RECOM: An Efficient Resistive Accelerator for Compressed Deep Neural Networks

Houxiang Ji* †, Linghao Song‡, Li Jiang†§, Hai(Halen) Li‡ and Yiran Chen‡§
*Zhiyuan College, Shanghai Jiao Tong University, Shanghai, China
†Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China
‡Department of Electrical and Computer Engineering, Duke University, Durham NC, U.S.A
Email: {jihouxiang, ljiang_cs}@sjtu.edu.cn, {linghao.song, hai.li, yiran.chen}@duke.edu

***Abstract*— Deep Neural Networks (DNNs) play a key role in prevailing machine learning applications. Resistive random-access memory (ReRAM) is capable of both computation and storage, contributing to the acceleration on DNNs by processing in memory. Besides, a significant amount of zero weights is observed in DNNs, providing a space to reduce computation cost further by skipping ineffectual calculations associated with them. However, the irregular distribution of zero weights in DNNs makes it difficult for resistive accelerators to take advantage of the sparsity as expected efficiently, because of its high reliance on regular matrix-vector multiplication in ReRAM. In this work, we propose ReCom, the first resistive accelerator to support sparse DNN processing. ReCom is an efficient resistive accelerator for compressed deep neural networks, where DNN weights are structurally compressed to eliminate zero parameters and become hardware-friendly. Zero DNN activation is also considered at the same time. Two technologies, Structurally-compressed Weight Oriented Fetching (SWOF) and In-layer Pipeline for Memory and Computation (IPMC), are particularly proposed. In our evaluation, ReCom can achieve 3.37x speedup and 2.41x energy efficiency compared to a state-of-the-art resistive accelerator.**

I. INTRODUCTION AND MOTIVATION

Deep Neural Networks (DNNs) are attaining fame in many applications including computer vision [1], natural language process and speech recognition [2] for unprecedented level of accuracy. DNNs are becoming larger and deeper, while higher speed is also desired. These computation- and memory-intensive DNNs aggravate burdens on the underlying hardware, limiting their deployment in resource constrained scenarios, such as mobile and embedded devices.

A. ReRAM-Based Neural Network Acceleration

Resistive random access memory (ReRAM) is a promising candidate to replace DRAM with superior characteristics like high density, low write energy and high endurance [3]. With crossbar architecture, ReRAM shows its great potential in neural network acceleration. As shown in Fig. 1, the input activation a_i can be converted into analog voltages applied on word-lines of the crossbars while the filter weights $w_{i,j}$ can be programmed into the cells. With bit-line current accumulation, a matrix-vector multiplication can be performed in such a direct and fast way. Several ReRAM-based work are proposed. PRIME [4] takes advantage of in-memory data movement and efficient ReRAM-based computation. ISAAC [5] explores in-situ processing approach and the ReRAM crossbars serve as analog dot product engines. Besides, PipeLayer [6] is another cutting-edge ReRAM-based accelerator for the deep learning. [7] also shows the potential for accelerating graph processing with ReRAM.

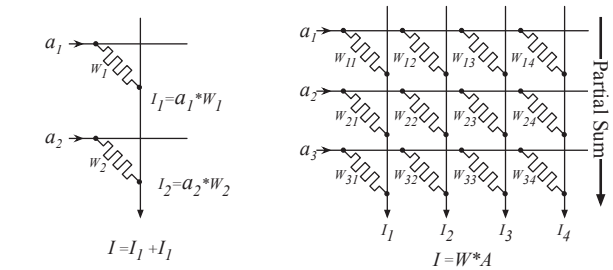

Fig. 1. Matrix-Vector Multiplication on ReRAM

B. Sparse Neural Network Acceleration

Recent researches show that a significant portion of filter weights and input activation in convolution layers are zero. Naturally, the multiplications and additions involving them do not contribute to the final result and thus can be skipped. Cnvlutin [8] focuses on the elimination of computation associated with zero activation ignoring further exploitation on zero weights. The zero weight/activation-aware architecture in

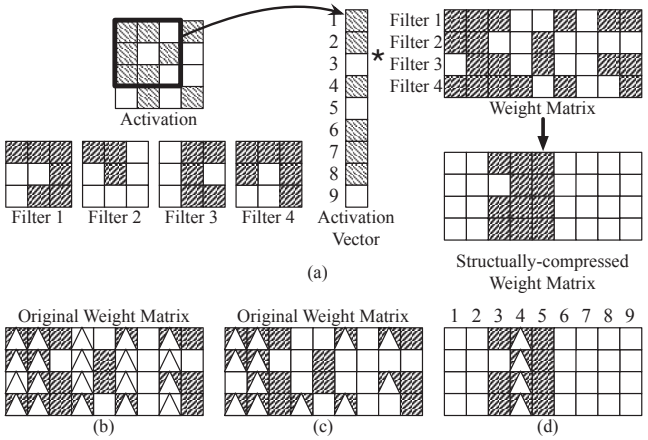

Fig. 2. (a) Conversion of a convolution to a matrix-vector multiplication and comparison of non-zeros (triangles) in computation between (b) [8], (c) [9] and (d) RECOM

§Corresponding authors are Li Jiang and Yiran Chen.

978-3-9819263-0-9/DATE18/©2018 EDAA

[9] aims at zero values in both filter weight and activation. The work ascribes the Cnvlutin's flaws to the synchronous execution and thus, allows each process engine performs a single convolution. Each engine skips multiplications and accumulations when any zero weight or activation appears.

To visualize the difference, we present an example in Fig.2. Weight matrix is spread by all filters in one layer with the number of filters $*$ the number of weights in a 3D filter. The block with lines means non-zero value while white blocks for zero. Blocks containing triangle represents the effectual computation. In Fig. 2 (b), i.e. Cnvlutin [8] skips the computation in column manner without looking into each single column. The work in (see Fig. 2 (c))[9] avoids every zero result calculation. But the performance is discounted by its lower parallelism.

C. Sparse Neural Network Acceleration with ReRAM

Previous accelerators focus on either the exploration of sparsity or utilization of ReRAM favorable crossbar architecture. Pipelayer [6], for example, leverages ReRAM cells to perform computation. And the large amount of ReRAM buffer in it takes much space on chip and causes huge energy consumption because of the zero activation in buffers. Besides, when ReRAM performs as the computation engine, the useless cost by zero weights is astonishing too.

Acceleration for DNNs with ReRAM encounters several problems because of random and irregular distribution of zero value in weight. Firstly, waste of cells is induced due to zero weights. Additional to the hardware waste, the processing on activation associated with zero weights also consumes enormous unnecessary energy. For example, when the values in column j of a weight matrix are all zeros, we will not program it to a crossbar. But the activation fetching for this column still goes on. Furthermore, the random memory access for weights or their activation severely prevents practical acceleration. More sophisticated design is required to solve these drawbacks.

In our RECOM, the *first* resistive accelerator utilizing the sparsity in neural networks, we deal with the incompatibility between ReRAM and irregular sparsity in DNNs by software-hardware co-design. The computation for RECOM is also displayed in Fig. 2 (d). Non-zero values only gather in some columns (e.g., column 3, 4, 5 in Fig. 2 (d)). Model compression and zero activation skipping are both employed in RECOM. Computation decreases sharply as shown in Fig. 2(b) [8], (c) [9] and (d) RECOM, resulting in 20, 12, and 4 multiplication operations, respectively. We also design two technologies, Structurally-compressed Weight Oriented Fetching (SWOF) and In-layer Pipeline for Memory and Computation (IPMC). The details of RECOM are discussed in the following section.

II. ARCHITECTURE

A. Overall Architecture

We propose the efficient ReRAM-based architecture, RECOM, leveraging ReRAM's computation capability to accelerate DNN processing. Fig. 3 illustrates the top-level design. A chip is mainly composed of a global SRAM on-chip buffer, 16 processing engines (PEs) connected in mesh, and an addressing unit. Each processing engine consists of several input buffers (IB) with position vectors (PV) for indexing, ReRAM crossbar arrays for computation and column fetching unit responsible for column-wise fetching. Besides, peripheral components like ADC, DAC, partial sum buffer, and rectified linear unit (ReLU) module are also equipped in PEs. The filter weights are programmed to ReRAM cells before reading in activation. Two kinds of input data reuse are achieved as mentioned in Eyeriss [10]: convolutional reuse and filter reuse. A specific group of crossbar arrays is allocated to each layer, supporting layer-wise pipeline [6].

B. Structural Compression on Weight Matrix

The matrix-vector multiplication in ReRAM has a regular structural manners. As we discussed in former section, deployment of sparse models on ReRAM faces with the problem of the irregular sparsity. Redundancy across filters and channels indicates the possible weight elimination. Arbitrary shapes of filters instead of fixed cuboid can potentially eliminate useless computations. Based on these observation, we use *Group Lasso* to obtain more "structured" weight matrix by zeroing out weights when training the neural networks [11]. We apply regularization on filter and channel dimension simultaneously because all-zero filter makes the corresponding channel useless in the next layer of the neural network. Filter-wise regularization can be viewed as the row reduction; while channel-wise and shape-wise regularization are column reduction in the weight matrix. Fig. 2 (d) shows a refined weight matrix with 3 effective columns remained.

C. Structurally-compressed Weight Oriented Fetching (SWOF)

Instead of passive reading, Structurally-compressed Weight Oriented Fetching (SWOF) positively fetches activation. SWOF locates and selects non-zero activation associated with non-zero weights in two steps: Row-SWOF and Column-SWOF. Row sparsity vector (RSV) and column sparsity vector (CSV) are used to record the sparsity. RSV and CSV record whether all elements in a row or column are eliminated after

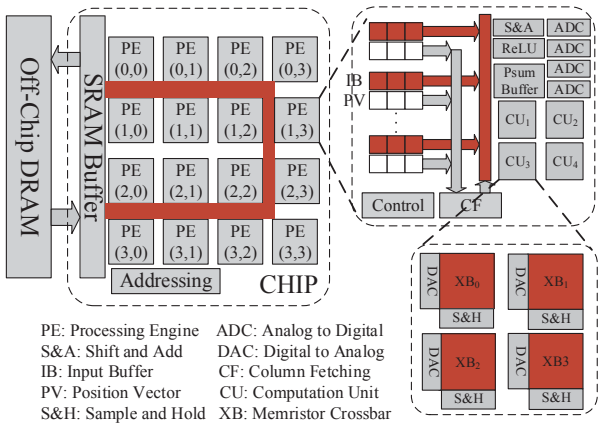

Fig. 3. RECOM Top-level Architecture

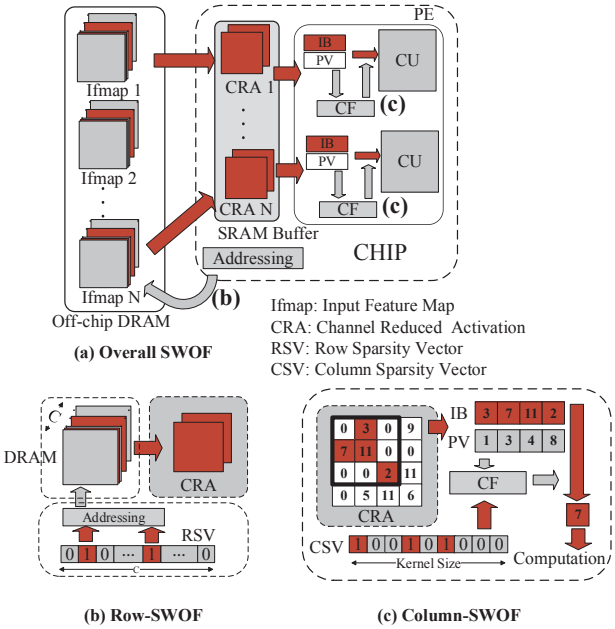

Fig. 4. Structurally-compressed Weight Oriented Fetching (SWOF)

compression separately (e.g., a "0" entry indicates all-zero row/column). Each layer has one RSV of M_l size and C_l CSV with the size $R_l \times S_l$. All the RSV and CSV are generated while compression, where M_l, R_l, S_l, C_l for filter number, filter height and width, and channel number in layer l.

The SWOF is displayed in Fig. 4. Row-SWOF works when fetching activation from off-chip DRAM to on-chip SRAM buffers–the most time- and energy-consuming step [10]. The elimination of j-th filter of i-th layer implies that weights in j-th channel in next layer are all zero. We do not need to fetch the activation in j-th channel of $(i+1)$-th layer anymore. The indexes of non-zero value in RSV are delivered to the addressing unit. We call the refined activation tiles as Channel Reduced Activation (CRA) in SRAM buffers. After activation loaded in, the activation tiles keep their original size equal to the filter size (filter height $*$ filter width). Then, Column-SWOF looks into each channel tiles. As the activation enters the input buffer (i.e., IB in Fig. 4 (a)), each value is checked. Only non-zero activation stay and their positions in filter are recorded in position vector (PV). Fetching unit (FU) compares the indexes of the remaining columns in CSV with those in PV. FU only delivers the activation with mutual indexes to CU.

D. In-layer Pipeline for Memory and Computation (IPMC)

As activation fetching (Memory) and computation can be decoupled and the sparsity in activation, i.e. zero activation, is also considerable, we propose In-layer Pipeline for Memory and Computation (IPMC) to further speed up the process. Default layer-wise pipeline in [6] accesses the memory and does computation in the same logical cycle. But the computation stage can be skipped when input activation are all zero in the buffer. We divide the process into memory access and computation. After SWOF, all activation in an input buffer is very likely to be dropped out, meaning no computation needed. The process engines move forward to the next buffer. Another benefit is the throughput improvement. In the case two consecutive activation vectors ($v1$, $v2$) are non-zeros (to be computed), the buffer holds fetched vector $v1$ in one cycle, but in the next cycle, this buffer is used to hold $v2$ while $v1$ is in computation simultaneously.

III. Evaluation

A. Benchmarks

We evaluated the proposed ReCom accelerator on two datasets: MNIST [12] and ImageNet [13]. Each 28×28 pixel handwritten digits grey image in MNIST is labeled with a number from 0 to 9. ImageNet is a large-scale dataset with over 15,000,000 images of roughly 22,000 categories. In our evaluation, we used the ILSVRC 2012, a subset of ImageNet with approximately 1000 categories and 1000 images in each category. We reshaped all the training images to the size of 256×256 pixel. The networks used in our evaluation are LeNet [12] on MNIST, AlexNet [1] and CaffeNet [14] on ImageNet. The typologies of the networks are summarized in Table I.

TABLE I
Network Topology
($FilterNumbers \times FilterHeight \times FilterWidth \times Channels$)

MNIST	LeNet	Conv 1	Conv 2
		$1 \times 5 \times 5 \times 20$	$50 \times 5 \times 5 \times 20$
ImageNet	AlexNet CaffeNet	Conv 1	Conv 2_1/2_2
		$96 \times 11 \times 11 \times 3$	$128 \times 5 \times 5 \times 48$
ImageNet	AlexNet CaffeNet	Conv 3	Conv 4_1/4_2
		$384 \times 3 \times 3 \times 256$	$192 \times 3 \times 3 \times 19$
ImageNet	AlexNet CaffeNet	Conv 5_1/5_2	–
		$128 \times 3 \times 3 \times 192$	

B. Evaluation Setup

In the evaluation, we trained the models on the prevailing platform Caffe [14]. We built a PipeLayer-like [6] accelerator as the baseline accelerator in our evaluation. The configuration of the simulation is shown in Table II. For ADC energy and latency, we obtain data from the latest survey [15]. We adopt the ReRAM read/write latency and energy from the work in [16], and memory timing from [3].

TABLE II
Simulation Configuration

DRAM Memory	128GB	SRAM Cache	128KB
Base Clock	1200MHz	Neuron Buffer Size	256B
Crossbar Size	64 × 64	Number of PEs	16
Number of Crossbar	256	DRAM Latency	15ns
ReRAM Write Energy	18.32pJ	ReRAM Read Energy	0.63pJ

C. Accuracy and Sparsity

The top-1 accuracy of default models(without regularization) and our models are: LeNet 98.47%, 98.46%; AlexNet 57.25%, 55.19%; CaffeNet 57.41%, 57.37%. And the sparsity(the ratio of zeros) of each layer is listed in Table III. CaffeNet and AlexNet are both trained on the ILSVRC 2012 dataset while with some differences: CaffeNet are not trained with the relighting data-augmentation and the order of pooling layer and normalization layer is switched. The row sparsity

TABLE III
WEIGHT MATRIX SPARSITY OF COMPRESSED MODELS

	Network	Conv1	Conv2	Conv3	Conv4	Conv5
Row	LeNet	0.00%	0.00%	–	–	–
	AlexNet	9.29%	12.11%	39.78%	46.32%	0.00%
	CaffeNet	0.00%	0.00%	1.57%	2.86%	0.00%
Column	LeNet	20.00%	94.80%	–	–	–
	AlexNet	0.00%	61.21%	77.11%	85.03%	81.17%
	CaffeNet	0.00%	23.50%	39.00%	38.50%	24.50%

represents the reduction in filter and the column sparsity represents the reduction in filter shape and channels. We can see that with decreasing accuracy, the sparsity increases sharply, especially comparing the AlexNet with CaffeNet.

D. Performance Results and Energy Saving

The average speedup of RECOM (with SWOF only, case 1) are 3.32x in LeNet, 2.81x in AlexNet and 1.39x in CaffeNet while for RECOM (with both SWOF and IPMC case 2), the speedups are 4.81x, 4.40x and 2.25x respectively. The overall GeoMean speedups of case 1 and case 2 are 2.16x and 3.37x. The speedups in case 1 varies from 1.00x to 8.95x in various layers of different models while case 2 varying from 1.33x to 10.66x. The increment illustrates our accelerator gains extra speedup with IPMC. The reduction in filter results in higher speedup than in filter shape and channel. Fig. 5 shows the speedup of RECOM compared with baseline accelerator.

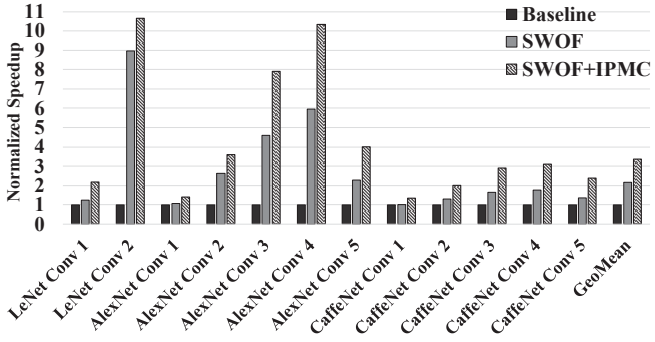

Fig. 5. Normalized Speedup

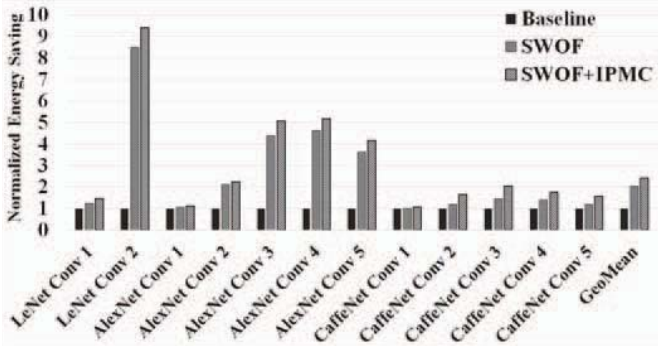

Fig. 6. Normalized Energy Saving

The energy saving of RECOM compared with baseline accelerator is shown in Fig. 6. The average energy saving of RECOM (with SWOF only) are 3.25x in LeNet, 2.77x in AlexNet and 1.23x in CaffeNet while in RECOM (with both SWOF and IPMC) the energy saving increases to 3.70x, 3.07x and 1.59x. With more energy cost on control, RECOMstill achieves a GeoMean energy saving of 2.03x (with SWOF only) and 2.41x (with both SWOF and IPMC). The energy saving ranges from 1.24x to 9.43x in LeNet, 1.04x to 5.18x in AlexNet and 1.00x to 2.04x in CaffeNet.

IV. CONCLUSION

In this paper, we proposed a novel resistive accelerator, RECOM, the *first* one to support the sparse DNN processing in ReRAM. Deep neural networks are compressed structurally by regularization on each layer when training with little or no loss in accuracy. Hardware-friendly and compressed weights are processed efficiently to exploit zero values while keeping high parallelism. Moreover, we fetch activation in a creative way called Structurally-compressed Weight Oriented Fetching (SWOF). In addition, we propose In-layer Pipeline for Memory and Computation (IPMC) to further speedup processing. Our experiments showed that the proposed architecture achieves 3.37x speedup (up to 10.66x), and 2.41x energy saving (up to 9.43x) over the state-of-the-art resistive accelerator architecture.

ACKNOWLEDGEMENTS

This research was partially supported by National Natural Science Foundation of China (Grant No. 61602300) and Shanghai Science and Technology Committee (Grant No. 15YF1406000). This work was also partially supported by NSF-1253424, NSF-1615475 and DOE-SC0018064.

REFERENCES

[1] A. Krizhevsky *et al.*, "Imagenet classification with deep convolutional neural networks," in *NIPS*, 2012.
[2] G. Hinton *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, 2012.
[3] C. Xu *et al.*, "Overcoming the challenges of crossbar resistive memory architectures," in *HPCA*, 2015.
[4] P. Chi *et al.*, "Prime: A novel processing-in-memory architecture for neural network computation in reram-based main memory," in *ISCA*, 2016.
[5] A. Shafiee *et al.*, "Isaac: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars," in *ISCA*, 2016.
[6] L. Song *et al.*, "Pipelayer: A pipelined reram-based accelerator for deep learning," in *HPCA*, 2017.
[7] L. Song *et al.*, "Graphr: Accelerating graph processing using reram," in *HPCA*, 2018.
[8] J. Albericio *et al.*, "Cnvlutin: Ineffectual-neuron-free deep neural network computing," in *ISCA*, 2016.
[9] D. Kim *et al.*, "A novel zero weight/activation-aware hardware architecture of convolutional neural network," in *DATE*, 2017.
[10] Y. H. Chen *et al.*, "Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks," in *ISCA*, 2016.
[11] W. Wen *et al.*, "Learning structured sparsity in deep neural networks," in *NIPS*, 2016.
[12] Y. Lecun *et al.*, "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE*, 1998.
[13] J. Deng *et al.*, "ImageNet: A Large-Scale Hierarchical Image Database," in *CVPR*, 2009.
[14] Y. Jia *et al.*, "Caffe: Convolutional architecture for fast feature embedding," in *MM*, pp. 675–678, ACM, 2014.
[15] B. Murmann, "Adc performance survey 1997-2017," *https://web.stanford.edu/ murmann/adcsurvey.html*, 2017.
[16] D. Niu *et al.*, "Design of cross-point metal-oxide reram emphasizing reliability and cost," in *ICCAD*, 2013.