Tiny but Accurate: A Pruned, Quantized and Optimized Memristor Crossbar Framework for Ultra Efficient DNN Implementation

Xiaolong Ma[†1], Geng Yuan[†1], Sheng Lin[1], Caiwen Ding[2], Fuxun Yu[3], Tao Liu[4], Wujie Wen[5], Xiang Chen[3], Yanzhi Wang[1]
[1]Northeastern University, [2]University of Connecticut, [3]George Mason University,
[4]Florida International University, [5]Lehigh University
E-mail: [1]{ma.xiaol, yuan.geng, lin.sheng,}@husky.neu.edu, [1]yanz.wang@northeastern.edu,
[2]caiwen.ding@uconn.edu, [3]{fyu2, xchen26}@gmu.edu, [4]tliu023@fiu.edu, [5]wuw219@lehigh.edu

Abstract— The memristor crossbar array has emerged as an intrinsically suitable matrix computation and low-power acceleration framework for DNN applications. Many techniques such as memristor-based weight pruning and memristor-based quantization have been studied. However, the high accuracy solution for the above techniques is still waiting for unraveling. In this paper, we propose a memristor-based DNN framework which combines both structured weight pruning and quantization by incorporating *ADMM* algorithm for better pruning and quantization performance. We also discover the non-optimality of the ADMM solution in weight pruning and the unused data path in a structured pruned model. We design a software-hardware co-optimization framework which contains the first proposed *Network Purification* and *Unused Path Removal* algorithms targeting on post-processing a structured pruned model after ADMM steps. By taking memristor hardware constraints into our whole framework, we achieve extreme high compression rate with minimum accuracy loss. For quantizing structured pruned model, our framework achieves nearly no accuracy loss after quantizing weights to 8-bit memristor weight representation. We share our models at anonymous link https://bit.ly/2VnMUy0**.**

1 Introduction

Structured weight pruning [1–3] and weight quantization [4,5] techniques are developed to facilitate weight compression and computation acceleration to solve the high demand for parallel computation and storage resources [6–8]. However, Even with compressed models, computation complexity still burden the overall performance of the state-of-the-art CMOS hardware applications.

To mitigate the bottleneck caused by CMOS-based DNN architectures [9, 10], the next-generation device/circuit technologies [11, 12] emerge with their highlighted non-volatility, high energy efficiency, in-memory computing capability and high scalability. Memristor crossbar device has shown its potential for bearing all these characteristic which makes it intrinsically suitable for large DNN hardware architecture design. Motivated by the fact that there is no precedent model that is structured pruned and quantized as well as satisfying memristor hardware constraints, in this work, a *memristor-based ADMM regularized optimization* method is utilized both on structured pruning and weight quantization in order to mitigate the accuracy degradation during extreme model compression. A structured pruned model can potentially benefit for high-parallelism implementation in crossbar architecture. Further more, quantized weights can reduce hardware imprecision during read/write procedure, and save more hardware footprint due to less peripheral circuits are needed to support fewer bits.

However, an ADMM pruning method [3] cannot fully exploit all redundancy in a neural network model. Therefore, we design a hardware-software co-optimization framework in which we investigate *Network Purification* and *Unused Path Removal* after the procedure of *structured weight pruning with ADMM*. Moreover, we utilize distilled knowledge from software model to guide quantization with memristor hardware constraint. To the best of our knowledge, we are the first to combine extreme structured weight pruning and weight quantization in a unified and systematic memristor-based framework. Also, we are the first to discover the redundant weights and unused path in a structured pruned DNN model and design a sophisticate co-optimization framework to boost higher model compression rate as well as maintain high network accuracy. By incorporating memristor hardware constraints in our model, our frameworks are guaranteed feasible for a real memristor crossbar device. Our contributions are as follows:

- We adopt ADMM for efficiently optimizing the non-convex problem and utilized this method on structured weight pruning.
- We systematically investigate the weight quantization on a pruned model with memristor hardware constraints.
- We design a software-hardware co-optimization framework in which *Network Purification* and *Unused Path Removal* are first proposed.

We evaluate our proposed memristor framework on different networks. We conclude that structured pruning method with *memristor-based ADMM regularized optimization* achieves high compression rate and desirable high accuracy. Hardware experimental results shows our memristor framework is very energy efficient and saves great amount of hardware footprint.

978-1-7281-4123-7/20/$31.00
[†]These authors contributed equally.

2 Background

2.1 Model Compression techniques for Crossbar Architecture

Heuristic weight pruning methods [15] are widely used in neuromorphic computing designs to reduce the weight storage and computing delay. [16] implemented weight pruning techniques on a neuromorphic computing system using irregular pruning caused unbalanced workload, greater circuits overheads and extra memory requirement on indices. To overcome the limitations, [17] proposed group connection deletion, which structually prunes connections to reduce routing congestion between memristor crossbar arrays.

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 [4]. [18] presented a technique to reduce the overhead of Digital-to-Analog Converters (DACs)/Analog-to-Digital Converters (ADCs) in resistive random-access memory (ReRAM) neuromorphic computing systems. They first normalized the data, and then quantized intermediary data to 1-bit value. This can be directly used as the analog input for ReRAM crossbar and, hence, avoids the need of DACs.

2.2 Memristor Crossbar Model

Memristor [11] crossbar is an array structure consists of memristors, horizontal Word-lines and Vertical Bit-lines, as shown in Figure 1. Due to its outstanding performance on computing matrix-vector multiplications (MVM), memristor crossbars are widely used as dot-product accelerator in recent neuromorphic computing designs [19]. By programming the conductance state (which is also known as "memristance") of each memristor, the weight matrix $\mathbf{W}$ can be mapped onto the memristor crossbar. Given the input voltage vector $\mathbf{V}_i$, the MVM output current vector $\mathbf{I}_j$ can be obtained in time complexity of $O(1)$.

2.3 Challenges in Memristor Crossbars Implementation and Mitigation Techniques

Different from the software-based designs, hardware imperfection is one of the key issues that causes the hardware non-ideal behaviors and needs to be considered in memristor-based designs. They are:

Process Variation mainly comes from the line-edge roughness, oxide thickness fluctuations, and random dopant variations [20]. Inevitably, process variation plays an increasingly significant role as the process technology scales down to nanometer level. In a DNN hardware design, the non-ideal behaviors caused by process variations may lead to an accuracy degradation.

State Drift is the phenomenon that the memristance would change after several reading opertions [21]. It is known that memristor is a thin-film device constructed by a region highly doped with oxygen vacancies and an undoped region. By nature, applying an electric field across the memristor over a period of time, the oxygen vacancies would migrate to the direction along with the electric field, which leads to the (memristance) state drift. Consequently, an error will incur when the state of memristor drifts to another state level.

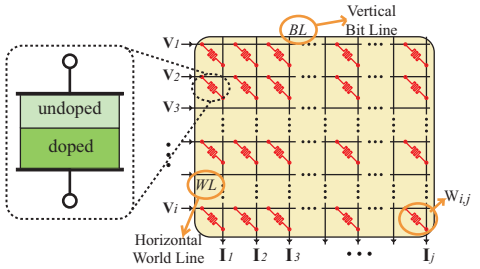

Figure 1: memristor and memristor crossbar

It has been proved that applying quantization on memristor-based designs can mitigate the undesired impacts caused by hardware imperfections [22].

3 A Memristor-Based Highly Compressed DNN Framework

The memristor crossbar structure has shown its potential for neuromorphic computing system compared to the CMOS technologies [16]. Due to great amount of weights and computations that involved in networks, an efficient and highly performed framework is needed to conquer the memory storage and energy consumption problems.

3.1 Problem Formulation

ADMM [23] is an advanced optimization technique which decompose an original problem into subproblems that can be solved separately and iteratively. By adopting *memristor-based ADMM regularized optimization*, the framework can guarantee the solution satisfying memristor hardware constraints while maintain high accuracy after pruning.

The *memristor-based ADMM regularized optimization* starts from a pre-trained full size DNN model without compression. Consider an N-layer DNNs, sets of weights of the i-th (CONV or FC) layer are denoted by $\mathbf{W}_i$. And the *loss function* associated with the DNN is denoted by $f\big(\{\mathbf{W}_i\}_{i=1}^N\big)$. The overall problem is defined by

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

Given the value of α_i, the memristor-based constraint set $\mathcal{P}_i = \{\mathbf{W}_i | \sum(\text{structured } \mathbf{W}_i \neq 0) \leq \alpha_i\}$ and $\mathcal{Q}_i$={the weights in the i-th layer are mapped to the quantization values}, where α_i is predefined hyper parameters. The general constraint can be extended in structured pruning such as filter pruning, channel pruning and column pruning, which facilitate high-parallelism implementation in hardware.

Similarly, for weight quantization, elements in $\mathcal{Q}_i$ are the solutions of $\mathbf{W}_i$. Assume set of $q_{i,1}, q_{i,2}, \cdots, q_{i,M_i}$ is the available memristor state value which is the elements in $\mathbf{W}_i$, where M_i denotes the number of available quantization level in layer i. Suppose $q_{i,j}$ indicates the j-th quantization level in layer i, which gives

$$q_{i,j} \in [-memr_{max}, -memr_{min}] \cup [memr_{min}, memr_{max}] \tag{2}$$

where $memr_{min}$, $memr_{max}$ are the minimum and maximum memristance value of a specified memristor device.

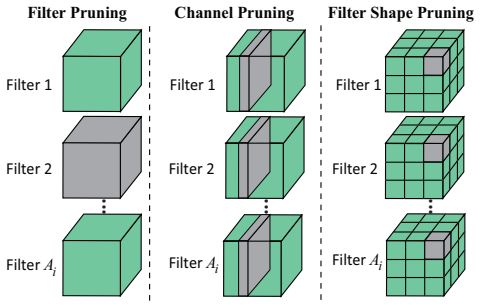

Figure 2: Illustration of filter-wise, channel-wise and shape-wise structured sparsities.

3.2 Memristor-based ADMM regularized optimization step

Corresponding to every memristor-based constraint set of $\mathcal{P}_\mathbf{i}$ and $\mathcal{Q}_\mathbf{i}$, a indicator functions is utilized to incorporate $\mathcal{P}_\mathbf{i}$ and $\mathcal{Q}_\mathbf{i}$ into objective functions, which are

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

for $i = 1, \ldots, N$. Substituting into (1) and we get

$$\begin{aligned} \underset{\{\mathbf{W}_i\}}{\text{minimize}} \quad & f\big(\{\mathbf{W}_i\}_{i=1}^N\big) + \sum_{i=1}^N g_i(\mathbf{Y}_i) + \sum_{i=1}^N h_i(\mathbf{Z}_i), \\ \text{subject to} \quad & \mathbf{W}_i = \mathbf{Y}_i = \mathbf{Z}_i,\ i = 1, \ldots, N, \end{aligned} \tag{3}$$

We incorporate auxiliary variables $\mathbf{Y}_i$ and $\mathbf{Z}_i$, dual variables $\mathbf{U}_i$ and $\mathbf{V}_i$, and the augmented Lagrangian formation $L_\rho\{\cdot\}$ of problem (3) is

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

The first term in problem (4) is the original DNN loss function, and the second and third term are differentiable and convex. As a result, subproblem (4) can be solved by stochastic gradient descent [24] similar to training the original DNN.

The standard ADMM algorithm [23] steps proceed by repeating, for $k = 0, 1, \ldots$, the following subproblems iterations:

$$\begin{aligned} \mathbf{W_i^{k+1}} := \underset{\{\mathbf{W_i}\}}{\text{minimize}} \ & L_\rho(\{\mathbf{W_i}\}, \{\mathbf{Y_i^k}\}, \{\mathbf{U_i^k}\}) \\ & + L_\rho(\{\mathbf{W_i}\}, \{\mathbf{Z_i^k}\}, \{\mathbf{V_i^k}\}) \end{aligned} \tag{5}$$

$$\begin{aligned} \mathbf{Y_i^{k+1}}, \mathbf{Z_i^{k+1}} := \underset{\{\mathbf{Y_i}, \mathbf{Z_i}\}}{\text{minimize}} \ & L_\rho(\{\mathbf{W_i^{k+1}}\}, \{\mathbf{Y_i}\}, \{\mathbf{U_i^k}\}) \\ & + L_\rho(\{\mathbf{W_i^{k+1}}\}, \{\mathbf{Z_i}\}, \{\mathbf{V_i^k}\}) \end{aligned} \tag{6}$$

$$\mathbf{U_i^{k+1}} := \mathbf{U_i^k} + \mathbf{W_i^{k+1}} - \mathbf{Y_i^{k+1}};\ \mathbf{V_i^{k+1}} := \mathbf{V_i^k} + \mathbf{W_i^{k+1}} - \mathbf{Z_i^{k+1}} \tag{7}$$

which (5) is the proximal step, (6) is projection step and (7) is dual variables update.

The optimal solution is the Euclidean projection of $\mathbf{W}_i^{k+1} + \mathbf{U}_i^k$ and $\mathbf{W}_i^{k+1} + \mathbf{V}_i^k$ onto $\mathcal{P}_i$ and $\mathcal{Q}_\mathbf{i}$. Namely, elements in solution that less than α_i will be set to zero. In the meantime, those kept elements are quantized to the closest valid memristor state value.

3.3 Memristor-Based Structured Weight Pruning

In order to accommodate high-parallelism implementation in hardware, we use structured pruning method [1] instead of the irregular pruning method [15] to reduce the size of the weight matrix while avoid extra memory storage requirement for indices. Figure 2 shows different types of structured sparsity which include filter-wise sparsity, channel-wise sparsity and shape-wise sparsity.

Figure 3 (a) shows the general matrix multiplication (GEMM) view of the DNN weight matrix and the different structured weight pruning methods. The structured pruning corresponds to removing rows (filters-wise) or columns (shape-wise) or the combination of them. We can see that after structured weight pruning, the remaining weight matrix is still regular and without extra indices.

Figure 3 (b) illustrate the memristor crossbar schematic size reduction from corresponding structured weight pruning and Figure 3 (c) shows physical view of the memristor crossbar blocks. A CONV layer has n filters, m channels which include total k columns, and is denoted as $\mathbf{W} \in \mathbb{R}^{n \times k}$. Due to the increasing reading/writing errors caused by expanding the memristor crossbar size, we limited our design by using multiple 128×64 [25] crossbars for all DNN layers. In Figure 3 (c), i, j denote columns and rows for each crossbar, X represent inputs and c is the column number which is also shown in Figure 3 (a). By easy calculation, one can derived that there's k/j different crossbars to store one filter's weights as a block unit. So there's total $p = n/j$ blocks to store $\mathbf{W} \in \mathbb{R}^{n \times k}$. 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.

4 Software-hardware Co-optimization

Due to the existence of the non-optimality of ADMM process and the accuracy degradation problem of quantizing sparse DNN, a software-hardware co-optimization framework is desired. In this section we propose: (i) network purification and unused path removal to efficiently remove redundant channels or filters, (ii) memristor model quantization by using distilled knowledge from helper models.

4.1 Network Purification and Unused Path Removal (P-RM)

Weight pruning with memristor-based ADMM regularized optimization can significantly reduce the number of weights while maintaining high accuracy. However, does the pruning process really remove all unnecessary weights?

From our analysis on the computing paradigm of DNN, we find that if a whole filter is pruned, then the generated feature maps by this filter will be "blank" (i.e., all zeros). If those "blank" features input to the next layer, then the corresponding channel weights in the next layer will become useless. By the same token, a pruned channel also causes the corresponding filter in previous layer a useless one. Figure 4 gives a clear illustration on the relationship between pruned filters/channels and correspond unused channels/filters.

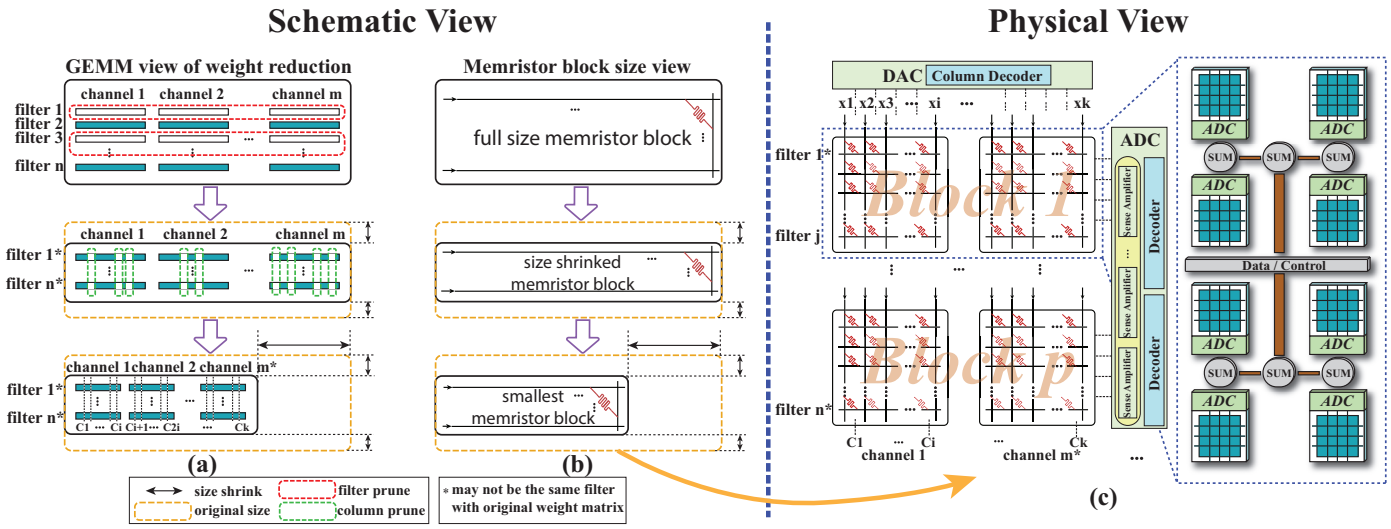

Figure 3: Structured weight pruning and reduction of hardware resources

To better optimize the unused path removal effect we discuss above, we derive an emptiness ratio parameter η to define what can be treated as an empty channel. Suppose Λ_i is the number of columns per channel in layer i, and j is channel index. We have

$$\eta_{i,j} = \Big[\sum_{k=1}^{\delta}(column_k != 0)\Big]/\delta \quad \delta \in \Lambda_i \tag{8}$$

If $\eta_{i,j}$ lower than a pre-defined threshold, we can assume that this channel is empty and thus prune every column in it even there are non-zero columns in it. However, if we remove every column in this circumstance, dramatic accuracy drop will occur and it will be hard to recover by retraining because some relatively "important" weights might be removed. So we design *Network Purification* algorithm dealing with the *non-optimality* problem of the ADMM process. We set-up a criterion parameter $\sigma_{i,j}$ to represent importance score of channel j, which is:

$$\sigma_{i,j} = \sum_{k=1}^{\delta} \|column_k\|_F^2 \quad \delta \in \Lambda_i \tag{9}$$

One can think of this process as if *collection evidence for whether channel j that contains one or several columns need to be removed*. A channel can only be treated as empty when both η and σ are under thresholds. *Network Purification* also works on purifying remaining filters and thus remove more unused path in the network. Algorithm 1 shows our generalized method of the P-RM method where $Th_1 \dots Th_4$ are hyper-parameter thresholds values.

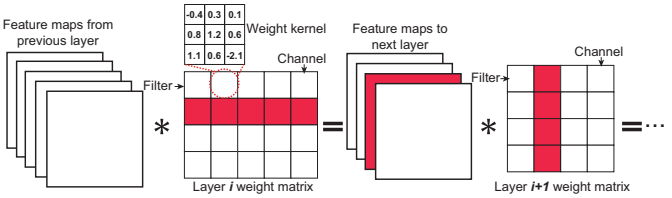

Figure 4: Unused data path caused by structured pruning

4.2 Memristor Weight Quantization

Traditionally, DNN in software is composed by 32-bit weights. But on a memristor device, the weights of a neural

Algorithm 1: Network purification & Unused path removal

Result: Redundant weights and unused paths removed
Load ADMM pruned model
δ = numbers of columns per channel
for $i \leftarrow 1$ *until last layer* **do**
  for $j \leftarrow 1$ *until last channel in layer*$_i$ **do**
    for each: $k \in \delta$ **and** $\|column_k\|_F^2 < Th_1$ **do**
      calculate: *equation* (8), (9);
    end
    if $\eta_{i,j} < Th_2$ **and** $\sigma_{i,j} < Th_3$ **then**
      prune($channel_{i,j}$)
      prune($filter_{i-1,j}$) **when** $i \neq 1$;
    end
  end
  for $m \leftarrow 1$ *until last filter in layer*$_i$ **do**
    if $filter_m$ **is** empty **or** $\|filter_m\|_F^2 < Th_4$ **then**
      prune($filter_{i,m}$)
      prune($channel_{i+1,m}$) **when** $i \neq$ last layer index;
    end
  end
end

network are represented by the memristance of the memristor (i.e. the memristance range constraint $\mathcal{Q}_i$ in ADMM process). Due to the limited memristance range of the memristor devices, the weight values exceeding memristance range cannot be represented precisely. Meanwhile, the write-on value and the exact value mismatch when mapping weights on memristor crossbar will also cause the reading mismatch if the amount of the value shift exceeds state level range.

In order to mitigate the memristance range limitation and the mapping mismatch, larger range between state level $(q_{i,1}, q_{i,2}, \cdots, q_{i,M_i})$ is needed which means fewer bits are representing weights. To better maintain accuracy, we use a pretrained high-accuracy teacher model to provide distillation loss to add on our memristor model (referred as student model) loss to provide better training performance. The loss of the student model is defined as

$$l_{student} = (1-\gamma)\mathbb{L}(p_s, p_r) + \gamma\mathcal{T}^2\mathbb{L}(p_s, p_t) \tag{10}$$

The $\mathbb{L}$ in first term in (10) is the memristor model (student) loss, and in second term is distillation loss between student and teacher. p_s and p_t are outputs of student and

Table 1: Structured weight pruning results on multi-layer network on MNIST, CIFAR-10 and ImageNet datasets. (P-RM: Network Purification and Unused Path Removal). Accuracies in ImageNet results are reported in ***Top-5*** accuracy.

Method	Original model Accuracy	Compression Rate Without P-RM	Accuracy Without P-RM	Prune Ratio With P-RM	Accuracy With P-RM	Weight Quantization Accuracy (8-bit)
MNIST						
Group Scissor [17]	99.15%	4.16×	99.14%	N/A	N/A	N/A
our LeNet-5	99.17%	23.18×	99.20%	**39.23×**	**99.20%**	**99.16%**
		34.46×	99.06%	***87.93×**	**99.06%**	**99.04%**
		45.54×	98.48%	**231.82×**	**98.48%**	**98.05%**
*numbers of parameter reduced: **25.2K**						
CIFAR-10						
Group Scissor [17]	82.01%	2.35×	82.09%	N/A	N/A	N/A
our ConvNet	84.41%	2.35×	84.55%	N/A	N/A	**84.33%**
		*2.93×	84.53%	N/A	N/A	**83.93%**
		5.88×	83.58%	N/A	N/A	**83.01%**
our VGG-16	93.70%	20.16×	93.36%	**44.67×**	**93.36%**	**93.04%**
				***50.02×**	**92.73%**	**92.46%**
our ResNet-18	94.14%	5.83×	93.79%	**52.07×**	**93.79%**	**93.71%**
		15.14×	93.20%	***59.84×**	**93.22%**	**93.27%**
*numbers of parameter reduced on *ConvNet*: **102.30K**, *VGG-16*: **14.42M**, *ResNet-18*: **10.97M**						
ImageNet ILSVRC-2012						
SSL [1] AlexNet	80.40%	1.40×	80.40%	N/A	N/A	N/A
our AlexNet	82.40%	4.69×	81.76%	**5.13×**	**81.76%**	**80.45%**
our ResNet-18	89.07%	3.02×	88.41%	**3.33×**	**88.36%**	**88.47%**
our ResNet-50	92.86%	2.00×	92.26%	**2.70×**	**92.27%**	**92.20%**
numbers of parameter reduced on *AlexNet*: **1.66M**, *ResNet-18*: **7.81M**, *ResNet-50*: **14.77M**						

Algorithm 2: Distillation Quantization

Result: distillation quantization with memristor hardware constraints

student ← model pruned and ready to apply quantization;

teacher ← model with a deeper structure and higher accuracy;

for *step* ← 1 *until* $l_{student}$ *converge* **do**

  $student_q = apply\_quantization(w_s, \mathcal{Q})$;

  calculate $\mathcal{T}^2\mathbb{L}(p_s, p_t)$ of $student_q$ & *teacher*;

  back propagate on *student* ← $\frac{\partial(\mathcal{T}^2\mathbb{L}(p_s,p_t))}{\partial(student_q)}$;

end

teacher and p_r is the ground-truth label. γ is a balancing parameter, and $\mathcal{T}$ is the temperature parameter.

5 Experimental Results

In this section, we show the experimental results of our proposed memristor-based DNN framework in which structured weight pruning and quantization with memristor-based ADMM regularized optimization are included. Our software-hardware co-optimization framework (i.e., *Network Purification* and *Unused Path Removal* (P-RM)) are also thoroughly compared. We test MNIST dataset on LeNet-5 and CIFAR-10 dataset using ConvNet (4 CONV layers and 1 FC layer), VGG-16 and ResNet-18, and we also show our ImageNet results on AlexNet, ResNet-18 and ResNet-50. The accuracy of pruned and quantized model results are tested based on our software models that incorporated with memristor hardware constraints. Models are trained on an eight NVIDIA GTX-2080Ti GPUs server using PyTorch API. Our memristor model on MATLAB and the NVSim [26] is used to calculate power consumption and area cost of the memristors and memristor crossbars. The 1R crossbar structure is used in our design. And we choose the memristor device that has $R_{on} = 1M\Omega$ and $R_{off} = 10M\Omega$. The memristor precision is 4-bit, which indicates that 16 state-levels can be represented by a single memristor device, and two memristors are combined to represent 8-bit weight in our framework. For the peripheral circuits, the power and area is calculated based on 45nm technology. And H-tree distribution networks are used to access all the memristor crossbars.

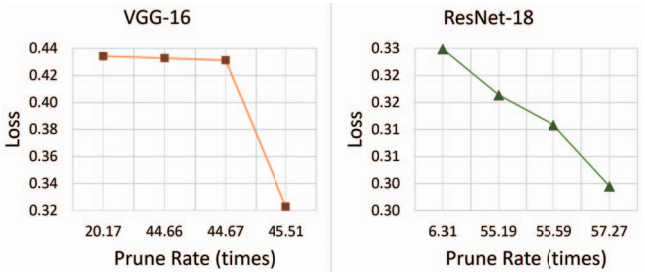

Figure 5: Effect of removing redundant weights and unused paths. (dataset: CIFAR-10; Accuracy: VGG-16-93.36%, ResNet-18-93.79%)

As shown in Table 1, we show groups of different prune ratios and 8-bits quantization with accuracies on each network structure. Figure 5 proves our previous arguments that ADMM's non-optimality exists in a structured pruned model. P-RM can further optimize the loss function. Please note all of the results are based on non-retraining process. Below are some results highlights on different dataset with different network structures.

MNIST. With LeNet-5 network, comparing to original accuracy (99.17%), our proposed P-RM framework achieve 231.82× compression with minor accuracy loss while other state-of-art compression rates are lossless. And no accuracy losses are observed after quantization on 40× and 88× models and only 0.4% accuracy drop on 231.82× model. On the other hand, Group Scissor [17] only has 4.16× compression rate.

CIFAR-10. Convnet structure are relative shallow so ADMM reaches a relative optimal local minimum, so post-processing is not necessary. But we still outperform Group Scissor [17] in accuracy (84.55% to 82.09%) when compression rate is same (2.35×). For larger networks, when a minor accuracy loss is allowed, our proposed P-RM method improves the prune ratio to 50.02× and 59.84× on VGG-16 and ResNet-18 respectively, and no obvious accuracy loss after quantization on pruned models.

ImageNet. AlexNet model outperform SSL [1] both

Table 2: Area/power comparison between models with and without P-RM on ResNet-18 and VGG-16 on CIFAR-10

	total area (mm^2)	total power (W)	accuracy
ResNet18 w/o P-RM 5.38x	0.235	3.359	93.79%
ResNet18 with P-RM 52.07x	0.042	0.585	93.79%
ResNet18 w/o P-RM 15.14x	0.117	1.622	93.20%
ResNet18 with P-RM 59.84x	0.041	0.556	93.22%
VGG16 93.36 w/o P-RM 20.16x	0.113	1.611	93.36%
VGG16 with P-RM 44.67x	0.056	0.824	93.36%
VGG16 with P-RM 50.02x	0.053	0.754	92.73%

in compression rate (4.69× to 1.40×) and network accuracy (81.76% to 80.40%), with or without P-RM. Our ResNet-18 and ResNet-50 models also achieve unprecedented 3.33× with 88.36% accuracy and 2.70× with 92.27% respectively. No accuracy losses are observed after quantization on pruned ResNet-18/50 models and around 1% accuracy loss on 5.13× compressed AlexNet model.

Table 2 shows our highlighted memristor crossbar power and area comparisons of ResNet-18 and VGG-16 models. By using our proposed P-RM method, the area and power of the 5.83× (15.14×) ResNet-18 model is reduced from $0.235mm^2$ ($0.117mm^2$) and $3.359W$ ($1.622W$) to $0.042mm^2$ ($0.041mm^2$) and $0.585W$ ($0.556W$), without any accuracy loss. For VGG-16 20.16× model, after using our P-RM method, the area and power is reduced from $0.113mm^2$ and $1.611W$ to $0.056mm^2$ ($0.053mm^2$) and $0.824W$ ($0.754W$), where the compression rate is achieved 44.67× (50.02×) with 0% (0.63%) accuracy degradation.

6 Conclusion

In this paper, we designed an unified memristor-based DNN framework which is tiny in overall hardware footprint and accurate in test performance. We incorporate ADMM in weight structured pruning and quantization to reduce model size in order to fit our designed tiny framework. We find the non-optimality of the ADMM solution and design *Network Purification* and *Unused Path Removal* in our software-hardware co-optimization framework, which achieve better results comparing to Gourp Scissor [17] and SSL [1]. On AlexNet, VGG-16 and ResNet-18/50, after structured weight pruning and 8-bit quantization, model size, power and area are significant reduced with negligible accuracy loss.

Acknowledgment

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

References

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

[2] X. Ma, G. Yuan, S. Lin, Z. Li, H. Sun, and Y. Wang, "Resnet can be pruned 60x: Introducing network purification and unused path removal (p-rm) after weight pruning," *arXiv preprint arXiv:1905.00136*, 2019.

[3] T. Zhang, K. Zhang, S. Ye, J. Li, J. Tang, W. Wen, X. Lin, M. Fardad, and Y. Wang, "Adam-admm: A unified, systematic framework of structured weight pruning for dnns," *arXiv preprint arXiv:1807.11091*, 2018.

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

[5] S. Lin, X. Ma, S. Ye, G. Yuan, K. Ma, and Y. Wang, "Toward extremely low bit and lossless accuracy in dnns with progressive admm," *arXiv preprint arXiv:1905.00789*, 2019.

[6] 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.

[7] H. Li, N. Liu, X. Ma, S. Lin, S. Ye, T. Zhang, X. Lin, W. Xu, and Y. Wang, "Admm-based weight pruning for real-time deep learning acceleration on mobile devices," in *Proceedings of the 2019 on Great Lakes Symposium on VLSI*, 2019.

[8] C. Ding, A. Ren, G. Yuan, X. Ma, J. Li, N. Liu, B. Yuan, and Y. Wang, "Structured weight matrices-based hardware accelerators in deep neural networks: Fpgas and asics," in *GLSVLSI*, 2018.

[9] C. Ding, S. Liao, Y. Wang, Z. Li, N. Liu, Y. Zhuo, C. Wang, X. Qian, Y. Bai, G. Yuan, X. Ma, etc. , "Circnn: accelerating and compressing deep neural networks using block-circulant weight matrices," *MICRO*. ACM.

[10] Y. Wang, C. Ding, Z. Li, G. Yuan S. Liao, X. Ma, B. Yuan, X. Qian, J. Tang, Q. Qiu, X. Lin, "Towards ultra-high performance and energy efficiency of deep learning systems: an algorithm-hardware co-optimization framework," *Thirty-Second AAAI Conference on Artificial Intelligence (AAAI)*. 2018.

[11] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, "The missing memristor found," *nature*, vol. 453, no. 7191, p. 80, 2008.

[12] X. Ma, Y. Zhang, G. Yuan, A. Ren, Z. Li, J. Han, J. Hu, and Y. Wang, "An area and energy efficient design of domain-wall memory-based deep convolutional neural networks using stochastic computing," in *ISQED*. IEEE, 2018.

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

[14] G. Yuan, C. Ding, R. Cai, X. Ma, Z. Zhao, A. Ren, B. Yuan, and Y. Wang, "Memristor crossbar-based ultra-efficient next-generation baseband processors," in *MWSCAS*, 2017.

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

[16] A. Ankit, A. Sengupta, and K. Roy, "Trannsformer: Neural network transformation for memristive crossbar based neuromorphic system design," in *Proceedings of ICCD*, 2017.

[17] 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.

[18] L. Xia, T. Tang, W. Huangfu, M. Cheng, X. Yin, B. Li, Y. Wang, and H. Yang, "Switched by input: power efficient structure for rram-based convolutional neural network," in *DAC*. ACM, 2016, p. 125.

[19] A. Shafiee, A. Nag, N. Muralimanohar, and et.al, "ISAAC: A Convolutional Neural Network Accelerator with In-Situ Analog Arithmetic in Crossbars," in *ISCA 2016*.

[20] 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," 2001.

[21] 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.

[22] 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.

[23] 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.

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

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

[26] X. Dong, C. Xu, S. Member, Y. Xie, S. Member, and N. P. Jouppi, "Nvsim: A circuit-level performance, energy, and area model for emerging nonvolatile memory," *IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS*.