

Received June 9, 2020, accepted June 22, 2020, date of publication July 9, 2020, date of current version July 22, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3008256

CxCNN: Towards the Use of Canonic Sign Digit Based Approximation for Hardware-Friendly Convolutional Neural Networks

MOHSIN RIAZ¹, REHAN HAFIZ¹, SALMAN ABDUL KHALIQ¹,
MUHAMMAD FAISAL¹, HAFIZ TALHA IQBAL¹, MOHSEN ALI¹,
AND MUHAMMAD SHAFIQUE², (Senior Member, IEEE)

¹Computer Engineering Department, Information Technology University, Lahore 54700, Pakistan

²Institute of Computer Engineering, Vienna University of Technology (TU Wien), 1040 Vienna, Austria

Corresponding author: Rehan Hafiz (rehan.hafiz@itu.edu.pk)

This work was supported in part by the HEC (NRPU Project), AxVision-Application-Specific and Data-Aware Approximate-Computing for Energy Efficient Image and Vision Processing Applications under Grant 10150.

ABSTRACT The design of hardware-friendly architectures with low computational overhead is desirable for low latency realization of CNN on resource-constrained embedded platforms. In this work, we propose CxCNN, a Canonic Sign Digit (CSD) based approximation methodology for representing the filter weights of pre-trained CNNs. The proposed CSD representation allows the use of multipliers with reduced computational complexity. The technique can be applied on top of state-of-the-art CNN quantization schemes in a complementary manner. Our experimental results on a variety of CNNs, trained on MNIST, CIFAR-10 and ImageNet datasets, demonstrate that our methodology provides CNN designs with multiple levels of classification accuracy, without requiring any retraining, and while having a low area and computational overhead. Furthermore, when applied in conjunction with a state-of-art quantization scheme, CxCNN allows the use of multipliers, which offer 77% logic area reduction, as compared to their accurate counterpart, while incurring a drop in Top-1 accuracy of just 5.63% for a VGG-16 network trained on ImageNet.

INDEX TERMS Convolution neural networks, dedicated accelerators, approximate computing, canonic sign digits.

I. INTRODUCTION AND RELATED WORK

Convolutional and Deep Neural Networks (DNNs) have achieved significant popularity in the artificial intelligence community for being particularly successful in challenging tasks such as handwritten digit recognition [1], object classification [2], image recognition [3], super-resolution [4]–[7] and autonomous driving [8]. Convolution layers of a DNN comprise of filters whose weights are learned during a *training phase* that typically involves backpropagation to minimize the classification error. The trained CNN is then utilized in the *inference phase* to perform the classification, recognition, or other tasks. Deep CNNs are thus characterized by a large number of compute-intensive convolution operations along-with enormous memory traffic and storage requirements [9]. GPUs have been particularly successful in

speeding up the inference and training phases of DNNs [10]. However, their high energy demand has resulted in a growing interest in the use of custom hardware designs, build using ASICs and FPGAs. In particular, the recent push towards emerging computing paradigms, such as Edge and Fog Computing, is pushing the boundaries of the extent of computations that can be realized on devices that are constrained with limited energy budget and are low on computational resources [11]–[14]. Recently, Approximate Computing is being employed to enable quality scalable designs on such resource-constrained devices by relaxing the bounds of precise computing and provide new opportunities for improving the area, energy, and performance efficiency of systems at the cost of reduced output quality [15], [16]. In the context of the realization of CNNs on such devices, researchers are particularly devising schemes to reduce the computational complexity of CNN accelerators by quantizing their filter weights. The reason is that the logic area and energy

The associate editor coordinating the review of this manuscript and approving it for publication was Mitra Mirhassani¹.

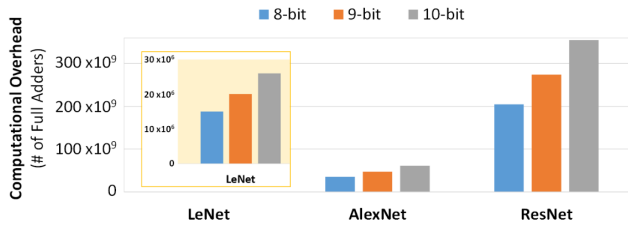


FIGURE 1. The figure shows the effect of bit-width scaling on the computational overhead of popular CNNs. Here *Computational Overhead* is defined as the number of full adders required to implement the convolution operation. A substantial increase in the computational overhead can be observed as we increase the bit-width. The inset provides a magnified view of computational overhead for LeNet.

requirements of a custom dedicated design scale with the *bit width* [17]. To demonstrate this, we evaluate the computational overhead of convolutional operations of a few popular CNN architectures for various bit-widths. For the sake of comparison, we define *computational overhead* as the number of full adders required to implement the convolution operations, assuming the use of Wallace trees [18] for the implementation of multipliers. Figure 1 plots the estimated *computational overhead* for LeNet [19], AlexNet [3] and ResNet-50 [20] for 8, 9 and 10 bit fixed point implementations. We note two observations. Firstly, the computational overhead increases substantially as the number of convolutional layers increase (ResNet-50 has 49 layers compared to 5 for AlexNet). Secondly, each additional bit in the bit-width of the design can cause a significant increase in computational overhead (30% for one additional design bit for ResNet-50). This increased computational complexity further adds to the area and latency overhead since each additional bit that is used to represent a filter weight results in the generation of an additional partial product that has to be added during the multiplication operation. Thus, a single-bit reduction in the bit-width of a fixed point CNN accelerator design can significantly reduce the area, compute energy, and latency.

A. STATE OF THE ART AND THEIR LIMITATIONS

A significant class of work related to the efficient implementation of CNNs aims at reducing the precision of convolutional filter weights to take benefit of arithmetic units with lower computational overhead [21], [22]. Binary $(-1, 1)$ and ternary $(-1, 0, 1)$ networks have been successfully explored in this context [23]–[25]; however, they require specialized retraining procedures to perform the backpropagation. Trinh et al. [26] proposed a 7-bit significant position encoding (SPE) scheme to achieve a 12.5 percent storage gain, as compared to an 8-bit fixed point binary representation. The technique made use of differential encoding and weight scaling to represent the filter weights using fewer non-zeros. However, due to the encoded nature of data representation, standard binary arithmetic was no more applicable for their computational units. In a recent effort, Gysel et al. [22] studied the effect of precision quantization, specifically in the context of dedicated architectures for CNNs. They proposed

three variants: a dynamic fixed-point representation, a mini float representation, and a multiplier-less scheme that utilized coefficients approximated with the power of 2. They observed that dynamic fixed-point representation coupled with retraining provides the best results. Note that all the quantization strategies mentioned above provide a *rigid* CNN model since the filter weights are required to be retrained for a particular quantization level. Thus, if it is needed to scale down the design to a lower energy mode, complete retraining has to be performed at a lower quantization level. Consequently, a separate corresponding set of weights has to be stored. This limitation makes the state-of-art schemes less favorable for quality-scalable accelerators that require configurable quality modes [27], [28] to exploit performance/area trade-off for resource-constrained devices. Thus, there is a need for a complementary scheme that builds upon the existing pre-trained network and tries to reduce further the computational overhead irrespective of the quantization method used. Furthermore, to enable quality scalable design, it is also desired that the scheme provides a systematic way to enable scalable designs with various accuracy levels that can be selected as per accuracy/resource requirements.

B. NOVEL CONTRIBUTIONS

In this paper, we propose CAxCNN, a Canonic Sign Digit based Approximation methodology for Convolutional Neural Networks. In particular, we analyze the use of our proposed Sign Digit (CSD) representation for the filter weights of CNNs and provide an associated error bound. The approximation aids in decreasing the number of non-zero terms in the CSD representation, subsequently resulting in arithmetic units with lower computational overhead. Our approach is complementary and can be applied to quantized weights obtained using the existing state of the art quantization schemes and do not require further retraining/fine-tuning. Furthermore, when employed with one of the proposed approximate CSD representation, CAxCNN can aid in the design of quality-scalable architectures by supporting multiple accuracy levels. Our CAxCNN tool-set is provided as an open-source contribution [29] to aid in reproducing the results and further research.

II. PROPOSED CAxCNN APPROXIMATION METHODOLOGY

In the following, we first provide a few relevant properties of CSD representation, followed by its relevance to CNNs. Then we present our approximate CSD representation and its associated error model. The section concludes with our CAxCNN methodology that aids in designing of hardware accelerators with reduced computational overhead.

A. CSD REPRESENTATION FOR FILTER WEIGHTS

Canonic sign digit representation uses ternary weights $\{-1, 0, 1\}$ to represent a binary number [30]. There are two important properties of CSD numbers: Firstly, for an N-bit binary number, its CSD representation has the least number

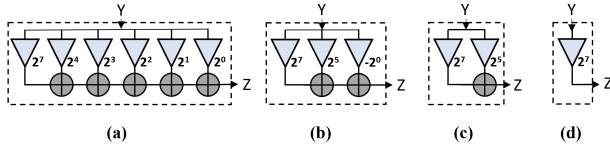


FIGURE 2. Multiplication of an input Y with a coefficient (with value 159) represented in (a) Binary (b) Accurate CSD (c) Approximated CSD ($\phi_x = 2$) and (d) Approximated CSD ($\phi_x = 1$) representation.

of non-zeros $(-1,1)$. Secondly, adjacent bits in the CSD representation cannot be both non-zero. An N -bit fixed-point binary number, P , with m integer and n fractional bits can be represented in an equivalent CSD representation as:

$$P = 2^{-n} \sum_{i=0}^N p_i 2^i \quad (1)$$

where $p_i \in \{-1, 0, 1\}$. Since CSD representation re-codes a binary number in a representation that requires ϕ (with $\phi < N$) non-zero digits, it helps reduce the computational cost of various arithmetic operations. This representation is particularly useful for the case when a number has to be multiplied with a constant since the number of partial products is equivalent to the reduced number of non-zeros (ϕ). These benefits can be exploited for the case of Deep CNNs since the filters weights are static (once trained) and hence can be converted to an equivalent CSD representation. The ternary representation of-course requires $N + 1 + \phi$ bits to be stored for each number: $N + 1$ bits to store the position while ϕ bits to store the polarity (-1 or $+1$) of non-zero bits. The benefit achieved in terms of reducing the computational complexity is illustrated in Fig. 2. Let us assume an input value Y being multiplied by a filter weight with value 159. Assuming $N=8$, the multiplier is represented as 10011111 in binary, and its dedicated circuit requires six shifts and five add operations (Fig 2(a)). The corresponding CSD representation for this constant multiplier is 10100001 with $\phi = 3$. Here, 1 represents a negative weight (-1) . It can be observed in fig 2(b) that CSD based representation results in a circuit that requires just ϕ shifts and $(\phi - 1)$ add operations for the same multiplication, thus reducing the overall computational overhead.

B. APPROXIMATE CANONIC SIGN DIGIT REPRESENTATION

Let P be a binary number accurately represented in CSD using ϕ non-zeros. Its corresponding approximate representation (CA_x) is formulated by representing it using ϕ_x non-zero ternary bits, where $(\phi_x < \phi)$. Fig 2 (c and d) illustrate two approximate CSD representations for a coefficient value of 159. We propose three approximation strategies, as described below:

- CA_{x_t} , Truncated CA_x : Provided a ϕ_x , only the most significant ϕ_x non-zero ternary digits are retained. Similar to a truncation operation, the rest of the least significant non-zero digits are discarded.

Algorithm 1 CA_{x_m} Approximation

```

Input:  $W$ : Weight Matrix,  $\phi_x$ : CSD Non-Zeros Limit,  $Qm.n$ : QFormat, m Integer and n Fractional bits
Output:  $W_{approx}$ : Approximated Weights
1: procedure  $CA_{x_m}(W_{in}, \phi_x, Qm.n)$ 
2:    $W_b \leftarrow toBinary(W, m, n)$ 
3:    $W_{CSD} \leftarrow toCSD(W_b)$ 
4:   for each  $w_{CSD}$  in  $W_{CSD}$  do
5:      $NZIndices \leftarrow NonZerosIndices(w_{CSD})$ 
6:      $w_{Approx} \leftarrow w_{CSD}$ 
7:      $NZCount \leftarrow CountNonZeros(w_{Approx})$ 
8:      $i \leftarrow 0$ 
9:     while  $NZCount > \phi_x$  do
10:       $w_{CSD}(NZIndices(i)) \leftarrow 0$ 
11:       $NZCount \leftarrow NZCount - 1$ 
12:       $i \leftarrow i + 1$ 
13:    end while
14:     $loss \leftarrow |w_{CSD} - w_{approx}|$ 
15:     $w_{approx} \leftarrow Minimize(loss, w_{in}, \phi_x, m, n)$ 
16:  end for
17: end procedure

```

- CA_{x_e} , Exhaustive CA_x : A truncated CSD representation CA_{x_t} , of a binary number, may not necessarily be its closest representation in ϕ_x non-zero ternary bits. Since there may exist multiple approximate CSD representations for a number, an analytical form solution may not be trivial. Thus, for CA_{x_t} we perform an exhaustive search across all possible ϕ_x non-zero digit CSD combinations to find the closest match with the least error.
- CA_{x_m} , Minimal Search CA_x : In minimal search approximation, we first compute CA_{x_t} , and then perform a localized search in the neighborhood of the number P to find a better candidate. This method reduces the search time as compared to CA_{x_e} while providing an approximate representation that is close to the original number. The procedure is described in Algorithm 1.

Since we shall be applying CA_x approximation to the filters of CNN, the floating-point weights are first converted to binary with m integer and n fractional bits. A matrix of such trained weights is passed to the CA_{x_m} Approximation procedure, along-with its fixed-point bit-width information (m, n) and the desired ϕ_x . The procedure converts the number to an equivalent CSD representation using [30]. For each number, the positions of non-zero digits are counted. The desired number(ϕ_x) of most significant non-zero digits are kept while rest are discarded by assigning a value of zero. This representation matches that of CA_{x_t} . A *loss function* is then computed that provides the measure of error incurred due to approximation. A *Loss Minimization* procedure, Algorithm 2, is then employed to search other CSD combinations that lie within a neighborhood of window defined by the loss function. Finally, the representation with minimum loss and with a count of non-zeros equal to or less than ϕ_x is returned as the approximated weight.

Algorithm 2 $CA_{x,m}$: Loss Minimization

Input: $loss$: Approximation error, w_{in} : Input weight, ϕ_x : Non-Zero Limit $Qm.n$: Q-Format, m Integer and n Fractional bits

Output: w_{approx} : Approximated CSD Weights

procedure Minimize($loss, w_{in}, \phi_x, Qm.n$)

- 1: Generate all CSDs in range $w_{in}-loss$ to $w_{in}+loss$
- 2: Remove Non-Zeros upto ϕ_x for all CSDs
- 3: Calculate $loss_{new}$ for each approximated CSD
- 4: Return CSD with minimum $loss_{new}$

end procedure

C. APPROXIMATION ERROR ANALYSIS OF CAxCNN

When a binary number P is represented using CA_{x_t} representation, an error is introduced due to the reduced number of non-zeros. The maximum error shall occur when the maximum number of non-zeros is packed towards the most significant bits. Since two consecutive digits cannot be non-zero, the maximum error that can occur in representing an N bit binary number using ϕ_x non-zero canonic sign digits is given below in equation 2.

$$E_{CA_{x_t}} \leq \begin{cases} 2^{-n} \sum_{i=0}^{\lceil \frac{N+1}{2} \rceil - \phi_x - 1} 2^{2i+1} & \text{if } N \text{ is odd} \\ 2^{-n} \sum_{i=0}^{\lceil \frac{N+1}{2} \rceil - \phi_x - 1} 2^{2i} & \text{if } N \text{ is even} \end{cases} \quad (2)$$

Here n is the number of fractional bits of the binary number P . This error bound applies to all the CSD approximations since CA_{x_t} incurs the maximum accuracy loss.

D. CAxCNN METHODOLOGY

Our CAxCNN methodology is illustrated in Fig.3 and described below. Libraries such as TensorFlow and Caffe are employed to train the CNN and learn the filter weights. These filter weights are typically learned as 32-bit floating-point numbers [17]. The ranges of these values are then analyzed (using tools such as MATLAB Fixed-Point Tool) to deduce the maximum number of integer bits (m) required to represent them. The decision to select the number of fractional bits (n) directly affects the precision of the network. Typically, the total bit-width ($m+n$) is capped to 32 or 16 bits, and then the fixed-point simulations are carried out (in tools such as MATLAB) to evaluate the drop in the accuracy of the network. This step can be followed by systematic bit-width reduction techniques, such as Ristretto [17], [22]. The Ristretto tool performs automatic network quantization by evaluating different bit-widths for number representation to find the right balance between compression rate and network accuracy. These steps (highlighted in red color in Fig.3) require fine-tuning/re-training of the learned weights. Once a viable fixed-point representation meeting the desired accuracy level has been found, the quantized filter weights are

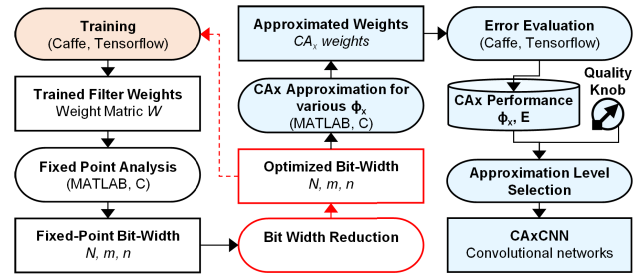


FIGURE 3. CAxCNN Methodology (highlighted in blue) for CSD based approximation of CNNs.

passed on to the CAxCNN methodology. Note that CAxCNN methodology (highlighted in blue) can be applied to both normal and reduced bit-width weights and is thus a complementary strategy. However, it is suggested to be employed after some automatic bit width reduction technique [17], [22]. First, CA_x approximations are applied to the filter weights for various values of ϕ_x . Thus, for each ϕ_x , we get a particular configuration of weights. Classification accuracy is then computed for each configuration. The configurations that provide accuracy values that are equal to or better than the minimum acceptable quality level (controlled via quality control knob) are all eligible for hardware implementation. The approximated weights result in a significant reduction in the computational overhead due to reduced logic area and latency. Thus the CA_x approximations provide a realizable quality control knob to gracefully trade-off quality vs. area/latency benefits.

This methodology is particularly suitable for recently proposed CNN accelerators exploiting *batch processing* [31] and *dynamic hardware reconfiguration* [32]. *Batch processing* involves processing the whole batch of input activations for a particular layer of CNN. Since memory bandwidth and storage requirement of the filter weights is averaged over the entire batch, the storage overhead associated with the ternary representation of the weights, as discussed in Section II.A is lowered. When combined with *dynamic hardware reconfiguration*, our CA_{x_t} approximate representation can be further exploited to design quality-scalable accelerators. This is because filter weights of varying accuracy levels can be formed by discarding the non-zeros digits beyond the ϕ_x most significant non-zeros. Accordingly, the related computational logic can be either power/clock gated for the case of ASICs or removed for the case of FPGAs. This is unlike schemes that require a different encoding for each approximate representation [26] or the ones that perform retraining for each accuracy level [17].

III. EXPERIMENTAL RESULTS AND DISCUSSION

In this section, we provide the results of our evaluation of the proposed CAx approximation schemes and CAxCNN methodology for four CNNs: A LeNet architecture (trained on MNIST dataset for handwritten number classification), a CIFAR10 network (trained on CIFAR-10 dataset for image classification) and, an AlexNet [3] and a VGG-16 [33]

TABLE 1. Absolute Mean Error, Max Error and Error Upper Bound for CA_x based approximations for various values of ϕ_x .

		Max Error	Mean Error	Error Upper-bound
CA_{x_t}	$\phi_x = 3$	0.0312	0.0147	0.125
	$\phi_x = 2$	0.0845	0.0170	0.625
	$\phi_x = 1$	0.3345	0.0714	2.625
CA_{x_m}	$\phi_x = 3$	0.0312	0.0147	0.125
	$\phi_x = 2$	0.0574	0.0163	0.625
	$\phi_x = 1$	0.2419	0.0621	2.625
CA_{x_e}	$\phi_x = 3$	0.0312	0.0147	0.125
	$\phi_x = 2$	0.0574	0.0163	0.625
	$\phi_x = 1$	0.2419	0.0621	2.625

TABLE 2. Time required for computing CA_x Approximations.

	Average Time (sec) (T_{CA_x})	Relative Time ($T_{CA_x}/T_{CA_{x_t}}$)
CA_{x_t}	0.0028	1x
CA_{x_m}	0.0690	25x
CA_{x_e}	0.2330	83x

network (trained on ImageNet dataset for image classification). To demonstrate that CA_x provides effective representation for CNN's filter weights, we compared the accuracy achieved to that of Gysel *et al.* who reported the accuracy of their quantized LeNet and CIFAR in [17] and for AlexNet in [22]. It is pertinent to note here that this comparison is provided for illustrating the robustness of CSD based approximation. In reality, our technique is complementary in a way that it is applied together with the quantization schemes. Thus, towards the end, we also provide the results of CAxCNN in a complementary setup where the proposed CA_x approximations are being applied to a VGG-16 network quantized using state-of-art dynamic fixed-point quantization [22].

A. EVALUATING THE CA_x APPROXIMATIONS

We apply the proposed CA_{x_t} , CA_{x_m} , and CA_{x_e} approximations to the filter weights of LeNet for various values of ϕ_x . We report the maximum and mean of the absolute error introduced to the trained filter representation in Table 1. For these results a LeNet implementation with Q4.4 representation for filter weights was used. It can be observed that CA_{x_m} provides the same error performance as that of CA_{x_e} . CA_{x_t} approximation is providing slightly elevated levels of error; however, the numerical results are still within the theoretical error bound provided by equation 2.

For a large CNN, approximations can take a considerable time to be computed as the number of filters is large. In order to assess the computational overhead of the proposed CA_x approximation schemes, we report the average time to approximate a LeNet filter weight in Table 2. The scripts were run on a Core i5 machine with 16 GB RAM, using MATLAB. CA_{x_m} and CA_{x_e} require 25x and 83x more time as compared with CA_{x_t} for the respective approximation to be computed. CA_{x_m} can thus be conveniently applied instead of the time-consuming CA_{x_e} since both introduce the same level of errors.

CA_x representation allows the use of multipliers with low computational overhead (as illustrated in Section II.B) at the cost of approximation. To evaluate this benefit, we compared the synthesis results of an accurate 8-bit Booth multiplier to that of a corresponding multiplier design considering $\phi_x = 3, 2$ and 1, respectively. Table 3 reports the number of BELs, LUTs, and the corresponding latency. The designs were synthesized on a Xilinx Virtex-5 XC5VLX20T FPGA using Xilinx ISE 14.7 IDE. It can be observed that CA_x provides 45%, 77%, and 97% area benefits in terms of BEL usage, and 17%, 29% and, 55% improvement in latency for $\phi_x = 3, 2$ and 1 respectively. This improvement is primarily due to the reduced number of shifters and adders required for CSD based binary arithmetic. Table 3 also reports the error introduced due to the approximate representation of the multiplier. For this, we performed exhaustive simulations over all the possible input combinations of an 8-bit multiplier and analyzed the Mean Absolute Error (MAE), the Worst Case Error (WCE), and the Mean Absolute Percentage Error (MAPE) incurred. It can be observed that even with $\phi_x = 2$, the MAE value is 499, with a MAPE of only 2.95%. Since CA_x is an approximate representation, to further assess the area and latency benefit per unit accuracy-drop, we compared our performance to that of three state-of-art approximate multipliers [34]–[36]. Open-source implementations of these multipliers were utilized for reproducible results. Since these multipliers provide multiple configurations, we selected a variant from each of these that either provided a comparable value of latency or error. For DRUM [34], a configuration of DRUM (8,4) was utilized. For [35], the Mult8 \times 8Cc implementation was used that consists of a combination of 4 \times 2 and 4 \times 4 approximate multipliers along with approximate addition. For [36] the online implementation of logMultK_w with w = 5 was utilized. In Table 3, the Blue, Green, and Red colors highlight the best, 2nd best and 3rd best value within each column, respectively (excluding Booth-8 multiplier). It can be observed that while DRUM [34] provides a latency that is comparable with $\phi_x = 3$, the error incurred is much higher. Similarly, $\phi_x = 2$ representation results in a multiplier design that has lower latency and error (MAE, MAPE) as compared to Mult8 \times 8Cc [35] and logMultK_w [36]. The area requirement of the multipliers that are designed for the proposed CA_x representation is also lower as compared to the approximate multipliers being evaluated. We, however, emphasize here that CA_x is a representation scheme and hence does not directly compete with the existing approximate multipliers. The design of approximate multipliers may also be tailored for CA_x representation to take benefit of the reduced number of partial products that are inherently provided by the canonic representation.

B. EVALUATING CAxCNN METHODOLOGY

We applied the CAxCNN methodology to LeNet and CIFAR10. The classification accuracy (%age) for various accuracy modes is provided in Table 4 for all the three CA_x approximations. It can be observed that the classification

TABLE 3. Comparison of synthesis results and error evaluation for 8×8 multiplication. The table provides results for an accurate 8×8 Booth multiplier and compares against the multiplier required for various values of ϕ_x considering CAx approximations. The table also provides the synthesis and error results for a few state-of-the-art approximate multipliers. The Blue, Green, and Red colors highlight the best, 2nd best and 3rd best value within each column, respectively (excluding Booth-8 multiplier). $\phi_x = 2$ representation results in a multiplier design that provides the best area, latency, and MAE values.

	Synthesis Results			Error		
	BEL	LUT	Latency (ns)	MAE	WCE	MAPE (%)
Booth-8	44	26	7.063	0	0	0
$\phi_x = 3$	24	23	5.813	71	1275	0.37
$\phi_x = 2$	10	9	4.965	499	5355	2.95
$\phi_x = 1$	1	0	3.133	3023	21675	18.72
DRUM(8,4) [34]	59	45	7.504	925	7425	5.84
Mult8x8Cc [35]	62	56	5.086	1590	8288	12.94
logMultK_w [36]	97	74	9.262	828	5257	4.79

TABLE 4. Classification Accuracy (%age) of LeNet and CIFAR10 for various ϕ_x .

	ϕ_x	CA_{x_t}	CA_{x_m}	CA_{x_c}
		Caffe LeNet 8bit Baseline 98.72	$\phi_x = 4$	98.72
	$\phi_x = 3$	98.72	98.72	98.72
	$\phi_x = 2$	98.72	98.72	98.72
	$\phi_x = 1$	98.68	98.70	98.70
Caffe CIFAR10 8bit Baseline 81.94	$\phi_x = 4$	81.94	81.94	81.94
	$\phi_x = 3$	81.82	81.82	81.82
	$\phi_x = 2$	81.32	81.52	81.52
	$\phi_x = 1$	66.08	76.16	76.16

TABLE 5. Classification Accuracy (%age) for CAxCNN based CNNs.

		LeNet	CIFAR10	AlexNet
Dynamic Fixed-Point [17], [22]		98.76	81.40	56.41
Fixed-Point		98.72	81.94	56.41
CAxCNN with ϕ_x	Booth-8	98.72	81.94	56.41
	3	98.72	81.82	56.39
	2	98.72	81.32	56.12
	1	98.68	66.08	50.04

accuracy of LeNet is marginally affected by the approximations. Even for CIFAR10, which is a more extensive network, all the three CA_x approximation schemes provide graceful degradation of accuracy as we move from $\phi = 3$ to 1.

CAx approximations can thus be utilized as an efficient quality control knob for quality scalable design as CAxCNN avoids further retraining/fine-tuning. The methodology can, therefore, be used to design run-time accuracy-configurable hardware accelerators by gating the logic that relates to the truncated CSD bit. The associated routing and control overheads have to be, however, evaluated in detail.

We also compared the classification accuracy of LeNet, CIFAR10, and AlexNet quantized using the Dynamic Fixed-Point scheme of Ristretto [22] and those approximated using CA_{x_t} based CAxCNN. The results are reported in Table 5. Accuracy values for 8 bit fixed-point implementations are also provided. It can be observed that for all the three CNNs, we are achieving accuracy values that are on par with the state-of-the-art. Furthermore, there is again a

TABLE 6. Classification Accuracy (%age) for CAxCNN applied on VGG-16 [33] with quantized coefficients generated by Ristretto [22] tool using Dynamic Fixed-Point quantization scheme.

	Pre-Trained	Ristretto	CAxCNN with ϕ_x		
			3	2	1
CA_{x_e}	67.92	66.84	63.12	63.08	57.60
CA_{x_t}			63.12	62.90	54.50

graceful degradation in quality, as we move from $\phi_x = 3$ to 1, even for the relatively deeper AlexNet. The drop in Top-1 accuracy is just 0.03% and 0.51% for $\phi_x = 3$ and 2, respectively, for AlexNet. The motivation for this comparison was to demonstrate that a small number of non-zeros in CSD can represent CNNs with reasonable accuracy.

C. CAxCNN APPROXIMATION WITH RISTRETTO QUANTIZATION

CAxCNN is a complementary technique and can be applied together with state-of-the-art quantization schemes. To demonstrate this, we apply CAxCNN on the filter weights of the VGG-16 network that has already quantized using the dynamic fixed-point quantization scheme of Ristretto [22]. Table-6 reports the classification accuracy for various values of ϕ_x . It can be observed that there is a graceful degradation in the accuracy as we move from $\phi_x = 3$ to 1. Specifically, the drop in top-1 accuracy is 5.63.% for a CAx configuration with $\phi_x = 2$, which provided 77% area benefit as per Table-3. Thus, CAxCNN provides efficient representation to enable the design of accelerators with lower computational complexity.

IV. CONCLUSION AND FUTURE WORK

A CAxCNN methodology, based upon an approximated CSD representation for the filter weights of CNNs was proposed. This methodology aids in the development of hardware accelerators for CNNs with low computational overhead. CAxCNN provides approximation schemes with various approximation levels, and their associated error bound. Evaluations on LeNet, CIFAR, AlexNet, and VGG-16 demonstrate that CAx based approximations provide low computational overhead with acceptable quality. The methodology can be employed to design quality-scalable accelerators by exploiting batch processing and dynamic reconfiguration with CA_{x_t} approximation.

REFERENCES

- [1] Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, and L. D. Jackel, "Handwritten digit recognition with a back-propagation network," in *Proc. Adv. Neural Inf. Process. Syst.*, 1990, pp. 396–404.
- [2] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 779–788.
- [3] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1–9.

- [4] L. Zhou, Z. Wang, Y. Luo, and Z. Xiong, "Separability and compactness network for image recognition and superresolution," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 11, pp. 3275–3286, Nov. 2019.
- [5] K. Jiang, Z. Wang, P. Yi, G. Wang, T. Lu, and J. Jiang, "Edge-enhanced GAN for remote sensing image superresolution," *IEEE Trans. Geosci. Remote Sens.*, vol. 57, no. 8, pp. 5799–5812, Aug. 2019.
- [6] Z. Wang, P. Yi, K. Jiang, J. Jiang, Z. Han, T. Lu, and J. Ma, "Multi-memory convolutional neural network for video super-resolution," *IEEE Trans. Image Process.*, vol. 28, no. 5, pp. 2530–2544, May 2019.
- [7] P. Yi, Z. Wang, K. Jiang, Z. Shao, and J. Ma, "Multi-temporal ultra dense memory network for video super-resolution," *IEEE Trans. Circuits Syst. Video Technol.*, Jul. 1, 2020, doi: 10.1109/TCSVT.2019.2925844.
- [8] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Müller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba, "End to end learning for self-driving cars," 2016, *arXiv:1604.07316*. [Online]. Available: <http://arxiv.org/abs/1604.07316>
- [9] 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.
- [10] NVIDIA DGX-1 Artificial Intelligence System. Accessed: Sep. 16, 2017. [Online]. Available: <http://images.nvidia.com/content/technologies/deep-learning/pdf/Datasheet-DGX1.pdf>
- [11] S. Wu, D. Hu, S. Ibrahim, H. Jin, J. Xiao, F. Chen, and H. Liu, "When FPGA-accelerator meets stream data processing in the edge," in *Proc. IEEE 39th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jul. 2019, pp. 1818–1829.
- [12] S. Du, T. Huang, J. Hou, S. Song, and Y. Song, "FPGA based acceleration of game theory algorithm in edge computing for autonomous driving," *J. Syst. Archit.*, vol. 93, pp. 33–39, Feb. 2019.
- [13] C. Lammie, A. Olsen, T. Carrick, and M. R. Azghadi, "Low-power and high-speed deep FPGA inference engines for weed classification at the edge," *IEEE Access*, vol. 7, pp. 51171–51184, 2019.
- [14] X. Xu, Y. Ding, S. X. Hu, M. Niemier, J. Cong, Y. Hu, and Y. Shi, "Scaling for edge inference of deep neural networks," *Nature Electron.*, vol. 1, no. 4, p. 216, 2018.
- [15] S. Behroozi, J. Li, J. Melchert, and Y. Kim, "SAADI: A scalable accuracy approximate divider for dynamic energy-quality scaling," in *Proc. 24th Asia South Pacific Design Autom. Conf.*, Jan. 2019, pp. 481–486.
- [16] D. Wu, T. Chen, C. Chen, O. Ahia, J. S. Miguel, M. Lipasti, and Y. Kim, "SECO: A scalable accuracy approximate exponential function via cross-layer optimization," in *Proc. IEEE/ACM Int. Symp. Low Power Electron. Design (ISLPED)*, Jul. 2019, pp. 1–6.
- [17] P. Gysel, M. Motamedi, and S. Ghiasi, "Hardware-oriented approximation of convolutional neural networks," 2016, *arXiv:1604.03168*. [Online]. Available: <https://arxiv.org/abs/1604.03168>
- [18] J. Fadavi-Ardekani, "M* N booth encoded multiplier generator using optimized Wallace trees," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 1, no. 2, pp. 120–125, Jun. 1993.
- [19] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
- [20] 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.
- [21] S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan, "Deep learning with limited numerical precision," in *Proc. 32nd Int. Conf. Mach. Learn. (ICML)*, 2015, pp. 1737–1746.
- [22] P. Gysel, J. Pimentel, M. Motamedi, and S. Ghiasi, "Ristretto: A framework for empirical study of resource-efficient inference in convolutional neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 11, pp. 5784–5789, Nov. 2018.
- [23] 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*. [Online]. Available: <http://arxiv.org/abs/1602.02830>
- [24] M. Kim and P. Smaragdus, "Bitwise neural networks," 2016, *arXiv:1601.06071*. [Online]. Available: <http://arxiv.org/abs/1601.06071>
- [25] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-Net: Imagenet classification using binary convolutional neural networks," in *Proc. Eur. Conf. Comput. Vis.*, Springer, 2016, pp. 525–542.
- [26] H.-P. Trinh, M. Duranton, and M. Paindavoine, "Efficient data encoding for convolutional neural network application," *ACM Trans. Archit. Code Optim. (TACO)*, vol. 11, no. 4, p. 49, 2015.
- [27] M. Shafique, R. Hafiz, M. U. Javed, S. Abbas, L. Sekanina, Z. Vasicek, and V. Mrazek, "Adaptive and energy-efficient architectures for machine learning: Challenges, opportunities, and research roadmap," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI (ISVLSI)*, Jul. 2017, pp. 627–632.
- [28] M. Alioto, V. De, and A. Marongiu, "Energy-quality scalable integrated circuits and systems: Continuing energy scaling in the twilight of Moore's law," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 8, no. 4, pp. 653–678, Dec. 2018.
- [29] Caxcnn Library. Accessed: Jul. 8, 2020. [Online]. Available: <http://vispro.itu.edu.pk/open-source-lib/>
- [30] G. A. Ruiz and M. Granda, "Efficient canonic signed digit recoding," *Microelectron. J.*, vol. 42, no. 9, pp. 1090–1097, Sep. 2011.
- [31] Y. Shen, M. Ferdman, and P. Milder, "Escher: A CNN accelerator with flexible buffering to minimize off-chip transfer," in *Proc. IEEE 25th Annu. Int. Symp. Field-Program. Custom Comput. Mach. (FCCM)*, Apr. 2017, pp. 93–100.
- [32] M. Putic, A. Buyuktosunoglu, S. Venkataramani, P. Bose, S. Eldridge, and M. Stan, "DyHard-DNN: Even more DNN acceleration with dynamic hardware reconfiguration," in *Proc. 55th ACM/ESDA/IEEE Design Autom. Conf. (DAC)*, Jun. 2018, pp. 1–6.
- [33] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*. [Online]. Available: <http://arxiv.org/abs/1409.1556>
- [34] 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.
- [35] S. Ullah, S. Rehman, B. S. Prabakaran, F. Kriebel, M. A. Hanif, M. Shafique, and A. Kumar, "Area-optimized low-latency approximate multipliers for FPGA-based hardware accelerators," in *Proc. 55th ACM/ESDA/IEEE Design Autom. Conf. (DAC)*, Jun. 2018, pp. 1–6.
- [36] 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.



MOHSIN RIAZ received the B.S. degree in electrical engineering from the University of Engineering and Technology (UET), Lahore, Pakistan, in 2009, and the M.S. degree from the Lahore University of Management and Sciences (LUMS), in 2014. He has worked as a Research Associate with the Vision Processing Laboratory (Vispro), Information Technology University (ITU), Pakistan, in 2018. His research interests include hardware optimization for deep learning architectures and digital system design.



REHAN HAFIZ received the Ph.D. degree in EE from the University of Manchester, U.K., in 2008. He has served as an Assistant Professor with the School of Electrical Engineering and Computer Science, National University of Sciences and Technology, from 2008 to 2015. He is currently a Professor with the Computer Engineering Department, Information Technology University (ITU), Lahore. He founded and directed the Vision Image and Signal Processing (VISpro) Laboratory that focuses on vision system design, development of power efficient architectures, design of approximate computing-based hardware accelerators, FPGA-based designs, multi-projector and immersive display technologies, and applied image and video processing. He holds several patents in USA, South Korean, and Pakistan patent office. He has published several articles related to custom processor design, application specific processor designing, video stabilization, multi-projector rendering, and approximate computing.



SALMAN ABDUL KHALIQ received the B.S. degree in electrical engineering from the Pakistan Institute of Engineering and Applied Sciences (PIEAS), Islamabad, in 2014, and the M.S. degree in electrical engineering from Information Technology University (ITU), Lahore, in 2019. He was a Graduate Research Assistant with the Vision Processing Laboratory. His research interests include hardware optimization for deep learning architectures and digital system design.



MOHSEN ALI received the Ph.D. degree in the area of computer vision from the CISE, University of Florida. He is currently an Assistant Professor with Information Technology University (ITU) and the Co-Founder of the Intelligent Machines Laboratory. IML has been established with the objective to allow researchers and engineers working in machine learning, computer vision, and robotics to work together and solve real-world problems. His current research interests include theoretical and practical problems entailing computer vision and machine learning; and apply them in the field of video and image analysis, remote sensing, and affective computing. His group works extensively in deep learning both extending theoretical foundations and applying them to create practical solutions. He is also a Fulbright Scholar, an alumnus of PUCIT and LUMS. His research articles have been published in top ranking venues of computer vision research.



MUHAMMAD FAISAL received the B.S. degree in computer science from COMSATS University, Pakistan, in 2016, and the M.S. degree from Information Technology University (ITU), Pakistan, in 2018. He is currently working as a Research Associate with the Vision Processing (VisPro) Laboratory. Prior to joining VisPro, he has worked as a Graduate Research Assistant with the Intelligent Machines Laboratory. His research interests include computer vision and deep learning.



MUHAMMAD SHAFIQUE (Senior Member, IEEE) received the Ph.D. degree in computer science from the Karlsruhe Institute of Technology (KIT), Germany, in January 2011. He was with Streaming Networks Pvt., Ltd., where he was involved in research and development of video coding systems for several years. He has been a Full Professor of Computer Architecture and Robust Energy-Efficient Technologies (CARE-Tech.) with the Institute of Computer Engineering, TU Wien, Austria, since November 2016. His research interests include computer architecture, power-/energy-efficient systems, robust computing, hardware security, brain-inspired computing trends like neuromorphic and approximate computing, embedded artificial intelligence, hardware and system-level design for machine learning, emerging technologies and nanosystems, FPGAs, MPSoCs, and embedded systems. His research has a special focus on cross-layer modeling, design, and optimization of computing and memory systems, as well as their deployment in use cases from the Internet of Things (IoT), cyber-physical systems (CPS), and ICT for development (ICT4D) domains. He holds one U.S. patent and has (co)authored six books, more than ten book chapters, and over 200 papers in premier journals and conferences. He is a Senior Member of the IEEE Signal Processing Society (SPS) and a member of the ACM, SIGARCH, SIGDA, SIGBED, and HIPEAC. He has given several Keynote, Invited Talks, and Tutorials. He has served on the TPC of numerous prestigious IEEE/ACM conferences. He has received the 2015 ACM/SIGDA Outstanding New Faculty Award, six gold medals in his educational career, and several best paper awards and nominations at prestigious conferences like CODES+ISSS, DATE, DAC, and ICCAD, the Best Master Thesis Award, the DAC'14 Designer Track Best Poster Award, the IEEE Transactions on Computers Feature Paper of the Month Awards, and the Best Lecturer Award. He has also organized many special sessions at premier venues and served as the Guest Editor for the *IEEE Design and Test Magazine* and the IEEE TRANSACTIONS ON SUSTAINABLE COMPUTING.



HAFIZ TALHA IQBAL received the B.S. degree in electrical engineering from the National University of Sciences and Technology (NUST) Islamabad, in 2017. He is currently pursuing the M.S. degree in electrical engineering with Information Technology University (ITU), Lahore. He was a Research Assistant with LUMS and a Graduate Fellow with the VISpro Laboratory, ITU. His research interests include approximate computing, wearable devices, and digital system design.

...