An Ultra-Efficient Memristor-Based DNN Framework with Structured Weight Pruning and Quantization Using ADMM

Geng Yuan[†]
Northeastern University
Boston, USA
yuan.geng@husky.neu.edu

Xiaolong Ma[†]
Northeastern University
Boston, USA
ma.xiaol@husky.neu.edu

Caiwen Ding
Northeastern University
Boston, USA
ding.ca@husky.neu.edu

Sheng Lin
Northeastern University
Boston, USA
lin.sheng@husky.neu.edu

Tianyun Zhang
Syracuse University
Boston, USA
tzhan120@syr.edu

Zeinab S. Jalali
Syracuse University
Syracuse, USA
zsaghati@syr.edu

Yilong Zhao
Shanghai JiaoTong University
Shanghai, China
ylzhao2018@gmail.com

Li Jiang
Shanghai JiaoTong University
Shanghai, China
jiangli@cs.sjtu.edu.cn

Sucheta Soundarajan
Syracuse University
Syracuse, USA
susounda@syr.edu

Yanzhi Wang
Northeastern University
Boston, USA
yanz.wang@northeastern.edu

***Abstract*—The high computation and memory storage of large deep neural networks (DNNs) models pose intensive challenges to the conventional Von-Neumann architecture, incurring substantial data movements in the memory hierarchy. The memristor crossbar array has emerged as a promising solution to mitigate the challenges and enable low-power acceleration of DNNs. Memristor-based weight pruning and weight quantization have been seperately investigated and proven effectiveness in reducing area and power consumption compared to the original DNN model. However, there has been no systematic investigation of memristor-based neuromorphic computing (NC) systems considering both weight pruning and weight quantization. In this paper, we propose an unified and systematic memristor-based framework considering both structured weight pruning and weight quantization by incorporating *alternating direction method of multipliers* (ADMM) into DNNs training. We consider hardware constraints such as crossbar blocks pruning, conductance range, and mismatch between weight value and real devices, to achieve high accuracy and low power and small area footprint. Our framework is mainly integrated by three steps, i.e., memristor-based ADMM regularized optimization, masked mapping and retraining. Experimental results show that our proposed framework achieves 29.81× (20.88×) weight compression ratio, with 98.38% (96.96%) and 98.29% (97.47%) power and area reduction on VGG-16 (ResNet-18) network where only have 0.5% (0.76%) accuracy loss, compared to the original DNN models. We share our models at anonymous link** http://bit.ly/2Jp5LHJ .

I. Introduction

With the rise of artificial intelligence, Deep Neural Networks have been widely used thanks to their high accuracy, excellent scalability, and self-adaptiveness [1]. DNN models are becoming deeper and larger, and are evolving fast to satisfy the diverse characteristics of broad applications. The high computation and memory storage of DNN models pose intensive challenges to the conventional Von-Neumann architecture, incurring substantial data movements in memory hierarchy.

To achieve high performance and energy efficiency, hardware acceleration of DNNs is intensively studied both in academia and industry [2–9]. DNN model compression techniques, including weight pruning [10–15] and weight quantization [16–18], are developed to facilitate hardware acceleration by reducing storage/computation in DNN inference with negligible impact on accuracy. However, as Moore's law is coming to an end [19], the acceleration of the conventional Von-Neumann architecture is limited to some extent.

To further mitigate the intensive computation and memory storage of DNN models, the next-generation device/circuit technologies beyond CMOS and novel computing architectures beyond the traditional Von-Neumann machine are investigated. The crossbar array of the recently discovered memristor devices (i.e., memristor crossbar) can be utilized to perform matrix-vector multiplication in the analog domain and solve systems of linear equations in $O(1)$ time complexity [20, 21]. Ankit et al. [22] implemented weight pruning techniques to NC systems using memristor crossbar arrays, which reduces the area (energy) consumption compared to the original network. However, for hardware implementations on on-chip neuromorphic computing systems, there are several limitations: (i) unbalanced workload; (ii) extra memory footprint on indices; (iii) irregural memory access. These will cause the circuit overheads in hardware implementations. To address these limitations, Wang. et al. [23] proposed group connection deletion, which prunes connections to reduce routing congestion between crossbar arrays.

On the other hand, Zhang. et al. [24] discussed the effectiveness of using the quantized conductance in memristor in multi-level logics. Song. et al. [25] investigated the generation of quantization loss in the memristor-based NC systems and its impacts on computation accuracy, and proposed a regularized offline learning method that can minimize the impact of quantization loss during neural network mapping. Weight quantization can mitigate hardware imperfection of memristor including state drift and process variations, caused by the imperfect fabrication process or by the device feature itself.

Because weight pruning and weight quantization techniques leverage different sources of redundancy, they may be combined to achieve higher DNN compression. However, there has been no systematic investigation of this effect in the memristor-based NC systems considering both weight pruning and weight quantization. In this paper, we propose an unified and systematic memristor-based framework considering both structured weight pruning and weight quantization, by incorporating ADMM into DNNs training. We consider hardware constraints such as crossbar blocks pruning, conductance range, and mismatch between weight value and real devices, to achieve high accuracy and low power and small area footprint. Our proposed framework can better mitigate the inaccuracy caused by the hardware imperfection compared to only weight quantization method [24, 25]. It contains *memristor-based ADMM regularized optimization*, *masked mapping* and *retraining steps*, which can guarantee the solution feasibility (satisfying all constraints) and provide high solution quality (maintaining test accuracy) at the same time. The contributions of this paper include:

- We systematically investigate the combination of structured weight pruning and weight quantization techniques leveraging different sources of redundancy, to achieve higher DNN compression ratio and low power and area in the domain of memristor-based NC systems.
- We adopt ADMM, an effective optimization technique for

[†]These authors contributed equally.

978-1-7281-2954-9/19/$31.00 ©2019 IEEE

general and non-convex optimization problems, to jointly optimize weight pruning and weight quantization problems during training for higher model accuracy.

We evaluate our proposed framework on different networks. Experimental results show that our proposed framework can achieve 29.81×(20.88×) weight compression ratio, with 98.38% (96.96%) and 98.29% (97.47%) power and area reduction on VGG-16 (ResNet-18) network where only have 0.5% (0.76%) accuracy loss, compared to the original DNN models.

II. Background on Memristors

A. Memristor Crossbar Model

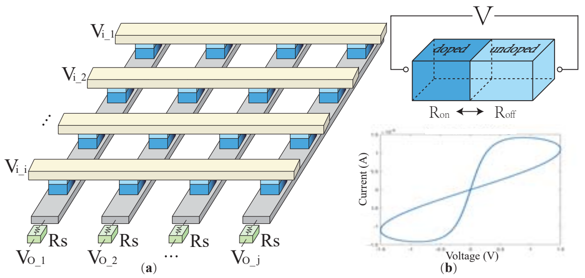

Fig. 1: (a) Memristor crossbar performs maxtrix-vector multiplication. (b) Memristor model and its V-I curve.

Memristor has shown remarkable characteristics as one of the most promising emerging technologies as shown in Figure 1 [26]. The memristor has many promising features, such as non-volatility, low-power, high integration density, and excellent scalability. Memristors can be formed as a crossbar structure [27], as shown in Figure 1. Each pair of horizontal Word-line (WL) and vertical Bit-line (BL) is connected across a memristor device. Given the input voltage vector $\mathbf{v}_i$ and the weight matrix $\mathbf{W}$ which can be constructed by a preprogramed crossbar array, the matrix multiplication result $\mathbf{v}_o$ can be easily obtained by measuring the current across the resistor R_S. By nature, the memristor crossbar array is attractive for matrix computations with high degree of parallelism which can achieve the time complexity of $O(1)$. Based on this superior feature, the memristor-based computing system can provide a promising solution to reduce the latency and improve the energy efficiency of neuromorphic computation.

B. Hardware Imperfection of Memristor Crossbars and Mitigation Techniques

Hardware imperfection of Memristor is mainly caused by the imperfect fabrication process or by the device feature itself. These significant issues cannot be ignored in hardware design, which is different from the software-based system design.

1) State Drift: Memristor device consists of a thin-film structure, and the film is divided into two regions. One region is highly doped with oxygen vacancies and another region is an undoped. Applying an electric field across the device over time would lead to the migration of oxygen vacancies and change the memristance state, which is called state drift [28]. Thus, after a certain number of read operations, the resistance of the device will drift caused by the accumulative effect of applying the same direction voltage. As a result of the state drift effect, the imprecision will be incurred when the memristor's state drifts to the other state levels.

2) Process Variation.: Process variation is also phenomenal as the process technology scales to nanometer level. It mainly comes from the line-edge roughness, oxide thickness fluctuations, and random dopant variations that affect the memristor device performance [29]. The process variation will cause the hardware non-ideal behavior, which usually means the accuracy degradation [30].

It can be observed that quantization on resistance values plays an important role in dealing with hardware imperfections. However, the prior work on mitigating the effect of hardware imperfections are mainly *ad hoc*, lacking a systematic, algorithm-hardware co-optimization framework to improve overall resilience. Our proposed framework can mitigate the inaccuracy caused by the hardware imperfection, while achieves high hardware efficiency as well.

III. A Unified and Systematic Memristor-Based Framework for DNNs

The memristor crossbar structure has shown promising features in neuromorphic computing systems compared to the traditional CMOS technologies[22]. However, as DNN goes deeper and deeper, the massive weight computation and weight storage introduce severe challenges in neuromorphic computing system hardware implementations. On the other hand, to systematically address hardware imperfections of memristor crossbars, in this paper, we propose a integrated memristor-based framework

A. Unified and Systematic Memristor-Based Framework using ADMM

1) Connection to ADMM: ADMM [31] is a powerful optimization tool, by decomposing an original problem into two subproblems that can be solved separately and iteratively. Consider the optimization problem $\min_{\mathbf{x}} \; f(\mathbf{x}) + g(\mathbf{x})$. In ADMM, the problem is first re-written as

$$\min_{\{\mathbf{x},\mathbf{z}\}} \; f(\mathbf{x}) + \mathbf{g}(\mathbf{z}), \quad \text{subject to } \mathbf{x} = \mathbf{z} \tag{1}$$

Next, by using augmented Lagrangian [31], the above problem is decomposed into two subproblems on $\mathbf{x}$ and $\mathbf{z}$. The first is $\min_{\mathbf{x}} \; f(\mathbf{x}) + q_1(\mathbf{x})$, where $q_1(\mathbf{x})$ is a quadratic function. As $q_1(\mathbf{x})$ is convex, the complexity of solving subproblem 1 is the same as minimizing $f(\mathbf{x})$. Subproblem 2 is $\min_{\mathbf{z}} \; g(\mathbf{z}) + q_2(\mathbf{z})$, where $q_2(\mathbf{z})$ is again a quadratic function. The two subproblems will be solved iteratively until convergence is achieved [32].

2) Unified Memristor-Based Framework: There is a difficulty in using ADMM directly due to the non-convex nature of the objective function for DNN training, and thereby lacking of any guarantees on solution feasibility and solution quality. It becomes even more challenging when incorporating ADMM into training the memristor-based DNN model, where we need to consider hardware constraints such as crossbar blocks pruning, conductance range, and mismatch between weight value and real devices. To overcome this challenge, we proposed an unified memristor-based framework including *memristor-based ADMM regularized optimization*, *masked mapping* and *retraining steps*, which can guarantee the solution feasibility (satisfying all constraints) and provide high solution quality (maintaining test accuracy) at the same time.

First, the *memristor-based ADMM regularized optimization* starts from a pre-trained DNN model without compression. Consider an N-layer DNNs, sets of weights and biases of the i-th (CONV or FC) layer are denoted by $\mathbf{W}_i$ and $\mathbf{b}_i$, respectively. And the *loss function* of the N-layer DNN is denoted by $f\big(\{\mathbf{W}_i\}_{i=1}^N, \{\mathbf{b}_i\}_{i=1}^N\big)$. Combining the task of memristor-based structured pruning and weight quantization, the overall problem is defined by

$$\begin{aligned} \underset{\{\mathbf{W}_i\},\{\mathbf{b}_i\}}{\text{minimize}} \quad & f\big(\{\mathbf{W}_i\}_{i=1}^N, \{\mathbf{b}_i\}_{i=1}^N\big), \\ \text{subject to} \quad & \mathbf{W}_i \in \mathbf{P}_i, \; \mathbf{W}_i \in \mathbf{Q}_i, \; i = 1, \ldots, N. \end{aligned} \tag{2}$$

Given the value of α_i, the set $\mathbf{P}_i = \{\mathbf{W}_i | card(supp(\mathbf{W}_i)) \leq \alpha_i\}$ reflects the constraint for memristor-based structured weight pruning, where 'card' refers to cardinality and 'supp' refers to the support set. Elements in $\mathbf{P}_i$ are the solution of $\mathbf{W}_i$ satisfying the number of non-zero elements (after structured pruning and memristor crossbar mapping) in $\mathbf{W}_i$ is limited by α_i for layer i. Similarly, elements in $\mathbf{Q}_i$ are the solutions of $\mathbf{W}_i$, in which elements in $\mathbf{W}_i$ assume one of $q_{i,1}, q_{i,2}, \cdots, q_{i,M_i}$ (memristor state values), where M_i denotes the number of available quantization level in layer i. Please note that the $q_{i,j}$ value indicates the j-th quantization level in layer i, and $q_{i,j} \in [-cond_{max}, -cond_{min}] \cup [cond_{min}, cond_{max}]$, where $cond_{min}$, $cond_{max}$ are the minimum and maximum valid conductance value of a specified memristor device. More specifically, we use indicator functions to incorporate the memristor-based structured pruning and weight quantization constraints into the objective function, which are

$$g_i(\mathbf{W}_i) = \begin{cases} 0 & \text{if } \mathbf{W}_i \in \mathbf{P}_i, \\ +\infty & \text{otherwise,} \end{cases} h_i(\mathbf{W}_i) = \begin{cases} 0 & \text{if } \mathbf{W}_i \in \mathbf{Q}_i, \\ +\infty & \text{otherwise,} \end{cases}$$

for $i = 1, \ldots, N$. Then the original problem (2) can be equivalently rewritten as

$$\underset{\{\mathbf{W}_i\},\{\mathbf{b}_i\}}{\text{minimize}} \quad f\big(\{\mathbf{W}_i\}_{i=1}^N, \{\mathbf{b}_i\}_{i=1}^N\big) + \sum_{i=1}^N g_i(\mathbf{W}_i) + \sum_{i=1}^N h_i(\mathbf{W}_i). \quad (3)$$

We incorporate auxiliary variables $\mathbf{Y}_i$ and $\mathbf{Z}_i$, dual variables $\mathbf{U}_i$ and $\mathbf{V}_i$, then apply ADMM to decompose problem (3) into three subproblems. After that, We solve these subproblems iteratively until the convergence. Assume in iteration k, the first subproblem is

$$\begin{aligned} \underset{\{\mathbf{W}_i\},\{\mathbf{b}_i\}}{\text{minimize}} \quad & f\big(\{\mathbf{W}_i\}_{i=1}^N, \{\mathbf{b}_i\}_{i=1}^N\big) + \sum_{i=1}^N \frac{\rho_i}{2} \|\mathbf{W}_i - \mathbf{Y}_i^k + \mathbf{U}_i^k\|_F^2 \\ & + \sum_{i=1}^N \frac{\rho_i}{2} \|\mathbf{W}_i - \mathbf{Z}_i^k + \mathbf{V}_i^k\|_F^2, \end{aligned} \quad (4)$$

The first term in problem (4) is the differentiable (non-convex) loss function of the DNN, while the other quadratic terms are convex. As a result, this subproblem can be solved by stochastic gradient descent (e.g., the ADAM algorithm [33]) similar to training the original DNN.

The solution $\{\mathbf{W}_i\}$ of subproblem 1 is denoted by $\{\mathbf{W}_i^{k+1}\}$. Then we aim to derive $\{\mathbf{Z}_i^{k+1}\}$ and $\{\mathbf{Y}_i^{k+1}\}$ in subproblem 2 and 3. Thanks to the characteristics in combinatorial constraints (the memristor-based structured pruning and weight quantization constraints), the optimal, analytical solution of the two subproblems are Euclidean projections. We can prove that the projections are: keeping α_i elements with largest magnitudes and setting remaining weights to zero; and to quantize every weight element to the closest valid memristor state value. Finally, we update dual variables $\mathbf{U}_i$ and $\mathbf{V}_i$ according to ADMM rule [31] and thereby complete the k-th iteration in *memristor-based ADMM regularized optimization*.

Masked Mapping and Retraining: We first perform the Euclidean projection (mapping) on the derived $\mathbf{W}_i$ to guarantee that at most α_i values in each layer are non-zero. Since the zero weights will not be mapped on the memristor crossbar, we can mask the zero weights and retrain the DNN with non-zero weights using training sets. Particularly, this retraining step is similar to the *ADMM regularized hardware optimization* step, but only the memristor weight quantization constraints need to be satisfied. In this way test accuracy can be partially restored.

B. Memristor-Based Structured Pruning & Quantization

1) Memristor-Based Structured Weight Pruning: In order to be hardware-friendly, we use structured pruning method [11] instead of the irregular pruning method [10] to reduce weights parameters. There are different types of structured sparsity, filter-wise sparsity, channel-wise sparsity, shape-wise sparsity as shown in Figure 2. In the proposed framework, We incorporate structured pruning in ADMM regularization, where memristor features are considered. Compared to [23], our proposed method can better explore the sparsity on weight matrices, with negligible accuracy degradation, resulting in better area saving and lower power consumption.

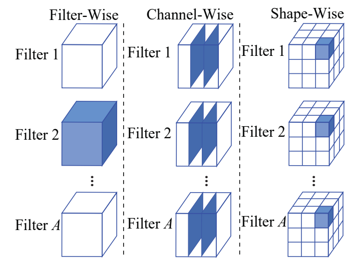

Fig. 2: Illustration of filter-wise, channel-wise and shape-wise structured sparsities.

To better illustrate how structured pruning saves the memristor crossbars, we transform the weight tensors of a CONV layer to general matrix multiplication (GEMM) format [34]. As shown in Figure 3 (a) (GEMM view), the structured pruning corresponds to reducing rows or columns. The three structured sparsities, along with their combinations, will reduce the weight dimension in GEMM while maintaining a full matrix. Indices are not needed and weight quantization will be better supported.

Figure 3 (b) shows a memristor implementation size view and memristor crossbar area reduction on different types of sparsities. By applying filter (row) pruning and shape/channel (column) pruning, as shown in the top of Figure 3(b), either blocks of memristor crossbar or numbers of memristor crossbars can be saved compared to the original design.

The Figure 4 shows how we map the weight parameters on the memristor crossbars. As shown in the GEMM view in Figure 3 (a), assuming a CONV layer has n filters, m channels (including k columns of weights), denoted as $\mathbf{W} \in \mathbb{R}^{n \times k}$. Generally, the size of a single memristor crossbar is limited because the reading and writing error will increase by using

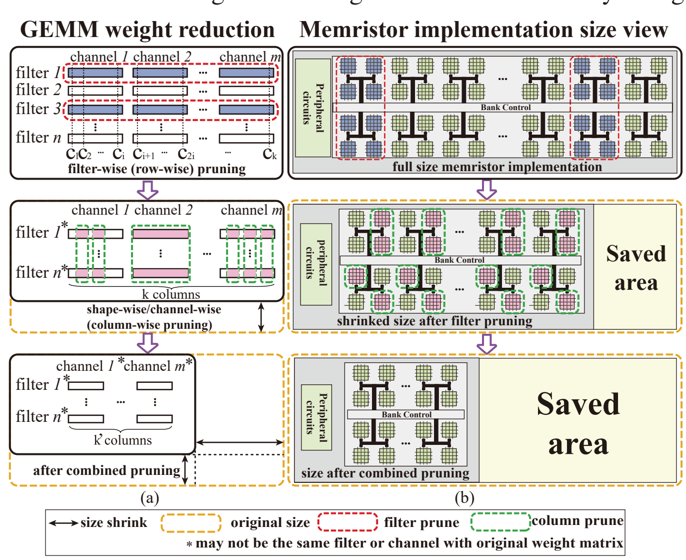

Fig. 3: Structured weight pruning and reduction of hardware resources

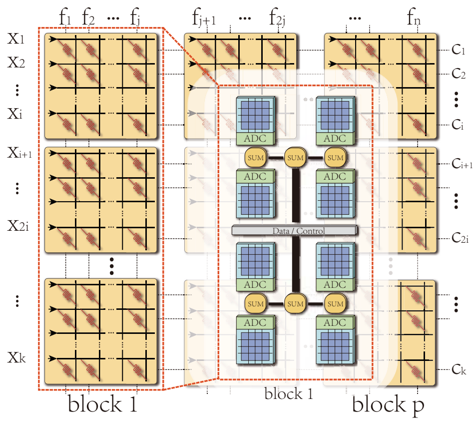

Fig. 4: Weights Mapping on Memristor crossbars

larger crossbar size [35]. Thus, multiple memristor crossbars are used to accommodate the large size weight matrix. To maintain accuracy, the single memristor crossbar size in our design is no larger than 128×64 [36] and is identical for all DNN layers. As shown in Figure 4, each crossbar has i rows and j columns. We use X and f to represent the inputs and filters, where c represents the column of weights as shown in Figure 3 (a). Since there are k weights in a filter, we need to use the columns at same position from at least k/j different crossbars to store one filter's weights. Therefore, j filters can be fully mapped on those crossbars as one block shown in Figure 4. There are n filters in total, therefore we need at least $p = n/j$ blocks to fully map the whole weight matrix ($\mathbf{W} \in \mathbb{R}^{n \times k}$). Within each block, the outputs of each crossbar will be propagated through an ADC. Then We column-wisely sum the intermediate results of all crossbars.

2) Memristor-Based Weight Quantization: The weights of the DNNs are represented by the conductance of memristors on memristor crossbars and the output of the memristor crossbars can be obtained by measuring the accumulated current. Due to the limited conductance range of the memristor devices, the weight values exceeding conductance range cannot be represented precisely. On the other hand, within the conductance range, accuracy loss also exists because of the mismatch between weight values and real memristor devices.

To mitigate the limitation of conductance range, we incorporate the conductance range constraint of the memristor device (i.e., $\mathbf{Q}_i \in [cond_{min}, cond_{max}]$) into DNNs training. To mitigate the accuracy degradation caused by the weight mismatch, we incorporate the constraint of conductance state levels (i.e., $q_{i,1}, q_{i,2}, \cdots, q_{i,M_i} \in [cond_{min}, cond_{max}]$) into DNNs training. Here set $\mathbf{Q}_i = \{$ the value of every element is one of the values in $q_1, q_2, \cdots, q_M\}$ to represent the constraint, and $q_1, q_2, \cdots, q_M$ are all available quantized states. Theoretically, the conductance of the memristor can be set to any states within its available range. In reality, the memristor conductance states are limited by the resolution that the peripheral write and read circuitry can provide. Generally speaking, more state levels require more sophisticated peripheral circuitry. In order to reduce the overheads caused by the peripheral circuitry and satisfy the robustness of the whole system, the conductance range will be quantized to several distinctive state levels and represented by discrete states.

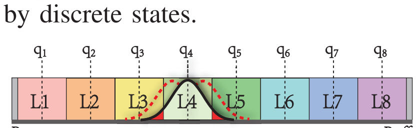

Fig. 5: Multi-level Memristor Storing 3-bit weight

Figure 5 illustrates an example of an 8-level (3bits) memristor with linear conductance level (where it may behave as nonlinear in real designs [37]). The distribution curve shows a possible range that the memristor state might be actually set to, when the writing target state is q_4. An error will incur (hardware imprecision) when the actual written state is different from the target state level. In order to minimize the error caused by the hardware imprecision, in our constraint of conductance state levels, we set the quantized values as the mean value of each state level. To optimize the overall performance, the number of memristor state levels is also considered in our proposed framework. By quantizing the weights to fewer bits while maintain the overall accuracy, we can further improve the performance since fewer state levels provide longer distance for single state, resulting in better error resilience and reducing the hardware imprecision.

Another advantage is the design area and power consumption can be reduced by quantizing the weights to fewer bits. According to the state-of-the-art design of neuromorphic computing using memristor, a practical assumption is that the memristor cell can represent 16 weight levels (4-bit weights) [38]. To ensure a relatively high accuracy, usually two (or more) memristors are bundled to represent weights with high resolution (more bits) [39]. On the other hand, since the memristor device only has positive conductance value while the weights are positive or negative, we use different memristor crossbars to represent the positive weights and negative weights separately. As an illustration in Figure 6, a 9-bit weight value can be represented using a 8-bit positive block and a 8-bit negative block, where each of the 8-bit block is formed by two 4-bit memristor crossbars. In general, the cost of the ADCs and other peripheral circuits will grow exponentially for adding every extra bit precision. Thus, the overhead of the peripheral circuit can be significantly reduced by quantizing the weights to fewer bits. The total design area and power consumption can be reduced as well.

Figure 7 shows the weights distribution of a CONV layer in ResNet18 using CIFAR-10 dataset before (b) and after (a) quantization, after structured weight pruning. For a 5-bit quantization using our proposed method, the weights are quantized into 32 different levels within memristor's valid conductance range.

IV. Experimental Results

In this section, we evaluate our systematic structured weight pruning and weight quantization framework on MNIST dataset using LeNet-5 network structure and CIFAR-10 dataset using ConvNet (4 CONV layers and 1 FC layer), VGG-16 and ResNet-18 network structures. All models are designed using PyTorch API and oriented to match the memristor's physical characteristics. Our hardware performance results such as

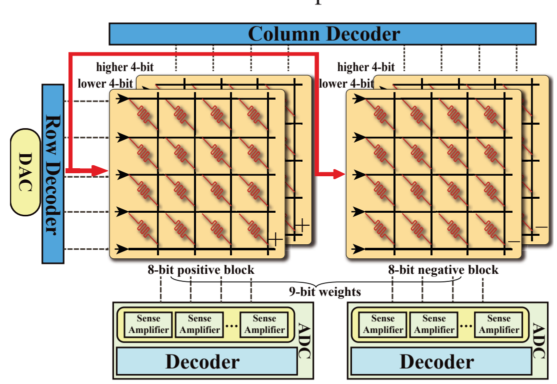

Fig. 6: Represent Weights Using Multi-Memristor Crossbars

TABLE I: Structured Weight pruning results on multi-layer network on MNIST, CIFAR-10 datasets (*calculation is based on bolded results)

Method	Structured Weight Pruning Statistics (9-bit)				Quantization & Accuracy		
	Original Accuracy	Pruned Accuracy	Crossbar Area Saved	Compression Ratio	7-bit	6-bit	5-bit
MNIST							
Group Scissor [23]	99.15%	99.14%	75.94%	4.16×	-	-	-
our LeNet-5	**99.17%**	**99.15%**	**94.34%**	**17.69×**	**98.97%**	**99.03%**	**99.03%**
		99.02%	97.30%	37.06×	98.85%	98.82%	98.77%
		98.33%	99.05%	105.52×	98.28%	98.10%	98.23%
*numbers of parameter reduced: **8.65K**							
CIFAR-10							
Group Scissor [23]	82.01%	82.09%	57.45%	2.35×	-	-	-
our ConvNet	**84.41%**	84.55%	57.45%	2.35×	84.18%	83.50%	80.81%
		84.53%	**65.87%**	**2.93×**	**83.73%**	**82.25%**	**80.41%**
		83.58%	82.99%	5.88×	83.00%	81.54%	78.18%
our VGG-16	**93.70%**	93.76%	89.26%	9.31×	93.67%	93.64%	93.26%
		93.36%	**96.65%**	**29.81×**	**92.97%**	**92.51%**	**91.21%**
our ResNet-18	**94.14%**	93.79%	91.49%	11.75×	93.68%	93.25%	92.92%
		93.20%	**95.21%**	**20.88×**	**93.13%**	**92.65%**	**92.44%**
*numbers of parameter reduced on *ConvNet*: **102.30K**, *VGG-16*: **13.98M**, *ResNet-18*: **10.46M**							

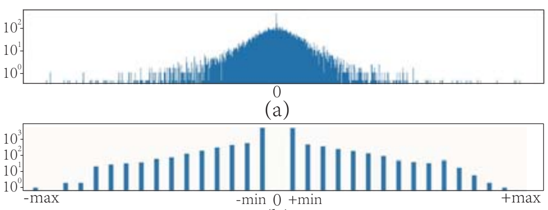

Fig. 7: Distribution of the weights: (a)before quantization, (b) after 5-bit quantization

power consumption, area cost of the memristor device and its peripheral circuits are simulated by using NVSim [40] and our MATLAB model. In our memristor model, $R_{min} = 1M\Omega$, $R_{max} = 10M\Omega$, with 4-bit precision, and the peripheral circuits are using 45nm technology. We use 128×64 crossbar size on ResNet-18 and VGG-16, where ConvNet and LeNet-5 uses 32×32 crossbar size. The experiments are done on an eight NVIDIA GTX-2080Ti GPUs server.

In this work, multiple 9-bit non-pruned models on different networks are used as our original DNN models, and results of structured weight pruning using our original DNN models show that, on memristor LeNet-5 model, we achieve 17.69× weight reduction without accuracy loss, 37.06× weight reduction with negligible accuracy loss and 105.52× weight reduction within 1% accuracy loss. Meanwhile we shrink memristor crossbar area by more than 94%. On muti-layers CNN for CIFAR-10, we achieve a higher accuracy compared with [23]. On deeper neural network structures such as VGG-16 and ResNet-18, we manage to compress each model unprecedentedly by 29.81× with negligible accuracy loss and 20.88× within 1% accuracy loss, respectively. We manage to save more crossbar area compared with [23], and reduce 96.65% of the crossbar area for VGG-16 and 95.21% for ResNet-18. The experimental results illustrate great potential for incorporating ADMM into structured weight pruning and quantization techniques on memristor-based DNN design, which will tremendously reduce the area and power consumption.

A. Experimental Results on Structured Weight Pruning

In our experiment, we compare our proposed framework with Group Scissor [23] as shown in Table I. Please note that we only prune CONV layers because they perform most of the FLOPs in the network calculation. On MNIST dataset, our original CNN model achieves 99.17% accuracy, and 99.15% accuracy with structured weight pruning. We also reduce our model size using extreme prune configuration, the results shows our method gets 98.33% accuracy when we compress our model by 105.52×.

On CIFAR-10 dataset, we construct different networks to test our method. Compared with the Group Scissor [23], we not only achieve higher test accuracy using same compression ratio (2.35×), but also manage to maintain same accuracy with a even higher compression ratio (2.93×). For deeper network structures like VGG-16 and ResNet-18, we introduce such high regular sparsity into networks without accuracy degradation. Our framework reduces 13.98M and 10.46M weight volume for VGG-16 and ResNet-18 respectively.

B. Experimental Results on Weight Quantization for Memristor Crossbar Mapping

From the discussion in Section III-B.2, we can see that fewer bits can reduce the power as well as the memristor crossbar area. However, quantizing weights to some specific values will cause non-negligible accuracy degradation. In this paper, to mitigate accuracy degradation, we adopt ADMM to dynamically optimize well-leveled groups of weights which can be actually mapped on the memristors. By including memristor characteristics as discussed in Section III-B.2, our quantization process does not map weights to zero, and our state levels are zero-symmetric. Table I shows different configurations for weight quantization and Figure 8 shows the power and area. Experimental results demonstrate that our framework maintains high weight prune ratio and fewer bits with promising test accuracy. According to 6-bit quantization results in Table I, there is only 0.1% accuracy degradation after quantizing LeNet-5 on 17.69× model, and only 0.2% accuracy degradation after quantizing the 105.52× model. For a larger dataset such as CIFAR-10, a shallow ConvNet will introduce around 1.0% accuracy degradation for our designed configuration (2.35×) and 2.0% accuracy degradation on a 5.88× compressed model, however as the network structure getting deeper, the accuracy drops around 0.1% in a 9.31× compressed VGG-16 model and 0.5% in a 11.75× compressed ResNet-18 model, and as the compression ratio gets larger, accuracy drops 0.8% in a 29.81× compressed VGG-16 model and 0.6% in a 20.88× compressed ResNet-18 model.

As shown in Figure 8, fewer bits represenation results in less power consumption and smaller area footprint, because the overhead of peripheral circuits such as ADCs and DACs

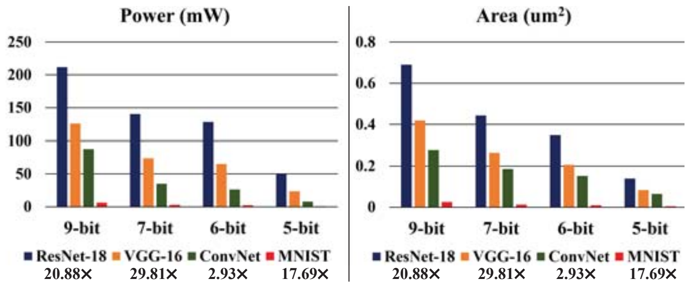

Fig. 8: Total power and area reduction on compressed models using different quantization bits and networks

will significantly decrease by lower the computing precision. There is a tremendous power and area reduction using the 5-bit quantization, since all memristor crossbars for higher bits representations are no longer needed. Beside the power and area reduction, fewer bits representation mitigates hardware imperfection of memristor including state drift and process variations. Compared to original DNN models, our 5-bit quantization models can achieve the largest power (area) reduction as 96.95% (97.46%), 98.38% (98.28%), 95.91% (89.74%) and 96.96% (93.97%) on ResNet-18, VGG-16, ConvNet and LeNet-5, respectively, among different bits representations.

V. Conclusion

In this paper, we propose an unified and systematic memristor-based framework with both structured weight pruning and weight quantization by incorporating ADMM into DNNs training. Three steps are mainly incorporated in our framework, which are *memristor-based ADMM regularized optimization*, *masked mapping* and *retraining*. We evaluate our proposed framework on different networks, and for each network, several pruning and quantizaiton scenarios are tested. On LeNet-5 and ConvNet, we can easily achieve better results comparing to Gourp Scissor. On VGG-16 and ResNet-18, after structured weight pruning and quantization, significant weight compression ratio, power and area reduction 5-bit weight representation can achieve significant power and area reduction network where only result in negligible accuracy loss, compared to the original DNN models.

Acknowledgment

This work is funded by National Science Foundation CCF-1637559. We thank all anonymous reviewers for their feedback.

References

[1] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[2] C. Ding, S. Liao, Y. Wang, Z. Li *et al.*, "Circnn: accelerating and compressing deep neural networks using block-circulant weight matrices," in *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*. ACM, 2017, pp. 395–408.

[3] Y. Wang, C. Ding, and et al., "Towards ultra-high performance and energy efficiency of deep learning systems: an algorithm-hardware co-optimization framework," *AAAI2018*, Feb 2018.

[4] C. Ding, A. Ren, and et al., "Structured weight matrices-based hardware accelerators in deep neural networks," *Proceedings of GLSVLSI 18*, 2018.

[5] X. Ma, Y. Zhang, and et al., "An area and energy efficient design of domain-wall memory-based deep convolutional neural networks using stochastic computing," *2018 19th ISQED*, Mar 2018.

[6] A. Shrestha, H. Fang, Q. Wu, and Q. Qiu, "Approximating back-propagation for a biologically plausible local learning rule in spiking neural networks," in *ICONS*, 2019, (in press).

[7] H. Fang, A. Shrestha, De Ma, and Q. Qiu, "Scalable noc-based neuromorphic hardware learning and inference," in *2018 IJCNN*, July 2018.

[8] H. Fang, A. Shrestha, Z. Zhao, Y. Wang, and Q. Qiu, "A general framework to map neural networks onto neuromorphic processor," in *20th ISQED*, March 2019, pp. 20–25.

[9] H. Li, N. Liu, and et al., "Admm-based weight pruning for real-time deep learning acceleration on mobile devices," in *Proceedings of GLSVLSI*. ACM, 2019.

[10] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *NeurIPS*, 2015.

[11] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li, "Learning structured sparsity in deep neural networks," in *NeurIPS*, 2016, pp. 2074–2082.

[12] T. Zhang, K. Zhang, and et al., "Adam-admm: A unified, systematic framework of structured weight pruning for dnns," *arXiv preprint arXiv:1807.11091*, 2018.

[13] X. Ma, G. Yuan, and et al., "Resnet can be pruned 60x: Introducing network purification and unused path removal (p-rm) after weight pruning," *arXiv preprint arXiv:1905.00136*, 2019.

[14] S. Ye, X. Feng, and et al., "Progressive dnn compression: A key to achieve ultra-high weight pruning and quantization rates using admm," *arXiv preprint arXiv:1903.09769*, 2019.

[15] W. Niu, X. Ma, Y. Wang, and B. Ren, "26ms inference time for resnet-50: Towards real-time execution of all dnns on smartphone," *arXiv preprint arXiv:1905.00571*, 2019.

[16] E. Park, J. Ahn, and S. Yoo, "Weighted-entropy-based quantization for deep neural networks," in *CVPR*, 2017.

[17] J. Wu, C. Leng, Y. Wang, Q. Hu, and J. Cheng, "Quantized convolutional neural networks for mobile devices," in *CVPR*, 2016.

[18] S. Lin, X. Ma, and et al., "Toward extremely low bit and lossless accuracy in dnns with progressive admm," *arXiv preprint arXiv:1905.00789*, 2019.

[19] M. M. Waldrop, "The chips are down for moores law," *Nature News*, vol. 530, no. 7589, 2016.

[20] L. Chua, "Memristor-the missing circuit element," *IEEE Transactions on circuit theory*, vol. 18, no. 5, pp. 507–519, 1971.

[21] G. Yuan, C. Ding, and et al., "Memristor crossbar-based ultra-efficient next-generation baseband processors," in *MWSCAS*. IEEE, aug 2017.

[22] A. Ankit, A. Sengupta, and K. Roy, "Transnformer: Neural network transformation for memristive crossbar based neuromorphic system design," in *Proceedings of the 36th International Conference on Computer-Aided Design*. IEEE Press, 2017, pp. 533–540.

[23] Y. Wang, W. Wen, B. Liu, D. Chiarulli, and H. Li, "Group scissor: Scaling neuromorphic computing design to large neural networks," in *DAC*. IEEE, 2017.

[24] Y. Zhang, N. I. Mou, P. Pai, and M. Tabib-Azar, "Quantized current conduction in memristors and its physical model," in *SENSORS, 2014 IEEE*. IEEE, 2014, pp. 819–822.

[25] C. Song, B. Liu, W. Wen, H. Li, and Y. Chen, "A quantization-aware regularized learning method in multilevel memristor-based neuromorphic computing system," in *2017 NVMSA*. IEEE, 2017.

[26] A. G. Radwan, M. A. Zidan, and K. N. Salama, "HP Memristor mathematical model for periodic signals and DC," in *2010 53rd IEEE International Midwest Symposium on Circuits and Systems*, aug 2010.

[27] M. Hu, H. LI, Q. Wu, and G. S. Rose, "Hardware realization of bsb recall function using memristor crossbar arrays," in *DAC Design Automation Conference 2012*, 2012, pp. 498–503.

[28] J. J. Yang, M. D. Pickett, X. Li, D. A. Ohlberg, D. R. Stewart, and R. S. Williams, "Memristive switching mechanism for metal/oxide/metal nanodevices," *Nature Nanotechnology*, 2008.

[29] S. Kaya, A. R. Brown, A. Asenov, D. Magot, e. D. LintonI, T.", and C. Tsamis, "Analysis of statistical fluctuations due to line edge roughness in sub-0.1μm mosfets," in *Simulation of Semiconductor Processes and Devices 2001*. Springer Vienna, 2001, pp. 78–81.

[30] S. Pi and et al., "Cross point arrays of 8 nm x 8 nm memristive devices fabricated with nanoimprint lithography," *Journal of Vacuum Science & Technology B: Microelectronics and Nanometer Structures*, 2013.

[31] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein *et al.*, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine learning*, 2011.

[32] H. Ouyang, N. He, L. Tran, and A. Gray, "Stochastic alternating direction method of multipliers," in *ICML*, 2013, pp. 80–88.

[33] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[34] S. Chetlur, C. Woolley, P. Vandermersch, J. Cohen, J. Tran, B. Catanzaro, and E. Shelhamer, "cudnn: Efficient primitives for deep learning," *arXiv preprint arXiv:1410.0759*, 2014.

[35] M. Hu, J. P. Strachan, Zhiyong Li, R. Stanley, and Williams, "Dot-product engine as computing memory to accelerate machine learning algorithms," in *2016 ISQED*. IEEE, mar 2016, pp. 374–379.

[36] M. Hu, C. E. Graves, and et al., "Memristor-Based Analog Computation and Neural Network Classification with a Dot Product Engine," *Advanced Materials*, 2018.

[37] J. Lin, L. Xia, Z. Zhu, H. Sun, Y. Cai, H. Gao, M. Cheng, X. Chen, Y. Wang, and H. Yang, "Rescuing memristor-based computing with non-linear resistance levels," in *DATE 2018*, 2018.

[38] M. Courbariaux, J. P. David, and Y. Bengio, "Training deep neural networks with low precision multiplications," in *ICLR*, 2015.

[39] P. Chi, S. Li, C. Xu, T. Zhang, and *et al.*, "Prime: A novel processing-in-memory architecture for neural network computation in reram-based main memory," in *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture*, 2016.

[40] X. Dong, C. Xu, and et al., "Nvsim: A circuit-level performance, energy, and area model for emerging nonvolatile memory," *IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS*, vol. 31, no. 7, 2012.