Structured Pruning of RRAM Crossbars for Efficient In-Memory Computing Acceleration of Deep Neural Networks

Jian Meng, *Graduate Student Member, IEEE*, Li Yang, *Graduate Student Member, IEEE*, Xiaochen Peng, *Graduate Student Member, IEEE*, Shimeng Yu, *Senior Member, IEEE*, Deliang Fan, *Member, IEEE*, and Jae-Sun Seo, *Senior Member, IEEE*

(Invited Paper)

***Abstract*—The high computational complexity and a large number of parameters of deep neural networks (DNNs) become the most intensive burden of deep learning hardware design, limiting efficient storage and deployment. With the advantage of high-density storage, non-volatility, and low energy consumption, resistive RAM (RRAM) crossbar based in-memory computing (IMC) has emerged as a promising technique for DNN acceleration. To fully exploit crossbar-based IMC efficiency, a systematic compression design that considers both hardware and algorithm is necessary. In this brief, we present a system-level design considering the low precision weight and activation, structured pruning, and RRAM crossbar mapping. The proposed multi-group Lasso algorithm and hardware implementations have been evaluated on ResNet/VGG models for CIFAR-10/ImageNet datasets. With the fully quantized 4-bit ResNet-18 for CIFAR-10, we achieve up to 65.4× compression compared to full-precision software baseline, and 7× energy reduction compared to the 4-bit unpruned RRAM IMC hardware with 1.1% accuracy loss. For the fully quantized 4-bit ResNet-18 model for ImageNet dataset, we achieve up to 10.9× structured compression with 1.9% accuracy degradation.**

***Index Terms*—Convolutional neural networks, hardware accelerator, in-memory computing, structured pruning, resistive RAM.**

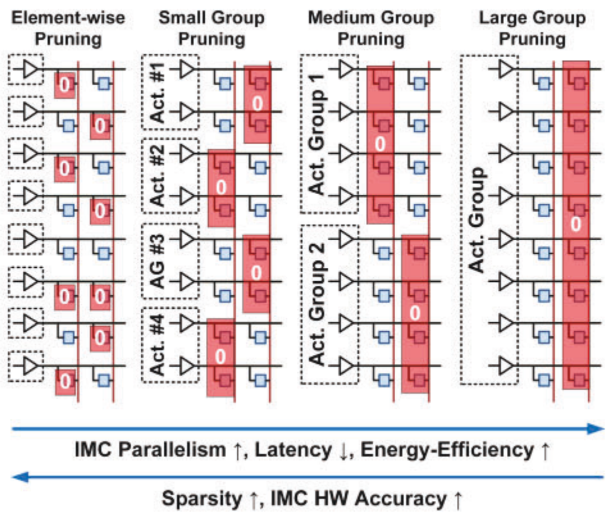

Fig. 1. For different pruning group sizes, trade-offs exist for IMC hardware design on energy, area, and accuracy.

I. Introduction

DEEP neural networks (DNNs), including convolutional neural networks (CNNs), have been very successful with high accuracy for many computer vision applications [1]. However, the high requirement for hardware resources hinders efficient deployment of DNNs on hardware devices. To achieve high energy-efficiency and density, many prior works investigated DNN model pruning and compression. Non-structured pruning [2] can achieve high element-wise sparsity with the cost of a large amount of index memory storage [3] and irregular memory access for hardware computation. Alternatively, structured pruning can efficiently eliminate the weights in a structured manner during training [4], [5] to improve the targeted hardware's energy-efficiency.

With the property of naturally supporting parallel dot-product operations, non-volatile memory (NVM) crossbar based in-memory computing (IMC) has emerged as a promising technique [6] for fast DNN computations. Among different NVMs, RRAM devices can store multiple levels in one cell [7]–[9], which provides dense storage for fast in-memory parallel computations. However, it becomes very inefficient to apply random sparsity patterns resulted from non-structured pruning to a fixed RRAM crossbar structure (Fig. 1). If the IMC operation happens on a column basis, it will be much more efficient to prune out the entire/partial column in a structured manner. However, as shown in Fig. 1, smaller group sizes achieve higher sparsity compared with the large-sized

Manuscript received February 5, 2021; accepted March 8, 2021. Date of publication March 26, 2021; date of current version April 30, 2021. This work was supported in part by National Science Foundation under Grant 1652866 and Grant 1715443; in part by Semiconductor Research Corporation AIHW Program; and in part by the C-BRIC/ASCENT, two of six centers in JUMP, a Semiconductor Research Corporation Program sponsored by DARPA. This brief was recommended by Associate Editor J. K. Eshraghian. *(Corresponding author: Jae-Sun Seo.)*

Jian Meng, Li Yang, Deliang Fan, and Jae-Sun Seo are with the School of Electrical, Computer and Energy Engineering, Arizona State University, Tempe, AZ 85287 USA (e-mail: jaesun.seo@asu.edu).

Xiaochen Peng and Shimeng Yu are with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332 USA (e-mail: shimeng.yu@ece.gatech.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCSII.2021.3069011.

Digital Object Identifier 10.1109/TCSII.2021.3069011

groups [10]. Correspondingly, small-sized group will restrict the number of rows that can be activated simultaneously, which requires more number of cycles to go through the same crossbar array but also affects the DNN accuracy achievable by a given ADC precision at the periphery due to different dynamic range of the IMC results.

There have been several prior works that studied crossbar-based acceleration for DNNs. Crossbar-aware pruning [11] presented pruning schemes that consider the crossbar structure, but requires a ~3X overhead for the number of RRAM cells to support the proposed dataflow as well as a large number of non-compute crossbars for the data buffer. In [12], 8-bit MobileNet and VGG models were deployed to the tiled-based RRAM architecture, but DNN pruning was not investigated with the RRAM-based accelerator design. RRAM crossbar based quantization/pruning has been jointly investigated [13], [14], but the high precision weight (e.g., 8-bit or 16-bit) and high precision activation (e.g., 16-bit or full precision) causes large storage and communication overhead. In a recent crossbar-based pruning work [15], accuracy and energy have been reported, but the hardware results did not provide detailed resource consumption based on a specific type of IMC device.

In this work, we propose a hardware-friendly RRAM crossbar based pruning technique using 2-bit-per-cell RRAM devices, where only non-zero weights are stored. The algorithm has been evaluated on various-sized CNN models with CIFAR-10 and ImageNet datasets, showing a large compression rate compared to the full-precision software baseline. We use the circuit-level simulator NeuroSim [16] to evaluate the hardware performance, including accuracy and the detailed resource consumption. The main contributions of this brief are:

- New DNN training algorithm that prunes weights in variable column-group sizes considering crossbar hardware mapping and memory storage efficiency, together with low-precision quantization (e.g., 4-bit) of both activations and weights.
- Hardware implementation that exploits group-wise weight sparsity in newly compressed DNNs.
- We evaluated the accuracy, energy, and area characterization of the DNN inference accelerator design. We achieve up to 7× energy reduction compared to the 4-bit unpruned model with minimum accuracy loss.

II. Hardware-Aware Structured Pruning Algorithm

A. Prior Works on Group Lasso Based Pruning

Wen *et al.* [4] applied group Lasso [17] as a regularization term in the loss function to exploit the structured sparsity during DNN training. Suppose the weight matrix of the DNN model with layer L can be expressed as multiple of 4-D tensors $W_l \in \mathbf{R}^{N_l \times C_l \times K_l \times K_l}$, where N_l, C_l, K_l represents the dimensionality of output channel, input channel and each single kernel, respectively. The loss function can be constructed as:

$$\hat{\mathcal{L}} = \mathcal{L}(f(x, \{W\}_{l=1}^{L}), y) + \lambda \sum_{l=1}^{L} \sum_{g=1}^{G_l} ||W_{l,g}||_2, \tag{1}$$

where $\mathcal{L}$ represents the loss of the DNN model f regarding the targeted output y, and G_l represents the number of pre-defined groups in layer l. The penalty level λ can be tuned based on

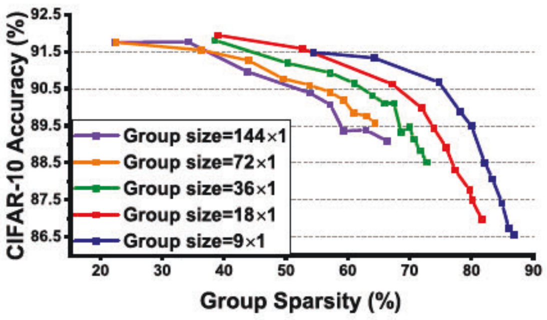

Fig. 2. For different group sizes, group sparsity vs accuracy of ResNet-20 CNN on CIFAR-10 dataset are shown.

the datasets and group sizes. Group Lasso generates structured sparsity for the DNN based on flexibly defined group $W_{l,g}$.

B. Computation-Efficiency-Aware Deployment

Selecting a suitable group size is critical for both algorithm and hardware design. Pruning out large-sized groups (e.g., filter-wise pruning) generates less sparsity compared to the smaller one [10]. This is also observed in our experiments shown in Fig. 2. It is also unrealistic to map the entire filter as a single RRAM column because various layer sizes will lead to inconsistent sub-array sizes or utilization. On the other hand, pruning and deploying small-sized groups to the small sub-arrays will introduce higher sparsity and less analog noise with IMC during inference. However, turning on the limited number of rows results in system-level latency increase, and small-sized sub-arrays will cause complex circuits design, leading to high power and area consumption. Thus, the question arises: *How to deploy group-wise sparsity on RRAM based IMC hardware, while keeping a balance between hardware efficiency and DNN accuracy?*

In this work, the targeted group size was selected between the filter-wise pruning and channel-wise pruning to ensure uniform group size across all layers. To resolve the "group size vs. performance" question, we mapped multiple same size small groups to the single column. In our design, mapping 16, 8, and 4 input channels of single convolution filter (with 3 × 3 kernel size) are equivalent to filling 100%, 50%, and 25% of a 144 × 1 RRAM column, corresponding to 144 × 1, 72 × 1, and 36 × 1 group sizes, respectively. Fig. 3 shows an example when the group size is 72×1. The partial sums will be computed separately by activating the part of the rows during each clock cycle. With the less amount of activated rows, the analog computing noise will be kept small with the cost of the increased latency overhead (e.g., 2X for a 72 × 1 group). The high sparsity generated by the small-sized group pruning introduces larger resource reduction and alleviates the latency overhead by removing a large amount of computation.

C. Proposed Pruning Algorithm With Column Group Matching

Fig. 3 shows an example where a RRAM array is divided into two groups of partial columns. The conventional group Lasso algorithm (Eq. (1)) penalizes all groups together without considering this spatial separation, leading to memory waste caused by the lopsided number of unpruned groups between the top half and bottom half of the crossbar array (Fig. 5).

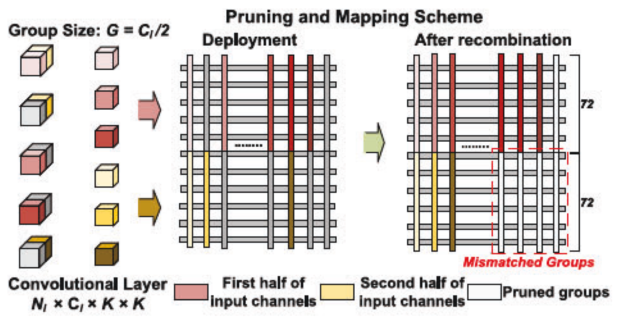
Pruning and Mapping Scheme
Group Size: $G = C_l/2$
Deployment
After recombination
72
72
Mismatched Groups
Convolutional Layer $N_l \times C_l \times K \times K$
First half of input channels
Second half of input channels
Pruned groups

Fig. 3. The overall mapping scheme when the pruning group size is half of the sub-array column. The conventional group Lasso algorithm will cause the lopsided number of unpruned groups after the group-wise recombination.

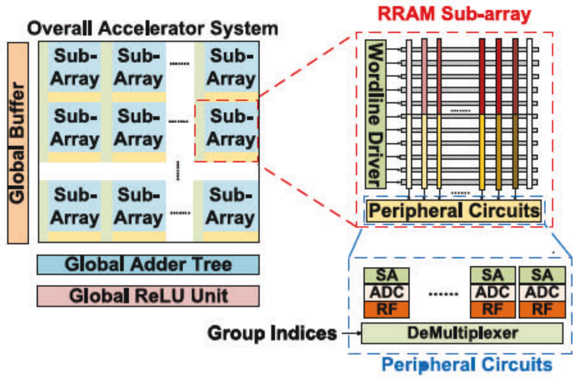

Fig. 4. System-level design of the accelerator. Each layer's weights are mapped to different number of sub-arrays.

We propose a crossbar-aware structured pruning method to resolve this issue. Motivated by the weight penalty clipping method (WPC) proposed by Yang *et al.* [5], we penalize the different groups of partial columns individually with its penalty clipping threshold value δ_l^i (Eq. (2)) and penalty level λ_i.

$$\delta_l^i = a \cdot \frac{1}{G_l^i} \sum_{g=1}^{G_l^i} ||W_{l,g}||_2 \tag{2}$$

Equation (3) describes the loss function of the proposed multi-group Lasso algorithm, in which N represents the number of partial column groups (e.g., $N = 2$ when group size is 72×1), and a controls the clipping intensity.

$$\hat{\mathcal{L}} = \mathcal{L}(f(x, W), y) + \sum_{i=1}^{N} \lambda_i (\sum_{l=1}^{L} \sum_{g=1}^{G_l^i} min(||W_{l,g}||_2, \delta_l^i)) \tag{3}$$

With the carefully selected λ_i, we achieved significant mismatch reduction, as shown in Fig. 5. More specifically, δ_l^i determines whether the group Lasso should penalize the group or not, so pruning does not shrink the weight distribution exceedingly to distort the quantization.

III. CNN Hardware Accelerator Design

Fig. 4 shows the top-level block diagram of the CNN accelerator with compressed IMC RRAM crossbar tiles. Each tile includes multiple RRAM sub-arrays to store the compressed weights and compute parallel dot-product operations. The CNN accelerator is performed on a layer-by-layer basis, without aggressive pipelining between consecutive images, which will put more overhead on the activation storage. The input register loads the input data from the global buffer, while the output register accumulates partial sums obtained from sub-arrays then sends the results back to the global buffer. The global buffer stores inputs and activations, and a global control unit generates proper instructions to RRAM tiles. Besides the MAC operations computed by the RRAM sub-arrays, activation, normalization, and pooling operations are performed by separate computation modules.

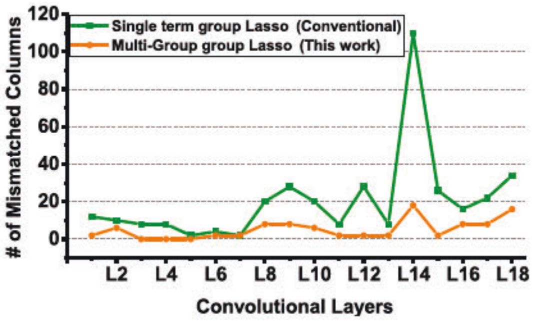

Fig. 5. For ResNet-20 CNN and 72×1 group size, the number of mismatched columns (2-bit per cell) is reduced by applying multi-group Lasso algorithm.

To implement 4-bit quantized target CNNs, we use 2-bit per cell HfO_2 1T1R RRAM devices, characterized from [18] and projected to 32nm CMOS node. According to the benchmark results from the NeuroSim framework [16], adapting small R_{on} (e.g., tens of kΩ) while avoiding the large voltage drop will introduce a large area overhead for the 1T1R device-based design. In our design, we choose the resistance levels to be 100kΩ, 400kΩ, 700kΩ, and 1MΩ with 2-bit per cell RRAM devices and 0.5V read voltage. Table I summarizes the detailed hardware characteristics modeled by NeuroSim [16]. 4-bit activations are mapped across multiple cycles (depending on the group size) with digital input voltages onto the word-line (WL) of the RRAM sub-array. We set the memory size as 144×32 to avoid the memory waste in shallow layers. This work focuses on hardware-based DNN compression, and we did not consider the conductance instability and retention drifting of the filamentary analog synaptic device in our current design. We assume that techniques such as the refreshing mechanism [19], [20] will be applied to mitigate the accuracy degradation caused by the conductance instability.

In each RRAM sub-array, we use a switch matrix to activate the WLs in parallel; each column is connected to a 5-bit or 6-bit successive approximation register (SAR) analog-to-digital converter (ADC). The shift-add modules are used to shift and accumulate (1) the partial sums of the 4-bit sequential input voltages during multiple cycles and (2) the partial sums from two RRAM columns for 4-bit weights. Since the column groups are compressed, we employ additional de-multiplexers after the shift-add modules, which convey the compressed partial sums from column groups to the corresponding peripheral circuits for ensuing CNN operations. As shown in Table I, the de-multiplexers consume a tiny portion of the total resources. The accelerator also includes adder trees, and ReLU units, to support various DNN structures and sizes. The global buffer stores the largest feature map across the whole network to reduce expensive off-chip memory access.

TABLE I
HARDWARE SPECIFICATION SIMULATED BY NEUROSIM FOR 4-BIT RESNET-20 WITH 72 × 1 GROUP

RRAM Sub-Array			
Components	Area (μm^2)	Latency (ns)	Energy (pJ)
Memory Array (144 × 32)	75.5		
Switch Matrix (WL and SL)	457.3	2	1.1
SAR ADC (5-bit)	8,409.3	12	40.4
Shift-Add-Input	1,412.9	0.8	6.8
Shift-Add-Weight (2 col use 1)	825.8	0.8	1.0
Total	11,180.8	15.6	49.3
Peripheral Circuits			
DeMUX 1-to-4	7.9	0.001	0.01
DeMUX 1-to-8	57.2	0.001	0.1
DeMUX 1-to-16	111.0	0.001	0.2
1 stage AdderTree (128 units)	2,510.3	0.3	4.4
2 stage AdderTree (128 units)	7,740.1	0.6	13.7
3 stage AdderTree (128 units)	18,408.8	1.0	32.6
Global Buffer (32 × 32 × 16 × 4)	173,266	0.9/bit/access	0.003/bit/access
ReLU (128 units)	939.5	0.5	0.9

TABLE II
STRUCTURED PRUNING AND QUANTIZATION RESULTS OF RESNET-20, VGG-8, AND RESNET-18 FOR CIFAR-10/IMAGENET DATASETS

Models & Dataset	Method & Precision	Group Size	Inference Accuracy	Group Sparsity	Comp. Rate	No. of Params
ResNet-20 & CIFAR10	FP32	-	91.95%	-	1.00×	0.27M
	This work (4-bit)	144 × 1	91.24%	34.54%	12.22×	0.18M
		72 × 1	91.03%	51.56%	16.51×	0.13M
		36 × 1	90.88%	56.43%	18.36×	0.12M
ResNet-18 & CIFAR10	FP32	-	94.75%	-	1.00×	11.17M
	This work (4-bit)	144 × 1	93.61%	86.14%	57.72×	1.75M
		72 × 1	93.51%	87.76%	65.35×	1.52M
		36 × 1	93.56%	90.52%	84.38×	1.15M
VGG-8 & CIFAR10	FP32	-	94.07%	-	1.00×	12.97M
	This work (4-bit)	144 × 1	92.94%	65.79%	57.06×	1.82M
		72 × 1	92.98%	74.24%	74.07×	1.41M
		36 × 1	93.59%	73.40%	72.61×	1.43M
ResNet18 & ImageNet	FP32	-	69.76%	-	1.00×	11.69M
	This work (4-bit)	144 × 1	67.89%	19.24%	9.91×	9.50M
		72 × 1	67.50%	23.77%	10.49×	9.08M
		36 × 1	67.82%	26.41%	10.87×	8.78M

IV. EXPERIMENTAL RESULTS

In this section, we present the experimental results of our proposed algorithm and hardware design. The algorithm was validated on the CIFAR-10 and ImageNet datasets. The compression process was fine-tuned from the pre-trained full precision model with the SGD optimizer. Note that some prior works did not quantize the first, last, or residual layers [21], [22]. In our work, however, all DNN layers are fully quantized to 4-bit activation and 4-bit weights with the PACT quantizer [21] to achieve less statistical writing error (inside the memory cell) with low precision weights [23]. We utilized the circuit-level simulator NeuroSim [16] to estimate the area, latency, and energy of the proposed hardware design.

A. Software Results

As shown in Table II, we investigated group sizes of 144 × 1, 72 × 1, and 36 × 1 with with compact residual networks (e.g., ResNet-20 with 0.27 million parameters) and large size DNN models (e.g., ResNet-18/VGG-8 with 11.17/12.97 million parameters). The proposed algorithm compressed ResNet-20 by 18.36× and ResNet-18/VGG-8 by 65.4 × /74.1×, respectively, with around only 1% accuracy degradation. Since VGG-8 has a large portion of the fully-connected weights, so we also applied the channel-wise pruning on the first fully-connected layer with the proposed method.

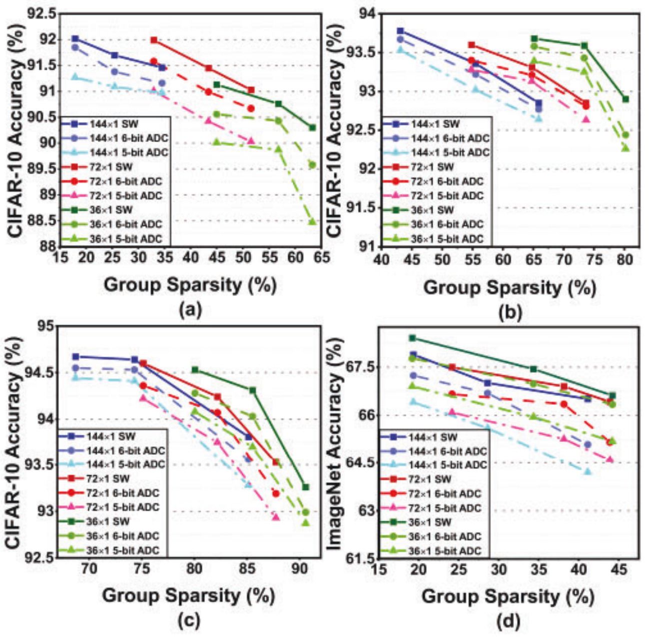

Fig. 6. Hardware inference accuracy with pruned models: (a) ResNet-20, (b) VGG-8, (c) ResNet-18 on CIFAR-10, and (d) ResNet-18 on ImageNet.

We also evaluated the fully quantized 4-bit ResNet-18 for ImageNet dataset with different group sizes. The proposed algorithm compressed the model by 10.9× with 1.94% accuracy degradation. Compared to the CIFAR-10 experiments, the ImageNet classification task is much more complicated, and the number of the redundant weights in DNNs is less. Consequently, it becomes more difficult to exploit the sparsity [24], as shown in Table II.

B. Hardware Results

We mapped the pruned CNN models to the designed RRAM based accelerator, and evaluated the hardware DNN accuracy and energy/latency/area using NeuroSim [16]. Fig. 6 shows the sparsity vs. hardware accuracy for different DNNs for CIFAR-10/ImageNet datasets. 6-bit ADC provides only marginally better accuracy than 5-bit ADC, but incurs larger area/power overhead, thus we used 5-bit ADC for the remaining results.

Since the total number of operations reduced significantly after pruning, we report the energy per image instead of TOPS/W. We choose the hardware deployment of the unpruned 4-bit CNNs as the baseline for comparison. Fig. 7 shows the energy/latency/area for different DNNs for CIFAR-10 dataset. With our proposed pruning scheme, compared to each of the hardware baseline, we reduce energy consumption by 37.0%, 80.3%, and 90.2%, and total area consumption by 33.3%, 81.0%, and 82.0% with ResNet-20, VGG-8, and ResNet-18 models, respectively. The IMC latency overhead caused by computation with different sizes of groups can be alleviated by high sparsity, and the high sparsity substantially reduces the total energy and area of the accelerator.

For a similar sparsity level, 36 × 1 groups have the largest latency/energy cost, while 144 × 1 groups have the largest area consumption. The system achieved the best tradeoff between energy efficiency and inference accuracy with the 72 × 1 group size (Fig. 7). With the selected group size, Table III shows the comparison to relevant prior works. Crossbar-aware pruning [11] and fine-grid pruning [14]. Compared

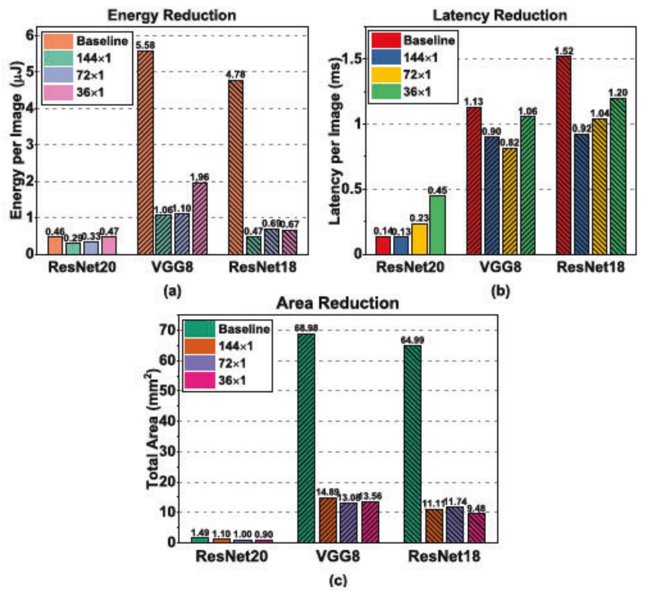

Fig. 7. System-level hardware results on (a) energy, (b) latency, and (c) area.

TABLE III
COMPARISON WITH PRIOR WORKS BASED ON 72 × 1 GROUP SIZE

Models & Dataset	Methods	Structured Sparsity	Inference Acc.	Model Precision
VGG-8	Crossbar-Aware [11]	50.0%	93.8%	32-bit
CIFAR-10	**This work**	**54.8%**	**93.8%**	**4-bit**
ResNet-56	Fine-Grid [14]	23.5%	92.5%	16-bit
CIFAR-10	**This work**	**33.4%**	**92.8%**	**4-bit**
VGG-16	Fine-Grid [14]	34.9%	93.1%	16-bit
CIFAR-10	**This work**	**55.8%**	**93.7%**	**4-bit**

to the full-precision VGG-8 model, our compressed model with 4-bit precision achieves similar inference accuracy with 4.8% structure sparsity improvements. Instead of introducing a large amount of non-computed cells, our proposed mapping scheme deployed the compressed model densely and efficiently. Therefore, the overall memory cost will be reduced. For both VGG-16 and ResNet-56, the proposed method outperformed the fine-grid pruning [14] with 20.9% and 9.9% additional structural sparsity along with 4× precision reduction, while maintaining similar inference accuracy. With our proposed method and selected group size, the resulting sparsity and low precision model will eventually lead to the optimal solution between the resource consumption and accuracy.

V. CONCLUSION

In this brief, we presented the multi-group Lasso algorithm that efficiently maps pruned DNNs onto RRAM crossbars and evaluated the RRAM-based hardware accelerator design. Considering the rigid crossbars employed for IMC RRAM hardware, we investigated the tradeoffs between the hardware performance and software accuracy, and found the optimal pruning group size. The proposed hardware accelerator achieved up to 7 × /6× energy/area reduction with ResNet/VGG models for CIFAR-10 dataset with minimum accuracy degradation. The algorithm was also verified via 4-bit ResNet-18 on ImageNet dataset with >10× compression and minimum accuracy degradation.

REFERENCES

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE/CVF CVPR*, 2016, pp. 770–778.

[2] S. Park, J. Lee, S. Mo, and J. Shin, "Lookahead: A far-sighted alternative of magnitude-based pruning," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2020, pp. 1–20. [Online]. Available: https://openreview.net/pdf?id=ryl3ygHYDB

[3] P. Wang, Y. Ji, C. Hong, Y. Lyu, D. Wang, and Y. Xie, "SNrram: An efficient sparse neural network computation architecture based on resistive random-access memory," in *Proc. 55th ACM/ESDA/IEEE Design Autom. Conf. (DAC)*, 2018, pp. 1–6.

[4] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li, "Learning structured sparsity in deep neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 2074–2082.

[5] L. Yang, Z. He, and D. Fan, "Harmonious coexistence of structured weight pruning and ternarization for deep neural networks," in *Proc. Assoc. Adv. Artif. Intell. (AAAI)*, 2020, pp. 6623–6630.

[6] N. Verma *et al.*, "In-memory computing: Advances and prospects," *IEEE Solid-State Circuits Mag.*, vol. 11, no. 3, pp. 43–55, 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8811809

[7] E. R. Hsieh *et al.*, "High-density multiple bits-per-cell 1T4R RRAM array with gradual SET/RESET and its effectiveness for deep learning," in *Proc. IEEE Int. Electron Devices Meeting (IEDM)*, 2019, pp. 35.6.1–35.6.4.

[8] S. Yin *et al.*, "Monolithically integrated RRAM- and CMOS-based in-memory computing optimizations for efficient deep learning," *IEEE Micro*, vol. 39, no. 6, pp. 54–63, Nov./Dec. 2019.

[9] W. He *et al.*, "2-bit-per-cell RRAM-based in-memory computing for area-/energy-efficient deep learning," *IEEE Solid-State Circuits Lett.*, vol. 3, pp. 194–197, Jul. 2020. [Online]. Available: https://ieeexplore.ieee.org/document/9145716

[10] H. Mao *et al.*, "Exploring the granularity of sparsity in convolutional neural networks," in *Proc. IEEE/CVF CVPR Workshops*, 2017, pp. 1927–1934.

[11] L. Liang *et al.*, "Crossbar-aware neural network pruning," *IEEE Access*, vol. 6, pp. 58324–58337, 2018.

[12] Q. Wang, X. Wang, S. H. Lee, F.-H. Meng, and W. D. Lu, "A deep neural network accelerator based on tiled RRAM architecture," in *Proc. IEEE Int. Electron Devices Meeting (IEDM)*, 2019, p. 14.

[13] X. Ma *et al.*, "Tiny but accurate: A pruned, quantized and optimized memristor crossbar framework for ultra efficient DNN implementation," in *IEEE Asia South Pac. Design Autom. Conf. (ASP-DAC)*, 2020, pp. 301–306.

[14] C. Liu, F. Yu, Z. Qin, and X. Chen, "Enabling efficient ReRAM-based neural network computing via crossbar structure adaptive optimization," in *Proc. ACM/IEEE Int. Symp. Low Power Electron. Design*, 2020, pp. 133–138.

[15] L. Jiang *et al.*, "PIM-Prune: Fine-grain DCNN pruning for crossbar-based process-in-memory architecture," in *Proc. IEEE/ACM Design Autom. Conf.*, 2020, pp. 1–6.

[16] X. Peng, S. Huang, Y. Luo, X. Sun, and S. Yu, "DNN+NeuroSim: An end-to-end benchmarking framework for compute-in-memory accelerators with versatile device technologies," in *Proc. IEEE Int. Electron Devices Meeting (IEDM)*, 2019, pp. 32.5.1–32.5.4.

[17] M. Yuan and Y. Lin, "Model selection and estimation in regression with grouped variables," *J. Royal Stat. Soc. B, Stat. Methodol.*, vol. 68, no. 1, pp. 49–67, 2006.

[18] W. Wu *et al.*, "A methodology to improve linearity of analog RRAM for neuromorphic computing," in *Proc. IEEE Symp. VLSI Technol.*, 2018, pp. 103–104.

[19] Y. Xiang *et al.*, "Impacts of state instability and retention failure of filamentary analog RRAM on the performance of deep neural network," *IEEE Trans. Electron Devices*, vol. 66, no. 11, pp. 4517–4522, Nov. 2019.

[20] M. Cheng *et al.*, "TIME: A training-in-memory architecture for RRAM-based deep neural networks," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 38, no. 5, pp. 834–847, May 2019.

[21] J. Choi, S. Venkataramani, V. Srinivasan, K. Gopalakrishnan, Z. Wang, and P. Chuang, "Accurate and efficient 2-bit quantized neural networks," in *Proc. Conf. Syst. Mach. Learn. (SysML)*, 2019, pp. 1–12. [Online]. Available: https://mlsys.org/Conferences/2019/doc/2019/168.pdf

[22] W. Jiang, W. Wang, and S. Liu, "Structured weight unification and encoding for neural network compression and acceleration," in *Proc. IEEE/CVF CVPR Workshops*, 2020, pp. 3068–3076.

[23] J. Lin *et al.*, "Rescuing memristor-based computing with non-linear resistance levels," in *Proc. IEEE DATE*, 2018, pp. 407–412.

[24] M. Zhu and S. Gupta, "To prune, or not to prune: Exploring the efficacy of pruning for model compression," 2017. [Online]. Available: arXiv:1710.01878.