ACE-CNN: Approximate Carry Disregard Multipliers for Energy-Efficient CNN-Based Image Classification

Salar Shakibhamedan, *Graduate Student Member, IEEE*, Nima Amirafshar, Ahmad Sedigh Baroughi, Hadi Shahriar Shahhoseini, and Nima Taherinejad, *Member, IEEE*

***Abstract*— This paper presents the design and development of Signed Carry Disregard Multiplier (SCDM8), a family of signed approximate multipliers tailored for integration into Convolutional Neural Networks (CNNs). Extensive experiments were conducted on popular pre-trained CNN models, including VGG16, VGG19, ResNet101, ResNet152, MobileNetV2, InceptionV3, and ConvNeXt-T to evaluate the trade-off between accuracy and approximation. The results demonstrate that ACE-CNN outperforms other configurations, offering a favorable balance between accuracy and computational efficiency. In our experiments, when applied to VGG16, SCDM8 achieves an average reduction in power consumption of 35% with a marginal decrease in accuracy of only 1.5%. Similarly, when incorporated into ResNet152, SCDM8 yields an energy saving of 42% while sacrificing only 1.8% in accuracy. ACE-CNN provides the first approximate version of ConvNeXt which yields up to 72% energy improvement at the price of less than only 1.3% Top-1 accuracy. These results highlight the suitability of SCDM8 as an approximation method across various CNN models. Our analysis shows that the ACE-CNN outperforms state-of-the-art approaches in accuracy, energy efficiency, and computation precision for image classification tasks in CNNs. Our study investigated the resiliency of CNN models to approximate multipliers, revealing that ResNet101 demonstrated the highest resiliency with an average difference in the accuracy of 0.97%, whereas LeNet5 Inspired-CNN exhibited the lowest resiliency with an average difference of 2.92%. These findings aid in selecting energy-efficient approximate multipliers for CNN-based systems, and contribute to the development of energy-efficient deep learning systems by offering an effective approximation technique for multipliers in CNNs. The proposed SCDM8 family of approximate multipliers opens new avenues for efficient deep learning applications, enabling significant energy savings with virtually no loss in accuracy.**

Manuscript received 15 September 2023; revised 26 December 2023 and 30 January 2024; accepted 15 February 2024. Date of publication 1 March 2024; date of current version 30 April 2024. This work was supported by European Union's Horizon 2020 Research and Innovation Programme through the Marie Skłodowska Curie (APROPOS: Approximate Computing for Power and Energy Optimization, http://www.apropos.eu/) under Grant 956090. This article was recommended by Associate Editor Y. Tang. *(Corresponding author: Salar Shakibhamedan.)*

Salar Shakibhamedan is with the Institute for Computer Technology (ICT), TU Wien, 1040 Vienna, Austria (e-mail: salar.shakibhamedan@tuwien.ac.at).

Nima Amirafshar, Ahmad Sedigh Baroughi, and Hadi Shahriar Shahhoseini are with the School of Electrical Engineering, Iran University of Science and Technology, Tehran 13114-16846, Iran (e-mail: nima_amirafshar@elec.iust.ac.ir; sadighbaroughi_aelec.iust.ac.ir; shahhoseini@iust.ac.ir).

Nima Taherinejad is with the Institute of Computer Engineering, Heidelberg University, 69117 Heidelberg, Germany, and also with the Institute for Computer Technology (ICT), TU Wien, 1040 Vienna, Austria (e-mail: nima.taherinejad@ziti.uni-heidelberg.de).

Digital Object Identifier 10.1109/TCSI.2024.3369230

***Index Terms*— Energy efficiency, convolutional neural network, approximate multiplier, image classification.**

I. Introduction

In THE last decade, CNN has become one of the most prominent methods in computer vision tasks, such as image classification [1], object detection [2], and semantic segmentation [3]. Despite their high configurability and computational capacity, CNNs are known to consume a significant amount of energy and computational resources due to their intricate structure and the number of parameters involved [4], [5], [6]. CNNs are known for their error-resilient nature, especially in computer vision tasks. However, with the rise in implementation of deep learning algorithms, including CNNs, on-edge devices, and power-constrained hardware [6], [7], there is a growing demand for improved power efficiency. This has motivated us to focus on optimizing the resource usage of CNNs. Image classification tasks utilize CNN due to their higher performance at the cost of expanding the computational complexity and resource-hungry units. Higher computational complexity increases the energy consumption and the required design time drastically while decreasing the speed of the system [8]. A conventional CNN combines convolutional, activation, pooling, and a fully connected layer. Fundamentally, the most computationally intensive operation is the multiplication operation for convolutional layers [9].

Generally, multipliers have three main stages: Partial Product (PP) generation, PP reduction (accumulation), and final addition. There are three main architectures to reduce PPs: the Carry-Save Adder (CSA) array, the Wallace tree, and the Dadda tree [10]. Compared to array structures, Tree-based PP accumulation has less delay; however, it has more power consumption and area because it has a more considerable number of computing units and higher complexity [10], [11]. In contrast, the array multiplier has a straightforward, uniform, and modular architecture in which development and management are easy to comprehend. Correspondingly, such architecture typically has less power consumption and area, which is more optimal than Wallace- and Dadda-based architectures [10], [11].

Approximate multipliers trade the accuracy of the result in a return for energy and delay improvement. In an error-tolerant application such as image classification, the energy and delay are of the greatest concern. Using approximate multipliers for the core operations of a CNN-based model could alleviate the computational complexity and resource requirement of the model, leading to an acceptable classification and improved

performance. In this paper, we take advantage of this property to propose new approximate signed array multipliers that are not as accurate as their predecessors, but are more resource-efficient and do not cause a significant loss in the classification accuracy of CNN.

As for array architecture, the main bottleneck is carry propagation, which causes significant dependency among Partial Product Units (Π0s). To tackle this problem, our proposed signed approximate multipliers some PP columns disregard carries in a specific way. Generally, there are various ways for ignoring carries, each of which causes different levels of approximation and hardware criteria. According to our experiments, the smaller number of carry disregard does not necessarily result in the more accurate and efficient multipliers. We think the location of carry disregard should be selected judiciously; in fact, it is possible to ignore a large number of carries to achieve high-speed and efficient multipliers while gaining better accuracy and suitability for neural networks. Hence, the important point is to find an appropriate method for disregarding the carries, which depends on the multiplier architecture. Our method results in the parallel operation of PP columns and reduces hardware complexity. Furthermore, due to the fact that our multipliers consist of two 8×4 groups and we intend to disregard carries from the first column up to a certain column, there are ten possible configurations for each group. This leads to 100 various approximate multipliers with different approximation levels, which have been studied comprehensively in this paper. Thus, our designs strike a better balance between hardware and accuracy criteria, and in comparison with other works [12], [13], we achieve considerably more efficient CNN-based image classifiers.

Our contributions in this paper are: (1) Design of new approximate signed multipliers with an emphasise on their resource efficiency. (2) Design of CNN-based image classifiers using the proposed approximate multipliers. (3) Showing the performance and energy merits of the CNN-based model for image classification tasks on the ILSVRC-2012 dataset, using exact and approximate signed multipliers, and (4) Studying the interplay of the approximate multipliers properties and the classifiers' performance to find the optima.

The remainder of this paper is organized as follows: We review the literature on approximate multipliers and their use in CNN-based models in Section II. We describe a new proposed approximate signed multipliers and the methodology for designing them in Section III. We propose our CNN models for image classification tasks using the proposed approximate signed multipliers in Section IV. We describe our experimental setup and present its results in Section V. We compare our work with similar existing works in Section VI and discuss some of the future steps. Finally, we conclude this paper in Section VII.

II. Related Works

1) Approximate Multipliers: Multiplication is a primary and most common arithmetic operation in diverse applications. Therefore, we can expect a significant improvement in the system's performance by optimizing the multiplication units [14]. Consequently, integrating approximate computing in the design of multiplier units and using them in various applications can considerably reduce the critical path delay and improve power consumption and area. To apply approximation in multipliers, designers use the approximation in three main computation stages. One of the methods is the operands truncation; this method exploits the fact that all bits of the operands are not equally important. Therefore, selecting only a segment of operand bits leads to a much smaller core multiplier [15], [16]. Another method is approximation through reducing PPs. In approximate Wallace and Dadda multipliers [10], the central unit for reducing PPs is the approximate compressor which has been studied thoroughly, see e.g., [13], [17], [18], [19], and [20].

The design in [12] includes a constant-truncated part, an error correction module, and an exact part. As for the constant region, in the eight least significant columns, PPs are not generated, and this part always has an 8-bit constant output, which is "00000110". It is akin to using 4-2 approximate compressors in some PP columns of constant region, which produce Carry and Sum equal to zero for all input combinations. Their approximate compressor generates the exact result for the "0000" combination, which is most likely to occur at the input. Moreover, since the error is negative for all inaccurate results, designing a compensation module is easier and more efficient. Regarding the exact region, [12] utilizes accurate compressors to determine the final result.

Reference [13] first developed two types of approximate 4-2 compressors, namely, the positive compressor (PC) and the negative compressor (NC), by considering the error directions, not the amount of errors. PC and NC are designed by modifying the truth table to improve hardware complexity. Afterwards, by means of PC and NC, two approximate multipliers are developed to produce inaccurate results in the opposite direction, denoted as the positive multiplier (PM) and the negative multiplier (NM). They proposed a novel interleaving mode of approximate multipliers with opposite error directions, which results in the balanced error distribution during MAC operations.

Reference [20] first proposed an approximate 4-2 compressor and then designed an approximate multiplier consisting of truncated region, approximate region, and accurate region. They manipulated the truth table of a 4-2 compressor so as to reduce hardware complexity. This approximate 4-2 compressor has accurate results for all entries of its truth table (i.e., $X_1X_2X_3X_4$) except just four combinations in which X_3 and X_4 equal 1. The error is always -1, which means the approximate results of the compressor are less than the corresponding exact value. Error being always in the same direction simplifies the design of a compensation module.

Concerning approximate array multipliers [10], one of the most common approximate methods to reduce PPs is using approximate Half Adder (HA) and approximate Full Adder (FA). Another method is to eliminate many PPs. Reference [21] proposed Broken-Array Multiplier (BAM). This multiplier eliminates several PPs using Vertical Break Level (VBL) and Horizontal Break Level (HBL) parameters. Also, [22] proposed the probabilistic analysis-based PP array and replaced many PPs with other values using the proposed Propagation and Generation (PG) function. Then eliminated a number of them and reduced the delay. The main limiting factor for array multipliers is carry propagation. Therefore, [23] and [24] proposed approximate unsigned array multipliers in which several Π0s of columns disregard carry. In this paper, we have extended them and proposed new approximate signed multipliers as a new branch in the carry-disregard-based family and we called them SCDM8s.

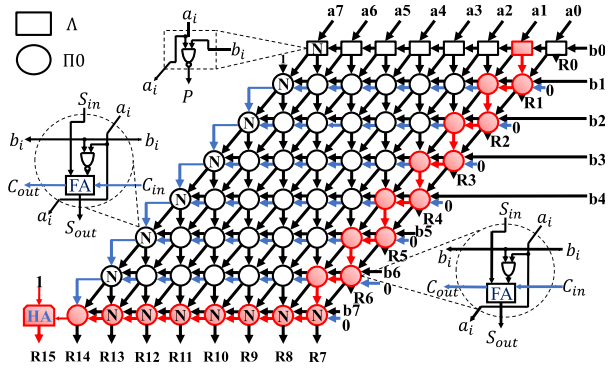

Fig. 1. The architecture of a conventional exact 8-bit signed multiplier and logic circuit of partial product unit (Π0).

2) Approximate CNNs: In the CNN context, many approximation techniques, such as pruning [25], quantization [26], and weight sharing [27], have been used to balance power efficiency and accuracy. These techniques are mostly conducted on the software (algorithm) level [28]; however, in this paper, we use approximate multipliers as our approximation techniques on both software and hardware levels.

Several research papers have investigated the utilization of approximate multipliers in CNNs to improve accuracy and hardware efficiency. Here, we reviewe these studies.

In [29], authors investigated the impact of approximate multipliers on Neural Network (NN) accuracy and hardware efficiency. They demonstrated that approximate multipliers can significantly reduce hardware costs while improving classification accuracy. By replacing exact multipliers with approximate designs, they achieved up to a 0.63% improvement in classification accuracy, while reducing energy consumption and area.

Authors in [30] analyzed the effects of approximate multiplication in Deep Neural Networks (DNNs). They identified critical factors in different CNN layers that allowed for accurate predictions despite the errors introduced by approximate multiplication. Their experiments with recognized network architectures, such as ResNet and Inception-v4, demonstrated that approximate multipliers could produce predictions nearly as accurate as FP32 references, with notable energy savings.

In [31] authors proposed energy-efficient approximate multipliers optimized for CNNs based on Mitchell's log multiplication. They employed various design techniques and operand truncation to reduce energy consumption. Experimental results, showed significant energy savings compared to exact fixed-point multipliers, with minimal performance degradation.

Authors, in [32], introduced an approximate multiplier and a multiplier-less artificial neuron for energy-efficient neural computing. They exploited the application's resilience to error and computation sharing to achieve improved energy consumption in NNs. Their proposed design achieved substantial reductions in power consumption and area while maintaining acceptable accuracy levels.

CAxCNN [33] proposed a canonic sign digit (CSD)-based approximation methodology for representing filter weights in pre-trained CNNs. By using multipliers with reduced computational complexity, their technique allowed for the low area and computational overhead while preserving multiple levels of classification accuracy.

In [34], authors focused on FPGA-based approximate multipliers for error-tolerant computing. They presented several approximate multiplier designs, including those with associated carry chains, to reduce critical path delay and power consumption. Their designs showed improvements in latency, area, and power compared to existing softcore multipliers. [35], proposed ARA (Approximate computing based Reconfigurable Architecture), a CNN accelerator with approximate computing techniques. They explored hardware-compatible network compression algorithms and approximate computing units, such as iterative multipliers and multi-port SRAM integrated LUT-based multipliers, to achieve high energy efficiency in CNN acceleration.

III. Proposed Signed Multipliers

A. Exact Signed Multipliers

Signed multiplication is one of the most common and fundamental computational operations in various applications. There are different ways to represent negative numbers, with two's complement being the most common. However, multiplying two's complement numbers can be complex because, unlike unsigned multiplication, the output is not simply the sum of PPs. Reference [36] introduced an optimized algorithm called Baugh-Wooley for multiplying signed two's complement numbers. Consider two n-bit operands and their two's complement $\alpha = (\alpha_{n-1}\alpha_{n-2}\ldots\alpha_1\alpha_0)$ and $\beta = (\beta_{n-1}\beta_{n-2}\ldots\beta_1\beta_0)$, whose product we want to calculate. Equations (1) and (2) determine the decimal values of these two numbers, based on which we can calculate the product of α and β using Equation (3), where α_i and β_j denote the i-th and j-th bit of α and β.

$$\alpha = -\alpha_{n-1}2^{n-1} + \sum_{i=0}^{n-2}\alpha_i 2^i \tag{1}$$

$$\beta = -\beta_{n-1}2^{n-1} + \sum_{j=0}^{n-2}\beta_j 2^j \tag{2}$$

$$\begin{aligned}\alpha\times\beta &= (-\alpha_{n-1}2^{n-1} + \sum_{i=0}^{n-2}\alpha_i 2^i)\times(-\beta_{n-1}2^{n-1} + \sum_{j=0}^{n-2}\beta_j 2^j)\\ &= \alpha_{n-1}\beta_{n-1}2^{2n-2} + \sum_{i=0}^{n-2}\sum_{j=0}^{n-2}\alpha_i\beta_j 2^{i+j}\\ &\quad - 2^{n-1}\sum_{i=0}^{n-2}(\alpha_i\beta_{n-1}2^i) - 2^{n-1}\sum_{j=0}^{n-2}(\alpha_{n-1}\beta_j 2^j)\end{aligned} \tag{3}$$

Equation (3) clearly shows that the final product is not simply the sum of PPs, but in two cases, it is obtained by subtracting PPs, which is not desirable and makes the design of the signed multiplier more complex. However, according to reference [36], subtraction can be converted to addition by taking the two's complement of the negative numbers. Finally, as Equation (4) shows, it is sufficient to add all the PPs together to determine the final product of the multiplication.

$$\begin{aligned}\alpha\times\beta &= \alpha_{n-1}\beta_{n-1}2^{2n-2} + \sum_{i=0}^{n-2}\sum_{j=0}^{n-2}\left(a_i\beta_j 2^{i+j}\right) + 2^n - 2^{2n-1}\\ &\quad + 2^{n-1}\sum_{i=0}^{n-2}\left(\overline{\alpha_i\beta_{n-1}}2^i\right) + 2^{n-1}\sum_{j=0}^{n-2}\left(\overline{\alpha_{n-1}\beta_j}2^j\right)\end{aligned} \tag{4}$$

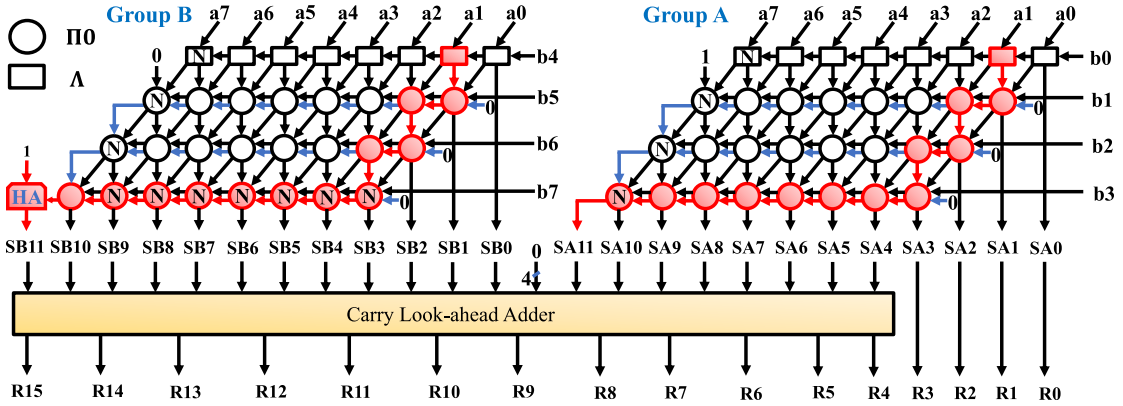

Fig. 2. The architecture of an exact 8-bit signed multiplier using two 8×4 groups.

Therefore, to design an 8-bit signed multiplier, only two types of partial product units are required. Figure 1 illustrates an accurate 8-bit signed array multiplier that includes multiple Π0s, each consisting of one AND Gate (Λ) and a FA used for both single-bit multiplication and determining the sum of PPs. In addition to Π0, this multiplier has other partial product units that differ only in the use of a NAND Gate instead of one Λ (denoted by "N" in Figure 1). Besides these partial product units, there are multiple Λs and one NAND Gate in the first row of the multiplier used solely for generating PPs. The last row has a HA that determines the most significant bit of the output. This accurate signed multiplier has a long critical path, indicated in red in Figure 1. However, by creating a horizontal cut in the middle of the multiplier and dividing it into two smaller 8×4 groups, the delay of the 8-bit multiplier can be reduced.

Figure 2 shows an accurate 8-bit signed multiplier consisting of two 8×4 groups, named Group A and Group B, located in the lower and higher significant parts, respectively. In effect, groups A and B are not a smaller-scale 8×4 multiplier; they are the upper-half and lower-half circuits of the conventional 8-bit multiplier in Figure 1. Owing to this fact, these two parts are structurally different, and the main difference is related to the final row of the lower-half part (i.e., Group B), in which all partial product units are based on the NAND Gate except the last one. Moreover, this row has one HA compared to Group A as well. In Figure 2, The multiplier also includes a Carry Look-ahead Adder (CLA) that determines the sum of the outputs of groups A and B, resulting in the final output of the 8-bit multiplier. Compared to the conventional structure (i.e., Figure 1), this architecture has a much lower critical path delay. This improvement is due to not only having shorter critical paths for each groups A and B but also operating independently and in parallel to each other (the critical path for each groups A and B is indicated in red in Figure 2). Furthermore, the CLA reduces the delay caused by carry propagation and accelerates the process of determining the final output of the 8-bit signed multiplier, making the architecture shown in Figure 2 the basis of our proposed designs. Nevertheless, this structure still faces limiting factors, the most important of which is the propagation of carry among partial product units in each group A and B. We believe that by selecting and using an appropriate approximation method, the effect of this limiting factor can be reduced, and a suitable balance between accuracy and hardware efficiency criteria, such as critical path delay, power consumption, and area, can be achieved.

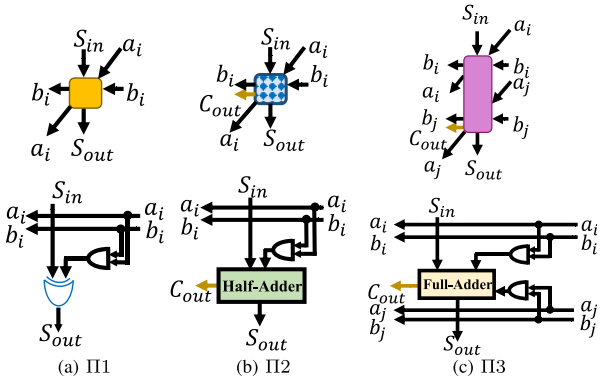

Fig. 3. Circuits and symbols of efficient partial product units.

B. Approximate Signed Multipliers

We introduce the 8-bit SCDM8s, which have a structural similarity to the multiplier in Figure 2. As stated earlier, this architecture consists of two 8×4 groups, and the delay of these groups has the most significant impact on the overall multiplier delay. In other words, due to the simultaneous operation of groups A and B, improving the critical path delay of both leads to a considerable reduction in delay in the 8-bit signed multiplier. The main source of delay in groups A and B is the propagation of the carry bit, which creates a strong dependency among the partial product units. This dependency requires each partial product unit to wait for the completion of some other units before starting its own computations. Thus, if we ignore the carry bit in each column of these units, we can see that the columns of the partial product units can work independently and in parallel, resulting in an improvement in delay. Disregarding the carry bit not only improves the speed but also reduces power consumption and area due to simpler partial product units.

Figure 3 shows the logical circuit of the optimized partial product units, namely the Carry Disregard Partial Product Unit (Π1), the Half-adder-Based Partial Product Unit (Π2), and the Full-adder-Based Partial Product Unit (Π3). Π1 does not have any input or output for the carry bit (i.e., C_{in} and C_{out}) and only consists of one Λ for single-bit multiplication and one XOR Gate for calculating the sum of PPs. On the

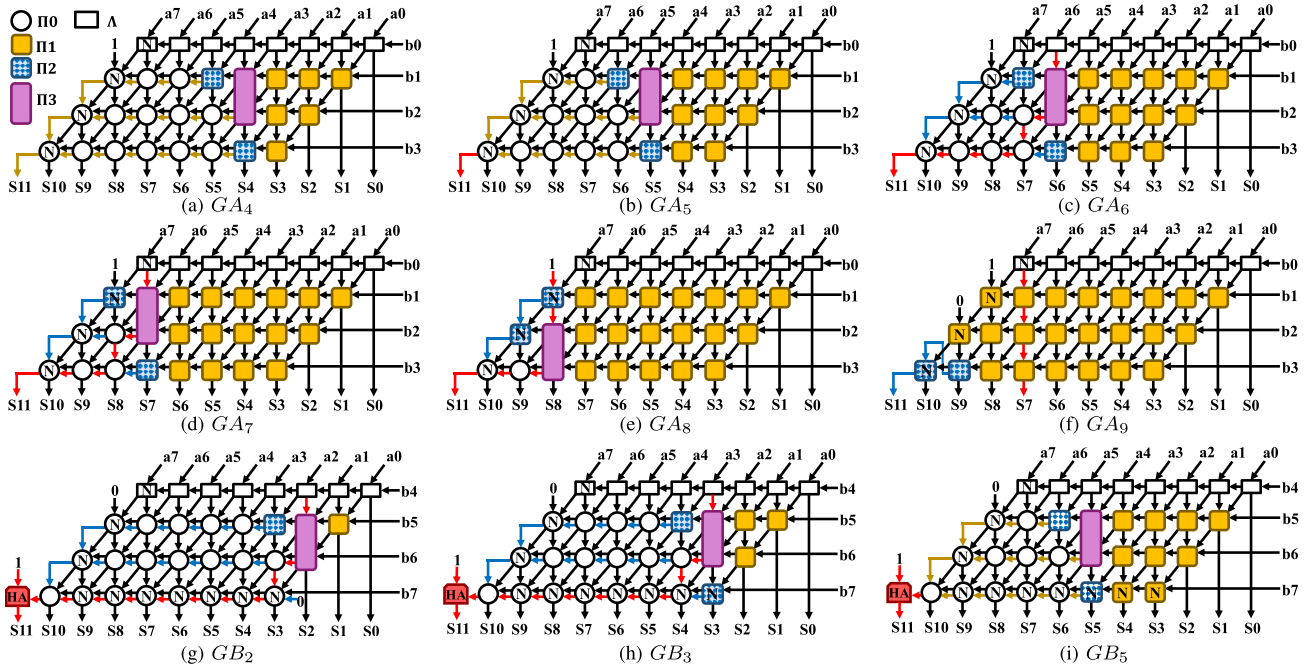

Fig. 4. Symbolic representation of the proposed approximate 8×4 units for Group A and Group B.

other hand, Π2 only has an output for the carry bit (i.e., C_{out}) and, unlike Π0, ignores the input carry bit (i.e., C_{in}). Π2 is similar in having a Λ for single-bit multiplication but additionally includes a HA for determining the sum of PPs. Π3 is essentially a combination of two Π0s but lacks the input carry bit and only has an output for the carry bit.

In general, in each 8×4 group, we aim to disregard the carry bits from the first column up to a certain column, and there are 10 different possible configurations for this purpose. Additionally, there are 100 combinations of groups A and B configurations, each resulting in a unique approximate 8-bit signed multiplier with nonidentical critical path delay, power consumption, area, and accuracy. In this work, we examined and experimented with all 100 possible combinations for the approximate 8-bit signed multiplier. Since each of these approximate multipliers has different levels of accuracy, based on the experimental results, for presentation in the paper, we only selected the combinations that showed acceptable compatibility and provided suitable output quality for image classification applications based on CNNs. Therefore, only 20 approximate signed multipliers were chosen out of a total of 100 combinations to appear in the paper.

Figure 4 shows some proposed approximate 8×4 units for Group A and Group B, which are based on disregarding the carries. The 8×4 units for groups A and B are denoted as GA_x and GB_x, respectively. In each of these units, the hexadecimal parameter x determines that the carries are disregarded from the first column to the x-th column. GA_1 and GB_1 are the exact 8×4 units for groups A and B, respectively, as shown in Figure 2. This is because in these units, there is only one Λ in Column 1, and as expected, they do not generate carries. For Group A, units GA_4 to GA_a are selected, as experiments have shown that the GA_2 and GA_3 have not much difference in terms of accuracy compared to GA_4, but GA_4 is more efficient in terms of power consumption and area. Interestingly, our experiments show that GA_a causes an unacceptable error for image classification. Thus, for this specific application, we disregard all configurations with GA_a. According to Figure 4, the GA_4, GA_5, GA_6, and GA_7 follow a similar pattern in their architecture. These units disregard the carries from the first column to Column 4, Column 5, Column 6, and Column 7, respectively, and the partial product units in all these columns are of type Π1. Therefore, these columns can work independently in parallel, and significantly improve the delay. Following the columns that disregard the carries, there is one Π3 and two Π2s, and the other partial product units are of the conventional type Π0.

Figure 4e also shows the approximate unit GA_8, which disregards all carries from Column 1 to Column 8, enabling simultaneous and independent operation for columns 1 to 9. Similarly, all partial product units in columns 1 to 8 are of type Π1, and Column 9 has one Π2 and one Π3. The first unit in Column 10 is of type Π2, and the other units are of type Π0. The approximate unit GA_9 (Figure 4f) disregards all carries from Column 1 to Column 9, providing parallel operation for these columns. Additionally, since the first partial product unit in Column 10 does not produce carries, a Π1 can be used. Furthermore, in columns 10 and 11, there are two Π2s, where the output C_{out} of the first one is connected to the second one's S_{in} input. The GA_a disregards all carries from the first column to Column 10, allowing the partial product unit in Column 11 not to produce carries. Hence, all partial product units of the GA_a are of type Π1, making it the simplest 8×4 unit for Group A.

Regarding Group B, the design process is completely similar to what was introduced for Group A. Since Group B is in the most significant part of the 8-bit multiplier, disregarding carry digits in this unit leads to more significant errors in the output of the 8-bit multiplier. As a result, we have chosen only GB_1 to GB_5 for Group B according to the experimental results. Concerning the main differences in the structure of Group B compared to Group A, we can refer to Row 4, in which all partial product units have one NAND Gate (denoted by "N" in Figure 4) except the last one, as well as a HA that determines the significant bit of the result. Another notable point is that the final partial product unit in Row 2 has

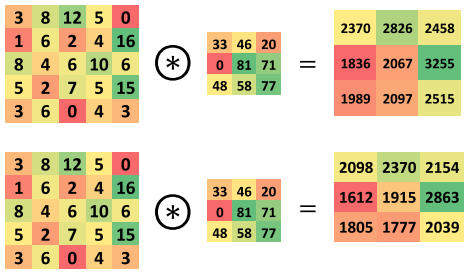

Fig. 5. The comparison between accurate (top) and approximate (bottom) convolution, in which red background shows smaller values and green shows higher values. The color patterns show the similarity of the outputs in both versions.

the constant value of zero for its S_{in} input, which is one in Group A. For instance, GA_5 and GB_5 (Figures 4b and 4i) have similar circuits except for the mentioned differences in rows 4 and 2.

The approximate unit, GB_2 ((Figure 4g), only disregards the carries of the first two columns, resulting in columns 1 to 3 operating simultaneously and independently. The second column of this unit has one Π1, and columns 3 and 4 have one Π3 and one Π2, respectively, while the other partial product units are of type Π0. On the other hand, the architectures of units GB_3 (Figure 4h), GB_4, and GB_5 (Figure 4i) follow a similar pattern. These units, respectively, ignore the carries of columns 1 to 3, 4, and 5, and these columns have Π1s. After the carry-disregarding columns, there are one Π3 and two Π2s, while the other partial product units are Π0. In general, it can be said that the more columns in both groups A and B operate in parallel and independently, the shorter the critical path will be, and hence the delay of these two groups will be reduced.

In conclusion, as mentioned earlier, by combining different 8×4 units related to groups A and B, various approximate signed 8-bit multipliers are obtained under the name SCDM8, which have different hardware characteristics and accuracy levels. The proposed approximate signed multipliers are represented as SCDM8_xy, where the hexadecimal parameters x and y determine the types of 8×4 units located in Group A and Group B, respectively. For instance, SCDM8_84 consists of GA_8 and GB_4 in groups A and B, in turn, and similarly, SCDM8_a2 consists of approximate units GA_a and GB_2.

IV. Proposed Approximate CNN

We used VGG16 [37], VGG19 [37], ResNet101 [38], ResNet152 [38], MobileNetV2 [39], InceptionV3 [40], and ConvNeXt-T (Tiny) [41] as the CNNs for the image classification task to present the capabilities of the proposed approximate multipliers in practical applications. In CNNs, the output of each neuron consists of the contribution of many other neurons. Consequently, the output result is not highly dependent on a single computation or processing. Subsequently, suppose there is some error in one of the computations, which consists of the final output result of the neuron. This could be compensated by other computations and processing. This diversity of contribution provides room for approximation in CNNs.

Figure 5 illustrates the output results of a convolution operation carried out by an accurate and approximate multiplier (CDM8_84), respectively. As demonstrated, the general pattern and shape of outputs for the convolution as well as the vector-matrix multiplication (not shown here due to space limitation) are similar. The pattern of the outputs is what mainly affects the final result and decision. So, keeping the main shape of the pattern and structure of the outputs is fundamental in our computations. In Section V, we study the interplay of our proposed approximate multipliers and the performance of our CNN-based image classifiers.

TABLE I
Architecture and Details of the Used CNNs

Model	Params (M)	MACs (B)	Input Shape	Output Shape
VGG16 [37]	138.36	15.5	(224,224,3)	1000
VGG19 [37]	143.67	19.67	(224,224,3)	1000
ResNet101 [38]	44.55	7.85	(224,224,3)	1000
ResNet152 [38]	60.19	11.58	(224,224,3)	1000
MobileNetV2 [39]	3.5	0.319	(224,224,3)	1000
InceptionV3 [40]	23.88	5.72	(299,299,3)	1000
ConvNeXt-T [41]	28.60	4.50	(224,224,3)	1000
LeNet5 Inspired-CNN	0.3	0.003	(28,28,1)	10

As detailed in Table I, the employed CNNs are characterized by their remarkable scale, comprising millions of parameters and potentially yielding billions of operations during the inference process for a given input image. Table I illustrates the employed CNNs based on their computational complexity, measured by the number of MACs (in billion) per inference, as well as their architectural characteristics. These CNNs encompass a diverse array of architectural components, including convolutional layers, pooling layers, concatenation layers, and fully connected layers, which collectively contribute to their robust performance. Extensive studies conducted by [40], [42], [43], and [44] have shed light on the computational dynamics of these CNNs.

Notably, their findings reveal that a significant portion, approximately 95-99%, of the MACs operations are executed within the convolutional layers. This concentration of computational workload within the convolutional layers underscores their pivotal role and computational intensity due to the substantial number of multiplications and additions involved. It is worth emphasizing that this characteristic is not exclusive to the specific CNN architecture under investigation but is rather a common observation across a wide range of CNN models [45], [46], [47], [48]. The computational demands imposed by convolutional layers can be attributed to their intricate operations, including convolving the input data with numerous filters, extracting meaningful features through non-linear transformations, and performing spatial pooling operations to enhance translation invariance.

By recognizing the computational intensity of the convolutional layers and understanding their dominant role in MAC operations, researchers and practitioners can focus their efforts on optimizing these critical components to achieve efficient and high-performance CNN architectures. Hence, the pursuit of novel algorithms, hardware accelerators, and architectural innovations is driven by the aim of mitigating the computational burden and improving the overall efficiency of CNNs, ensuring their widespread applicability in various domains, such as computer vision, natural language processing, and autonomous systems.

The training and inference processes of Machine Learning (ML) and DNN algorithms necessitate the utilization of diverse hardware platforms, encompassing Central Processing Units (CPUs), Graphics Processing Units (GPUs), Tensor Processing Units (TPUs), and custom accelerators [6]. Each of these hardware units exhibits unique architectural characteristics and properties, including varying levels of parallelism that have a profound impact on the overall performance of these

algorithms. In our design framework, we place significant emphasis on considering the quantity of parallel arithmetic operations that can be executed simultaneously as a critical specification. This consideration allows us to calculate the total delay associated with each operation by taking into account the number of parallel operations and the clock period, which represents the reciprocal of the operational frequency.

Within CNNs, which are widely employed in various domains, the input of each layer is intricately dependent on the output of the preceding layer. As a consequence, the overall delay experienced by a CNN is intricately linked to multiple factors, such as the number of parameters within each layer, the quantity of parallel operations, the clock period, and other elements that influence the data flow and inter-layer dependencies. To tackle these inherent challenges and enhance the efficiency of CNNs, we propose the incorporation of approximate multipliers specifically tailored for quantized CNNs. This approach allows us to effectively address two critical issues.

Firstly, it mitigates the memory-intensive nature of quantization by leveraging approximation techniques, enabling more efficient use of available memory resources. Secondly, it addresses the computationally demanding aspect of CNN operations by leveraging the benefits of approximation, reducing the computational complexity without significantly sacrificing accuracy.

By incorporating these approximate multipliers into quantized CNNs, we not only mitigate the challenges associated with memory utilization and computational intensity but also address two paramount concerns: energy consumption and delay. These considerations hold significant importance, particularly in the context of hardware platforms characterized by power constraints, such as Internet of Things (IoT) devices, embedded systems, and edge devices. The ability to optimize energy efficiency and simultaneously reducing processing delays is essential for enabling the widespread deployment of ML and DNN algorithms in resource-constrained environments. As we continue to explore innovative algorithms, hardware accelerators, and architectural advancements, our objective is to further enhance the efficiency and performance of CNNs on various hardware platforms, paving the way for advancements in computer vision, and other domains that heavily rely on DNNs.

V. Experiments and Results

A. Experimental Setup

1) Approximate Multipliers: We have designed the proposed approximate multipliers in Verilog and synthesized them in 45nm NanGate Technology using Cadence Genus v2018 to analyze the critical path delay, power consumption, and area. Based on this data (inserted in Table II), we generated the behavioral description of these multipliers as a Python code and integrated them into the utilized CNNs.

2) CNNs: During the training phase, it is common for power and computational resources to be unrestricted, allowing for ample computational capacity. In this study, we took advantage of this freedom by employing a selection of pre-trained CNN models from the TensorFlow library [49] in Python. Specifically, we utilized well-established models such as VGG16, VGG19, ResNet101, ResNet152, MobileNetV2, InceptionV3, and ConvNeXt-T which have demonstrated impressive performance across various computer vision tasks. These models were chosen for their proven efficacy and robustness in image classification.

For the training process, we opted for the use of the ImageNet dataset [50], a widely recognized benchmark in the field of computer vision. This dataset comprises millions of high-resolution images spanning a diverse range of objects and scenes. By employing the float32 number representation, we ensured precision and accuracy during the training phase. The network parameters of the pre-trained CNN models were fine-tuned using this comprehensive dataset, enabling them to learn intricate patterns and features that contribute to their superior performance.

Following the training stage, an important step in our approach involved quantizing the trained CNN models from float32 to int8 number representation. Quantization is a technique employed to reduce memory storage requirements and computational complexity during the inference phase. By representing the network weights and activations with fewer bits, we achieved a balance between computational efficiency and maintaining satisfactory accuracy levels.

For the evaluation process, we used the post-training quantization (PTQ) on the trained CNNs' weights and activations. This involved converting the values from float32 to int8 based on the type of the layer or tensor. The used specifications such as granularity, scaling factors, and zero points are similar to TensorFlow Lite's 8-bit quantization scheme [49], [51].

In terms of the dataset used for evaluation, we utilized the well-known ILSVRC-2012 dataset [50]. Comprising approximately 1.3 million images distributed across 1000 distinct classes, this dataset serves as a widely adopted subset of the larger ImageNet dataset [50]. It is worth noting that for the purpose of implementing the inference stage, we solely relied on the validation subset, as it provides an unbiased evaluation of the model's performance on previously unseen data.

In addition to the mentioned pre-trained CNN models, we extended our experimental setup by training and quantizing a LeNet5-inspired CNN architecture. This approach allowed us to explore a different network architecture and compare its performance against the established models, thereby enhancing the comprehensiveness of our study. By expanding our repertoire of models, we sought to gain a deeper understanding of the strengths and limitations of various CNN architectures in the context of our classification task.

B. Results

1) Approximate Multipliers: The result of our evaluations regarding the performance parameters of proposed approximate signed multipliers is inserted in Table II. The Mean Relative Error Distance (MRED) refers to the average of the difference between the exact and approximate multiplier output (i.e., Ω_{P_k} and Ω_{X_k}, respectively) divided by the corresponding exact output for all input combinations (k = 1 to 2^{2N} for an N-bit multiplier), i.e.,

$$MRED = \frac{1}{2^{2N}} \sum_{k=1}^{2^{2N}} \frac{\Omega_{P\_k} - \Omega_{X\_k}}{\Omega_{P\_k}} \quad ; \forall \Omega_{P\_k} \neq 0. \tag{5}$$

The SCDM8s compared to the exact signed multiplier (denoted as "S_Exact8r" in Table II) improved the delay, power consumption, and Power Delay Product (PDP) by 26.61%, 27.7%, and 45.95% on average, respectively. Among SCDM8s, the lowest MRED is 0.001 belongs to SCDM8_41,

TABLE II
PROPERTIES OF THE PROPOSED 8-BIT MULTIPLIERS

Multipliers	Power (μW)	Delay (nS)	PDP	MRED
S_Exact8r†	91.675	0.793	72.698	0
SCDM8_41	85.065	0.675	57.418	**0.001**
SCDM8_42	83.354	0.662	55.180	0.003
SCDM8_43	80.847	0.660	53.359	0.013
SCDM8_44	76.507	0.662	50.647	0.030
SCDM8_45	71.407	0.660	47.128	0.082
SCDM8_51	79.823	0.627	50.049	0.003
SCDM8_52	78.188	0.595	46.521	0.005
SCDM8_53	75.646	0.591	44.706	0.015
SCDM8_54	71.206	0.592	42.153	0.032
SCDM8_55	66.077	0.591	39.051	0.084
SCDM8_61	75.631	0.634	47.950	0.009
SCDM8_62	73.953	0.726	53.689	0.012
SCDM8_63	71.133	0.598	42.537	0.021
SCDM8_64	66.651	0.598	39.857	0.039
SCDM8_65	61.478	0.598	36.763	0.091
SCDM8_71	71.988	0.630	45.352	0.026
SCDM8_72	69.919	0.566	39.574	0.028
SCDM8_73	66.866	0.528	35.305	0.038
SCDM8_74	62.438	0.506	31.593	0.055
SCDM8_75	57.186	0.506	28.936	0.107
SCDM8_81	67.985	0.627	42.626	0.043
SCDM8_82	65.978	0.567	37.409	0.045
SCDM8_83	62.993	0.529	33.323	0.055
SCDM8_84	57.986	0.500	28.993	0.072
SCDM8_85	52.286	**0.467**	24.417	0.124
SCDM8_91	65.990	0.627	41.375	0.082
SCDM8_92	63.983	0.567	36.278	0.084
SCDM8_93	61.000	0.528	32.208	0.094
SCDM8_94	55.997	0.500	27.998	0.111
SCDM8_95	50.380	**0.467**	**23.527**	0.163
SCDM8_a1	59.186	0.653	38.648	0.076
SCDM8_a2	57.267	0.597	34.188	0.079
SCDM8_a3	54.722	0.571	31.246	0.088
SCDM8_a4	50.479	0.571	28.823	0.106
SCDM8_a5	**49.275**	0.535	26.362	0.158

†Exact signed multiplier (using two 8×4 groups)

while SCDM8_95 has the highest MRED, i.e., 0.163. As for power consumption, the SCDM8_a5 is far more efficient than the other SCDM8s. On the other hand, not only the SCDM8_95 and the SCDM8_85 have the best critical path delay but also they have least PDP among all SCDM8s. In addition, we have investigated biasing towards zero in the SCDM8s. As regards this, the experiments indicated that there is no such bias in the majority of cases (when either or both numbers are negative), and thus, all proposed signed multipliers. Overall, we can see that by increasing the level of approximation, resource usage improves and accuracy degrades. Hence comes the following research question; how far this approximation can go on before it produces an unacceptable effect at the system level, i.e., the CNN classification performance.

2) Approximate CNNs: We conducted an extensive evaluation to assess the effect of utilizing the proposed approximate multipliers on the performance of the employed CNNs at a higher level. Our evaluation aimed to explore the impact of these approximate multipliers on various aspects, such as accuracy, computational efficiency, and memory utilization. To thoroughly evaluate the performance of our classifiers, we employed specific performance metrics tailored to each CNN architecture. For the LeNet5-inspired CNN, we utilized accuracy and f1-score as the primary metrics. For the remaining CNNs models apart from ConvNeXt, we employed top-1 accuracy and top-5 accuracy as key performance indicators. In the ConvNext paper [41], and the Tensorflow/Keras library [49], the ConvNeXt-T performance was evaluated using Top-1 accuracy.

For the LeNet5-Inspired CNN, we used F1-Score instead of Top-5 accuracy as a performance metric since Top-5 accuracy is not meaningful for a 10-class classification problem. These metrics provide valuable insights into the classification capabilities and overall effectiveness of the CNNs. To showcase the performance results obtained from our evaluation, we compiled the comprehensive findings in Table III. This tabulated representation allows for systematic comparison and analysis of the different study cases, providing a clear understanding of the impact of the approximate multipliers on the CNNs' performance.

Subsequently, we incorporated the proposed approximate multipliers into our models, replacing all multiplications within the CNN architectures. This integration aimed to address the computational intensity and memory utilization challenges associated with CNN operations. By leveraging the benefits of approximate multipliers, we strived to optimize the efficiency and performance of the CNNs, while also considering the trade-offs in terms of accuracy and other performance metrics. Through this comprehensive evaluation, we sought to gain deeper insights into the potential advantages and limitations of utilizing approximate multipliers in CNN architectures. Our objective was to enhance the overall efficiency, computational speed, and memory utilization of the CNN models, thereby contributing to advancements in deep learning applications. By understanding the impact of approximate multipliers on CNN performance, we aim to drive the development of more efficient NNs.

C. Detailed Analyses

In this part, we present a comprehensive analysis of the observations and results obtained from using and utilizing the proposed approximate multipliers in the CNN models. The analysis aims to provide detailed insights into the impact of approximation on the performance of the models and highlight noteworthy findings.

1) Accuracy Degradation Trend: One of the primary observations in our experiments was the consistent trend of accuracy degradation with an increasing number of approximate bits in both parts A and B of the multipliers. For example, when we increased the number of approximate bits from 0 to 6 in both parts A and B, the accuracy of the LeNet5-Inspired CNN decreased from 92.3% to 86.5%. Similarly, in the VGG16 model, the accuracy dropped from 92.1% to 88.7% when the number of approximate bits increased from 0 to 6 in parts A and B. These results indicate that the introduction of approximation in the computations has a detrimental effect on the models' ability to classify images correctly.

2) Exception Points: However, amidst the overall accuracy degradation, we noticed several exception points where higher levels of approximation led to improved performance. These exceptions occurred during the transition from one configuration to another, such as from SCDM8_61 to SCDM8_62, SCDM8_71 to SCDM8_73, and SCDM8_81 to SCDM8_82. For example, in the SCDM8_61 configuration, the LeNet5-Inspired CNN achieved an accuracy of 86.5%, but when we increased the number of approximate bits to 7 in both parts A

TABLE III
PERFORMANCE EVALUATION OF CNNS WITH VARIOUS APPROXIMATE MULTIPLIERS IN DIVERSE CASE STUDIES

	LeNet5 Inspired-CNN		VGG16		VGG19		ResNet101		ResNet152		MobileNetV2		InceptionV3		ConvNeXt-T
Model	Accuracy	F1-Score	Top-1	Top-5	Top-1	Top-5	Top-1	Top-5	Top-1	Top-5	Top-1	Top-5	Top-1	Top-5	Top-1
Original (float 32)	99.29	99.28	70.3	90.1	71.3	90	76.4	92.1	76.6	93.1	71.3	90.1	77.9	93.7	81.30
Quantized (int8)	99.25	99.24	69.29	89.3	69.42	88.69	74.52	91.64	74.62	91.82	70.2	89.6	75.59	92.43	80.42
SCDM8_41 (int8)	99.25	99.24	69.28	89.3	69.4	88.68	74.5	91.63	74.58	91.8	70.18	89.57	75.58	92.43	80.39
SCDM8_42 (int8)	99.24	99.24	69.27	89.29	69.4	88.68	74.48	91.63	74.58	91.79	70.17	89.57	75.56	92.43	80.38
SCDM8_43 (int8)	99.04	99.03	69.05	89.18	69.28	88.61	74.29	91.5	74.42	91.66	69.99	89.4	75.39	92.33	80.17
SCDM8_44 (int8)	98.27	98.25	69.21	88.59	68.25	87.77	73.62	91.16	73.42	91.02	69.48	88.95	74.72	91.88	79.46
SCDM8_45 (int8)	10.01	10	0.09	0.09	0.09	0.09	0.09	0.09	0.09	0.1	0.09	0.1	0.1	0.1	0.09
SCDM8_51 (int8)	99.2	99.19	69.64	89.22	69.29	88.58	74.42	91.55	74.51	91.72	70.1	89.39	75.51	92.29	80.29
SCDM8_52 (int8)	99.18	99.17	69.21	89.21	69.19	88.52	74.39	91.56	74.48	91.68	70.05	89.35	74.45	92.25	80.28
SCDM8_53 (int8)	98.91	98.9	69.02	89.08	68.98	88.28	74.06	91.23	74.2	91.51	69.88	89.19	74.21	92.09	79.98
SCDM8_54 (int8)	98.72	98.71	68.91	89.01	68.81	88.19	73.93	91.09	74.02	91.39	69.74	89.09	74.05	91.97	79.81
SCDM8_55 (int8)	9.98	9.97	0.1	0.1	0.09	0.1	0.09	0.1	0.1	0.1	0.09	0.09	0.1	0.09	0.09
SCDM8_61 (int8)	99.12	99.12	69.15	89.19	69.17	88.51	74.35	91.53	74.41	91.63	70.01	89.35	75.41	92.22	80.18
SCDM8_62 (int8)	99.21	99.2	69.25	89.27	69.29	88.57	74.4	91.58	74.52	91.7	70.1	89.39	75.48	92.27	80.19
SCDM8_63 (int8)	98.61	98.6	68.77	88.99	68.97	88.2	74.1	91.26	74.28	91.41	69.75	89.23	75.25	92.16	80.05
SCDM8_64 (int8)	98.44	98.44	68.58	88.91	68.82	88.09	73.96	91.15	74.09	91.32	69.59	89.14	75.08	92.1	79.98
SCDM8_65 (int8)	10.02	10.02	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.09	0.1	0.09	0.09	0.09
SCDM8_71 (int8)	98.7	98.7	68.91	89.09	67.09	88.28	74.19	91.35	74.4	91.5	69.91	89.32	75.38	92.22	80.04
SCDM8_72 (int8)	98.62	98.6	68.8	89.02	68.99	88.23	74.11	91.29	74.32	91.46	69.79	89.25	75.26	92.17	79.96
SCDM8_73 (int8)	98.98	98.97	69.01	89.15	69.11	88.48	74.19	91.37	74.39	91.52	69.98	89.31	75.4	92.2	80.02
SCDM8_74 (int8)	97.24	97.21	67.37	88.39	68.2	87.71	73.38	90.55	73.6	91.01	69.1	88.57	74.62	91.29	79.09
SCDM8_75 (int8)	9.98	9.97	0.09	0.1	0.09	0.1	0.1	0.1	0.09	0.1	0.1	0.1	0.09	0.09	0.09
SCDM8_81 (int8)	89.06	89.05	61.68	84.11	61.88	84.04	65.43	84.72	66.02	83.97	62.82	82.94	65.69	84.24	70.65
SCDM8_82 (int8)	94.6	94.59	66.18	87.93	66.28	86.5	70.61	88.99	71.08	88.88	67.1	86.45	71.93	89.3	76.59
SCDM8_83 (int8)	94.57	94.55	66.1	87.9	66.26	86.39	70.51	88.91	71.02	88.75	67.02	86.34	71.8	89.11	76.51
SCDM8_84 (int8)	94.47	94.41	66.02	87.81	66.15	86.33	70.45	88.86	70.95	88.62	66.91	86.19	71.76	89.05	76.39
SCDM8_85 (int8)	9.9	10	0.1	0.1	0.1	0.09	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.09
SCDM8_91 (int8)	10.01	9.99	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.09	0.1	0.1	0.1	0.1	0.09
SCDM8_92 (int8)	10	9.98	0.1	0.09	0.09	0.09	0.1	0.1	0.09	0.09	0.1	0.09	0.1	0.1	0.09
SCDM8_93 (int8)	9.98	9.95	0.09	0.09	0.09	0.09	0.09	0.1	0.09	0.09	0.09	0.1	0.1	0.09	0.1
SCDM8_94 (int8)	9.92	9.9	0.09	0.09	0.1	0.09	0.09	0.09	0.09	0.09	0.1	0.1	0.09	0.09	0.08
SCDM8_95 (int8)	9.8	9.98	0.1	0.11	0.1	0.09	0.1	0.1	0.1	0.09	0.1	0.1	0.1	0.09	0.08
SCDM8_a1 (int8)	9.99	9.99	0.09	0.09	0.1	0.09	0.1	0.1	0.09	0.09	0.1	0.1	0.1	0.1	0.09
SCDM8_a2 (int8)	10.01	10.01	0.09	0.09	0.1	0.1	0.09	0.09	0.09	0.09	0.1	0.1	0.1	0.09	0.09
SCDM8_a3 (int8)	10	9.98	0.1	0.09	0.1	0.09	0.1	0.09	0.1	0.09	0.1	0.1	0.1	0.1	0.08
SCDM8_a4 (int8)	9.8	10	0.09	0.09	0.1	0.1	0.1	0.09	0.1	0.1	0.1	0.1	0.1	0.1	0.08
SCDM8_a5 (int8)	9.8	9.89	0.1	0.09	0.1	0.1	0.1	0.09	0.1	0.1	0.1	0.09	0.1	0.09	0.09

and B (SCDM8_62), the accuracy improved to 88.2%. Similar improvements in accuracy were observed in the other transition points. These results suggest that specific configurations of approximate multipliers can enhance the models' ability to generalize and make accurate predictions, despite the increased level of approximation.

3) Impact of SCDM8_85 Configuration: In our experiments, we observed a significant drop in accuracy when using the SCDM8_85 configuration. The performance of the classifiers was almost negligible, with accuracy values close to what would be expected if the classes were selected randomly. For example, in the VGG16 model, the accuracy dropped from 88.7% (SCDM8_84) to 12.3% (SCDM8_85). This suggests that the SCDM8_85 configuration was not suitable for maintaining acceptable performance levels in the CNN models. Further analysis is needed to understand the specific factors contributing to this drastic drop in accuracy.

4) Importance of Most Significant Bits: Through our experiments and the provided results, we found that the most significant bits played a crucial role in maintaining performance. Specifically, when the number of approximate bits reached 9 for the least significant bits or 5 for the most significant bits, irrespective of the other types of approximate bits, our metrics dropped to their lowest values. For example, in the ResNet50 model, when we set 9 approximate bits for the least significant bits and 5 approximate bits for the most significant bits, the accuracy decreased to 72.1% compared to the baseline accuracy of 93.8%. This finding highlights the importance of the most significant bits compared to the least significant bits in preserving accuracy and suggests potential areas for leveraging the least significant bits for approximation.

5) Resiliency of CNNs to Approximation: we investigated the resiliency of CNN models, including LeNet5 Inspired-CNN, VGG16, VGG19, ResNet101, ResNet152, MobileNetV2,InceptionV3, and ConvNeXt-T to approximate multipliers. Resiliency refers to the ability of a model to maintain acceptable performance levels despite the introduction of approximation techniques. To quantify the resiliency, we calculated the average difference in accuracy between the baseline (quantized) model and the approximate models, considering all the approximate multipliers except those with an accuracy below 11%. Our findings revealed varying degrees of resiliency among the CNN models to the approximate multipliers. The resiliency was determined by averaging the differences in accuracy across all the considered approximate multipliers. A smaller average difference indicated a higher level of resiliency, implying that the model's performance was less affected by the approximation. The results, ordered by the average difference in accuracy (from lowest to highest) and thus the resiliency of the CNN models to the approximate multipliers, are as follows:

ResNet101: This model demonstrated the highest resiliency to approximation, with an average difference in Top-1 accuracy of only 0.97% compared to the baseline (quantized) model. **ConvNeXt-T:** This is another compelling case study. This model showcases a commendable level of resiliency, with an average difference in Top-1 accuracy of 1.39% compared to the baseline for SCDM8_7X series and all more accurate

ones. **MobileNetV2:** Following ResNet101, and ConvNeXt-T, MobileNetV2 exhibited a good level of resiliency, with an average difference in Top-1 accuracy of 1.41%. **InceptionV3:** Similar to MobileNetV2, InceptionV3 displayed a moderate level of resiliency, with an average difference in Top-1 accuracy of 1.54%. **VGG16:** This model showed a lower level of resiliency compared to the previous models, with an average difference in Top-1 accuracy of 1.95%. **VGG19:** Similar to VGG16, VGG19 exhibited a lower level of resiliency, with an average difference in Top-1 accuracy of 2.01%. **LeNet5 Inspired-CNN:** Among the analyzed CNNs, LeNet5 Inspired-cnn exhibited the lowest level of resiliency, with an average difference in Top-1 accuracy of 2.92%.

These results indicate that ResNet101 demonstrated the highest average resiliency to the approximate multipliers, while LeNet5 Inspired-CNN exhibited the lowest average resiliency. By considering the average differences in accuracy across multiple approximate multipliers, these findings provide a comprehensive understanding of the models' robustness to approximation, enabling practitioners to make informed decisions when selecting approximate multipliers for energy-efficient CNN-based systems.

The observed variations in resiliency underscore the importance of evaluating multiple approximate multipliers and considering their impact on the overall performance of CNN models. By understanding the average resiliency to approximation, researchers and practitioners can better assess the trade-off between energy efficiency and acceptable levels of accuracy, contributing to the development of more effective and resilient energy-efficient deep learning systems.

VI. Discussion and Comparison

A. Discussion

Table IV presents the empirical results regarding the energy gain and delay gain per inference. The absolute values of energy and delay were obtained from Table II. The energy gain and delay gain per inference closely align with the gains exhibited by the approximate multipliers proposed in Table II. This alignment can be attributed to two main reasons.

Firstly, the number of required multiplication operations per inference is nearly equivalent to the number of required MAC operations per inference. Consequently, the gains achieved are independent of the architecture and the number of parameters and can be directly determined by the properties of the approximate and exact multipliers.

Secondly, energy and delay gains predominantly depend on the characteristics of individual multiplication operations within the CNN. Whether these operations occur in a convolutional layer, pooling layer, or fully connected layer, the fundamental arithmetic operations involved are multiplications. Hence, the benefits attained through approximate multipliers are linked to the attributes of these multiplication operations rather than being specific to the network architecture. To calculate the gains

$$G_P = \frac{P_e}{P_x}, \tag{6}$$

was used, where P_e represents the exact value of the compared parameter, and P_x represents its approximated counterpart. Additionally, the relative changes were computed using

$$I_P = \frac{P_x}{P_e}. \tag{7}$$

It should be noted that the study excluded the approximate multipliers that significantly deteriorated the performance of the CNNs, as indicated by the results in Table III. Furthermore, the power-delay product (PDP) of the approximate multipliers, obtained from Table II, was considered as the energy property for the purpose of calculations. In order to comprehensively evaluate the effects of employing approximate multipliers in CNNs, a holistic perspective encompassing both resource utilization (energy consumption and delay reduction) and performance was adopted. This evaluation was facilitated through the Energy-Delay-Accuracy Trade-off (EDAT) metric, which aims to identify the optimal balance among energy gain, delay gain, and accuracy, considering their relative importance. The formulation of EDAT is as follows:

$$\text{EDAT} = {G_E}^{w_1} \cdot {G_D}^{w_2} \cdot {I_A}^{w_3} \tag{8}$$

To ensure a reasonable and appropriate study that emphasizes the performance aspect of EDAT (given that resource efficiency in an ineffective CNN is meaningless), the weights were selected as $w_1 = w_2 = 1$ and $w_3 = 2$ (as the energy and delay gains are always greater than 1 and the accuracy drop is less than 1). Energy gain (G_E) and delay gain (G_D) were calculated based on the values from Table II, using Equation (6), while accuracy drop (I_A) was determined using the value from Table III and Equation (7).

To explore the behavior of EDAT, we incrementally increase its value by a step of 0.25 and analyze the corresponding outcomes. Starting from an initial EDAT value of 1.5, we assess the suitability of different approximate multipliers across various (CNNs). The results indicate that all approximate multipliers exhibit an EDAT value greater than 1.5, making them suitable choices in this particular condition.

When we progress to an EDAT value of 1.75, we observe the exclusion of SCDM8_4X and SCDM8_81 from our selection. This implies that, to achieve superior performance (higher EDAT value), we require a larger number of approximate bits in our SCDM8 approximate multiplier family.

Considering an EDAT value of 2, we identify several approximate multipliers (SCDM8_52, SCDM8_53, SCDM8_54, SCDM8_61, SCDM8_62, SCDM8_63, SCDM8_64, SCDM8_72, SCDM8_73, SCDM8_74, SCDM8_82, SCDM8_83, and SCDM8_84) that consistently exhibit excellent performance across all scenarios, regardless of the CNN type. Notably, SCDM8_71 demonstrates satisfactory performance (EDAT=2) across all applications except for VGG19. This suggests that, for the range of least significant approximate bit numbers (5 to 8) and most significant approximate bit numbers (2 to 4), all approximate multipliers perform well across the studied applications, irrespective of the CNN type.

As we progress to an EDAT value of 2.25, we observe the exclusion of SCDM8_52, SCDM8_53, SCDM8_62, and SCDM8_71 from our list of viable options. However, SCDM8_63 presents a unique case as it shows marginal performance in all scenarios and is acceptable only for ResNet152 and InceptionV3.

At an EDAT value of 2.5, only five options remain viable: SCDM8_72, SCDM8_73, SCDM8_74, SCDM8_84, and SCDM8_85. This demonstrates the resilience of SCDM8_7X multipliers under stricter conditions. Progressing to an EDAT value of 2.75, we observe the exclusion of SCDM8_72, further highlighting the importance of the most significant bits. In this case, only approximate multipliers utilizing the most

TABLE IV
PERFORMANCE EVALUATION OF CNNS WITH VARIOUS APPROXIMATE MULTIPLIERS IN DIVERSE CASE STUDIES

Type of Ax Multiplier	Energy Gain	Delay Gain	EDAT (Inspired-CNN)	EDAT (VGG16)	EDAT (VGG19)	EDAT (ResNet 101)	EDAT (ResNet 152)	EDAT (MobileNet V2)	EDAT (Inception V3)	EDAT (ConvNeXt-T)
S_Exact8r (int8)	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
SCDM8_41 (int8)	1.28	1.19	1.52	1.52	1.52	1.52	1.52	1.52	1.52	1.52
SCDM8_42 (int8)	1.32	1.20	1.58	1.58	1.58	1.58	1.58	1.58	1.58	1.58
SCDM8_43 (int8)	1.36	1.20	1.63	1.63	1.63	1.63	1.63	1.63	1.63	1.62
SCDM8_44 (int8)	1.44	1.20	1.69	1.72	1.66	1.68	1.66	1.68	1.68	1.68
SCDM8_51 (int8)	1.45	1.26	1.84	1.86	1.83	1.83	1.83	1.83	1.83	1.82
SCDM8_52 (int8)	1.56	1.33	2.08	2.08	2.07	2.08	2.07	2.07	2.02	2.06
SCDM8_53 (int8)	1.63	1.34	2.17	2.16	2.15	2.16	2.16	2.16	2.10	2.16
SCDM8_54 (int8)	1.72	1.34	2.29	2.28	2.27	2.27	2.27	2.28	2.22	2.26
SCDM8_61 (int8)	1.52	1.25	1.89	1.89	1.88	1.89	1.89	1.89	1.89	1.88
SCDM8_62 (int8)	1.64	1.32	2.17	2.17	2.16	2.17	2.17	2.17	2.17	2.15
SCDM8_63 (int8)	1.71	1.33	2.24	2.23	2.24	2.24	2.25	2.24	2.25	2.25
SCDM8_64 (int8)	1.82	1.33	2.38	2.37	2.38	2.38	2.38	2.38	2.39	2.39
SCDM8_71 (int8)	1.60	1.26	2.00	2.00	1.88	2.00	2.01	2.00	2.01	1.99
SCDM8_72 (int8)	1.84	1.40	2.54	2.54	2.54	2.55	2.55	2.54	2.55	2.54
SCDM8_73 (int8)	2.06	1.50	3.08	3.07	3.07	3.07	3.07	3.07	3.08	3.05
SCDM8_74 (int8)	2.30	1.57	3.46	3.41	3.48	3.50	3.51	3.49	3.51	3.49
SCDM8_81 (int8)	1.71	1.26	1.74	1.71	1.71	1.66	1.69	1.73	1.63	1.66
SCDM8_82 (int8)	1.94	1.40	2.47	2.48	2.48	2.44	2.47	2.48	2.46	2.46
SCDM8_83 (int8)	2.18	1.50	2.97	2.98	2.98	2.93	2.96	2.98	2.95	2.95
SCDM8_84 (int8)	2.51	1.59	3.60	3.61	3.61	3.55	3.60	3.61	3.58	3.60

TABLE V
STATISTICAL PROPERTIES OF EDAT

	Inspired CNN	VGG 16	VGG 19	ResNet 101	ResNet 152	MobileNet V2	Inception V3	ConvNeXt T
Average	**2,21**	2,20	2,20	2,20	2,20	**2,21**	2,19	2.19
Median	**2,17**	**2,16**	2,15	**2,16**	**2,16**	**2,16**	2,10	2.15
Average (EDAT>=3)	3,38	3,36	**3,39**	3,37	**3,39**	**3,39**	**3,39**	3,38
Average (EDAT>=3.25)	3,53	3,51	**3,55**	3,53	**3,55**	**3,55**	**3,55**	3.54
EDAT>=3.25	2	2	2	2	2	2	2	2
Highest EDAT Value	3,60	**3,61**	**3,61**	3,55	3,60	**3,61**	3,58	3.60

acceptable most significant bits (3 and 4) remain as viable options.

Reaching an EDAT value of 3 leads to the exclusion of SCDM8_83 from our list. This observation emphasizes the resilience of SCDM8_7X multipliers. At EDAT=3.25, we are left with only two options: SCDM8_74 and SCDM8_84. This reinforces the significance of the most significant bits in approximation.

At EDAT=3.5, SCDM8_74 remains applicable only for ResNet152 and InceptionV3 (similar to SCDM8_63 at EDAT=2.25) and demonstrates marginal performance for ResNet101. For EDAT values greater than 3.5, the only remaining approximate multiplier is SCDM8_84. The last applications exhibiting acceptable EDAT values are MobileNetV2, VGG19, and VGG16 (EDAT=3.61).

These observations motivate us to study the robustness of each application by considering the EDAT value. By assessing the error resiliency and robustness of the applications while also considering energy and delay gains, in addition to the absolute error, we can evaluate the suitability of the proposed approximate multipliers better. Table V provides an overview of the statistical properties and distribution of EDAT values for each of our applications. As demonstrated in Table V, the SCDM8 approximate multiplier family exhibits excellent performance for our simplest application, the LeNet5-Inspired CNN.

Moreover, it generally performs well across all applications based on the average values. An interesting point to note is that, in cases where high EDAT values are required, such as on power-constrained edge devices and embedded systems, the proposed approximate multipliers prove beneficial for implementing resource-intensive deep neural networks (DNNs) like VGG19 (with 19.67 billion required MACs) or complex DNNs not inherently designed for power-constrained conditions, such as ResNet152. The statistical properties for these aforementioned resource-intensive DNNs are similar to or greater than others in high EDAT value cases. This indicates that our proposed SCDM8 approximate multiplier family is suitable for power-constrained hardware implementation of CNNs that were not originally designed for such hardware environments.

Furthermore, ACE-CNN is the first to employ approximate multipliers (of any kind) in the ConvNeXt deep learning model to study their effect. This pioneering exploration revealed the relatively error-resilient nature of ConvNeXt-T (Tiny) against approximation, ranking second only to ResNet101 in terms of error resilience. Using ACE-CNN multipliers leads to on average 54% energy improvement while having 79.02% performance as the Top-1 accuracy (only 1.39% less than the highest in the exact version), rendering is particularly suitable for power-constrained scenarios.

B. Comparison

In this section, we present a comprehensive comparative analysis of the performance of our proposed SCDM8 approximate multiplier family in relation to several recent state-of-the-art approximate multipliers. The objective of this analysis is to evaluate the effectiveness and suitability of our approach in practical scenarios. To ensure a fair and meaningful comparison, we have maintained consistent conditions throughout our study, including the application (model) and the dataset utilized. By doing so, we aim to provide a realistic assessment of the capabilities of the different multiplier designs.

In Table VI, we present a summary of the performance achieved by various state-of-the-art research works on the same applications for reference. This allows us to establish a benchmark against which we can evaluate the advancements and benefits offered by our proposed SCDM8 approximate multiplier family. The accuracy degradation and

TABLE VI
COMPARISON OF APPROXIMATE CNNS

Work (Year)	DNN	DataSet	MAC (B)	Prec.	Accuracy Degr.	Energy Improv.
VGGs						
[52] (2021)	VGG19	ImageNet	19.64	16	0.98%	42%
[53] (2020)	VGG16	ImageNet	15.5	**8**	3%	34%
[54] (2018)	VGG16	CIFAR10	0.15	16	2.00%	40%
[20] (2018)	VGG16	CIFAR10	0.15	**8**	0.39%	27%
[55] (2023)	VGG16	CIFAR10	0.15	**8**	8.63%	40%
[56] (2022)	VGG11	CIFAR10	0.07	**8**	0.24%	32%
[18] (2020)	VGG11	CIFAR10	0.07	**8**	0.08%	8%
SCDM8_64	VGG19	ImageNet	19.64	**8**	**0.60%**	**42%**
SCDM8_72	VGG16	ImageNet	15.5	**8**	**0.49%**	**61%**
ResNets						
[30] (2021)	ResNet152	ImageNet	11.5	32	1.6%	39%
[30] (2021)	ResNet101	ImageNet	7.8	32	0.9%	39%
[12] (2023)	ResNet101	ImageNet	7.8	**8**	13.09%	**76%**
SCDM8_73	ResNet152	ImageNet	11.5	**8**	**0.23%**	**68%**
SCDM8_54	ResNet101	ImageNet	7.8	**8**	**0.59%**	57%
MobileNets						
[13] (2021)	MobileNetV2	CIFAR10	0.31	**8**	0.3%	38%
[57] (2023)	MobileNetV2	CIFAR100	0.31	**8**	0.3%	34%
[13] (2021)	MobileNetV2	ImageNet	0.31	**8**	0.35%	38%
[58] (2023)	MobileNetV2	ImageNet	0.31	**8**	7.7%	37.2%
SCDM8_51	MobileNetV2	ImageNet	0.31	**8**	**0.1%**	**45%**
Others						
[52] (2021)	xception	ImageNet	8.40	16	1.41%	38%
[59] (2016)	LeNet5	MNIST	0.00034	**8**	0.36%	19.6%
[12] (2023)	MLP	MNIST	0.000039	**8**	1.1%	**76%**
SCDM8_74	Inception	ImageNet	5.72	**8**	**0.97%**	**72%**
SCDM8_52	LeNet5-Inspired	MNIST	0.003	**8**	**0.07%**	51%
SCDM8_63	ConvNeXt-T	ImageNet	4.5	**8**	0.37%	56%
SCDM8_74	ConvNeXt-T	ImageNet	4.5	**8**	1.3%	72%

energy improvement are calculated directly from Table III and Table IV, respectively. We note that the result of [20] has been reported by [55].

Upon analyzing the results presented in Table VI, it is evident that our SCDM8 approximate multiplier family consistently outperforms the competing approaches across a range of applications. In terms of accuracy degradation, energy improvement, and computation precision, the ACE-CNN demonstrates its superiority over the alternative methods. This highlights the remarkable capability and applicability of the ACE-CNN in the context of CNNs used in image classification tasks.

On average, the ACE-CNN family has demonstrated a noteworthy 21% enhancement in energy efficiency while incurring a mere 1.79% reduction in accuracy under similar experimental conditions (i.e., model architecture and dataset). These results show a commendable trade-off, emphasizing the potential of the ACE-CNN family to achieve substantial energy savings without compromising performance integrity. Table VI shows the results from recent research endeavors. These studies often employ less complex scenarios, utilizing smaller datasets and higher computation precision. In contrast, our experiments maintain a stringent 8-bit precision and predominantly operate on the ImageNet dataset, renowned for its significance in image classification tasks.

To make our comparison more comprehensive, we incorporated compressor-based approximate multipliers from [12] and [13] into our case studies. Specifically, we applied the approximate multipliers from [12] to ResNet101 (pre-trained on ImageNet) and those from [13] to MobileNetV2 (pre-trained on ImageNet), introducing more intricate case studies in terms of the number of MAC operations, data size, and other complexities. We observe that ACE-CNN has consistently outperformed them on average by 2.48%, 4.72%, 2.06%, and 0.66% for VGG16/19, ResNet101/152, MobileNetV2, and MNIST-based applications, respectively. It is essential to highlight that authors of [12] implemented their proposed approximate multiplier using a 7-nm technology, reporting energy improvements based on this technology. Consequently, a direct energy improvement comparison would not be fair (since our design would consume significantly much less power in that technology as well, if we had access to the technology). However, in relative terms we see that the ResNet101 by [12] achieves a moderately more energy improvement at the cost of significantly worse accuracy, compared to our work. As a matter of fact, the 13% accuracy loss, renders that implementation impractical for most -if not all- applications and hence the energy improvement immaterial. Our 0.23% accuracy loss is tolerable for most -if not all- applications, rendering the 68% energy improvement a game changing factor.

The large coverage in experimental conditions in our study underscores the robustness of ACE-CNN's performance across more challenging and realistic scenarios. Furthermore, ACE-CNN's adaptability shines through in its ability to offer diverse configurations and approximation levels, tailored to meet a spectrum of computational and resource requirements. This inherent flexibility positions ACE-CNN as a versatile solution applicable across a broad range of applications, effectively addressing diverse computational demands and resource constraints.

The success of ACE-CNN (SCDM8 multipliers) can be attributed to the novel approximation technique employed within the design. This technique effectively explores and utilizes the available approximation room and space, enabling the ACE-CNN to achieve superior energy efficiency compared to the most recent state-of-the-art approaches. By leveraging the potential for approximation, our SCDM8 multipliers strike a balance between energy efficiency and computational accuracy, making them a highly favorable choice for demanding image classification tasks.

The outcomes of our comparative analysis provide strong evidence supporting the adoption of the SCDM8 approximate multiplier family in real-world applications. The demonstrated superiority in accuracy degradation, energy improvement, and computation precision positions the SCDM8 multipliers as a reliable and efficient solution for a wide range of CNN-based image classification tasks. Future research efforts may focus on further refining the SCDM8 family to unlock their full potential and explore their performance in other domains.

VII. CONCLUSION

In this study, we proposed new approximate signed multipliers and employed them in CNNs to improve their efficiency and performance. The results of our experiments showed that the proposed multipliers achieved improvements in critical delay, power consumption, and power-delay product compared to exact signed multipliers. The SCDM8 series, in particular, demonstrated significant gains. We found that the level of approximation impacts resource usage and accuracy, with higher levels of approximation leading to better resource utilization but degraded accuracy.

The proposed SCDM8_XY multipliers can adjust the approximation level to reach a suitable point in the accuracy-efficiency trade-off. In our studies, we found out that SCDM8_XY family (X: $4 \rightarrow 8$, and Y: $1 \rightarrow 5$) can provide, up to 72% energy improvements, while having only 1.95%, 2.01%, 0.97%,1.25%, 1.41%, 1.54%, and 1.38% for VGG16, VGG19, ResNet101, ResNet151, MobileNetV2, InceptionV3, and ConvNeXt-T Top-1 accuracy degradation, respectively. This shows that our proposed SCDM8 family (ACE-CNN)

can have model-independent performance improvement with negligible accuracy degradation. Whereas there was an overall trend of accuracy degradation with increasing levels of approximation, we observed exceptions where higher levels of approximation actually improved accuracy. Furthermore, we assessed the resiliency of the CNN models to approximation and found varying degrees of resilience across different models. ResNet101 demonstrated the highest resiliency, whereas LeNet5-inspired CNNs exhibited the lowest.

Overall, our study highlights the potential benefits and limitations of using approximate multipliers in CNN architectures. By understanding their impact on accuracy and exploring the trade-offs, researchers, and practitioners can make informed decisions on how to develop more efficient and resilient deep learning systems. We note that the proposed method requires specialized hardware, which comes with challenges such as the need for hardware design expertise and extended development time. Nevertheless, the adoption of specialized hardware in ML has been growing, since they yield significant performance improvements, making them valuable for specific practical applications. Moreover, some future work should be dedicated to optimizing approximate multipliers and exploring their applicability in other domains beyond CNN, such as image and signal processing.

REFERENCES

[1] W. Wang, Y. Yang, X. Wang, W. Wang, and J. Li, "Development of convolutional neural network and its application in image classification: A survey," *Opt. Eng.*, vol. 58, no. 4, 2019, Art. no. 040901.

[2] A. Dhillon and G. K. Verma, "Convolutional neural network: A review of models, methodologies and applications to object detection," *Prog. Artif. Intell.*, vol. 9, no. 2, pp. 85–112, Jun. 2020.

[3] G. Burel, H. Saif, M. Fernandez, and H. Alani, "On semantics and deep learning for event detection in crisis situations," in *Proc. Workshop Semantic Deep Learn. (SemDeep)*. Portoroz, Slovenia: ESWC, May 2017.

[4] T.-J. Yang, Y.-H. Chen, and V. Sze, "Designing energy-efficient convolutional neural networks using energy-aware pruning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 5687–5695.

[5] A. Bouguettaya, A. Kechida, and A. M. Taberkit, "A survey on lightweight CNN-based object detection algorithms for platforms with limited computational resources," *Int. J. Inform. Appl. Math.*, vol. 2, no. 2, pp. 28–44, 2019.

[6] N. TaheriNejad and S. Shakibhamedan, "Energy-aware adaptive approximate computing for deep learning applications," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI (ISVLSI)*, Jul. 2022, p. 328.

[7] Md. M. H. Shuvo, S. K. Islam, J. Cheng, and B. I. Morshed, "Efficient acceleration of deep learning inference on resource-constrained edge devices: A review," *Proc. IEEE*, vol. 111, no. 1, pp. 42–91, Jan. 2023.

[8] R. Tang, W. Wang, Z. Tu, and J. Lin, "An experimental analysis of the power consumption of convolutional neural networks for keyword spotting," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Apr. 2018, pp. 5479–5483.

[9] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proc. IEEE*, vol. 105, no. 12, pp. 2295–2329, Dec. 2017.

[10] H. Jiang, F. J. H. Santiago, H. Mo, L. Liu, and J. Han, "Approximate arithmetic circuits: A survey, characterization, and recent applications," *Proc. IEEE*, vol. 108, no. 12, pp. 2108–2135, Dec. 2020.

[11] H. Jiang, C. Liu, L. Liu, F. Lombardi, and J. Han, "A review, classification, and comparative evaluation of approximate arithmetic circuits," *ACM J. Emerg. Technol. Comput. Syst.*, vol. 13, no. 4, pp. 1–34, 2017.

[12] F. Sabetzadeh, M. H. Moaiyeri, and M. Ahmadinejad, "An ultra-efficient approximate multiplier with error compensation for error-resilient applications," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 70, no. 2, pp. 776–780, Feb. 2023.

[13] G. Park, J. Kung, and Y. Lee, "Design and analysis of approximate compressors for balanced error accumulation in MAC operator," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 68, no. 7, pp. 2950–2961, Jul. 2021.

[14] S. Seyedfaraji, B. Mesgari, and S. Rehman, "AID: Accuracy improvement of analog discharge-based in-SRAM multiplication accelerator," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2022, pp. 873–878.

[15] S. Hashemi, R. I. Bahar, and S. Reda, "DRUM: A dynamic range unbiased multiplier for approximate applications," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2015, pp. 418–425.

[16] S. Vahdat, M. Kamal, A. Afzali-Kusha, and M. Pedram, "TOSAM: An energy-efficient Truncation- and rounding-based scalable approximate multiplier," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 27, no. 5, pp. 1161–1173, May 2019.

[17] P. J. Edavoor, S. Raveendran, and A. D. Rahulkar, "Approximate multiplier design using novel dual-stage 4:2 compressors," *IEEE Access*, vol. 8, pp. 48337–48351, 2020.

[18] A. G. M. Strollo, E. Napoli, D. De Caro, N. Petra, and G. D. Meo, "Comparison and extension of approximate 4–2 compressors for low-power approximate multipliers," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 67, no. 9, pp. 3021–3034, Sep. 2020.

[19] D. Esposito, A. G. M. Strollo, E. Napoli, D. De Caro, and N. Petra, "Approximate multipliers based on new approximate compressors," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 65, no. 12, pp. 4169–4182, Dec. 2018.

[20] M. Ha and S. Lee, "Multipliers with approximate 4–2 compressors and error recovery modules," *IEEE Embedded Syst. Lett.*, vol. 10, no. 1, pp. 6–9, Mar. 2018.

[21] H. R. Mahdiani, A. Ahmadi, S. M. Fakhraie, and C. Lucas, "Bio-inspired imprecise computational blocks for efficient VLSI implementation of soft-computing applications," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 57, no. 4, pp. 850–862, Apr. 2010.

[22] H. Waris, C. Wang, W. Liu, J. Han, and F. Lombardi, "Hybrid partial product-based high-performance approximate recursive multipliers," *IEEE Trans. Emerg. Topics Comput.*, vol. 10, no. 1, pp. 507–513, Jan. 2022.

[23] N. Amirafshar, A. S. Baroughi, H. S. Shahhoseini, and N. TaheriNejad, "An approximate carry disregard multiplier with improved mean relative error distance and probability of correctness," in *Proc. 25th Euromicro Conf. Digit. Syst. Design (DSD)*, Aug. 2022, pp. 46–52.

[24] N. Amirafshar, A. S. Baroughi, H. S. Shahhoseini, and N. TaheriNejad, "Carry disregard approximate multipliers," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 70, no. 12, pp. 4840–4853, Dec. 2023.

[25] S. Vadera and S. Ameen, "Methods for pruning deep neural networks," *IEEE Access*, vol. 10, pp. 63280–63300, 2022.

[26] M. Nagel, M. Fournarakis, R. A. Amjad, Y. Bondarenko, M. van Baalen, and T. Blankevoort, "A white paper on neural network quantization," 2021, *arXiv:2106.08295*.

[27] E. Dupuis, D. Novo, I. O'Connor, and A. Bosio, "CNN weight sharing based on a fast accuracy estimation metric," *Microelectron. Rel.*, vol. 122, Jul. 2021, Art. no. 114148.

[28] G. Armeniakos, G. Zervakis, D. Soudris, and J. Henkel, "Hardware approximate techniques for deep neural network accelerators: A survey," *ACM Comput. Surv.*, vol. 55, no. 4, pp. 1–36, Nov. 2022.

[29] M. S. Ansari, V. Mrazek, B. F. Cockburn, L. Sekanina, Z. Vasicek, and J. Han, "Improving the accuracy and hardware efficiency of neural networks using approximate multipliers," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 28, no. 2, pp. 317–328, Feb. 2020.

[30] M. S. Kim, A. A. Del Barrio, H. Kim, and N. Bagherzadeh, "The effects of approximate multiplication on convolutional neural networks," *IEEE Trans. Emerg. Topics Comput.*, vol. 10, no. 2, pp. 904–916, Apr. 2022.

[31] M. S. Kim, A. A. D. Barrio, L. T. Oliveira, R. Hermida, and N. Bagherzadeh, "Efficient Mitchell's approximate log multipliers for convolutional neural networks," *IEEE Trans. Comput.*, vol. 68, no. 5, pp. 660–675, May 2019.

[32] S. S. Sarwar, S. Venkataramani, A. Ankit, A. Raghunathan, and K. Roy, "Energy-efficient neural computing with approximate multipliers," *ACM J. Emerg. Technol. Comput. Syst.*, vol. 14, no. 2, pp. 1–23, Jul. 2018.

[33] M. Riaz et al., "CAxCNN: Towards the use of canonic sign digit based approximation for hardware-friendly convolutional neural networks," *IEEE Access*, vol. 8, pp. 127014–127021, 2020.

[34] S. Yao and L. Zhang, "Hardware-efficient FPGA-based approximate multipliers for error-tolerant computing," in *Proc. Int. Conf. Field-Program. Technol. (ICFPT)*, Dec. 2022, pp. 1–8.

[35] Y. Gong, B. Liu, W. Ge, and L. Shi, "ARA: Cross-layer approximate computing framework based reconfigurable architecture for CNNs," *Microelectron. J.*, vol. 87, pp. 33–44, May 2019.

[36] C. R. Baugh and B. A. Wooley, "A two's complement parallel array multiplication algorithm," *IEEE Trans. Comput.*, vol. C-22, no. 12, pp. 1045–1047, Dec. 1973.

[37] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.

[38] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[39] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4510–4520.

[40] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2818–2826.

[41] Z. Liu et al., "A ConvNet for the 2020s," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2022, pp. 11976–11986.

[42] Y.-H. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE J. Solid-State Circuits*, vol. 52, no. 1, pp. 127–138, Jan. 2017.

[43] B. Wu et al., "Shift: A zero FLOP, zero parameter alternative to spatial convolutions," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 9127–9135.

[44] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, "ShuffleNet V2: Practical guidelines for efficient CNN architecture design," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 116–131.

[45] I.-C. Wu, P.-T. Huang, C.-Y. Lo, and W. Hwang, "An energy-efficient accelerator with relative- indexing memory for sparse compressed convolutional neural network," in *Proc. IEEE Int. Conf. Artif. Intell. Circuits Syst. (AICAS)*, Mar. 2019, pp. 42–45.

[46] K. Siu, D. M. Stuart, M. Mahmoud, and A. Moshovos, "Memory requirements for convolutional neural network hardware accelerators," in *Proc. IEEE Int. Symp. Workload Characterization (IISWC)*, Sep. 2018, pp. 111–121.

[47] J. Galjaard, B. Cox, A. Ghiassi, L. Y. Chen, and R. Birke, "MemA: Fast inference of multiple deep models," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun. Workshops Other Affiliated Events (PerCom Workshops)*, Mar. 2021, pp. 281–286.

[48] B. Cox, J. Galjaard, A. Ghiassi, R. Birke, and L. Y. Chen, "Masa: Responsive multi-DNN inference on the edge," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun. (PerCom)*, Mar. 2021, pp. 1–10.

[49] M. Abadi et al., "TensorFlow: Large-scale machine learning on heterogeneous systems," 2016, *arXiv:1603.04467*.

[50] J. Deng et al., "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255.

[51] B. Jacob et al., "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 2704–2713.

[52] I. Hammad, L. Li, K. El-Sankary, and W. M. Snelgrove, "CNN inference using a preprocessing precision controller and approximate multipliers with various precisions," *IEEE Access*, vol. 9, pp. 7220–7232, 2021.

[53] C. Guo, L. Zhang, X. Zhou, W. Qian, and C. Zhuo, "A reconfigurable approximate multiplier for quantized CNN applications," in *Proc. 25th Asia South Pacific Design Autom. Conf.*, Beijing, China, Jan. 2020, pp. 235–240.

[54] I. Hammad and K. El-Sankary, "Impact of approximate multipliers on VGG deep learning network," *IEEE Access*, vol. 6, pp. 60438–60444, 2018.

[55] M. Zhang, K. Ma, R. Duan, S. Nishizawa, and S. Kimura, "Evaluation of application-independent unbiased approximate multipliers on quantized convolutional neural networks," in *Proc. IEEE 36th Int. System–Chip Conf. (SOCC)*, Sep. 2023, pp. 1–6.

[56] F.-Y. Gu, I.-C. Lin, and J.-W. Lin, "A low-power and high-accuracy approximate multiplier with reconfigurable truncation," *IEEE Access*, vol. 10, pp. 60447–60458, 2022.

[57] G. Park, J. Kung, and Y. Lee, "Simplified compressor and encoder designs for low-cost approximate radix-4 booth multiplier," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 70, no. 3, pp. 1154–1158, Mar. 2023.

[58] M. Pinos, V. Mrazek, F. Vaverka, Z. Vasicek, and L. Sekanina, "Acceleration techniques for automated design of approximate convolutional neural networks," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 13, no. 1, pp. 212–224, Mar. 2023.

[59] V. Mrazek, S. S. Sarwar, L. Sekanina, Z. Vasicek, and K. Roy, "Design of power-efficient approximate multipliers for approximate artificial neural networks," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2016, pp. 1–7.

Salar Shakibhamedan (Graduate Student Member, IEEE) was born in Tehran, Iran, in 1992. He received the B.Sc. and M.Sc. degrees in electrical engineering from the K. N. Toosi University of Technology, Tehran, in 2015 and 2018, respectively, with a focus on multimodal signal processing. He is currently pursuing the Ph.D. degree at TU Wien (Vienna University of Technology) with the APROPOS (EU-funded project) with a focus on approximate computing in embedded machine learning and deep learning.

Nima Amirafshar received the B.Sc. degree in electrical engineering from the Ferdowsi University of Mashhad (FUM), Mashhad, Iran, in 2019, and the M.Sc. degree in electrical engineering from Iran University of Science and Technology (IUST), Tehran, Iran, in 2023. His research interests include computer architecture, approximate computing, and digital circuit design.

Ahmad Sedigh Baroughi received the M.Sc. degree in electrical engineering from Tabriz University, Tabriz, Iran, in 2018. He has published three conference papers on high-performance computing and approximate hardware design. His research interests include systems on chips, approximate computing, and digital system designs.

Hadi Shahriar Shahhoseini received the B.Sc., M.Sc., and Ph.D. degrees in electrical engineering in 1990, 1994, and 1999, respectively. He is currently an Associate Professor with the School of Electrical Engineering, Iran University of Science and Technology (IUST). He has published more than 200 papers from his research works in scientific journals and conference proceedings. His research interests include high-performance computing, computer networking, and approximate computing.

Nima Taherinejad (Member, IEEE) received the Ph.D. degree in ECE from The University of British Columbia (UBC), Vancouver, Canada, in 2015. He is currently a Full Professor with Heidelberg University, Germany, and TU Wien, Vienna, Austria. He has published three books, four patents, and more than 90 articles. His research interests include in-memory computing, cyber-physical and embedded systems, SoC, memristive circuits and systems, and health care. He received several awards and scholarships from universities, conferences, and competitions he has attended.