experiments on the ImageNet dataset based on GoogLeNet. The benefit of weight freezing in boosting the performance of binary and ternary weight networks was demonstrated by increasing the accuracy by freezing randomly selected weights during fine-tuning.

However, considering that the selection of weight freezing was executed in a stochastic manner without adhering to a particular reference point, it is posited that employing weight freezing predicated on the reference point of fine-grained weight freezing could potentially facilitate the optimization of computational reduction or enhancement of accuracy. Furthermore, compared to the parameters in FP32, binarized weights are expected to change less frequently because of their insensitivity to value changes during updates when using a small learning rate (LR) that does not cross the threshold of the sign function.

III. SIGN-FLIPPING-AWARE OPTIMIZATION

Unlike CNNs, which consist of FP32 parameters, BNNs employ latent weights and a sign function to convert them into binary values during the training phase. Early BNN studies categorized binary mapping into two types: stochastic and deterministic. However, the computational cost and complexity associated with probabilistic binarization render this unreasonable, leading to the development of BNN training primarily based on deterministic approaches. Thus, we shall outline the problem definition and suggest a technique for addressing the updating rules during the training of deterministic binarization.

A. PROBLEM DEFINITION

Latent weights, also known as real-valued or floating-point weights, play an essential role in BNN training. Although BNNs use binary weights during forward propagation to minimize computational and memory requirements, real-valued latent weights are used during the weight update process to maintain a higher precision representation of the weights.

The weight update process in BNNs typically involves the following steps:

1) The network performs forward and backward propagation with binary weights and activations, thereby allowing for XNOR operations.
2) Gradient calculation using latent weights: The gradients were computed using real-valued latent weights to maintain precision during the update process.
3) The latent weights are updated using the computed gradients and an optimization algorithm, such as stochastic gradient descent (SGD) or one of its variants.
4) Binary weights are derived from the updated latent weights by applying a binarization function that typically involves thresholding the latent weights to produce binary values (-1 or 1).

The latent weight update process directly influenced the SFR change in each binary layer. When the latent weights are updated, the resulting binary weights cannot change their signs unless the updated latent weights cross the binarization threshold. Therefore, understanding the relationship between latent weights and SFR in BNNs is crucial for optimizing the training process and enhancing network performance. Developing strategies to control layer freezing by adapting the updating rule based on SFR can contribute to more stable and efficient training of resource-constrained devices.

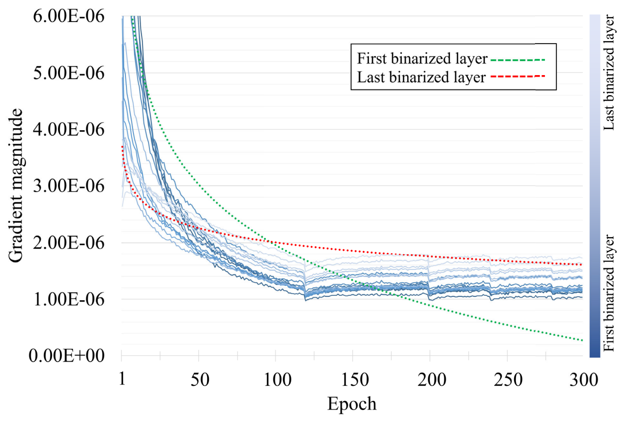

FIGURE 2. Tendency investigation with average of absolute gradient magnitude value for each layer throughout the epoch in ResNet20 model.

As illustrated in Fig. 2, the average of absolute magnitude for the gradients reaches the saturation point during the training process, resulting in minimal weight updates during the latter of training. The first binarized layer undergoes frequent updates in the early stage of training, manifested as sign flips. In contrast, as the training extends over a prolonged duration, the last binarized layer exhibits a higher propensity for weight updates than the first. In BNNs, the sensitivity to weight changes is diminished by the sign function, rendering the weight updates computationally useless unless the sign changes beyond the zero threshold.

B. SFR IN BNNS

SFR is quantifies the rate at which the binary weights in a BNN change their signs during the training process. The SFR of the l-th layer at epoch e is calculated as the ratio of the number of weight updates that result in a sign conversion to the total number of weight updates as follows:

$$SFR_l^e = \frac{\sum (W_{sign}^e \oplus W_{sign}^{e-1})}{n(W_{sign}^e)} \times 100\% \tag{1}$$

where W_{sign}^e represents the signed weight set W at present epoch e and W_{sign}^{e-1} represents the previous epoch weight set. The increased in the SFR derived from Eq. (1) signifies that the direction of the weight update undergoes frequent alterations, which that can potentially engender instability during training.

As the training progresses, the model converges towards an optimal point, and as shown in Fig. 3, the difference in the sign of the weight resulting from the latent weight update accounted for a negligible proportion at each layer. Freezing layers with low SFR that have a negligible effect on the gradient computation during backward propagation and updates provides an opportunity to significantly reduce the training computation of BNNs.

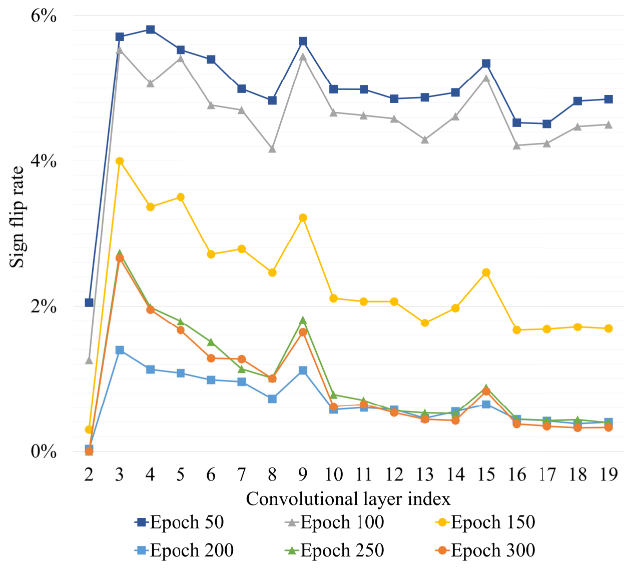

FIGURE 3. SFR for ResNet20 on CIFAR-10.

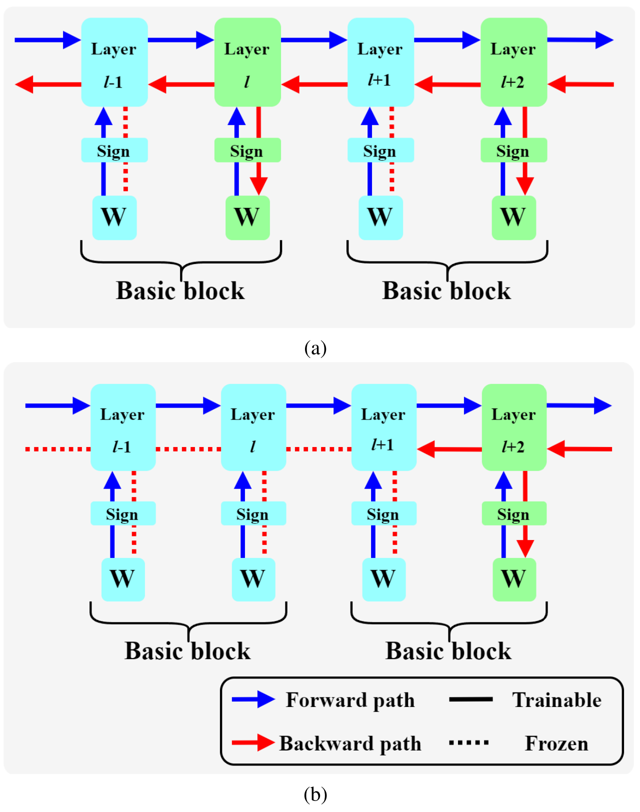

FIGURE 4. Binarized layer freezing flow: (a) only frozen weight updating, (b) layer freezing when former layers are not to be updated for diminishing backward computation.

To enhance the computational efficiency of BNN training, we implemented layer freezing for SFR of 0% or below the user-defined threshold, as shown in Fig. 4-(a). In addition, to maximize the computational reduction achieved by the proposed method, we employed backward blocking when all previous layers were frozen, as shown in Fig. 4-(b). The

TABLE 1. Average SFR tendency according to datasets and network sizes.

Dateset	Network	Average SFR for binary convolutional layer			
		Term 1 (25%)	Term 2 (50%)	Term 3 (75%)	Term 4 (100%)
CIFAR-10	ResNet20	4.55	3.78	3.18	0.77
	ResNet32	4.19	3.30	2.57	0.47
CIFAR-100	ResNet20	4.56	4.04	3.53	1.07
	ResNet32	4.49	3.83	3.12	0.77
ImageNet	ResNet18	5.63	4.36	3.37	0.79
	ResNet34	4.95	3.14	2.04	0.29
	ResNet50	2.01	0.86	0.43	0.26

detailed procedure for freezing the layers is presented in Algorithm 1.

Algorithm 1 Training Binarized Neural Network With SFAO

```
Input: $W$, FP32 weight set; $W_b$, binarized weight set; $A$,
    activation set; $e$, number of iterations; $l$, layer index;
    $SFR_l^e$, SFR for layer $l$ at iteration $e$; $SFR_{th}$, do not update
    threshold SFR value; $LR_e$, learning rate at iteration $e$
Output: $M_B$, trained target BNN model
1:  for $e \leftarrow 0$, iterations do
        (1) forward computation.
2:      Run forward computation of $M_B$ simultaneously.
3:      $W_b = \text{sign}(W) \cdot \frac{1}{n}\sum_n |W|$.
4:      Calculate $A_{e+1} = \text{popcount}(W_b \odot A_e)$.
        (2) backward and gradient computation.
5:      Calculate $SFR_l^e$ using Eq. (1).
6:      Compute network loss $\mathcal{L}$.
7:      if $SFR_l^e < SFR_{th}$ then
8:          updating flag = False
9:      end if
10:     if updating flag == True then
11:         $W_{e+1} \leftarrow W_e - LR_e \cdot \frac{\partial \mathcal{L}}{\partial W}$
12:     end if
13: end for
14: return trained BNN model $M_B$
```

C. SFR VARIABLES BASED ON DATASET AND NETWORK SIZE

1) HEURISTIC APPROACH

To examine the fluctuations in the SFR of the baseline network throughout the training process, PyTorch, a deep neural network framework based on Python, was employed to gauge the SFR across 300 epochs on CIFAR-10 and CIFAR-100 and 128 epochs on the ImageNet dataset [15], [37], as listed in Table 1.

The SFR value, as ascertained through the experimental results, exhibited fluctuations contingent on the complexity of the network and dataset. However, there was a concurrent convergence of the SFR to a specific value was seen as the loss value approached saturation, particularly in the CIFAR-10 datasets, where a notable key point was achieved at the 160 th epochs, with the SFR registering 0%. These findings indicate that the network reached a state of equilibrium concerning the weight updates, indicating the stabilization of the training.

Furthermore, employing a step LR schedule plays a pivotal role in determining SFR tendencies. Specifically, at the juncture of 150 epochs, an LR of 1-e3 is instituted, substantially diminishing the likelihood of alterations in the sign flip. This reduction in the propensity for sign flip changes can be attributed to the significantly small LR, which induces only minute updates in the latent weights.

By contrast, when considering larger datasets, the SFR did not reach 0% despite the saturation of the loss values. Table 1 illustrates this phenomenon, where CIFAR-100 and ImageNet exhibit loss saturation of 1.2 and 2.2, respectively; however, the SFR for each layer does not reach 0%. This result resembles the challenges encountered in the LR on the CIFAR-10 dataset. Therefore, we experimented with substantial adjustments to the LR to observe its impact on the SFR. Although these adjustments influenced the SFR for each layer, they did not improve accuracy.

2) SFR ADAPTIVE LAYER-WISE EARLY-STOP

To mitigate overfitting and minimize excessive training iterations, CNNs employ an early-stop mechanism that terminates training if no significant loss decreases beyond a predefined patience value. In contrast, BNNs do not require a precise training continuation factor based on loss values. Hence, an early-stop approach for BNNs exploiting SFR is proposed that allows direct insights into weight changes during training. By monitoring the SFR, we can identify the optimal stopping point, ensuring effective convergence of the BNN while avoiding unnecessary iterations that lead to overfitting.

Rather than relying solely on the loss employed as the criterion for early-stopping in traditional CNNs, SFR was introduced as the stopping criterion, quantifying the frequency at which the binary weights change their signs during training. Therefore, by monitoring the SFR as it changes at every epoch, we can identify an appropriate epoch to halt training, ensuring that the network is closer to a local minimum.

The customized early-stop approach for BNNs based on SFR (SFRES) are divided into three main parts:

Algorithm 2 Training Binarized Neural Network With SFRES

```
Input: $W$, FP32 weight set; $W_b$, binarized weight set; $A$, activation set; $e$, number of iterations; $l$, layer index; $SFR_l^e$, SFR for layer $l$ at iteration $e$; $MA[l]_e$, SFR moving average for layer $l$ at epoch $e$; $LR_e$, learning rate at iteration $e$
Output: $M_B$, trained target BNN model
1:  for $e \leftarrow 0$, iterations do
        (1) forward computation.
2:      Run forward computation of $M_B$ simultaneously.
3:      $W_b = \text{sign}(W) \cdot \frac{1}{n}\sum_n |W|$.
4:      Calculate $A_{e+1} = \text{popcount}(W_b \odot A_e)$.
        (2) backward and gradient computation.
5:      Calculate $SFR_l^e$ using Eq. (1).
6:      Compute network loss $\mathcal{L}$.
7:      if $|MA[l]_e - MA[l]_{e-1}| < delta$ then
8:          $patient++$
9:      end if
10:     if $patient > 5$ then
11:         updating flag = False
12:     end if
13:     if updating flag == True then
14:         $W_{e+1} \leftarrow W_e - LR_e \cdot \frac{\partial \mathcal{L}}{\partial W}$
15:     end if
16: end for
17: return trained BNN model $M_B$
```

1) Calculate the SFR for each layer.
2) The SFR values calculated for each epoch are compared to determine whether a weight update should be performed for a specific layer.
3) Perform backward blocking, which increases computational efficiency when all preceding layers have their weights frozen.

Explicitly customized for the BNNs, the proposed early-stop approach considers the distinctive characteristics of binary weights and determines a training termination point, at which these weights become marginal and remain unchangeable. This mitigates the risk of premature termination or prolonged training beyond the optimal point, thereby enhancing the convergence and alleviating the overall computational cost of BNNs.

$$MA[l]_e = \sum_{e=0}^{m-1} SFR_l^e \tag{2}$$

$$\text{if} \quad MA[l]e - MA[l]e - 1 < delta, \quad \text{then} \quad patient++ \tag{3}$$

Moving averages (MA) have proven effective in smoothing the metrics over time, delivering a more reliable indication of when to halt training, mainly when the metric contains anomalous values. The SFR calculation for layer l in epoch e relies on a user-defined window size, denoted by m, as described in (2). The window size determines the range, over which the MA is computed, thereby contributing to the overall reliability of the training.

To determine whether the current layer was trained using the MA value for the SFR, we compare the MA value in the current epoch, $MA\,[l]_e$, with the MA value in the previous epoch, $MA\,[l]_{e-1}$, as shown in (3). Subsequently, we choose updated whether to update the weights based on the specified *SFR* and SFR_{th}. The complete training process of the SFRES is presented in Algorithm 2.

IV. EVALUATION

Through comprehensive analysis and experimentation on both network performance and hardware perspectives, we examine the impact of layer freezing on the overall training process. This evaluation focused on quantifying the improvements in the training speed achieved by implementing the sign-flipping-aware updating rule in the BNNs. By comparing the total instruction count of the proposed layer-freezing method with that of the baseline, we ascertained its efficacy in reducing the training duration. The results also investigated the potential trade-offs, limitations, and challenges associated with layer freezing, providing insights into its applicability and benefits for improving the efficiency of BNN training.

A. IMAGE CLASSIFICATION

In the context of image classification, we conducted experiments on two of our proposed methods, SFRES and SFAO. Firstly, the experiments on SFRES are designed to check the differences with the general optimizer for FP32, and the experiments on SFAO are compared by referring to the experimental results of related studies.

1) EXPERIMENTAL SETUP

To assess the efficacy of layer freezing using the sign-flipping-aware updating rule, we established experimental environments using PyTorch 1.3.1, CUDA 10.2, and CUDNN 7.6.5. The comparative analysis involved renowned datasets, such as CIFAR-10, CIFAR-100, and ImageNet [d,e]. Additionally, ReActNet [3] was employed as a state-of-the-art BNN methodology to benchmark the proposed techniques.

The hyperparameters were carefully configured throughout the experiments on the image classification applications. The weight decays were assigned to 1e-4. LRs of 1e-1, 1e-2, and 1e-3 at the 1st, 150th, and 225th epochs, respectively. These meticulously selected settings facilitate a comprehensive evaluation of the sign-flipping-aware updating rule, specifically within the context of the considered datasets and methodologies.

2) RESULTS

As discussed in Section III-C, the SFR exhibited varying saturation points according to the network depth and dataset. Hence, we conducted experiments utilizing using various *patience* and *delta* to establish an SFRES that caters to the specific characteristics of each situation.

TABLE 2. Experimental results for SFRES based on a given network model on CIFAR-10 dataset.

BNN methodology	Network	Avg. SFR (%)	Top-1 accuracy (%)	The early-stop epoch point
ReActNet	ResNet20	2.39	85.62	277
	ResNet32	1.72	86.75	280
	ResNet44	1.24	87.46	285

TABLE 3. Experimental results for SFRES based on a given network model on CIFAR-100 dataset.

BNN methodology	Network	Avg. SFR (%)	Top-1 accuracy (%)	The early-stop epoch point
ReActNet	ResNet20	2.38	55.55	288
	ResNet32	1.98	57.58	288
	ResNet44	1.86	59.26	278

TABLE 4. Experimental results for SFRES based on a given network model on ImageNet dataset.

BNN methodology	Network	Avg. SFR (%)	Top-1 accuracy (%)	The early-stop epoch point
ReActNet	ResNet18	2.78	57.31	109
	ResNet34	1.08	63.91	118
	ResNet50	0.22	60.04	116

In our experiments with CIFAR-10, SFRES terminated learning before 300 epochs, which falls short of the specified breakpoint, as listed in Table 2. Subsequently, we continued the experiments without applying SFRES, and despite the increase in top-1 accuracy averaging 0.77%, the performance degradation caused by SFRES decreased, particularly with an increase in network depth.

The experiments conducted on CIFAR-100 with the same network depth as CIFAR-10, as depicted in Table 3, exhibit a similar performance degradation tendency similar to that of the CIFAR-10 experiment.

For the experiments on the ImageNet dataset listed in Table 4, we chose 128 epochs as the final termination point, owing to the size of the dataset. As the depth of the network and dataset size increase, the change in the average SFR became blunted, and the number of epochs that terminated earlier became faster. This is because as the depth of the network increases, the representational capabilities of the network improve, although the probability of gradient vanishing increases, making it difficult to expect performance improvement. Within the same epoch, owing to shortcuts in the network, the first layer of each residual block had a high SFR, whereas the second layer of the block showed an average difference of 3.8 times compared to the SFR of the first layer, which we interpreted as a reason for the insignificant improvement in accuracy.

The BNN was trained on the ImageNet dataset under the equivalent condition of using pretrained FP32 weights as initial weights for a fair comparison with other state-of-the-art techniques. In this experiment, layer freezing was employed when the SFR reached 1% or less. Unlike the RPR [36] method, which freezes a certain percentage of random weights without considering the SFR of each layer, SFAO selectively freezes layers based on their infrequent weight updates determined by SFR calculations. This approach prevents unnecessary updates by hindering network performance improvement. The experimental results presented in Table 5 demonstrate the effectiveness of the SFAO in maintaining performance while reducing unnecessary updates.

TABLE 5. Experimental results and a comparison to related work on ImageNet.

Network	Method	Param. precision	Top-1 accuracy (%)
ResNet18	Baseline	FP32	69.76
ResNet18	ADMM [38]	Binary	64.80
ResNet18	XNOR-Net [2]	Binary	60.80
ResNet18	RPR [36]	Binary	64.62
ResNet18	SFAO (ours)	Binary	64.97
ResNet50	Baseline	FP32	76.15
ResNet50	ADMM [38]	Binary	68.70
ResNet50	XNOR-Net [2]	Binary	63.90
ResNet50	RPR [36]	Binary	65.14
ResNet50	SFAO (ours)	Binary	65.01

B. TRAINING COMPARISON FOR COMPUTATION WITH LAYER FREEZING

Beyond image classification, we extended our experiment to include an object detection application utilizing a BNN as the backbone network. However, our experiment with object detection on the COCO dataset [39] revealed several challenges. mAP, a crucial metric for object detection performance following binarization, is required to improve its usability. Considering the aforementioned limitations, it is conceivable to argue that SFR experiments with a binarization backbone do not result in consistent and reliable findings. The substantial likelihood of deriving inaccurate inferences from unreliable data led us to consider the risk unacceptably high. Given these circumstance, we leave SFR experiments on object detection for future work and opt for an alternative approach to evaluate the experiment from a hardware-centric standpoint.

TABLE 6. Experimental setup for system-level architectural simulator gem5.

gem5 full system mode system configurations	
CPU core model	Minor (In-Order)
ISA	ARMv8 (aarch64)
CPU core frequency	2.3 GHz
Supply voltage VDD	1.0 V
L1 data/instruction cache size	32 KB
L2 cache size	1 MB
Memory model	8 GB DDR4, 2400MHz
Memory bus width	16 b

1) EXPERIMENTAL SETUP

Analyzing MAC operations with PyTorch provides valuable insight into computational variation. However, actual resource utilization, which considers hardware-specific factors, is also important. Therefore, we identified improvements in layer freezing by using a system-level architectural simulator, gem5, with an integrated energy model to obtain hardware-friendly results. Using gem5, the total instruction count can be assessed in detail by type, thereby enhancing the validity of the findings.

The experimental environment for gem5 involve CPU-based training with the system configuration shown in Table 6. DarkNet [40], a C++-based artificial intelligence learning framework, was utilized as the benchmark for implementing the proposed updating rule. The experiments were conducted on an ARM ISA running Ubuntu 18.04 and Linux kernel 4.9.92.

2) TOTAL INSTRUCTION

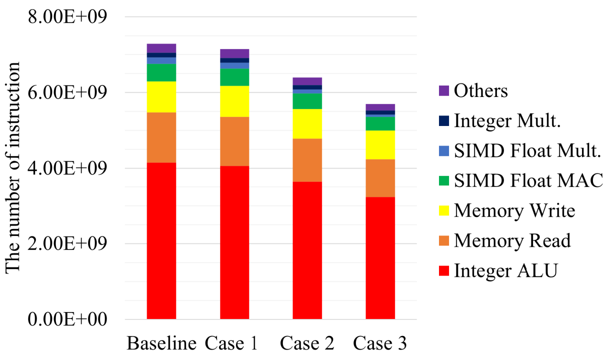

FIGURE 5. The number of instructions for ResNet20 on CIFAR-10 dataset.

TABLE 7. Various frozen layer cases at thresholds below 2% SFR.

Case	Frozen layer	Epoch	Top-1 accuracy (%)	Ratio to baseline (%)
Baseline	None	300	86.51	-
Case 1	7th	242	83.78	1.96
Case 2	7th,12-14th,17-19th	261	84.82	12.24
Case 3	7-19th	286	85.98	21.89

The instances in which the SFR dropped below the 2% threshold were categorized into three cases, and the distribution of instruction types is shown in Fig. 5. Among these, the most significant reduction was observed for the Integer ALU instruction type, exhibiting a decline of 22.16%. Correspondingly, Case 1 demonstrated a 1.96% reduction in the total number of instructions, Case 2 experienced a more substantial decrease of 12.24%, and Case 3 exhibited the most considerable reduction at 21.89%, all relative to the baseline. Notably, despite these changes, the impact on Top-1 accuracy, a pivotal metric for network performance, was minimal, declining by 0.44%, as shown in Table 7, thereby confirming the marginal extent of the accuracy reduction.

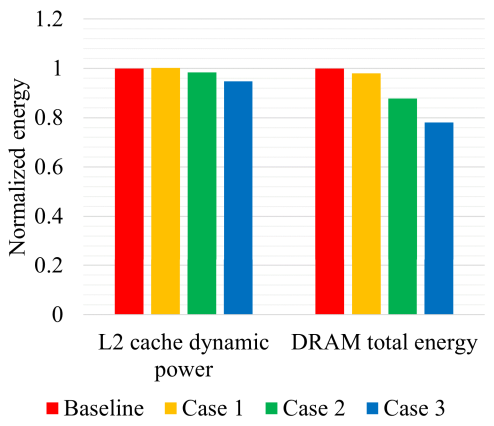

FIGURE 6. Energy consumption for ResNet20 on CIFAR-10 dataset.

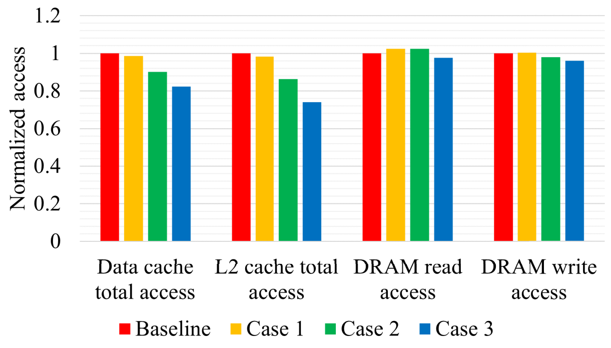

FIGURE 7. Normalized overall accesses for ResNet20 on CIFAR-10 dataset.

3) ENERGY CONSUMPTION

As shown in Fig. 6, the evaluation of the total energy consumption attributable to L2 cache and DRAM showed an energy reduction of 25.93% and 21.89%, respectively, and to further explore the investigation, we plotted the normalized number of accesses to cache and DRAM for each scenario, as shown in Fig. 7. Within the memory hierarchy framework, it is evident that Case 3 demonstrates a significant reduction in L2 cache accesses, amounting to a remarkable 25.93% decrease under the specified SFR.

V. DISCUSSION

SFAO for BNNs reveals its potential to reduce the number of instructions up to 21.89% during training while slightly dropping 0.44% Top-1 accuracy on the CIFAR-10 dataset. Based on the sign flip rate, the early stopping mechanism further optimized the training process, particularly with image classification. When contrasted with related work, our approaches, aligning with some established findings, emerged as a unique and efficient alleviation offering a 25.93% notable energy reduction in L2 cache dynamic power.

While the results are promising, the primary reliance on the application of image classification may not capture the complexities of real-world scenarios such as object detection and unsupervised learning. The robustness and scalability of SFAO in more extensive and diverse datasets remain areas for further exploration. Additionally, integrating adaptive learning mechanisms, which modulate hyperparameters based on input data [41], could further enhance the efficacy of SFAO.

Future work includes the amalgamation of SFAO with adaptive learning and its application in diverse BNN architectures. A particularly intriguing direction is the potential use of SFAO for training on embedded systems such as mobile, IoT, and individualized devices for privacy.

VI. CONCLUSION

The increasing number of AI applications and parameters calls for lighter networks. Typically, BNNs focus on improving the inference stage performance. Adopting a contrasting perspective, we implemented an alternative methodology that concentrated on lightweight BNN training. A new weight-updating rule was devised that exploits latent weight and employs the sign function for binarization. We effectively reduced backpropagation costs by incorporating a sign-flip threshold and implementing sequential layer freezing. Our experimental findings revealed that incorporating layer pruning during BNN training led to a marginal accuracy decline of 0.44% on the CIFAR-10 dataset. However, this strategy substantially reduced the computational cost, amounting to an impressive 21.89%. These findings highlight the potential of the proposed approach for enhancing the efficiency of BNNs and addressing the need for more computationally efficient models.

BNNs characterized by binary parameters have been primarily established using fine-tuning techniques for object detection applications. Additionally, gradual quantization methods applied to FP32 parameters have been employed to overcome the inherent limitations of the representational capacity compared to FP32 parameters. However, these approaches incur increased computational requirements and complexities, exceeding those associated with FP32 neural networks in binarization. Therefore, there is a pressing need for further investigation into BNN training methodologies, particularly in object detection, while accounting for the unique characteristics of binary parameters.

REFERENCES

[1] 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. (ECCV)*, 2018, pp. 722–737.

[2] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-Net: ImageNet classification using binary convolutional neural networks," in *Computer Vision—ECCV*. 2016, pp. 525–542.

[3] Z. Liu, Z. Shen, M. Savvides, and K. Cheng, "ReActNet: Towards precise binary neural network with generalized activation functions," in *Proc. ECCV*, 2020, pp. 2980–2988.

[4] P. Wang, Q. Hu, Y. Zhang, C. Zhang, Y. Liu, and J. Cheng, "Two-step quantization for low-bit neural networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4376–4384.

[5] Q. Hu, P. Wang, and J. Cheng, "From hashing to CNNs: Training binary weight networks via hashing," in *Proc. AAAI Conf. Artif. Intell.*, vol. 32, 2018, pp. 3247–3254.

[6] J. Choi, Z. Wang, S. Venkataramani, P. I.-J. Chuang, V. Srinivasan, and K. Gopalakrishnan, "PACT: Parameterized clipping activation for quantized neural networks," 2018, *arXiv:1805.06085*.

[7] D. Zhang, J. Yang, D. Ye, and G. Hua, "LQ-Nets: Learned quantization for highly accurate and compact deep neural networks," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 365–382.

[8] M. Shen, X. Liu, R. Gong, and K. Han, "Balanced binary neural networks with gated residual," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2020, pp. 4197–4201.

[9] X. He, Z. Mo, K. Cheng, W. Xu, Q. Hu, P. Wang, Q. Liu, and J. Cheng, "ProxyBNN: Learning binarized neural networks via proxy matrices," in *Proc. 16th Eur. Conf. Comput. Vis. (ECCV)*, Glasgow, U.K. Springer, Aug. 2020, pp. 223–241.

[10] M. Lin, R. Ji, Z. Xu, B. Zhang, Y. Wang, Y. Wu, F. Huang, and C.-W. Lin, "Rotated binary neural network," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 7474–7485.

[11] P. Pham, J. A. Abraham, and J. Chung, "Training multi-bit quantized and binarized networks with a learnable symmetric quantizer," *IEEE Access*, vol. 9, pp. 47194–47203, 2021.

[12] H. Qin, R. Gong, X. Liu, M. Shen, Z. Wei, F. Yu, and J. Song, "Forward and backward information retention for accurate binary neural networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 2247–2256.

[13] J. Y. Kang, C. H. Ryu, and T. H. Han, "Binarized neural network with parameterized weight clipping and quantization gap minimization for online knowledge distillation," *IEEE Access*, vol. 11, pp. 8057–8064, 2023.

[14] Y. Bengio, N. Léonard, and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," 2013, *arXiv:1308.3432*.

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, vol. 25, Dec. 2012, pp. 1097–1105.

[16] 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*.

[17] A. Bulat and G. Tzimiropoulos, "XNOR-Net++: Improved binary neural networks," 2019, *arXiv:1909.13863*.

[18] 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*.

[19] R. Ding, T.-W. Chin, Z. Liu, and D. Marculescu, "Regularizing activation distribution for training binarized deep networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 11400–11409.

[20] Y. Xu, X. Dong, Y. Li, and H. Su, "A main/subsidiary network framework for simplifying binary neural networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 7147–7155.

[21] J. Gu, C. Li, B. Zhang, J. Han, X. Cao, J. Liu, and D. Doermann, "Projection convolutional neural networks for 1-bit CNNs via discrete back propagation," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, 2019, pp. 8344–8351.

[22] Z. Liu, Z. Shen, S. Li, K. Helwegen, D. Huang, and K.-T. Cheng, "How do Adam and training strategies help BNNs optimization," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 6936–6946.

[23] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood, "The gem5 simulator," *ACM SIGARCH Comput. Archit. News*, vol. 39, no. 2, pp. 1–7, 2011.

[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.*, Dec. 2015, pp. 3123–3131.

[25] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks: Training deep neural networks with weights and activations constrained to $+1$ or -1," 2016, *arXiv:1602.02830*.

[26] Z. Xu, M. Lin, J. Liu, J. Chen, L. Shao, Y. Gao, Y. Tian, and R. Ji, "ReCU: Reviving the dead weights in binary neural networks," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 5178–5188.

[27] T. Zhao, X. He, J. Cheng, and J. Hu, "BitStream: Efficient computing architecture for real-time low-power inference of binary neural networks on CPUs," in *Proc. 26th ACM Int. Conf. Multimedia*, Oct. 2018, pp. 1545–1552.

[28] Q. Hu, G. Li, P. Wang, Y. Zhang, and J. Cheng, "Training binary weight networks via semi-binary decomposition," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 637–653.

[29] Y. Wu, Y. Wu, R. Gong, Y. Lv, K. Chen, D. Liang, X. Hu, X. Liu, and J. Yan, "Rotation consistent margin loss for efficient low-bit face recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 6865–6875.

[30] F. Zhu, R. Gong, F. Yu, X. Liu, Y. Wang, Z. Li, X. Yang, and J. Yan, "Towards unified INT8 training for convolutional neural network," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 1966–1976.

[31] Y. He, X. Zhang, and J. Sun, "Channel pruning for accelerating very deep neural networks," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 1398–1406.

[32] S. Ge, Z. Luo, S. Zhao, X. Jin, and X.-Y. Zhang, "Compressing deep neural networks for efficient visual inference," in *Proc. IEEE Int. Conf. Multimedia Expo (ICME)*, Jul. 2017, pp. 667–672.

[33] J. Wen, B. Zhang, Y. Xu, J. Yang, and N. Han, "Adaptive weighted non-negative low-rank representation," *Pattern Recognit.*, vol. 81, pp. 326–340, Sep. 2018.

[34] X. Xiao, T. B. Mudiyanselage, C. Ji, J. Hu, and Y. Pan, "Fast deep learning training through intelligently freezing layers," in *Proc. Int. Conf. Internet Things (iThings), IEEE Green Comput. Commun. (GreenCom), IEEE Cyber, Phys. Social Comput. (CPSCom), IEEE Smart Data (SmartData)*, Jul. 2019, pp. 1225–1232.

[35] L. F. Isikdogan, B. V. Nayak, C.-T. Wu, J. P. Moreira, S. Rao, and G. Michael, "SemifreddoNets: Partially frozen neural networks for efficient computer vision systems," in *Proc. 16th Eur. Conf. Comput. Vis. (ECCV)*, 2020, pp. 193–208.

[36] L. Cavigelli and L. Benini, "RPR: Random partition relaxation for training; binary and ternary weight neural networks," 2020, *arXiv:2001.01091*.

[37] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Univ. Toronto, Toronto, ON, Canada, Tech. Rep., 2009.

[38] C. Leng, Z. Dou, H. Li, S. Zhu, and R. Jin, "Extremely low bit neural network: Squeeze the last bit out with ADMM," in *Proc. AAAI Conf. Artif. Intell.*, vol. 32, 2018, pp. 3466–3473.

[39] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common object in context," in *Proc. 13th Eur. Conf. Comput. Vis. (ECCV)*, 2014, pp. 740–755.

[40] J. Redmon, "DarkNet: Open source neural networks in C," 2016. [Online]. Available: http://pjreddie.com/darknet/

[41] O. Tutsoy and C. Tanrikulu, "A machine learning-based 10 years ahead prediction of departing foreign visitors by reasons: A case on türkiye," *Appl. Sci.*, vol. 12, no. 21, p. 11163, Nov. 2022.

JU YEON KANG (Student Member, IEEE) received the B.S. degree in electronic engineering from the Tech University of Korea, Siheung, South Korea, in 2016. He is currently pursuing the M.S. and Ph.D. degrees in electrical and computer engineering with Sungkyunkwan University, Suwon, South Korea. His research interests include artificial intelligence, machine learning, and computer architecture.

CHANG HO RYU (Graduate Student Member, IEEE) received the B.S. degree in electronic engineering from Korea Aerospace University, Goyang, South Korea, in 2022. He is currently pursuing the M.S. and Ph.D. degrees in artificial intelligence with Sungkyunkwan University, Suwon, South Korea. His research interests include machine learning and computer architecture.

SUK BONG KANG (Student Member, IEEE) received the B.S. degree in electronic engineering from Sungkyunkwan University, Suwon, South Korea, in 2022, where he is currently pursuing the M.S. degree in electrical and computer engineering. His research interests include artificial intelligence, processing in memory, and computer architecture.

TAE HEE HAN (Senior Member, IEEE) received the B.S., M.S., and Ph.D. degrees in electrical engineering from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea, in 1992, 1994, and 1999, respectively. From 1999 to 2006, he was with the Telecom Research and Development Center, Samsung Electronics, where he developed 3G wireless, mobile TV, and mobile WiMax handset chipsets. From 2011 to 2013, he was a full-time Advisor on system ICs with Korean Government. Since March 2008, he has been with Sungkyunkwan University, Suwon, South Korea, as a Professor. His current research interests include SoC/Chiplet architectures for AI, advanced memory architecture, network-on-chip, and system-level design methodologies.