# RingCNN: Exploiting Algebraically-Sparse Ring Tensors for Energy-Efficient CNN-Based Computational Imaging

Chao-Tsung Huang

Department of Electrical Engineering National Tsing Hua University HsinChu, Taiwan, R.O.C. chaotsung@ee.nthu.edu.tw

Abstract—In the era of artificial intelligence, convolutional neural networks (CNNs) are emerging as a powerful technique for computational imaging. They have shown superior quality for reconstructing fine textures from badly-distorted images and have potential to bring next-generation cameras and displays to our daily life. However, CNNs demand intensive computing power for generating high-resolution videos and defy conventional sparsity techniques when rendering dense details. Therefore, finding new possibilities in regular sparsity is crucial to enable large-scale deployment of CNN-based computational imaging.

In this paper, we consider a fundamental but yet well-explored approach—algebraic sparsity—for energy-efficient CNN acceleration. We propose to build CNN models based on ring algebra that defines multiplication, addition, and non-linearity for n-tuples properly. Then the essential sparsity will immediately follow, e.g. n-times reduction for the number of real-valued weights. We define and unify several variants of ring algebras into a modeling framework, RingCNN, and make comparisons in terms of image quality and hardware complexity. On top of that, we further devise a novel ring algebra which minimizes complexity with component-wise product and achieves the best quality using directional ReLU. Finally, we design an accelerator, eRingCNN, to accommodate to the proposed ring algebra, in particular with regular ring-convolution arrays for efficient inference and on-the-fly directional ReLU blocks for fixed-point computation. We implement two configurations, n = 2 and 4 (50% and 75% sparsity), with 40 nm technology to support advanced denoising and super-resolution at up to 4K UHD 30 fps. Layout results show that they can deliver equivalent 41 TOPS using 3.76 W and 2.22 W, respectively. Compared to the real-valued counterpart, our ring convolution engines for n=2 achieve  $2.00\times$  energy efficiency and  $2.08\times$  area efficiency with similar or even better image quality. With n=4, the efficiency gains of energy and area are further increased to  $3.84\times$  and  $3.77\times$ with only 0.11 dB drop of peak signal-to-noise ratio (PSNR). The results show that RingCNN exhibits great architectural advantages for providing near-maximum hardware efficiencies and graceful quality degradation simultaneously.

Index Terms—convolutional neural network, computational imaging, regular sparsity, hardware accelerator

## I. INTRODUCTION

Convolutional neural networks (CNNs) have demonstrated their superiority in the fields of computer vision and compu-

This work was supported by the Ministry of Science and Technology, Taiwan, R.O.C., under Grant no. MOST 109-2218-E-007-034.

tational imaging. The former includes object recognition [18], [43] and detection [55]. The latter involves image denoising [49], [50], super-resolution (SR) [26], [31], [32], [47], and style transfer [24], [56]; in particular, denoising is the key to enhance low-light photography on mobile phones, and SR plays an important role for displaying lower-resolution contents on ultra-high-resolution (UHD) TVs. Although CNNs can be applied in both application fields, the computation schemes are quite different and so are their design challenges.

Recognition and detection CNNs aim to extract high-level features and usually process small images with a huge amount of parameters. In contrast, computational imaging ones need to generate low-level and high-precision details and often deal with much larger images with fewer parameters. For example, the state-of-the-art FFDNet for denoising [50] requires only 850K weights but can be used to generate 4K UHD videos at 30 fps. This will demand as high as 106 TOPS (tera operations per second) of computation, and the precision of multiplications could be at least 8-bit for representing sufficient dynamic ranges. Therefore, for computational imaging it is the intensive computation for rendering *fine-textured*, *high-throughput*, and *high-precision* feature maps to pose challenges for the deployment in consumer electronics.

Exploiting sparsity in computation is a promising way to reduce complexity for CNNs. Many approaches have been analyzed in detail, but most of them are discussed only for recognition and detection. The most common one is to explore natural sparsity for feature maps [3], [34] and/or filter weights [16], [38], [54]. Utilizing such sparsity, like unstructured pruning [17], will induce computation irregularity and thus significant hardware overheads. For example, the state-of-the-art SparTen [16] only delivers 0.43 TOPS/W on 45 nm technology for the dedicated designs to tame irregularity. If, instead, structured pruning [35], [40] is applied to improve regularity, model quality will then drop quickly. Thus, natural sparsity is hard to support high-throughput inference with low power consumption.

Another common approach is to explore the low-rank sparsity in over-parameterized CNNs by either decomposition [27], [30], [37] or model structuring [12], [19], [23], [41]. It aims

high compression ratios and approximates weight tensors by regular but radically-changed inference structures. This low-rank approximation works well for recognition CNNs which extract high-level features. But it could quickly deteriorate the representative power of computational imaging ones for generating local details. For example, merely applying depthwise convolution can lead to 1.2 dB of peak signal-to-noise ratio (PSNR) drop for SR networks [21]. Therefore, low-rank sparsity may not be suitable for fine-textured CNN inference.

A recent alternative for providing regular acceleration is to enforce full-rank sparsity on matrix-vector multiplications [13], [52], [53]. It partitions them into several  $n \times n$  sub-block multiplications and then replaces each one by a componentwise product between n-tuples. This is equivalent to a group convolution with data reordering [53]; therefore, for restoring representative power additional pre-/post-processing is required to mix information between components or groups. CirCNN [13] equivalently applies Fourier transform on each sub-block for this purpose by forcing weight matrices to be block-circulant. ShuffleNet [52] instead performs global channel shuffling, and HadaNet [53] adopts simpler Hadamard transform. However, the applicability of this approach is unclear for computational imaging because CirCNN aims very high compression ratios (66× for AlexNet [29]), and ShuffleNet and HadaNet focus only on bottleneck convolutions.

Lastly, a fundamental but less-discussed approach is to exploit algebraic sparsity. In contrast to using real numbers, CNNs can also be constructed by complex numbers [44] or quaternions [15], [39], [57]. By their nature, the number of real-valued weights can decrease two or four times, respectively. Moreover, their multiplications can be accelerated by fast algorithms. For example, the quaternion multiplication is usually expressed by a  $4 \times 4$  real-valued matrix and can be simplified into eight real-valued multiplications and some linear transforms [20]. Regarding activation functions, the realvalued component-wise rectified linear unit (ReLU) is mostly adopted, and its efficiency over complex-domain functions is demonstrated in [44]. Since this algebraic sparsity can reduce complexity with moderate ratios and high regularity, it is a good candidate for accelerating computational imaging. However, previous work only discusses the two traditional division algebras and thus poses strict limitations for implementation.

In this paper, we would like to lay down a more generalized framework—*RingCNN*—for algebraic sparsity to expand its design space for model-architecture co-optimization. Observing that division is usually not required by CNN inference, we propose to construct models by *ring*, a fundamental algebraic structure on *n*-tuples with definitions of multiplication and addition. In particular, we consider a bilinear formulation for ring multiplication to have transform-based fast algorithms and thus include full-rank sparsity into this framework. For constructing CNN models, we also equip non-linearity to the rings. Then several ring variants are defined properly and compared systematically for joint quality-complexity optimization.

This algebraic generalization also brings architectural insights on ring non-linearity. We observe that conventional

methods mostly adopt the component-wise ReLU for non-linearity and use the linear transforms in ring multiplication for information mixing. However, for fixed-point implementation these transforms will increase input bitwidths for the following component-wise products and bring significant hardware overheads. Inspired by this, we propose a ring with a novel directional ReLU to serve both non-linearity and information mixing. Then we can avoid the transforms before the products to eliminate the bitwidth-increasing overheads. Extensive evaluations will show that the proposed ring can achieve not only the best hardware efficiency for multiplications but also the best image quality for its compact structure for training.

Finally, we design an accelerator—eRingCNN—to utilize the proposed ring for high-throughput and energy-efficient CNN acceleration. For comparison purposes, we adopt eCNN [21], the state-of-the-art for computational imaging, as our architecture backbone. Then we devise highly-parallel ring-convolution engines for efficient inference and simply replace the real-valued counterparts in eCNN thanks to their regularity and similarity in computation. For the directional ReLU which involves two transforms, conventional MAC-based accelerators may need to perform quantization before each transform and cause up to 0.2 dB of PSNR drop. Instead, we apply an onthe-fly processing pipeline to avoid unnecessary quantization errors and facilitate fixed-point inference on 8-bit features. With 40 nm technology, we implement two sparsity settings, n=2 and 4, for eRingCNN to show the effectiveness.

In summary, the main contributions and findings of this paper are:

- We propose a novel modeling framework, RingCNN, to thoroughly explore algebraic sparsity. The corresponding training process, including quantization, is also established for in-depth quality comparisons.
- We propose a novel ring variant with a directional ReLU which achieves better image quality and area saving than complex field, quaternions, the rings alike to CirCNN and HadaNet, and all newly-discovered ones. Its image quality even outperforms unstructured weight pruning and sometimes, when n=2, can be better than real field.
- We design and implement accelerators with two configurations: eRingCNN-n2 (50% sparsity) and eRingCNN-n4 (75%). They can deliver *equivalent* 41 TOPS using only 3.76 W and 2.22 W, respectively, and support high-quality computational imaging at up to 4K UHD 30 fps.
- Our ring convolution engines achieve near-maximum hardware efficiencies ( $\cong n$ ). Layout results show that for n=2 they have  $2.00\times$  energy efficiency and  $2.08\times$  area efficiency compared to the real-valued counterpart. Those for n=4 can increase the corresponding efficiency gains to  $3.84\times$  and  $3.77\times$ , respectively.
- RingCNN models provide competitive image quality.
   Compared to the real-valued models for eCNN, those for eRingCNN-n2 even have an average PSNR gain of 0.01 dB and those for eRingCNN-n4 only drop by 0.11 dB. When serving Full-HD applications on eRingCNN, they can outperform the advanced FFDNet [50]

for denoising and SRResNet [31] for SR.

## II. MOTIVATION

We aim to enable next-generation computational imaging on consumer electronics by achieving high-throughput and high-quality inference with energy-efficient and cost-effective acceleration. However, computational imaging CNNs require dense model structures to generate fine-textured details. Thus before deploying any complexity-reducing method we need to examine the impact of image quality and the gain of computation complexity as a whole.

Without loss of generality, we demonstrate this qualitycomplexity tradeoff using the advanced model SRResNet as an example. In Fig. 1, two conventional sparsity techniques are examined. One is unstructured magnitude-based weight pruning for exploring natural sparsity. It shows graceful quality degradation when compression ratios are up to  $2\times$ ,  $4\times$ , and 8x. However, its irregular computation will erode the performance gain due to induced hardware overheads and load imbalance. For example, only 11.7% of power consumption and 5.6% of area are spent on MACs in the sparse tensor accelerator SparTen [16]. The other examined technique is depthwise convolution (DWC) which exploits low-rank sparsity. The quality drops very quickly and even can be worse than the old-fashioned VDSR [26]. As a result, weight pruning and DWC are unfavourable for computational imaging due to the computation irregularity and the quality distortion respectively.

A more straightforward approach is to reduce the model size in a compact way, and here we consider two cases: shrinking either model depth or feature channels. For SRResNet, the depth reduction causes sharp quality loss. In contrast, the channel reduction provides a good quality-complexity tradeoff which shows a similar trend as weight pruning and performs much better than DWC. In particular, this approach maintains high computation regularity and can be accelerated by energy-efficient dense tensor accelerators, such as eCNN [21] in which 94.0% of power consumption and 72.8% of area are spent on convolutions. Therefore, the compact model configurations should also be considered before applying sparsity.

In this paper, we would like to explore the possibility of having the quality of weight pruning and the regularity of compact modeling at the same time. We will approach this goal by using ring algebra for the elementary operations in CNNs. In this way, we can achieve *local sparsity* and assure *global regularity* simultaneously. Our results, RingCNN, for SRResNet are also shown in Fig. 1 to demonstrate the effectiveness. The details of our approach will be introduced in the following.

# III. RING ALGEBRA FOR NEURAL NETWORKS

Deep neural networks consist of many feed-forward layers. These layers are usually defined over real field  $\mathbb R$  and formulated by its three elementary operations: addition +, multiplication  $\cdot$ , and non-linearity f. With tensor extensions, we can have a common formulation for each l-th layer:

$$\mathbf{x}^{(l)} = \mathbf{f}^{(l)} (\mathbf{G}^{(l)} \mathbf{x}^{(l-1)} + \mathbf{b}^{(l)}),$$
 (1)

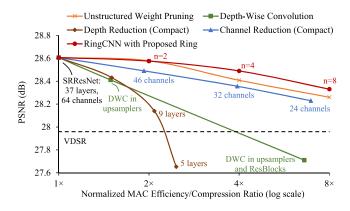


Fig. 1. Computation efficiency versus image quality. Different complexity-reducing methods are applied to SRResNet for four-times SR tasks (scaling up by four times for both of image width and height). The models are trained using the same training strategy. The image quality is measured by the averaged PSNR over test datasets Set5 [7], Set14 [48], BSD100 [36], and Urban100 [22].

where  $\mathbf{x}^{(l-1)}$ ,  $\mathbf{G}^{(l)}$ ,  $\mathbf{b}^{(l)}$ , and  $\mathbf{f}^{(l)}$  represent the input feature tensor, weight tensor, bias tensor, and non-linear tensor operation respectively. Conversely, as long as we define the three operations properly, we can construct neural networks at will using other algebraic structures.

An example for using complex field is shown in Fig. 2. Each complex number z can be expressed by either a complex form  $z_0 + z_1 i$  or an equivalent 2-tuple  $\binom{z_0}{z_1}$ . Then weight storage can be reduced by a half, and arithmetic computation can be accelerated by the complex multiplication algorithm, i.e. the complexity for each complex multiplication can be reduced from four real multiplications to three. In the following, we will first consider ring algebras to generalize this idea and define proper ring multiplication for discussion. Then we will analyze their demands of hardware resources, and, finally, propose a novel ring variant with directional non-linearity to maximize hardware efficiency.

## A. Ring Algebra

A ring R is a fundamental algebraic structure which is a set equipped with two binary operations + and  $\cdot$ . Here we consider the set of real-valued n-tuples, i.e.  $R = \{x = (x_0, ..., x_{n-1})^t \mid x_i \in \mathbb{R}\}$ . For clarity, x is a ring element and  $x_i$  is its real-valued *component*. And we simply use the component-wise vector addition for the ring addition +.

As for ring multiplication  $\cdot$ , it plays an important role for the properties of different rings. Given

$$z = g \cdot x \tag{2}$$

where  $z, g, x \in R$ , we consider it has a bilinear form to have a general formulation for fast algorithms, which will be discussed in Section III-B. In particular, the components of the three ring elements are related by

$$z_i = \sum_{j=0}^{n-1} \sum_{k=0}^{n-1} M_{ikj} g_k x_j, \tag{3}$$

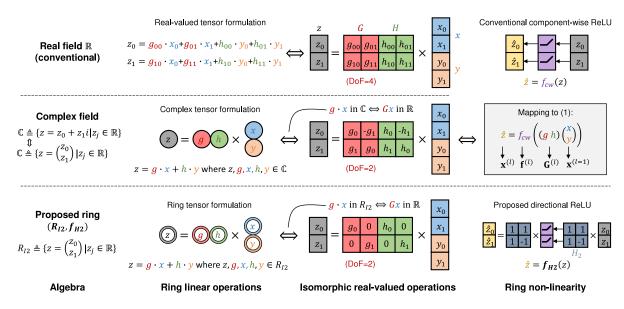


Fig. 2. A simple neural-network layer for four real-valued inputs  $(x_0, x_1, y_0, \text{ and } y_1)$  and two real-valued outputs  $(\hat{z}_0 \text{ and } \hat{z}_1)$  by (top) real field  $\mathbb{R}$ , (middle) complex field  $\mathbb{C}$ , and (bottom) a proposed 2-tuple ring  $(R_{I2}, f_{H2})$ . For the latter two algebras, the inputs are equivalently two 2-tuples (x and y) and the output becomes one 2-tuple  $(\hat{z})$ . Their tensor formulations,  $z = (g h) \begin{pmatrix} x \\ y \end{pmatrix}$  in  $\mathbb{C}$  or  $R_{I2}$ , have isomorphic operations in real field: z = Gx + Hy in  $\mathbb{R}$ . Then the degrees of freedom (DoF) in real numbers for each weight sub-matrix, e.g. G, are reduced from four  $(g_{00}, g_{01}, g_{10}, \text{ and } g_{11})$  to two  $(g_0 \text{ and } g_1)$ .

where M is a 3-D indexing tensor with only 1, 0, and -1 as its entries. In other words, the products of input ring components in form of  $g_k x_j$  are distributed to output components  $z_i$  through  $M_{ikj}$ . With the bilinear form (3), the ring multiplication (2) will be isomorphic to a matrix-vector multiplication

$$z = Gx, (4)$$

where the matrix G has entries  $G_{ij} = \sum_{k=0}^{n-1} M_{ikj} g_k$ . Without loss of generality, we will use g for filter weights and x for feature maps in the following.

After having definitions of + and  $\cdot$ , we still need to define a unary non-linear operation f for a ring to construct neural networks. A conventional choice, which is usually adopted by previous methods for full-rank or algebraic sparsity, is a component-wise ReLU

$$f_{cw}(x) = (\max(0, x_0), ..., \max(0, x_{n-1}))^t,$$
 (5)

where  $\max(0,\cdot)$  is the commonly-used real-valued ReLU.

# B. Fast Ring Multiplication

Now we will integrate transform-based full-rank sparsity into this framework. For the bilinear-form ring multiplication (3), from [46] we know that its optimal general fast algorithm over real field can be expressed by the following three steps

filter/data transform: 
$$\tilde{g} = T_q g, \tilde{x} = T_x x,$$
 (6)

component-wise product: 
$$\tilde{z} = \tilde{g} \circ \tilde{x}$$
 (on *m*-tuples), (7)

reconstruction transform: 
$$z = T_z \tilde{z}$$
, (8)

where  $T_g$  and  $T_x$  are  $m \times n$  transform matrices for g and x respectively, and  $T_z$  is  $n \times m$  for z. And  $\circ$  represents a component-wise product, i.e.  $\tilde{z}_i = \tilde{g}_i \tilde{x}_i$  for i = 0, 1, ..., m-1

for the three m-tuples  $\tilde{z}$ ,  $\tilde{g}$ , and  $\tilde{x}$ . If the transform matrices involve only simple coefficients, e.g.  $\pm 1$  or 0, then they can be implemented by adders, and the component-wise product will dominate computation complexity. In particular, the number of real-valued multiplications can be reduced from the general  $n^2$  for matrix G to m in (7). Therefore, the complexity of fast ring multiplication depends on how we decompose the indexing tensor M or its isomorphic matrix G into (6)-(8).

When G is diagonalizable over real field, i.e.  $G = T^{-1}DT$ , this complexity can be minimized as  $m = \operatorname{rank}(G)$  for  $\tilde{g} = \operatorname{diag}(D)$ . The proof is given in Appendix A. In this perspective, a ring  $R_H$  alike to HadaNet has a full-rank G, i.e.  $\operatorname{rank}(G) = n$ , which is diagonalized by Hadamard transform. Another example is a ring  $R_I$  equivalent to group convolution which applies component-wise products for a diagonal full-rank G, and its invertible T is simply the identity matrix I.

In contrast, if G is not diagonlizable over real field, we can instead apply the tensor rank decomposition for the indexing tensor M as mentioned in [20]. However, the complexity is usually larger than  $\operatorname{rank}(G)$  in this case, and the generic rank (grank) represents the lower bound for real-valued multiplications:  $m \geq \operatorname{grank}(M)$ . For example, the rotation matrix for complex field  $\mathbb C$  leads to three real-valued multiplications as its  $\operatorname{grank}(M) = 3$  while  $\operatorname{rank}(G) = 2$ , and the circulant matrix in  $\operatorname{CirCNN}$  also belongs to this category. The related properties of  $\mathbb C$  as well as  $R_H$  and  $R_I$  with ring dimension n=2 are as shown at the top of Table I.

#### C. Proper Ring Multiplication

In addition to the rings at hand, we would like to search more proper variants for in-depth analysis. We make three

TABLE I PROPERTIES OF RING ALGEBRAS.

	Ring Algebra				Fast Algorithm		Hardware	Hardware Efficiency (w.r.t. $\mathbb{R}$ )		
dim	Ring Variant	Unity	Isomorphic Matrix G	Non-linearity	Infomation Mixing	Transform Matrices	Real Mult. Number m	Weight Storage	Real Mult.	8-bit Complexity
	R <sub>H2</sub>	.1	Symmetric	Component-	Ring	Hadamard	2	Storage	2.0×	1.6×
	C C	$\binom{1}{0}$	Rotation (Symmetric + Signed S)	wise ReLU	Multiplication	Hadamard + one row/col.	3	2×	1.3×	1.1×
n=2	R <sub>I2</sub>	$\binom{1}{1}$	Diagonal	$f_{cw}$	None	Identity	2	(2-tuple ring)	2.0×	2.0×
	$(R_{12}, f_{H2})$	(1)	(Component-wise Product) <b>Directional</b> $f_{H2}$		onal $f_{\it H2}$	identity	2		2.0^	2.0^
	$R_{H4}$	grank-4 Permi	grank-4 Permutation (Symmetric)	Commonant	Dia -	Hadamard	1		4.0×	2.6×
	R <sub>O4</sub>	/1\	grank-4 Permutation + Signed S			Reflected Householder	1 4		4.0^	2.0^
	$R_{H4-I}/R_{H4-II}$	$R_{\text{tot}}/R_{\text{tot}} = 1/0$ grank-5 Permutation (+ S for $R_{\text{tot}}$ )		Ring Multiplication	Hadamard + one row/col.	-	4×	3.2×	2.1×	
n=4	$R_{O4-I}/R_{O4-II}$	(0)	grank-5 Permutation + Signed S	£	Multiplication	Householder + one row/col.	3	(4-tuple ring)		2.1^
	H		Rotation (Hamilton Product)	$J_{cw}$		Hadamard + four row/col.	8	(4-tuple fing)	2.0×	1.6×
	R <sub>I4</sub>	/¹\ Diagonal Nor		None	T.d. matitus	4		4.0×	4.0×	
	$(R_{I4}, f_{H4}/f_{O4})$	$\begin{pmatrix} \tilde{1} \\ 1 \end{pmatrix}$	(Component-wise Product)	Direction	al f <sub>H4</sub> /f <sub>O4</sub>	Identity	4		4.0×	4.0×

practical assumptions to confine the scope of discussion. The first one is exclusive sub-product distribution: each input sub-product  $g_k x_j$  in (3) is distributed to one output component  $z_i$  exclusively. It provides complete and non-redundant information mixing between ring components for maintaining compact model capacity. Then G has full rank and can be formulated by a sign matrix S and a permutation indexing matrix S:

$$G_{ij} = S_{ij}g_{P_{ij}}, (9)$$

where  $S_{ij} \in \{1,-1\}$ , and each row or column of P is a permutation of  $\{0,1,...,n-1\}$ . For example, the rotation matrix  $\begin{pmatrix} g_0 & -g_1 \\ g_1 & g_0 \end{pmatrix}$  for  $\mathbb C$  has  $S = \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}$  and  $P = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ .

Furthermore, given the existence of a ring unity 1, we consider the following explicit condition:

$$G = \begin{pmatrix} g_0 & & & \\ g_1 & g_0 & & \\ \vdots & & \ddots & \\ g_{n-1} & & g_0 \end{pmatrix} \text{ and } \mathbf{1} = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}. \tag{C1}$$

Without loss of generality, the condition on the first column of G and  $\mathbf{1}$  is drawn from the permutation definition of P and  $g \cdot \mathbf{1} = g$ . The diagonal of G is then derived by  $\mathbf{1} \cdot g = g$  which states that the isomorphic matrix of  $\mathbf{1}$  is the identity matrix, and therefore  $G = g_0 I$  if  $g = g_0 \mathbf{1}$ .

The second assumption is *commutativity*. It is not necessary for constructing neural networks, e.g. quaternions  $\mathbb H$  are not commutative. But it is sufficient to enable the demanded *associativity* for a ring, together with the exclusive sub-product distribution and an additional condition on commutative permutation. The details are discussed in Appendix B. Then, by examining the matrix form Gx = Xg for  $g \cdot x = x \cdot g$ , we have a cyclic-mapping condition for reducing candidates:

If 
$$P_{ij} = j'$$
, then  $P_{ij'} = j$  and  $S_{ij} = S_{ij'}$ . (C2)

Finally, the last assumption is that a smaller grank(M) is preferred for saving computations and leads to this rule:

$$\mbox{Consider only } S \in \mathop{\arg\min}_{S'} \mathop{\mathrm{grank}}(M(S';P)). \eqno(C3)$$

In practice, for each P satisfying (C1) and (C2) we ran the CP-ARLS algorithm [6] in MATLAB to evaluate

grank(M(S'; P)) for all possible S' and determined ring variants based on the results.

In the following, we consider moderate sparsity for computational imaging with n=2 and 4. We searched new ring variants as mentioned above and determined their transform matrices as discussed in Section III-B. Our findings are listed in Table I where we distinguish ring symbols by indicating nin the subscripts for clarity. For n=2, only  $R_{H2}$  and  $\mathbb{C}$  can satisfy . For n = 4, we found, by exhaustion, that there are two such non-isomorphic permutations. After applying (C3), the minimum grank(M) of them is found to be 4 and 5. The grank-4 permutation leads to two ring variants:  $R_{H4}$  and  $R_{O4}$ which are diagonalized respectively by Hadamard transform H and a reflected Householder matrix  $O = 2L_1(I - 2vv^t)$ where  $L_1 = \text{diag}((1-1-1-1)^t)$  and  $v = \frac{1}{2}(1111)^t$ . On the other hand, there are four grank-5 ring variants. Two of them,  $R_{H4-I}$  and  $R_{H4-II}$ , have transform matrices related to H, and the other two,  $R_{O4\text{-II}}$  and  $R_{O4\text{-II}}$ , are similarly connected to O. In particular,  $R_{H4-I}$  applies circular convolution as Cir-CNN and needs five real multiplications for complex Fourier transform. The details of isomorphic G and fast algorithms are summarized in Table II.

## D. Hardware Efficiency

Now we can systematically examine the benefits of these rings in terms of hardware resources. For concise hardware analysis, we assume that different algebraic structures have the same bitwidths for layer inputs and parameters. Then the weight storage is directly proportional to the degrees of freedom (DoF), and the multiplier complexity can be evaluated on the same basis. Regarding the amount of filter weights, a real-valued network would require  $n^2$  weights for an n-tuple pair of input and output features. But using n-tuple rings instead will only need n real-valued weights to represent the matrix G, i.e. DoF of G is reduced from  $n^2$  to n. Therefore, the efficiency of weight storage with respect to the real-valued networks is  $n \times$ , e.g.  $2 \times$  and  $4 \times$  for 2- and 4-tuple rings respectively. Similarly, the corresponding efficiency in terms of real-valued multiplications can be derived as  $n^2/m$ . In Table

D.					
Ring	Isomorphic Matrix	Filter	Data	Reconstruct.	
Variant	G	transform $T_g$	transform $T_x$	transform $T_z$	
R <sub>H2</sub>	$\begin{pmatrix} g_0 & g_1 \\ g_1 & g_0 \end{pmatrix}$	$H_2 = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$ Hadamard transform		$H_2^{-1} = \frac{H_2}{2}$	
C C	$ \begin{array}{c c}  & \text{HadaNet-alike} \\ \hline \mathbb{C} & \begin{pmatrix} g_0 & -g_1 \\ g_1 & g_0 \end{pmatrix} \\ \text{Complex field} \end{array} $		$\left(\frac{H_2}{1}\right)$	$\left(H_2^{-1} \begin{vmatrix} -1 \\ 0 \end{pmatrix}\right)$	
$R_{I2}$	$\begin{pmatrix} g_0 & 0 \\ 0 & g_1 \end{pmatrix}$		Identity $I_2 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$	0)	
$(R_{I2}, f_{H2})$	Group convolution		17		
R <sub>H4</sub>	$\begin{pmatrix} g_0 & g_1 & g_2 & g_3 \\ g_1 & g_0 & g_3 & g_2 \\ g_2 & g_3 & g_0 & g_1 \\ g_3 & g_2 & g_1 & g_0 \end{pmatrix}$ HadaNet-alike		transform	$H_4^{-1} = \frac{H_4}{4}$	
$R_{O4}$	$\begin{pmatrix} g_0 & g_1 & g_2 & g_3 \\ g_1 & g_0 & -g_3 & -g_2 \\ g_2 & -g_3 & g_0 & -g_1 \\ g_3 & -g_2 & -g_1 & g_0 \end{pmatrix}$	$O = \begin{pmatrix} 1 & -1 & -1 & -1 \\ 1 & -1 & 1 & 1 \\ 1 & 1 & -1 & 1 \\ 1 & 1 & 1 & -1 \end{pmatrix}$ Reflected Householder		$O^{-1} = \frac{O^t}{4}$	
R <sub>H4-I</sub>	$\begin{pmatrix} g_0 & \pm \mathbf{g_3} & g_2 & \pm \mathbf{g_1} \\ g_1 & g_0 & g_3 & g_2 \\ g_2 & \pm \mathbf{g_1} & g_0 & \pm \mathbf{g_3} \\ g_3 & g_2 & g_1 & g_0 \end{pmatrix}$	( H	$\frac{H_4}{0 \mp 1}$ I: blue	$\begin{pmatrix} H_4^{-1} & -1 \\ 0 \\ \pm 1 \\ 0 \end{pmatrix}$	
R <sub>H4-II</sub>	$\begin{pmatrix} g_2 - g_1 & g_0 - g_3 \\ g_3 & g_2 & g_1 & g_0 \end{pmatrix}$ I: CirCNN-alike	0 1	$\begin{pmatrix} n_4 & \pm 1 \\ 0 \end{pmatrix}$		
$R_{O4-I}$	$\begin{pmatrix} g_0 & \pm g_3 & g_2 & \pm g_1 \\ g_1 & g_0 & -g_3 & -g_2 \end{pmatrix}$	( (	$\begin{pmatrix} & & & -1 \\ & & & 0 \end{pmatrix}$		
R <sub>O4-II</sub>	$ \begin{pmatrix} g_2 & \overline{+}  \boldsymbol{g_1} & g_0 & \overline{+}  \boldsymbol{g_3} \\ g_3 & -g_2 & -g_1 & g_0 \end{pmatrix} $	$\left(\frac{O}{0 - 1 - 0 \mp 1}\right)$		$ \left(\begin{array}{c c} O^{-1} & 1 \\ 0 \\ \mp 1 \\ 0 \end{array}\right) $	
Н	$ \begin{pmatrix} g_1 & g_0 & g_3 & -g_2 \\ g_2 & -g_3 & g_0 & g_1 \\ g_3 & g_2 & -g_1 & g_0 \end{pmatrix} $ Ouaternion	$\left(\frac{H_4}{L_2}\right)$	$\left(\frac{H_4}{L_3}\right)$	$-L_1(H_4^{-1} -2I_4)$	
R <sub>I4</sub> (R <sub>I4</sub> , f <sub>H4</sub> ) (R <sub>I4</sub> , f <sub>O4</sub> )	$\begin{pmatrix} g_0 & 0 & 0 & 0 \\ 0 & g_1 & 0 & 0 \\ 0 & 0 & g_2 & 0 \\ 0 & 0 & 0 & g_3 \end{pmatrix}$	Identity $I_4 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$			
R <sub>14</sub> (R <sub>14</sub> , f <sub>144</sub>		$ \begin{pmatrix} g_3 & -g_2 & -g_1 & g_0 \\ g_0 & -g_1 & -g_2 & -g_3 \\ g_1 & g_0 & g_3 & -g_2 \\ g_2 & -g_3 & g_0 & g_1 \\ g_3 & g_2 & -g_1 & g_0 \end{pmatrix} $ $ \begin{matrix} Q_0 & 0 & 0 & 0 \\ 0 & g_1 & 0 & 0 \\ 0 & 0 & g_2 & 0 \\ 0 & 0 & 0 & g_2 \end{matrix} $	$ \begin{pmatrix} g_3 & -g_2 & -g_1 & g_0 \\ g_0 & -g_1 & -g_2 & -g_3 \\ g_1 & g_0 & g_3 & -g_2 \\ g_2 & -g_3 & g_0 & g_1 \\ g_3 & g_2 & -g_1 & g_0 \end{pmatrix} \begin{pmatrix} H_4 \\ \overline{L_2} \end{pmatrix} $ $ \begin{pmatrix} Q_0 & 0 & 0 & 0 \\ 0 & g_1 & 0 & 0 \\ 0 & 0 & g_2 & 0 \\ 0 & 0 & 0 & g_3 \end{pmatrix} $ $ \downarrow \qquad $	$ \begin{pmatrix} g_3 & -g_2 & -g_1 & g_0 \\ g_0 & -g_1 & -g_2 & -g_3 \\ g_1 & g_0 & g_3 & -g_2 \\ g_2 & -g_3 & g_0 & g_1 \\ g_3 & g_2 & -g_1 & g_0 \end{pmatrix}                                  $	

$$\text{Supplementary matrices:} \quad L_1 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}, L_2 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{pmatrix}, L_3 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

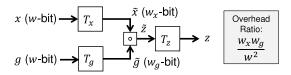


Fig. 3. Fixed-point computation of fast algorithms for ring multiplication.

I, only  $R_I$ ,  $R_H$ , and  $R_{O4}$  can reach the maximum efficiency,  $n \times$ , for full-rank G.

More importantly, for practical implementation we need to consider fixed-point computation and include the bitwidths of the multiplications for precise evaluations. Fig. 3 shows such an example for the fast algorithm (6)-(8). The main overheads brought by the transforms are the increased bitwidths for  $\tilde{x}$  and  $\tilde{g}$ , e.g.  $T_x$  and  $T_g$  will transform w-bit x and y into wider x-bit x and x-bit for evaluating the multiplier consider this factor, x-bit features and weights as shown in the rightmost column of Table I. In this case, only x-bit can reach the maximum efficiency for using identity transforms, and the other rings all suffer the corresponding overheads induced by their transforms. For example, x-bit x-bit features and x-bit features and x-bit features and weights as shown in the rightmost column of Table I. In this case, only x-bit features and weights as shown in the rightmost column of Table I. In this case, only x-bit features and weights as shown in the rightmost column of Table I. In this case, only x-bit features and weights as shown in the rightmost column of Table I. In this case, only x-bit features and weights as shown in the rightmost column of Table I. In this case, only x-bit features and weights as shown in the rightmost column of Table I. In this case, only x-bit features and x-bit features are features.

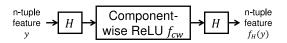


Fig. 4. Proposed directional ReLU  $f_H$ . H:  $n \times n$  Hadamard transform.

# E. Proposed Ring with Directional ReLU

The above discussions only involve linear operations of neural networks. For non-linearity, the component-wise ReLU  $f_{cw}$  is conventionally adopted even when we actually operate on n-tuples. As a result,  $R_I$  will have the worst model capacity, although it has the best hardware efficiency. It is because the information between different components of an n-tuple is not communicated or mixed, which is the same as the discussion on group convolution in [52]. This is also the reason why we assumed the complete information mixing property for searching ring multiplications in Section III-C. In the following, we apply algebraic-architectural co-design to have the hardware advantages of  $R_I$  while recovering the model capacity.

By examining the fast algorithm (6)-(8), we found that the information is in fact mixed by the transforms for data,  $T_x$ , and reconstruction,  $T_z$ . In addition, for neural networks this should be required only near non-linearity because cascaded linear operations will simply degrade to another single linear operator. Based on these two observations, we propose to mix information only before and after non-linearity and thus can adopt  $R_I$  for linear operations to have its architectural benefits. This proposal leads to a novel algebraic function for ring nonlinearity: directional ReLU  $f_{dir}(y) \triangleq U f_{cw}(Vy)$ , where U and V are two  $n \times n$  matrices for an input n-tuple y. It is equivalent to performing non-linearity in the directions of the row vectors of V, instead of the conventional standard axes, and then turning the axes to the column vectors of U. Thus the components of an n-tuple are considered as a whole, not separately, for non-linearity.

The computation of U and V induces complexity overheads. But they are only linearly proportional to the number of output channels, unlike the bitwidth-increased products in (7) which grow quadratically. To further reduce the overhead, we consider the simple Hadamard transform in Table II and propose a novel ring  $(R_I, f_H)$  with the directional ReLU as shown in Fig. 4:

$$f_H(y) \triangleq H f_{cw}(Hy).$$
 (10)

For n=4, another similar variant  $(R_{I4}, f_{O4})$  with  $f_{O4}(y) \triangleq Of_{cw}(Oy)$  is also possible. They have the same hardware advantages as  $R_I$  and possess better model capacity for additional information mixing. Note that for constructing neural networks they are different from  $R_H$  and  $R_{O4}$ , especially when skip connections exist or some convolutions are not followed by non-linearity.

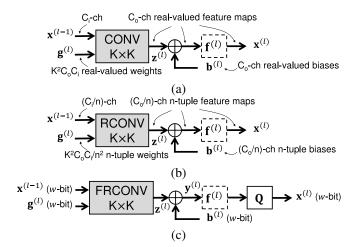


Fig. 5.  $K \times K$  convolution layers with (a) real-valued tensors, (b) n-tuple ring tensors, and (c) efficient implementation.  $C_i$ ,  $C_o$ : number of real-valued input and output channels.  $\mathbf{f}$ : non-linear tensor operation using element-wise f, which could appear or not (dash line) based on model structures.  $\mathbf{Q}$ : tensor quantization using element-wise quantization Q.

#### IV. RINGCNN MODELING

## A. Model Construction

We propose a unified RingCNN framework to include all the considered rings for in-depth comparisons on qualitycomplexity tradeoffs. By extending ring algebra to ring tensors  $\mathbf{z}$ ,  $\mathbf{g}$ , and  $\mathbf{x}$ , we formulate a  $K \times K$  ring convolution (RCONV):

$$\mathbf{z}[p, q, c_o] = \sum_{s, t, c_i} \mathbf{g}[s, t, c_i, c_o] \cdot \mathbf{x}[p - s, q - t, c_i], \quad (11)$$

where  $c_o$  and  $c_i$  are indexes for output and input n-tuple channels, p and q for feature positions, and s and t for weight positions. Then a real-valued convolution layer, either with non-linearity or not, can be converted into an RCONV layer as shown from Fig. 5(a) to (b). In this way, we can convert any existing real-valued model structure into an RingCNN alternative.

# B. Model Training

An RingCNN model can be treated as a conventional real-valued CNN if we implement it in form of the matrix-vector multiplication (4). Then the Backprop algorithm can flow gradients as usual without any special treatment. For the completeness of ring algebras, we can also represent the gradients in terms of ring operations and then express Backprop using only the ring terminology. For example, we have  $\nabla_x L = G^t \nabla_z L$  from (4) for a training loss L. Then  $\nabla_x L = g \cdot \nabla_z L$  for  $R_I$ ,  $R_H$ , and  $R_{O4}$  since G is symmetric for them. Similarly, the gradient  $\nabla_x L$  equals to  $g^c \cdot \nabla_z L$  for  $R_{H4\text{-}I}$  and  $g^* \cdot \nabla_z L$  for  $\mathbb{H}$ , where  $g^c$  and  $g^*$  represent circular folding and quaternion conjugate of g respectively. The same approach can be applied to express  $\nabla_g L$  in ring operations.

# C. Efficient Implementation

**Dynamic fixed-point quantization.** We prefer fixed-point computation for hardware implementation. It has been shown

effective to apply dynamic quantization with separate per-layer Q-formats [1] for real-valued feature maps and parameters [21]. We found that this approach also works well for the RingCNN models that adopt the component-wise ReLU. But when the directional ReLU is applied, image quality is deteriorated in many cases. It is because after this non-linearity different ring components have different dynamic ranges, and using one single Q-format for them causes large saturation errors. Therefore, for the directional ReLU we propose to use component-wise Q-formats for feature maps to address this issue. In other words, there are n different feature Q-formats in one layer, and each component of n-tuple features follows its corresponding Q-format.

**Fast algorithm.** We use the fast algorithm to formulate a fast ring convolution (FRCONV):

$$\mathbf{z}[p, q, c_o] = T_z \left( \sum_{s, t, c_i} \tilde{\mathbf{g}}[s, t, c_i, c_o] \circ \tilde{\mathbf{x}}[p - s, q - t, c_i] \right),$$
(12)

where  $\tilde{\mathbf{g}}$  and  $\tilde{\mathbf{x}}$  are the ring tensors after the transforms  $T_g$  and  $T_x$  respectively. For minimizing overheads, we avoid redundant transform operations by applying  $T_g$ ,  $T_x$ , and  $T_z$  only once for each of weight, input, and output ring elements respectively. Then each RCONV layer can then be efficiently implemented in hardware by applying FRCONV to its fixed-point model as shown in Fig. 5(c). Note that for  $R_I$  FRCONV is the same as RCONV for its identity transform matrices.

# V. ERINGCNN ACCELERATOR

To show the efficiency on practical applications, we further design an RingCNN accelerator, named eRingCNN, over the proposed ring  $(R_I, f_H)$ . For supporting high-throughput computational imaging, we use the highly-parallel eCNN as a backbone architecture and simply replace its real-valued convolution engine by a corresponding one for RCONV. This portability of linear operations is an advantage of algebraic sparsity, but we need a new and specific design for the directional ReLU. We implement two sparsity settings for n=2 and 4, and the details are introduced in the following.

**System diagram.** Fig. 6 presents the overall architecture. In one cycle, it can compute (32/n)-channel n-tuple output features from (32/n)-channel n-tuple inputs for  $4\times 2$  spatial positions. For both  $3\times 3$  and  $1\times 1$  convolution engines, the number of MACs is reduced by 50% and 75% for the settings n=2 and 4 respectively. Similarly, the size of the weight memory can be reduced by the same ratios, e.g. from 1280 KB in eCNN to 640 KB for n=2 and 320 KB for n=4. However, for simplicity the parameter compression in eCNN was not implemented; instead, we increase the size by  $1.5\times$  to 960 KB and 480 KB, respectively, to support large models. The rest architectural differences from eCNN are mainly on the designs of the RCONV engines and the novel directional ReLU

**RCONV engine.** To have local sparsity while maintaining global regularity, we increase the computing granularity from

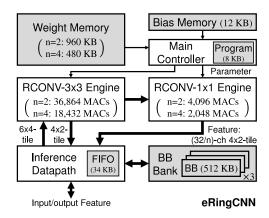


Fig. 6. System architecture of eRingCNN. (BB: image block buffer)

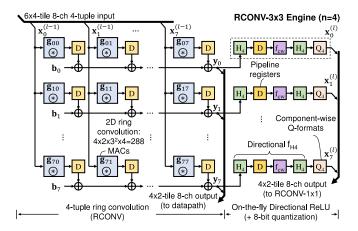


Fig. 7. RCONV engine for  $3\times3$  filters and 4-tuples. It processes eight 4-tuple input channels and generates eight 4-tuple output channels, which is equivalent to 32-channel real-valued inputs and outputs. It has 64 computing units (blue boxes) in which  $\circledast$  represents a 2D ring convolution unit which generates a 4×2-tile in one cycle, i.e.  $\mathbf{y}_i$  represents the 4-tuples without linearity in 4×2 spatial positions for the *i*-th output channel.

real numbers to n-tuple rings. Fig. 7 shows such a modification for the  $3\times3$  convolution engine with n=4. It is a channel-wise 2D computation array for 8-channel 4-tuple inputs and outputs. Each of the  $8\times8$  computing units is responsible for the 2D  $3\times3$  ring convolution for the corresponding input-output pair with ring tensor weights  $\mathbf{g}_{c_oc_i}\triangleq\mathbf{g}[:,:,c_i,c_o]$ . Thanks to  $(R_I,f_H)$ , it simply computes component-wise 2D convolutions for saving complexity. Finally, a novel directional ReLU block, including dynamic quantization with component-wise Q-formats, is devised to replace their real-valued counterparts.

**Directional ReLU unit.** It mixes information for RCONV outputs to recover model capacity; however, the mixing demands Hadamard transforms on high-bitwidth accumulated outputs, e.g. 24-bit for n=4. This induces two issues for conventional accelerator architectures. Firstly, the two transforms for  $f_H$  are likely to be implemented by the same fixed-point MACs for convolutions to meet the high computation throughput. But since the weights are only -1 and 1, the hardware efficiency would be low for the multipliers. Secondly and more importantly, the features before the Hadamard transforms

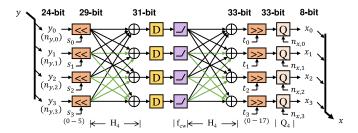


Fig. 8. On-the-fly directional ReLU for a 4-tuple (dashed box in Fig. 7). The input is  $y=(y_0,y_1,y_2,y_3)^t$ , and the output  $x=(x_0,x_1,x_2,x_3)^t$ . With component-wise Q-formats, their numbers of fractional bits are given as  $n_{y,i}$  and  $n_{x,i}$ , respectively. The numbers of shift bits are then derived as  $s_i=\max_{i'}n_{y,i'}-n_{y,i}$  and  $t_i=\max_{i'}n_{y,i'}-n_{x,i}$ . Green lines represent minus terms of the adders for Hadamard transform.

will need to be quantized for the MACs. We found that these additional quantizations, e.g. 24-bit to 8-bit for n=4, would cause up to 0.2 dB of PSNR drop for denoising and SR tasks.

Therefore, we propose an on-the-fly processing pipeline for this novel function, and Fig. 8 shows our implementation for n=4. It specifically implements the butterfly structures for Hadamard transforms to optimize hardware efficiency and keeps full-precision operations to preserve image quality. In this case, the internal bitwidths are up to 33-bit, in which the component-wise Q-formats contribute 5-bit for aligning components (through the left-shifters). This circuit is the major overhead for using  $(R_I, f_H)$  and also appears in the inference datapath for the non-linearity after skip or residual connections.

#### VI. EVALUATIONS

We show extensive evaluations for (A) ring algebras, (B) image quality on eRingCNN, and (C) hardware performance of eRingCNN. For clarity, the two sparsity configurations for eRingCNN are denoted by eRingCNN-n2 and eRingCNN-n4.

# A. Ring Algebras

Training setting and test datasets. For quality evaluations, we use the advanced ERNets for eCNN [21] as the real-valued backbone models. Then RingCNN models are converted from them as shown from Fig. 5(a) to (b). To fairly compare RingCNNs and real-valued CNNs, we evaluate their best performance by increasing their initial learning rates as high as possible before training procedures become unstable. Note that the real-valued ERNets in this paper will therefore perform better than those in [21] because of using higher learning rates. The models are trained using the lightweight settings as summarized in Table III if not mentioned. Finally, we test denoising networks on datasets Set5 [7], Set14 [48], and CBSD68 [36], and super-resolution ones on Set5, Set14, BSD100 [36], and Urban100 [22].

Quality comparison for different rings. Fig. 9 compares image quality in PSNR for the rings in Table I. When the component-wise ReLU is used,  $R_I$  performs the worst due to the lack of information mixing. The two traditional algebra alternatives  $\mathbb C$  and  $\mathbb H$  also do not perform well, considering

## TABLE III TRAINING SETTINGS.

Setting	Algebra	Model	Dataset	Patch	Patch #	Learning Rate
Lightweight	Real/Ring	SR/Dn	DIV2K	48×48	4.8M	4 · 10 <sup>-4</sup> · 2 <sup>-k</sup> , k=0-1
	Real/Ring	SR	DIV2K		9.6M 19.2M	4 · 10 <sup>-4</sup> · 2 <sup>-k</sup> , k=0-4
Polishment	Real	Dn	Waterloo	96×96		1 · 10 <sup>-3</sup> · 10 <sup>-k</sup> , k=0-2
	Ring	Dii				5 · 10 <sup>-4</sup> · 10 <sup>-k</sup> , k=0-2
Fine-tune for Quantization	Real/Ring	SR	DIV2K	96×96	3.2M	$1 \cdot 10^{-4} \cdot 2^{-k}$ , k=5-6
	Real/Ring	Dn	Waterloo	90^90	12.8M	1 · 10 <sup>-5</sup> · 2 <sup>-k</sup> , k=0-2

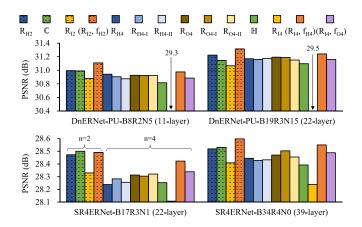


Fig. 9. PSNR comparison of different rings with (top) denoising model structure DnERNet-PU (PU: pixel unshuffle) and (bottom) four-times superresolution (SR $\times$ 4) SR4ERNet. The configurations are the same as the real-valued ERNets: ERModule number B, base pumping ratio R, and additional pumping layer number N. (Hatch pattern: 2-tuples; solid color: 4-tuples.)

more real-valued multiplications are required. Between the two grank-4 variants for n=4, the newly-discovered  $R_{O4}$  performs better than the HadaNet-alike  $R_{H4}$ . Similar results can be found for their corresponding grank-5 variants, e.g. the newly-discovered  $R_{O4\text{--}I}$  better than the CirCNN-alike  $R_{H4\text{--}I}$ . However, by using the directional ReLU, the proposed  $(R_I, f_H)$  can give better quality and constantly outperform the others. Since  $(R_{I4}, f_{O4})$  shows inferior quality, we therefore focus on  $(R_I, f_H)$  and adopt it for our implementation.

Ablation study between  $(R_I, f_H)$  and  $R_H$ . They share similar structures but have two major differences. First,  $(R_I, f_H)$  multiplies input features by weights  ${\bf g}$  directly while  $R_H$  does that after applying the filter transform. Second,  $(R_I, f_H)$  applies Hadamard transform only when non-linearity is required, but  $R_H$  always does that and results in a redundant structure. Therefore,  $R_H$  can imitate  $(R_I, f_H)$  by making up the differences: first training on transformed weights  $\tilde{{\bf g}}$  and then modifying model structures accordingly. Fig. 10(a) shows an example for modifying a residual block, and Fig. 10(b) illustrates typical PSNR results using two SR×4 networks as examples. Training on  $\tilde{{\bf g}}$  is occasionally helpful, but structure modification improves image quality most of the time. Therefore, the compact model structure is the main reason why  $(R_I, f_H)$  outperforms  $R_H$  for computational imaging.

Comparison with weight pruning. We also compare image quality between the proposed algebraic sparsity and the unstructured magnitude-based weight pruning. While RingCNNs

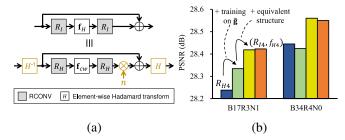


Fig. 10. Ablation study for  $(R_I, f_H)$ . (a) Equivalent residual block structures for using (top)  $(R_I, f_H)$  and (bottom)  $R_H$ . (b) PSNR for two SR4ERNet model structures (n=4).

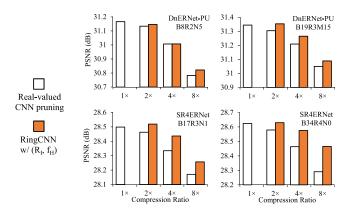


Fig. 11. Algebraically-sparse RingCNN versus unstructured weight pruning.  $(R_I, f_H)$ : n=2, 4, and 8 for  $2\times$ ,  $4\times$ , and  $8\times$  compressions. We use 200 more epochs for fine-tuning the weight-pruned models and, for fair comparisons, 100 more epochs for the original  $(1\times)$  CNN and RingCNNs.

are trained directly, real-valued CNNs are first pre-trained, then pruned, and finally fine-tuned. Fig. 11 shows the comparison results, and RingCNNs over  $(R_I, f_H)$  can deliver better image quality than the weight pruing for compression ratios  $2\times$ ,  $4\times$ , and  $8\times$ . In particular, the 2-tuple networks can even outperform the original  $(1\times)$  real-valued networks in many cases. This shows that the algebraic sparsity can serve strong prior for CNN models. As a result,  $(R_I, f_H)$  not only provides more regular structures but also achieves better quality for computational imaging. A case for recognition tasks, though not the focus of this paper, is also studied in Appendix C, where convolutions and corresponding non-linear functions are implemented with  $(R_I, f_H)$ , and batch normalization is remained as real-valued operations.

**Fixed-point implementation.** For comparisons in hardware efficiency, we implemented highly-parallel FRCONV engines, as depicted in Fig. 5(c) with non-linearity, for different rings. Their RTL codes are synthesized with 40 nm CMOS technology. For quality comparison, the models are quantized in 8-bit and then fine-tuned using the setting at the bottom of Table III. The quality loss due to quantization is similar for each ring, and Fig. 12 shows the comparison results. The area efficiencies are very close to the estimated 8-bit complexity in Table I because convolutions dominate the areas. The proposed  $(R_I, f_H)$  can provide the smallest area and the best quality

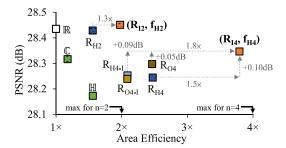


Fig. 12. Area efficiency after logic synthesis versus PSNR for 8-bit fixed-point implementation. These engines can compute one 3×3 convolution layer in one cycle for 32 input and 32 output channels of 8-bit real-valued features. Area efficiencies are calculated with respect to the real-valued engine, and PSNR is evaluated on SR4ERNet-B17R3N1.

TABLE IV PSNR PERFORMANCE AND COMPARISON FOR MODELS ON ERINGCNN.

	ERN	et@HD3	30 for	ERNe	ERNet@UHD30 for		
Application Scenario	eCNN eRingCNN		eCNN eRingCl		<b>CNN</b>		
	(real)	(n=2)	(n=4)	(real)	(n=2)	(n=4)	
Dn Model (σ=25)	Dı	nERNet-l	PU	DnERNet-PU			
on CBSD68	B19R3N15 (22-layer)		B8R2N5 (11-layer)		ayer)		
PSNR (dB)	31.36	31.36	31.27	31.19	31.17	31.00	
PSNR Gain to CBM3D (dB)	0.66	0.66	0.57	0.49	0.47	0.30	
PSNR Gain to FFDNet (dB)	0.15	0.15	0.06	-0.02	-0.04	-0.21	
SR×4 Model	S	SR4ERNet :			SR4ERNet		
on Set5/Set14/BSD100	B34R	4N0 (39-layer)		B17R3N1 (2:		layer)	
PSNR (dB)	29.47	29.51	29.47	29.40	29.40	29.33	
PSNR Gain to VDSR (dB)	0.58	0.63	0.59	0.52	0.52	0.44	
PSNR Gain to SRResNet (dB)	0.10	0.14	0.10	0.03	0.03	-0.04	

at the same time. Compared to the CirCNN-alike  $R_{H4\text{-I}}$  and HadaNet-alike  $R_{H4}$ , it has nearly 0.1 dB PSNR gain for the SR×4 task and provides  $1.8\times$  and  $1.5\times$  area efficiencies respectively. In summary,  $R_I$  can save area efficiently, and  $f_H$  can recover image quality significantly.

## B. Image Quality on eRingCNN

Training setting and model selection. To show competitive image quality, we further train models using the polishment setting in Table III with two large datasets, DIV2K [2] and Waterloo Exploration [33]. We consider two throughput targets for hardware acceleration: HD30 for Full HD 30 fps and UHD30 for 4K Ultra-HD 30 fps. For each throughput target and application scenario, we adopt the compact ERNet configuration for the real-valued eCNN in [21]. It has been optimized over model depth and width in terms of PSNR, and we build its corresponding RingCNN models with  $(R_I, f_H)$ .

Floating-point models. The PSNR results are shown in Table IV. The RingCNN models show significant gains over the traditional CBM3D [11] for denoising and VDSR for SR×4. Compared to the advanced FFDNet and SRResNet, the models for eRingCNN-n2 can outperform them with PSNR gains up to 0.15 dB at HD30 and have similar quality at UHD30. With 75% sparsity, the models for eRingCNN-n4 still give superior quality and only show noticeable PSNR inferiority for denoising at UHD30 due to the shallow layers.

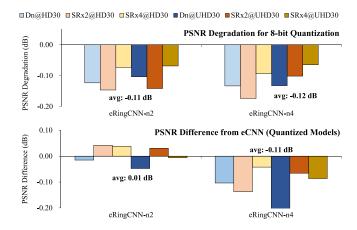


Fig. 13. Average PSNR results for six application targets: (top) PSNR degradation of 8-bit quantized ERNet models from float-point ones and (bottom) PSNR differences of quantized models for eRingCNN from those for eCNN. The test datasets for denoising (Dn) are Set5, Set14, and CBSD68, and those for super-resolution (SR) are Set5, Set14, BSD100, and Urban100.

**Dynamic fixed-point quantization.** On the top of Fig. 13, we show the effect of the 8-bit dynamic quantization which is used to save area. The quality degradation for ring tensors is around 0.11-0.12 dB of average PSNR drops, which is similar to the case of using real numbers. We also show the effect of applying sparse ring algebras (eCNN $\Rightarrow$ eRingCNN) at the bottom of Fig. 13. The degradation is not obvious for n=2, and the models for eRingCNN-n2 even outperform those for eCNN by 0.01 dB on average. For n=4, those for eRingCNN-n4 only suffer small PSNR degradation in 0.11 dB.

## C. Hardware Performance of eRingCNN

Implementation and CAD tools. We developed RTL codes in Verilog and verified the functional validity based on bit- and cycle-accurate simulations. Then the verified RTL codes were synthesized using Synopsys Design Compiler with TSMC 40 nm CMOS library. And SRAM macros were generated by ARM memory compilers. We used Synopsys IC Compiler for placement and routing and generating layouts for five well-pipelined macro circuits which constitute eRingCNN collectively. Finally, we performed time-based power consumption using Synopsys Prime-Time PX based on post-layout parasitics and dynamic signal activity waveforms from RTL simulation.

**Hardware performance.** We show the design configurations and layout performance in Table V. The areas of eRingCNN-n2 and eRingCNN-n4 are 33.73 and 23.36 mm<sup>2</sup> respectively, and the corresponding power consumptions are 3.76 and 2.22 W. They mainly differ in the ring dimension, and eRingCNN-n4 uses only a half number of MACs and a half size of weight memory compared to eRingCNN-n2.

**Area and power breakdown.** The details are shown in Table VI. For eRingCNN-n2, the convolution engines contribute 57.42% of area and 86.51% of power consumption for the highly-parallel computation. And for eRingCNN-n4 their contributions go down to 45.63% and 76.56%, respectively,

 $\label{table V} TABLE\ V$  Design configurations and layout performance of eRingCNN.

Design	eRingCNN-n2	eRingCNN-n4		
Technology	40 nm			
Frequency (MHz)	250			
Number of MAC	40,960	20,480		
SRAM (KB)	2,550	2,070		
Quantization Precision	Dynamic 8-bit			
Area (mm²)	33.73	23.36		
Power (W)	3.76	2.22		

TABLE VI Area and power breakdowns.

Computation/	eRingC	NN-n2	eRingCNN-n4		
Storage Unit	Area (mm²)	Power (W)	Area (mm²)	Power (W)	
RCONV-3×3	17.50 (51.86%)	3.02 (80.32%)	9.61 (41.13%)	1.56 (70.23%)	
RCONV-1×1	1.88 ( 5.56%)	0.23 ( 6.19%)	1.05 ( 4.50%)	0.14 ( 6.33%)	
Inference Datapath	3.85 (11.43%)	0.23 ( 6.08%)	4.38 (18.74%)	0.27 (12.19%)	
Main Controller	0.73 ( 2.15%)	0.00 ( 0.10%)	0.32 ( 1.36%)	0.00 ( 0.12%)	
Weight/Bias Memory	3.54 (10.50%)	0.05 ( 1.25%)	1.77 ( 7.57%)	0.02 ( 0.84%)	
Block Buffer Bank	6.24 (18.49%)	0.23 ( 6.07%)	6.24 (26.71%)	0.23 (10.28%)	
Total	33.73	3.76	23.36	2.20	

because of the saving of MACs. In addition, for a larger n the directional ReLU uses more adders and causes wider bitwidths. Therefore, for the RCONV-3×3 engines it occupies only 3.4% of area for eRingCNN-n2 but up to 8.9% for eRingCNN-n4. Accordingly, the inference datapath in eRingCNN-n4 is also 0.53 mm<sup>2</sup> larger than that in eRingCNN-n2.

Comparison with eCNN. As shown in Fig. 14, the RCONV engines reach near-maximum hardware efficiencies ( $\cong n$ ). Those in eRingCNN-n2 achieve  $2.08\times$  area efficiency and  $2.00\times$  energy efficiency. And those in eRingCNN-n4 further increase the efficiency gains of area and energy to  $3.77\times$  and  $3.84\times$ . The numbers for the whole accelerator are as high as  $1.64\times$  and  $1.85\times$  for eRingCNN-n2, and  $2.36\times$  and  $3.12\times$  for eRingCNN-n4. In addition, we compare their quality-energy tradeoff curves in Fig. 15. The eRingCNN accelerators show clear advantages over eCNN; in particular, the low-complexity eRingCNN-n4 is preferred when less energy is allowed to be consumed for generating one pixel. Finally, as eCNN, the eRingCNN accelerators demand only 1.93 GB/s of DRAM bandwidth for high-quality 4K UHD applications.

Comparison with Diffy. Table VII compares the hardware performance of Diffy $^1$  [34], another state-of-the-art accelerator for computational imaging, along with eCNN. Diffy applies optimization on bit-level computation which is hard to compare with eRingCNN directly; therefore, we perform comparison based on the same application target: FFDNet-level inference at Full-HD 20fps. In this case, the energy efficiencies of eRingCNN-n2 and eRingCNN-n4 over Diffy are  $2.71\times$  and  $4.59\times$  respectively by running at 167 MHz.

Comparison with SparTen, TIE, and CirCNN. Table VIII compares with the state-of-the-arts for different spar-

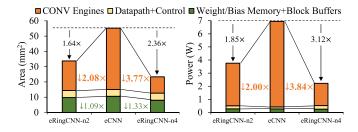


Fig. 14. Area (left) and power (right) comparison with eCNN.

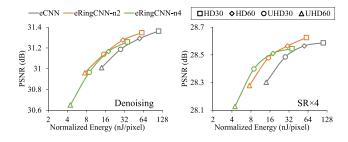


Fig. 15. Quality-energy comparison with eCNN: (left) denoising and (right) SR×4. The energy is normalized for generating one image pixel. Each accelerator forms its own curve with compact model configurations over different throughput targets.

sity approaches: SparTen (natural) [16], TIE (low-rank) [12], and CirCNN (full-rank). Here we compare synthesis results because only such numbers are reported for SparTen and CirCNN. For comparing over different compression ratios, we consider an equivalent throughput which corresponds to the computing demand of the target uncompressed or realvalued model. With only 2-4× compression, our eRingCNN accelerators already provide competitive energy efficiencies as equivalent 19.1-28.4 TOPS/W. In contrast, SparTen merely achieves 2.7 TOPS/W due to significant overheads for handling irregularity. Although TIE is very efficient for highlycompressed fully-connected (FC) layers, it shows inefficiency for the CONV layers with lower compression ratios. Finally, CirCNN only provides 10.0 TOPS/W using as high as 66× compression. The potential of algebraic sparsity is therefore demonstrated, in particular on moderately-compressed CNNs for computational imaging.

## VII. RELATED WORK

Low-rank sparsity. This line of research also provides regular structure for efficient hardware acceleration. One approach is low-rank approximation, including tensor-train (TT) [37], canonical polyadic (CP) [30], and Tucker [27]. Another one is building networks using low-rank structures, such as MobileNet (v1/v2) [19], [41] and SqueezeNet [23]. However, this sparsity mainly aims to provide high compression ratios, and the effect for computational imaging need further exploration

**Natural sparsity.** Exploiting such sparsity in features has been well-studied, e.g. in Cnvlutin [3], Cambricon-X [51], and Diffy [34]. And the sparsity in filter weights is usually explored by pruning [17], [35]. They can be further combined

 $<sup>^{1}</sup>$ We project the silicon area and power consumption of Diffy under 40 nm technology based on the scaling comparison of 65 nm in [45]:  $2.35 \times$  gate density and  $0.5 \times$  power consumption under the same operation speed.

Design	eCNN	Diffy		eRingCNN-n2	eRingCNN-n4
Technology	40 nm	65 nm (reported)	40 nm (projected)	40	nm
Frequency (MHz)	250	10	00	2:	50
Area (mm²)	55.23	29.22 12.43		33.73	23.36
Power (W)	6.94	13.58	6.79	3.76	2.22
SRAM for Parameters (KB)	1288	512		972	492
SRAM for Features (KB)	1570 528		28	1570	
Quantization	Dynamic 8-bit	Layer-wise (≤16-bit)		Dynamic 8-bit	
Model@Spec	ERNet @UHD30	FFDNet@HD20 VDSR@HD9		ERNet with RingCNN@UHD30	
Power (W) @ HD20, FFDNet-level	4.63	13.58 6.79		2.51 (2.71↓)	1.48 (4.59↓)

TABLE VIII
COMPARISON OF SPARTEN, TIE, AND CIRCNN (SYNTHESIS RESULTS).

Design	СизиТан	TI	E.	CirCNN	eRing	gCNN
Design	SparTen	11	E	CITCININ	n2	n4
Sparsity Type	Natural	Low-rank		Full-rank	Algebraic	
Technology	45nm	28nm		45nm	40nm	
Frequency (MHz)	800	1000		200	250	
Power (W)	Power (W) 0.12 0.10		0.08	2.14	1.45	
Quantization	8-bit	16-bit		16-bit	Dynamic 8-bit	
Application		Recog	nition		Comput. Imaging	
	3.1×	<7.4×	>4608×			
Compression Ratio	@VGG16	@VGG16	@ Four	66×	2×	4×
@ Model	CONV	CONV	FC	@AlexNet	@ERNet	@ERNet
	layers	layers	layers			
Equivalent Throughput (TOPS)	0.3 (:: 6.3× speedup)	0.2 (:: 6.72 frame/s)	7.64	0.8	41.0	
Equivalent Efficiency (TOPS/W)	2.7	1.9	72.9	10.0	19.1	28.4

as in SCNN [38], SparTen [16], and SmartExchange [54]. However, the natural irregularity could lead to high overheads for indexing circuits and load imbalance.

**Dense CNN accelerators.** There are many accelerators proposed for general-purpose inference with high parallelism, such as TPU [25], DNPU [42], ShiDianNao [14], and Eyeriss (v1/v2) [8], [9]. However, the computation sparsity for computational imaging were seldom exploited.

Block-based inference flows. They eliminate huge external memory bandwidth for feature maps, and two approaches were proposed to handle the boundary features across neighboring blocks. One is feature reusing, such as fused-layer [4] and Shortcut Mining [5], and the other one is recomputing, like eCNN [21]. In this paper, we adopt the latter only for the purpose of implementation and comparison.

# VIII. CONCLUSION

This paper investigates the fundamental but seldomexplored algebraic sparsity for accelerating computational imaging CNNs. It can provide local sparsity and global regularity at the same time for energy-efficient inference. We lay down the general RingCNN framework by defining proper ring algebras and constructing corresponding CNN models. By extensive comparisons with several rings, the proposed one with the directional ReLU achieves near-maximum hardware efficiency and the best image quality simultaneously. We also design two high-performance eRingCNN accelerators for verifying practical effectiveness. They can provide high-quality computational imaging at up to 4K UHD 30 fps while consuming only 3.76 W and 2.22 W, respectively. Based on these results, we believe that RingCNN exhibits great potentials for enabling next-generation cameras and displays with energy-efficient and high-performance computational imaging.

#### ACKNOWLEDGMENT

The author would like to thank the anonymous reviewers for their feedback and suggestions. I also would like to thank Chi-Wen Weng for his help on layout implementation.

## REFERENCES

- [1] "Q-format," in ARM Developer Suite—AXD and armsd Debuggers Guide. ARM Limited, 2001, ch. 4.7.9.
- [2] E. Agustsson and R. Timofte, "NTIRE 2017 challenge on single image super-resolution: Dataset and study," in *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2017.
- [3] J. Albericio, P. Judd, T. Hetherington, T. Aamodt, N. E. Jerger, and A. Moshovos, "Cnvlutin: Ineffectual-neuron-free deep neural network computing," in *Proceedings of the 43rd Annual ACM/IEEE International* Symposium on Computer Architecture (ISCA), 2016.
- [4] M. Alwani, H. Chen, M. Ferdman, and P. Milder, "Fused-layer CNN accelerators," in *Proceedings of the 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2016.
- [5] A. Azizimazreah and L. Chen, "Shortcut Mining: Exploiting cross-layer shortcut reuse in DCNN accelerators," in *IEEE International Symposium* on High Performance Computer Architecture (HPCA), 2019.
- [6] C. Battaglino, G. Ballard, and T. G. Kolda, "A practical randomized CP tensor decomposition," SIAM Journal on Matrix Analysis and Applications, vol. 39, 2018.
- [7] M. Bevilacqua, A. Roumy, C. Guillemot, and M.-L. A. Morel, "Low-complexity single-image super-resolution based on nonnegative neighbor embedding," in *British Machine Vision Conference (BMVC)*, 2012.
- [8] Y.-H. Chen, J. Emer, and V. Sze, "Eyeriss: a spatial architecture for energy-efficient dataflow for convolutional neural networks," in Proceedings of the 43rd Annual ACM/IEEE International Symposium on Computer Architecture (ISCA), 2016.
- [9] Y.-H. Chen, T.-J. Yang, J. Emer, and V. Sze, "Eyeriss v2: A flexible accelerator for emerging deep neural networks on mobile devices," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 9, 2019.
- [10] T.-W. Chin, R. Ding, C. Zhang, and D. Marculescu, "Towards efficient model compression via learned global ranking," in *IEEE/CVF Confer*ence on Computer Vision and Pattern Recognition (CVPR), 2020.
- [11] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising by sparse 3-D transform-domain collaborative filtering," *IEEE Transactions* on *Image Processing*, vol. 16, 2007.
- [12] C. Deng, F. Sun, X. Qian, J. Lin, Z. Wang, and B. Yuan, "TIE: Energy-efficient tensor train-based inference engine for deep neural network," in *Proceedings of the 46th Annual ACM/IEEE International Symposium on Computer Architecture (ISCA)*, 2019.
- [13] C. Ding, S. Liao, Y. Wang, Z. Li, N. Liu, Y. Zhou, C. Wang, X. Qian, Y. Bai, G. Yuan, X. Ma, Y. Zhang, J. Tang, Q. Qiu, X. Lin, and B. Yuan, "CirCNN: Accelerating and compressing deep neural networks using block-circulant weight matrices," in *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2017
- [14] Z. Du, R. Fasthuber, T. Chen, P. Ienne, L. Li, T. Luo, X. Feng, Y. Chen, and O. Temam, "ShiDianNao: Shifting vision processing closer to the sensor," in *Proceedings of the 42nd Annual ACM/IEEE International Symposium on Computer Architecture (ISCA)*, 2015.
- [15] C. J. Gaudet and A. S. Maida, "Deep quaternion networks," in *International Joint Conference on Neural Networks (IJCNN)*, 2018.
- [16] A. Gondimalla, N. Chesnut, M. Thottethodi, and T. N. Vijaykumar, "SparTen: A sparse tensor accelerator for convolutional neural networks," in *Proceedings of the 52nd Annual IEEE/ACM International* Symposium on Microarchitecture (MICRO), 2019.

- [17] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," in *International Conference on Learning Representations (ICLR)*, 2016.
- [18] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- [19] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient convolutional neural networks for mobile vision applications," arXiv:1704.04861, 2017.
- [20] T. D. Howell and J.-C. Lafon, "The complexity of the quaternion product," in TR 75-245. Cornell University, 1975.
- [21] C.-T. Huang, Y.-C. Ding, H.-C. Wang, C.-W. Weng, K.-P. Lin, L.-W. Wang, and L.-D. Chen, "eCNN: A block-based and highly-parallel CNN accelerator for edge inference," in *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2019.
- [22] J.-B. Huang, A. Singh, and N. Ahuja, "Single image super-resolution from tranformed self-exemplars," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.
- [23] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size," arXiv:1602.07360, 2017.</p>
- [24] J. Johnson, A. Alahi, and L. Fei-Fei, "Perceptual losses for realtime style transfer and super-resolution," in *European Conference on Computer Vision (ECCV)*, 2016.
- [25] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers, R. Boyle, P.-l. Cantin, C. Chao, C. Clark, J. Coriell, M. Daley, M. Dau, J. Dean, B. Gelb, T. V. Ghaemmaghami, R. Gottipati, W. Gulland, R. Hagmann, C. R. Ho, D. Hogberg, J. Hu, R. Hundt, D. Hurt, J. Ibarz, A. Jaffey, A. Jaworski, A. Kaplan, H. Khaitan, D. Killebrew, A. Koch, N. Kumar, S. Lacy, J. Laudon, J. Law, D. Le, C. Leary, Z. Liu, K. Lucke, A. Lundin, G. MacKean, A. Maggiore, M. Mahony, K. Miller, R. Nagarajan, R. Narayanaswami, R. Ni, K. Nix, T. Norrie, M. Omernick, N. Penukonda, A. Phelps, J. Ross, M. Ross, A. Salek, E. Samadiani, C. Severn, G. Sizikov, M. Snelham, J. Souter, D. Steinberg, A. Swing, M. Tan, G. Thorson, B. Tian, H. Toma, E. Tuttle, V. Vasudevan, R. Walter, W. Wang, E. Wilcox, and D. H. Yoon, "In-datacenter performance analysis of a tensor processing unit," in Proceedings of the 44th Annual ACM/IEEE International Symposium on Computer Architecture (ISCA), 2017.
- [26] J. Kim, J. K. Lee, and K. M. Lee, "Accurate image super-resolution using very deep convolutional networks," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- [27] Y.-D. Kim, E. Park, S. Yoo, T. Choi, L. Yang, and D. Shin, "Compression of deep convolutional neural networks for fast and low power mobile applications," in *International Conference on Learning Representations* (ICLR), 2016.
- [28] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," 2009.
- [29] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proceedings of the 25th International Conference on Neural Information Processing Systems* (NIPS), 2012.
- [30] V. Lebedev, Y. Ganin, M. Rakhuba, I. Oseledets, and V. Lempitsky, "Speeding-up convolutional neural networks using fine-tuned CPdecomposition," in *International Conference on Learning Representa*tions (ICLR), 2015.
- [31] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi, "Photo-realistic single image super-resolution using a generative adversarial network," in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.
- [32] B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee, "Enhanced deep residual networks for single image super-resolution," in *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2017.
- [33] K. Ma, Z. Duanmu, Q. Wu, Z. Wang, H. Yong, H. Li, and L. Zhang, "Waterloo Exploration Database: New challenges for image quality assessment models," *IEEE Transactions on Image Processing*, vol. 26, 2017.
- [34] M. Mahmoud, K. Su, and A. Moshovos, "Diffy: a déjà vu-free differential deep neural network accelerator," in *Proceedings of the 51st Annual*

- IEEE/ACM International Symposium on Microarchitecture (MICRO), 2018.
- [35] H. Mao, S. Han, J. Pool, W. Li, X. Liu, Y. Wang, and W. J. Dally, "Exploring the regularity of sparse structure in convolutional neural networks," in *IEEE Conference on Computer Vision and Pattern Recog*nition Workshops (CVPRW), 2017.
- [36] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *IEEE International Conference on Computer Vision (ICCV)*, 2001.
- [37] I. V. Oseledets, "Tensor-train decomposition," SIAM Journal on Scientific Computing, 2011.
- [38] A. Parashar, M. Rhu, A. Mukkara, A. Puglielli, R. Venkatesan, B. Khailany, J. Emer, S. W. Keckler, and W. J. Dally, "SCNN: An accelerator for compressed-sparse convolutional neural networks," in Proceedings of the 44th Annual ACM/IEEE International Symposium on Computer Architecture (ISCA), 2017.
- [39] T. Parcollet, M. Morchid, and G. Linarès, "Quaternion convolutional neural networks for heterogeneous image processing," in *IEEE International Conference on Acoustics, Speech and Signal Processing* (ICASSP), 2019.
- [40] A. Renda, J. Frankle, and M. Carbin, "Comparing rewinding and finetuning in neural network pruning," in *International Conference on Learning Representations (ICLR)*, 2020.
- [41] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.
- [42] D. Shin, J. Lee, J. Lee, and H.-J. Yoo, "DNPU: An 8.1TOPS/W reconfigurable CNN-RNN processor for general-purpose deep neural networks," in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2017.
- [43] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations (ICLR)*, 2015.
- [44] C. Trabelsi, O. Bilaniuk, Y. Zhang, D. Serdyuk, S. Subramanian, J. F. Santos, S. Mehri, N. Rostamzadeh, Y. Bengio, and C. J. Pal, "Deep complex networks," in *International Conference on Learning Representations (ICLR)*, 2018.
- [45] TSMC, 40nm Technology. [Online]. https://www.tsmc.com/english/dedicatedFoundry/technology/logic/l\_40nm, Retrieved 20 Nov 2020.
- [46] S. Winograd, Arithmetic Complexity of Computations. Cambridge University Press, 1980.
- [47] J. Yu, Y. Fan, and T. Huang, "Wide activation for efficient image and video super-resolution," in *British Machine Vision Conference (BMVC)*, 2019.
- [48] R. Zeyde, M. Elad, and M. Protter, "On single image scale-up using sparse-representations," in *Proceedings of the International Conference* on Curves and Surfaces, 2010.
- [49] K. Zhang, W. Zuo, Y. Chen, D. Meng, and L. Zhang, "Beyond a Gaussian denoiser: Residual learning of deep CNN for image denoising," *IEEE Transactions on Image Processing*, vol. 26, 2017.
- [50] K. Zhang, W. Zuo, and L. Zhang, "FFDNet: Toward a fast and flexible solution for CNN-based image denoising," *IEEE Transactions on Image Processing*, vol. 27, 2018.
- [51] S. Zhang, Z. Du, L. Zhang, H. Lan, S. Liu, L. Li, Q. Guo, T. Chen, and Y. Chen, "Cambricon-X: An accelerator for sparse neural networks," in Proceedings of the 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), 2016.
- [52] X. Zhang, X. Zhou, M. Lin, , and J. Sun, "ShuffleNet: An extremely efficient convolutional neural network for mobile devices," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.
- [53] R. Zhao, Y. Hu, J. Dotzel, C. D. Sa, and Z. Zhang, "Building efficient deep neural networks with unitary group convolutions," in *IEEE Con*ference on Computer Vision and Pattern Recognition (CVPR), 2019.
- [54] Y. Zhao, X. Chen, Y. Wang, C. Li, H. You, Y. Fu, Y. Xie, Z. Wang, and Y. Lin, "SmartExchange: Trading higher-cost memory storage/access for lower-cost computation," in *Proceedings of the 47th Annual ACM/IEEE International Symposium on Computer Architecture (ISCA)*, 2020.
- [55] Z.-Q. Zhao, P. Zheng, S.-T. Xu, and X. Wu, "Object detection with deep learning: A review," *IEEE Transactions on Neural Networks and Learning Systems*, 2019.
- [56] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-toimage translation using cycle-consistent adversarial networks," in *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[57] X. Zhu, Y. Xu, H. Xu, and C. Chen, "Quaternion convolutional neural networks," in European Conference on Computer Vision (ECCV), 2018.

#### APPENDICES

A. Minimal Algorithm for Diagonalizable G over  $\mathbb{R}$ 

**Theorem A.1.** Let m be the number of real-valued multiplications in a fast algorithm for an isotropic matrix G. Then

- (a) The lower bound of m is rank(G).
- (b) If G is diagonalizable over  $\mathbb{R}$ , there exists a minimal algorithm such that m = rank(G).

*Proof.* (a)  $m \ge \operatorname{rank}(T_z)$  by the dimension theorem and (8), and  $\operatorname{rank}(T_z) = \operatorname{rank}(G)$  by (8) and (4). Therefore, we have  $m \ge \operatorname{rank}(G)$ .

(b) Suppose  $G = T^{-1}DT$  with an invertible matrix T over real field and a diagonal matrix D which has entries with indeterminates  $g_i$ . We have

$$z = Gx = \underbrace{T^{-1}}_{=T_z} \underbrace{D}_{=\tilde{g}\circ} \underbrace{T}_x x, \tag{A-1}$$

and  $T_g$  can be derived by examining  $\tilde{g}_i = \sum_j (T_g)_{ij} g_j = D_{ii}$  for i = 0, ..., m - 1. This ring multiplication achieves the minimum complexity for  $m = \operatorname{rank}(D) = \operatorname{rank}(G)$ .

# B. Associativity from Commutativity

**Lemma B.1.** Let R be a ring with a bilinear-form multiplication, and  $a,b,c \in R$  with corresponding isomorphic matrices A, B, and C. The the multiplication associativity of R is equivalent to

$$C = AB$$
 if  $c = a \cdot b, \forall a, b$ .

*Proof.* Given  $c = a \cdot b$ , it is clear because  $\forall h \in R, a \cdot (b \cdot h) = (a \cdot b) \cdot h \iff \forall h \in R, ABh = Ch \iff C = AB$ .

**Lemma B.2.** Let R follow the exclusive sub-project distribution. Then

- (a)  $\forall a \in R$ , its isomorphic matrix A can be formulated by  $A = \sum_k a_k E_k$  where  $E_k$  is a signed permutation matrix.
- (b) For each standard-basis vector  $e_k = I_{:,k}$ ,  $E_k$  is its isomorphic matrix:  $e_k \cdot x = E_k x, \forall x \in R$ .

*Proof.* (a) From (9), we equivalently have  $A = S \circ (\sum_k a_k F_k)$  where  $F_k$  are permutation matrices with  $(F_k)_{ij} = \delta_{P_{ij}k}$ . Then we have  $A = \sum_k a_k E_k$  where  $E_k = S \circ F_k$ . Note that from (3)  $E_k$  can be directly derived from the indexing tensor M as  $(E_k)_{ij} = M_{ikj}$ .

(b) Let 
$$a = e_k$$
. Then  $A = E_k$  since  $a_j = \delta_{jk}$ .

**Theorem B.3.** The multiplication of R is associative if R has (i) the exclusive sub-product distribution, (ii) the commutative property of multiplication, and (iii) a commutative property of permutation matrices  $E_k$  such that  $E_k E_j = E_j E_k, \forall j, k$ .

*Proof.* Let  $a,b,c \in R$  and  $c=a \cdot b$ . By Lemma B.2, (i), and (ii), we have the j-th column of the isomorphic C as  $C_{:,j} = Ce_j = (a \cdot b) \cdot e_j = e_j \cdot (a \cdot b) = E_j Ab = \sum_k a_k E_j E_k b$ . Similarly,  $(AB)_{:,j} = ABe_j = a \cdot (b \cdot e_j) = a \cdot (e_j \cdot b) = AE_j b = a \cdot (e_j \cdot b)$ 

 $\sum_k a_k E_k E_j b$ . Then, we have C = AB by (iii) and thus the associative property of multiplication by Lemma B.1.

**Corollary B.3.1.** The multiplication of R is associative if R has the properties (i) and (ii) and the isomorphic matrix is diagonalizable over  $\mathbb{R}$ .

Proof. By Lemma B.2 and (i), for  $a \in R$  we consider its isomorphic matrix A which is diagonalizable:  $A = \sum_k a_k E_k = T^{-1}DT$  where D is a diagonal matrix in form of indeterminates  $a_k$  as D(a). We can diagonalize each permutation matrix  $E_j$  by setting  $a=e_j$  and have  $E_j=T^{-1}D(e_j)T$ . Then the commutative property of permutation matrices holds since the multiplication of diagonal matrices is commutative:  $E_k E_j = T^{-1}D(e_k)TT^{-1}D(e_j)T = T^{-1}D(e_k)D(e_j)T = T^{-1}D(e_j)D(e_k)T = E_jE_k$ . By Theorem B.3, the multiplication of R is thus associative.

Among the proper rings found in Section III-C,  $R_H$  and  $R_{O4}$  directly have the associative property of multiplication by Corollary B.3.1 since they are diagonalizable over  $\mathbb{R}$ . The rest of them ( $\mathbb{C}$ ,  $R_{H4\text{-II}}$ ,  $R_{H4\text{-II}}$ ,  $R_{O4\text{-II}}$ , and  $R_{O4\text{-II}}$ ) all possess the commutative property of permutation matrices and thus have the associative property as well by Theorem B.3.

## C. Comparison with Weight Pruning for Recognition

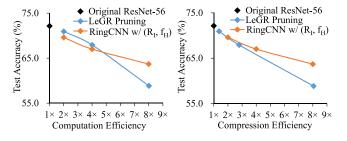


Fig. C-1. Computation efficiency (left) and weight compression efficiency (right) versus test accuracy for ResNet-56 on CIFAR-100 [28]. Note that ResNet-56 for CIFAR-100 has smaller-size feature maps for deeper layers, so the efficiencies of computation and compression are not the same for LeGR.

Recent pruning techniques for image recognition have mainly focused on structured pruning for their practical efficiency. In Fig. C-1, we compare RingCNN models to LeGR [10] which is the state-of-the-art for structured filter pruning. RingCNN models show their potential by outperforming LeGR for high computation efficiency (left) and for a wide range of compression efficiency (right). The study on hardware accelerators in this aspect will be our future work.