IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

ADMM-CP: ADMM-based Clustered Pruning on Crossbar Arrays for Memristor-based Inference Accelerator

Journal:	*Transactions on Computer-Aided Design of Integrated Circuits and Systems*
Manuscript ID	TCAD-2023-0282
Manuscript Type:	New Research Article
Date Submitted by the Author:	11-May-2023
Complete List of Authors:	feng, renhai; Tianjin University Li, Jiahang; Tianjin University
Keywords:	architecture, compression, low-power design, modeling, scheduling
Manuscript Subject:	Emerging Technologies and Applications

ADMM-CP: ADMM-based Clustered Pruning on Crossbar Arrays for Memristor-based Inference Accelerator

Renhai Feng, *Member, IEEE,* Jiahang Li

***Abstract*—Memristor crossbar array has emerged as a promising technology for deep neural networks acceleration. However, such memristive neuromorphic system will have a unacceptable power consumption and hardware area cost when we construct it from original neural network directly. This paper proposes a novel hardware-aware pruning algorithm called ADMM-CP to improve memristor-based inference accelerators' power-area efficiency, which is based on Alternating Direction Method of Multipliers (ADMM) and spectral clustering. ADMM-CP can completely translate the obtained network sparsity into the reduction of hardware area and the memristors needed while maintaining stable and effective pruning, which is difficult for the previously reported pruning algorithms developed for fully connected (FC) layers of memristor-based neural networks. Moreover, ADMM-CP is relatively insensitive to parameter initialization, which simplifies parameter selection. ADMM-CP is evaluated by transforming multilayer perceptron (MLP) and convolutional neural network (CNN topologies on different datasets (MNIST, SVHN, FASHION-MNIST, CIFAR-10 and CIFAR-100). Simulation results show that the proposed pruning algorithm reduces the area (energy) consumption by 55%–$64\%(58\%$–$65\%)$ of MLP networks and by 65.34%–$95.4\%(65.77\%$–$92.2\%)$ of fully-connected layers in CNN networks with respect to the original network implementations. The novelty of ADMM-CP is also demonstrated by comparison with recent same purposed framework TraNNsformer.**

***Index Terms*—Memristor crossbar array, pruning, ADMM.**

I. Introduction

MEMRISTOR-based neural networks is a promising technology for deep neural network (DNN) acceleration, especially using memristor crossbar array (MCA) topology for in-memory analog computation. With inherent storage-computing integration and massive parallel topology of MCA, significant speed-energy efficiency improvement can be achieved when implementing DNN inference with memristive technology, i.e., memristor-based inference accelerator [1], [2]. However, with the increasing scale and complexity of DNN, the hardware area and energy cost may be unacceptable [3] when we construct memristor-based neural networks based on the original DNN model directly. Thus, it is indispensable to perform DNN model compression before memristive technology realization. The main goal of model compression is to reduce model storage and speed up computation with negligible accuracy loss, for power-area efficient memristor-based inference accelerators.

Network pruning has been proven to be effective for network sparsification, which is vital to memristive implementation of neural networks. Han et al. [4] pioneered weight pruning, achieving $9\times$ weight reduction in AlexNet on ImageNet as an iterative heuristic method. Since then, researchers have popularized and improved the weight pruning method [5], [6] to further increase the pruning ratio and efficiency. However, most of these algorithms adopt a greedy and empirical approach which inevitably have their limitations. [7] proposed a systematic joint pruning quantization framework based on Alternating Direction Method of Multipliers (ADMM), which can make full use of model redundancy to achieve the maximum model compression ratio compared with the previous empirical methods. Afterward, thanks to the flexibility of ADMM framework, Ma et al. [8] introduced two auxiliary variables and attempted to perform pruning and quantization work simultaneously. However, non-structured pruning in [7], [8] caused irregular sparsity of pruned weight matrices, which is not suitable for crossbar array architecture and could not achieve noticeable improvement in power-area efficiency of memristor-based inference accelerators.

On the other hand, most existing structured pruning algorithms focused on convolution layer. However, the weights in fully-connected (FC) layers account for the vast majority of the network [6]. For power-area efficiency, developing a suitable pruning algorithm compatible with crossbar architectures for FC layers is necessary. [9] mainly focused on network clustering after pruning training, i.e., offline clustering. However, the network weights after pruning may not be suitable for clustering, which would lead to a poor clustering effect, leaving a large number of unclustered weights. As a result, the quality of clustering (the proportion of unpruned weights in the cluster) may be low. Recognizing this, Ankit et al. [3] proposed to prune FC layers with the particle size of crossbar array instead of synapse and dynamically prune the network from the perspective of network performance (accuracy) and clustering. In this way, the benefits of pruning and network sparsity can be retained at the hardware level. However, in [3], there were also inevitably unclustered synapses and even the quality of the formed cluster may be poor, especially for large crossbar size. For each individual network, [3] had to reevaluate most of the initialization for the best pruning effect, which results in huge complexity.

To overcome the challenges mentioned above, this paper performs clustered pruning (spectral clustering) on FC layers of networks in the framework of ADMM and proposes a novel hardware-aware pruning algorithm called ADMM-CP for power-area efficiency improvement. ADMM-CP produces a pruned network with structured sparsity based on the results

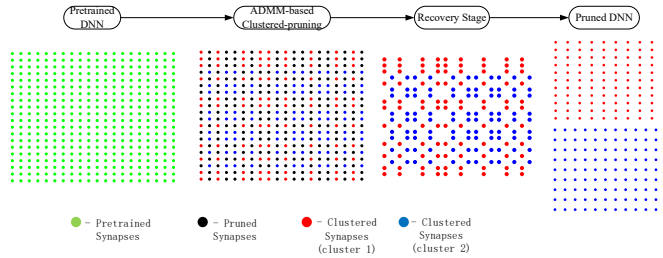

Fig. 1. Logical stages of ADMM-based clustered-pruning with examples of how each step affects the connectivity matrix.

of spectral clustering. It can completely translate the obtained network sparsity into the reduction of hardware area or the number of memristors needed while maintaining stable and good pruning effect with negligible accuracy loss. Moreover, our algorithm is relatively insensitive to parameter initialization, which simplifies parameter selection.

II. Hardware-aware Pruning algorithm based on ADMM and Spectral Clustering

In this section, a detailed discussion of a hardware-aware pruning algorithm based on ADMM and spectral clustering is given.

Fig. 1 shows the pruning process of the proposed ADMM-based clustered-pruning algorithm (ADMM-CP), which covers three stages: 1) pre-training; 2) ADMM-based clustered-pruning; 3) recovery stage. Subsection A is a detailed description of the three stages. Framework flexibility is discussed in subsection B.

A. ADMM-based clustered-pruning algorithm

Consider a DNN with N FC layers and L layers in total. The weight of the l-th layer is denoted by $\mathbf{W}_l$. The loss function associated with the DNN is $f(\{\mathbf{W}_l\}_{l=1}^L)$. The problem of clustered pruning can be formulated as an optimization problem:

$$\begin{aligned} &\min_{\{\mathbf{W}_l\}} f(\{\mathbf{W}_l\}_{l=1}^L) \\ &\text{s.}t. \mathbf{W}_l \in \boldsymbol{Q}_l \quad l = 1, ...N \end{aligned} \tag{1}$$

where the constraint set $\boldsymbol{Q}_l$ = {the nonzero weights can be clustered into several clusters}. Since (1) is non-convex with combinatorial constraints, it is infeasible to solve (1) using gradient descent methods directly. However, in the ADMM framework, the combinatorial constraints can be get rid of by defining indicator functions

$$g_l(\mathbf{W}_l) = \begin{cases} 0 & \mathbf{W}_l \in \boldsymbol{Q}_l \\ \infty & \mathbf{W}_l \notin \boldsymbol{Q}_l \end{cases} \tag{2}$$

for $l = 1, ..., N$. Incorporating auxiliary variables $\mathbf{H}_l$, problem (1) can be rewritten as

$$\begin{aligned} &\min_{\{\mathbf{W}_l\},\{\mathbf{H}_l\}} f(\{\mathbf{W}_l\}_{l=1}^L) + \sum_{l=1}^N g_l(\mathbf{H}_l)) \\ &\text{s.}t. \mathbf{W}_l = \mathbf{H}_l \quad l = 1, ..., N \end{aligned} \tag{3}$$

The augmented Lagrangian of (3) is

$$\begin{aligned} L(\{\mathbf{W}_l\}, \{\mathbf{H}_l\}) = f(\{\mathbf{W}_l\}_{l=1}^L) + \sum_{l=1}^N g_l(\mathbf{H}_l) + \\ \sum_{l=1}^N tr(\boldsymbol{\lambda}_l^T(\mathbf{W}_l - \mathbf{H}_l)) + \sum_{l=1}^N \frac{\rho_l}{2} \|\mathbf{W}_l - \mathbf{H}_l\|_F^2 \end{aligned} \tag{4}$$

where $tr(\cdot)$ calculates the trace of matrix, $\boldsymbol{\lambda}_l$ is Lagrange Multipliers and ρ_l is the penalty factor for the l_{th} FC layer. Let $\mathbf{U}_l = \frac{1}{\rho_l}\lambda_l$, (4) can be written as

$$\begin{aligned} L = f(\{\mathbf{W}_l\}_{l=1}^L) + \sum_{l=1}^N g_l(\mathbf{H}_l) + \\ \sum_{l=1}^N \frac{\rho_l}{2} \|\mathbf{W}_l - \mathbf{H}_l + \mathbf{U}_l\|_F^2 - \sum_{l=1}^N \frac{\rho_l}{2} \|\mathbf{U}_l\|_F^2 \end{aligned} \tag{5}$$

Via application of the augmented Lagrangian, (3) can be decomposed into two sub-problems using ADMM, which can be solved iteratively until convergence. The first sub-problem is

$$\min_{\{\mathbf{W}_l\}} f(\{\mathbf{W}_l\}_{l=1}^L) + \sum_{l=1}^N \frac{\rho_l}{2} \left\|\mathbf{W}_l - \mathbf{H}_l^k + \mathbf{U}_l^k\right\|_F^2 \tag{6}$$

In the objective function of (6), the first term a differentiable loss function of DNN. The second quadratic term is differentiable and convex, which can be regarded as regularizers. Therefore, (6) can be solved by stochastic gradient descent (SGD) as in original DNN training.

The second sub-problem is

$$\min_{\{\mathbf{H}_l\}} \sum_{l=1}^N g_l(\mathbf{H}_l) + \sum_{l=1}^N \frac{\rho_l}{2} \left\|\mathbf{W}_l^{k+1} - \mathbf{H}_l + \mathbf{U}_l^k\right\|_F^2 \tag{7}$$

Since $g_l(\cdot)$ is the indicator function of $\boldsymbol{Q}_l$, the analytical solution of (7) is

$$\mathbf{H}_l^{k+1} = \Gamma_{\boldsymbol{Q}_l}(\mathbf{W}_l^{k+1} + \mathbf{U}_l^k) \tag{8}$$

where $\Gamma_{\boldsymbol{Q}_l}(\cdot)$ is the Euclidean projection onto $\boldsymbol{Q}_l$. In the context of clustered pruning, the algorithm to obtain $\mathbf{H}_l^{k+1}$ is shown is Algorithm 1. The similarity matrix for spectral clustering is constructed with $\mathbf{Z}^k$ ($\mathbf{Z}^k = \mathbf{W}_l^{k+1} + \mathbf{U}_l^k$), where $\mathbf{Z}^k(i,j)$ represents the similarity between the i_{th} input node and the j_{th} output node. The graph nodes consist of the input nodes and output nodes of $\mathbf{Z}^k$.

In each iteration, a dual variable update step follows the two sub-problems

$$\mathbf{U}_l^{k+1} = \mathbf{U}_l^k + \mathbf{W}_l^{k+1} - \mathbf{H}_l^{k+1} \tag{9}$$

By iteratively solving sub-problems (6), (7) and updating the dual variable $\mathbf{U}_l$, we can finally obtain the clustered-pruning results. Fig. 2 illustrates the whole process of ADMM-CP. After ADMM-CP converges, the optimal clustered-pruned DNN topology can be obtained. According to the final clustering result, the unclustered weights in $\mathbf{W}_l$ will be pruned permanently while the remaining weights can form K_l dense clusters with rerouting between layers. This pruning operation will inevitably lead to accuracy loss. Hence a recovery stage is

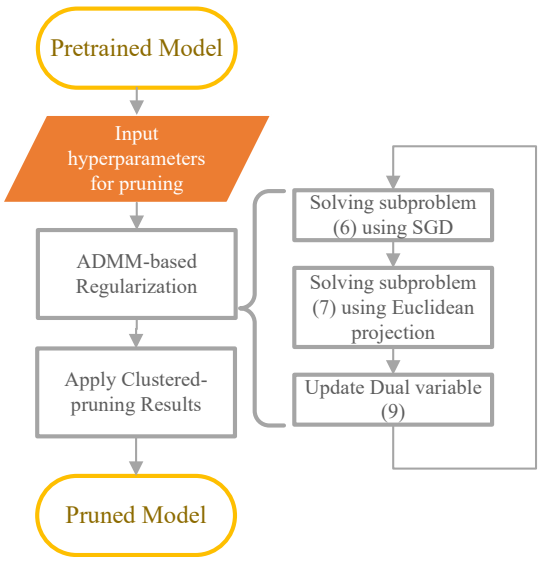

Fig. 2. Illustration for process of ADMM-CP.

vital to restore the DNN performance, which is similar to original DNN training. In this way, we significantly reduce weight redundancy while preserving the regularity of weight matrix, which is indispensable to improve the energy-area efficiency of memristor-based neural network inference accelerator.

Recall that FC layers account for most of the weights in DNN model. Since our primary goal is to reduce the weights (synapses) in DNN model for power-area efficiency, ADMM-CP is mainly intended to optimize or compress the topology of FC layers rather than CNN layers. Moreover, it is also possible for ADMM-CP to be combined with structured-pruning algorithms developed for convolutional layers [10]. Other compression algorithms, such as quantization technology [11] and Bayesian Weight Optimization [12], are also compatible with ADMM-CP to further compress the neural network model.

Algorithm 1 Algorithm to Solve (7)

Input: $\mathbf{W}_l^{k+1}$, $\mathbf{U}_l^k$, the number of clusters to construct K_l.
Output: the analytical solution of (7) $\mathbf{H}_l^{k+1}$, the spectral clustering result.

1: Construct the similarity matrix with $\mathbf{Z}^k(\mathbf{Z}^k = \mathbf{W}_l^{k+1} + \mathbf{U}_l^k)$;
2: Compute the degree matrix $\mathbf{D}$;
3: Compute the normalized Laplacian matrix $\mathbf{L}$;
4: Perform eigenvalue decomposition on $\mathbf{L}$;
5: Extract the eigenvectors corresponding to the K_l smallest eigenvalues to form $\widetilde{\mathbf{V}}$;
6: cluster the rows of $\widetilde{\mathbf{V}}$ with bi-Kmeans algorithm (each row of $\widetilde{\mathbf{V}}$ corresponds to a input/output node of $\mathbf{Z}^k$);
7: According to the clustering result, the unclustered weights $\mathbf{Z}^k(i,j)$ (i.e. the i_{th} input node and the j_{th} output node belong to different clusters) will be set to zero; The modified $\mathbf{Z}^k$ is the solution wanted $\mathbf{H}_l^{k+1}$;
8: Output $\mathbf{H}_l^{k+1}$ and the clustering result.

TABLE I
MLP BENCHMARKS

Application	Dataset	Layers	Neurons	Weights
Digit Recognition	MNIST	3	2410	2392800
House Recognition	SVHN	4	3610	4120800
Pattern Recognition	FASHION-MNIST	4	6010	10533600

B. Framework Flexibility

As shown in Fig. 2, hyperparameters are input to guide the training process. To obtain optimal DNN model for best power-area efficiency, several control knobs are exposed for adjustment, namely, cluster number K_l for each FC layer, penalty factor ρ_l for each FC layer, learning rate and recovery effort (epochs for recovery).

Among these control knobs, K_l and ρ_l have a relatively greater impact on training results. K_l determines the proportion of pruning for each layer to a large extent. While larger K_l leads to a higher pruning ratio, the accuracy may suffer from over pruned layers especially layers with high pruning sensitivity [13]. On the other hand, ρ_l influences the strength of ADMM regularization. In case of too small ρ_l, the algorithm may converge slow. In case of too large ρ_l, the algorithm will be overly inclined to clustered pruning, resulting in a dramatic loss in accuracy, which can not completely be restored in the recovery stage.

Steps to determine K_l and ρ_l are described as follows. Firstly, fix $K_l = 2$ and adjust ρ_l to ensure fast convergence with minimal accuracy loss. Secondly, increase K_l for each layer gradually until accuracy loss reaches the set threshold. According to the weight quantity of each layer, K_l of different layers have different priorities to be increased. The more weight parameters, the higher the priority. This is aimed at the highest pruning ratio while maintaining network accuracy. At first, increase K_l for the layer with the highest priority. If the final precision loss is lower than the threshold, keep the change of K_l; otherwise, cancel the change of K_l and mark this layer. The marked layer will not attempt to increase K_l. Then do the same for the next highest priority layer, and so on until you reach the lowest priority layer. Finally, loop through the process until all layers are marked. In this way, the optimal K_l for each layer is obtained.

III. Experimental Methodology

The proposed ADMM-CP algorithm was implemented using deep learning framework (Tensorflow). ADMM-CP is developed for topology optimization of FC layers. Therefore, the algorithmic benefits of ADMM-CP are studied on multi-layer perceptron (MLP) and convolutional neural network (CNN) topology, in which FC layers take over the vast majority of weights in network. ADMM-CP is applied on all the layers for MLPs. For CNNs, ADMM-CP is only applied on FC layers, leaving convolutional layers intact. To evaluate the performance of ADMM-CP, we apply ADMM-CP on MLPs ranging in the number of layers and layer sizes and CNNs with different complexity, as shown in TABLE I and TABLE II. For MLPs, a range of applications, i.e., digit recognition (MNIST dataset [14]), house number recognition (SVHN

TABLE II
CNN BENCHMARKS

Dataset	Architecture	Layers		Neurons	Weights
MNIST	Modified LeNet-5	total	5	13418	2700300
		FC	3	2410	2680800
CIFAR-10	AlexNet	total	8	152970	9638474
		FC	3	4106	6316042
CIFAR-100	VGG16	total	16	280676	20175012
		FC	3	4196	5451876

dataset [15]), and pattern recognition (FASHION-MNIST) are used for evaluation. For each application, the MLP architecture is adaptive to dataset complexity. To evaluate the optimization impact of ADMM-CP on CNNs, we choose LeNet-5 [14], AlexNet [16] and VGG16 on MNIST, CIFAR-10, and CIFAR-100 datasets respectively for experiment.

The system-level benefits of ADMM-CP are studied using the memristive crossbar architecture proposed in [17]. Memristor with Ag-Si memristive technology [18] is used for experiment. Our experiment is in light of [3].

IV. RESULTS

In this section, the various experimental results is shown to demonstrate the benefits of ADMM-CP at both algorithm level and system level for memristor-based inference accelerators. Section IV-A explains the algorithmic benefits of ADMM-CP. Evaluation results of ADMM-CP on MLP topology and CNN topology are shown in section IV-B and section IV-C respectively. The normalized values are used to report the energy and area benefits of ADMM-CP for more clear evaluation and comparison.

A. Algorithmic Superiority of ADMM-CP

TraNNsformer [3] improves the MCA utilization and reduces the fraction of unclustered synapses remaining after the training process in comparison to offline clustering [9]. However, there are inevitably unclustered synapses that require additional MCA to map. Moreover, the clusters generated by TraNNsformer [3] may contain a large number of unbalanced clusters (large aspect ratio), which will undermine the utilization of MCAs when they are mapped to small MCAs with uniform size. Thus, more MCAs are required to complete the mapping, which increases the circuit area and energy cost. Additionally, the parameter selection of TraNNsformer [3] may be difficult with complex DNN structure. Such restriction does exist in our proposed algorithm. No unclustered synapses remain after the training process in the proposed ADMM-CP framework. This means that the sparsity obtained from ADMM-CP can be completely translated into energy and area benefits for MCA architecture. Moreover, the cost caused by rerouting between layers of our proposed algorithm is relatively less than TraNNsformer [3]. In addition, the training effect of ADMM-CP is mainly influenced by two hyperparameters for each FC layer, i.e., K_l and ρ_l. With the same K_l and ρ_l, the training effect of ADMM-CP is relatively stable. In other words, ADMM-CP is insensitive to other control knobs unless these parameters are increased or decreased by orders of magnitude. This simplifies the parameter selection of ADMM-CP.

B. ADMM-CP on MLP Topology

In this subsection, the proposed ADMM-CP framework is evaluated for MLP topology on three datasets, as shown in TABLE I. The validation accuracy over training epochs on MNIST, SVHN, and FASHION-MNIST datasets under original training (referenced as "Original"), TraNNsformer, and ADMM-CP is shown in Fig. 4 (a), (b), and (c) of Appendix. A. For fair comparison between Original, TraNNsformer [3] and ADMM-CP, all DNNs are trained to close accuracy within equal training epochs and all the algorithms have converged. The pre-training, pruning, and recovery stages of ADMM-CP are merged in Fig. 3. At the beginning of training, ADMM-CP and Original are coincident, this is because ADMM-CP needs to go through a pre-training stage first. In the following clustered pruning stage, the accuracy rate curve of ADMM-CP drops sharply, and then gradually recovers to the same accuracy level as Original after the recovery stage. It can be seen that TraNNsformer and ADMM-CP require more training effort for convergence than original training due to the effort for pruning. However, we just need to increase the recovery epochs of ADMM-CP to achieve the accuracy levels of Original with tolerable loss or even slight improvement.

The pruning performance of Original, TraNNsformer, and ADMM-CP on MLP topology for each layer is shown in TABLE III, which shows the pruning ratio and the number of MCAs required for network mapping after pruning by the three methods of each FC layer. The higher the pruning ratio, the lower the number of memristors used, and the lower the power consumption from MCA. The number of MCAs required for network mapping will largely determine the circuit area and power consumption of peripheral circuits. The fewer the number of MCAs required, the smaller the circuit area and the lower power consumption of peripheral circuits.

With compared to Original and TraNNsformer, the pruning proportion of ADMM-CP is greater on the three datasets and the number of MCAs required for ADMM-CP is also less than Original and TraNNsformer on the whole. Overall, the pruning effect of ADMM-CP on MLP topology is better. The normalized area and energy costs of Original, TraNNsformer [3] and ADMM-CP on MNIST, SVHN , and FASHION-MNIST datasets are shown in Fig. 4 and Fig. 5 of Appendix. A respectively. The circuit area of memristive inference accelerator is mainly affected by the number of MCA required for network mapping after pruning, because the number of MCA determines the cost of supporting peripheral circuits, and the area of peripheral circuits is generally much larger than the area of MCA. The main power consumption of the memristive inference accelerator comes from MCAs and peripheral circuits. The higher the pruning ratio, the less the number of memristors to be used, the lower the power consumption from MCA, the less the number of MCA required for network mapping after pruning, the smaller the circuit area, and the lower the power consumption from peripheral circuits. The number of MCA required for network mapping

is also related to the shape of clusters obtained by clustering and is affected by the size of MCA in memristive inference accelerator.

The area and energy consumption have been normalized with respect to the Original area and energy consumption for each dataset. According to the experimental results above, ADMM-CP is overall better than TraNNsformer citeTraNNsFormer2020 on the two indicators, i.e. pruning ratio and the number of MCA required. Fig. 4 shows that ADMM-CP achieves 55.2%-64.6% area savings compared to Original across all datasets, higher than 37%-58.2% of TraNNsformer [3]. From Fig. 5, it can be seen that ADMM-CP achieves 58.1%-65.1% energy savings compared to Original across all datasets, higher than 31.3%-51.9% of TraNNsformer [3]. These results demonstrate the effectiveness of the proposed ADMM-CP framework on MLP topology in preserving the benefits of DNN sparsity at the hardware level.

C. ADMM-CP on CNN Topology

The proposed ADMM-CP framework is also evaluated for CNN topology, as shown in TABLE II. The validation accuracy over training epochs on the three CNN architectures under original training, TraNNsformer [3] and ADMM-CP is shown in Fig. 3 (d), (e), (f) of Appendix. A respectively. The accuracy curves of CNN topology are similar to the accuracy curves of MLP topology. ADMM-CP finally reaches the accuracy level produced by Original with the increase of training epochs. The pruning performance of Original, TraNNsformer [3], and ADMM-CP on CNN topology for each FC layer is shown in TABLE IV of Appendix. A. With compared to Original and TraNNsformer [3], the pruning proportion of ADMM-CP is greater across the three CNNs on the whole. The number of MCAs required for ADMM-CP is also less than Original and TraNNsformer [3] overall. The normalized area and energy costs of Original, TraNNsformer [3], and ADMM-CP on modified LeNet-5, AlexNet, and VGG16 are shown in Fig. 7 and Fig. 8 of Appendix. A respectively. Fig. 7 shows that ADMM-CP achieves 65.34%-95.4% area savings compared to Original across all datasets, significantly higher than 27.8%-76.1% of TraNNsformer [3]. From Fig. 8, it can be seen that ADMM-CP achieves 65.77%-92.2% area savings compared to Original across all CNN architecture, higher than 42.8%-84.3% of TraNNsformer [3]. These results further demonstrate the effectiveness of the proposed ADMM-CP framework on FC layers of CNN topology in preserving the benefits of DNN sparsity at the hardware level.

V. Conclusion

In this paper, a novel hardware-aware pruning algorithm named ADMM-CP based on ADMM and spectral clustering is proposed to improve power-area efficiency of memristor-based inference accelerators. Compared with previously reported pruning algorithms developed for fully connected layers of memristor-based neural networks, the obtained network sparsity of ADMM-CP can be completely translated into the reduction of hardware area or the number of memristors needed, thus improving power-area efficiency of memristor-based inference accelerators. Moreover, ADMM-CP is relatively insensitive to parameter initialization, which simplifies parameter selection. We evaluated ADMM-CP by transforming MLP and CNN topologies on different benchmark datasets. Simulation results show that ADMM-CP achieves significant improvement in power-area efficiency for memristor-based inference accelerator compared to original training. The novelty of ADMM-CP is also demonstrated by comparison with a recent same-purpose algorithm.

Acknowledgment

The authors would like to thank. . .

References

[1] P. Chi, S. Li, C. Xu, T. Zhang, J. Zhao, Y. Liu, Y. Wang, and Y. Xie, "Prime: A novel processing-in-memory architecture for neural network computation in reram-based main memory," in *ISCA 2016*, 2016, pp. 27–39.
[2] A. Ankit, I. El Hajj, S. Rahul Chalamalasetti, G. Ndu, M. Foltin, R. S. Williams, P. Faraboschi, W.-M. Hwu, J. Paul Strachan, K. Roy, and D. S. Milojicic, "Puma: A programmable ultra-efficient memristor-based accelerator for machine learning inference," in *ASPLOS*, 2019, pp. 715 – 731.
[3] A. Ankit, T. Ibrayev, A. Sengupta, and K. Roy, "TraNNsformer: Clustered pruning on crossbar-based architectures for energy-efficient neural networks," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 39, no. 10, pp. 2361 – 2374, 2020.
[4] S. Han, J. Pool, J. Tran, and W. J. Dally, "Learning both weights and connections for efficient neural networks," in *Adv. Neural Info. Process. Syst.*, 2015, pp. 1135 – 1143.
[5] K. Guo, S. Han, S. Yao, Y. Wang, Y. Xie, and H. Yang, "Software-hardware codesign for efficient neural network acceleration," *IEEE Micro*, vol. 37, no. 2, pp. 18 – 25, 2017.
[6] S. Han, X. Liu, H. Mao, J. Pu, A. Pedram, M. A. Horowitz, and W. J. Dally, "EIE: Efficient inference engine on compressed deep neural network," in *ISCA 2016*, 2016, pp. 243 – 254.
[7] A. Ren, T. Zhang, S. Ye, J. Li, W. Xu, X. Qian, X. Lin, and Y. Wang, "ADMM-NN: An algorithm-hardware co-design framework of dnns using alternating direction method of multipliers," in *ASPLOS*, 2019, pp. 925 – 938.
[8] G. Yuan, X. Ma, C. Ding, S. Lin, T. Zhang, Z. S. Jalali, Y. Zhao, L. Jiang, S. Soundarajan, and Y. Wang, "An ultra-efficient memristor-based dnn framework with structured weight pruning and quantization using admm," in *ISLPED 2019*, vol. 2019-July, 2019.
[9] L. Liang, L. Deng, Y. Zeng, X. Hu, Y. Ji, X. Ma, G. Li, and Y. Xie, "Crossbar-aware neural network pruning," *IEEE Access*, vol. 6, pp. 58 324 – 58 337, 2018.
[10] S. Lin, R. Ji, Y. Li, C. Deng, and X. Li, "Toward compact convnets via structure-sparsity regularized filter pruning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 2, pp. 574 – 588, 2020.
[11] E. Park, J. Ahn, and S. Yoo, vol. 2017-January, 2017, pp. 7197 – 7205.
[12] Y. Zhou, X. Hu, L. Wang, G. Zhou, and S. Duan, "Quantbayes: Weight optimization for memristive neural networks via quantization-aware bayesian inference," *IEEE Trans. Circuits Syst. I*, vol. 68, no. 12, pp. 4851 – 4861, 2021.
[13] V. Lebedev and V. Lempitsky, "Fast convnets using group-wise brain damage," in *CVPR*, vol. 2016-December, 2016, pp. 2554 – 2564.
[14] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278 – 2323, 1998.
[15] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Ng, "Reading digits in natural images with unsupervised feature learning," *NIPS*, Jan. 2011.
[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84 – 90, 2017.
[17] A. Ankit, A. Sengupta, P. Panda, and K. Roy, "RESPARC: A reconfigurable and energy-efficient architecture with memristive crossbars for deep spiking neural networks," vol. Part 128280, 2017.

[18] B. Rajendran, Y. Liu, J.-S. Seo, K. Gopalakrishnan, L. Chang, D. J. Friedman, and M. B. Ritter, "Specifications of nanoscale devices and circuits for neuromorphic computational systems," *IEEE Trans. Electron Devices*, vol. 60, no. 1, pp. 246 – 253, 2013.

APPENDIX A
SUPPLEMENTARY EXPERIMENTAL RESULTS

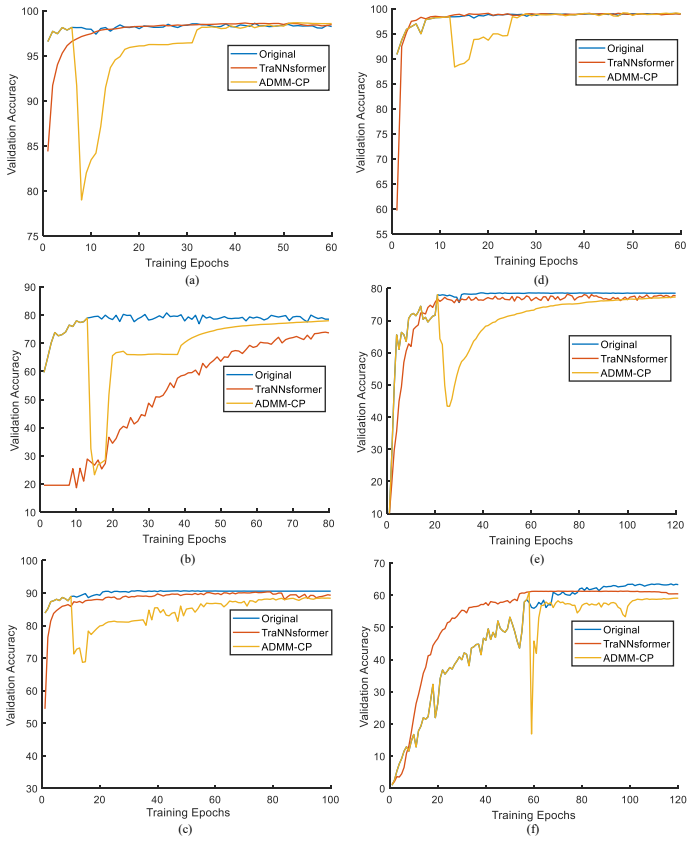

Fig. 3. The validation accuracy over training epochs for each MLP and CNN topology under original training, TraNNsformer and ADMM-CP. (a) MLP on MNIST, (b) MLP on SVHN, (c) MLP on FASHION-MNIST, (d) Modified LeNet-5 on MNIST, (e) AlexNet on CIFAR-10, (f) VGG16 on CIFAR-100.

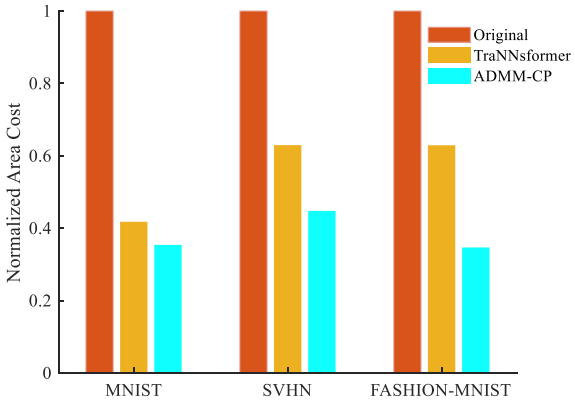

Fig. 4. The normalized area cost of Original, TraNNsformer and Proposed on MNIST and SVHN dataset for MLP topology.

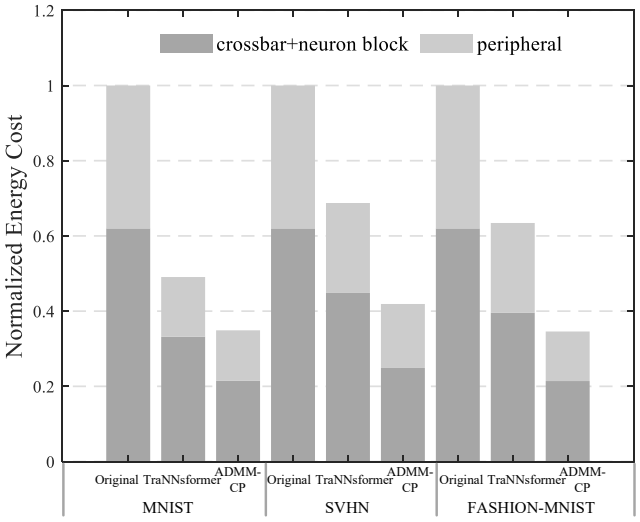

Fig. 5. The normalized energy cost of Original, TraNNsformer and Proposed on MNIST, SVHN and FAHION-MNIST dataset for MLP topology.

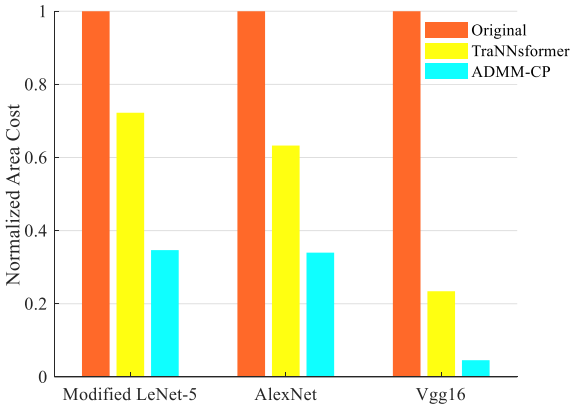

Fig. 6. The normalized area cost of Original, TraNNsformer and Proposed on Modified LeNet-5, AlexNet and VGG16.

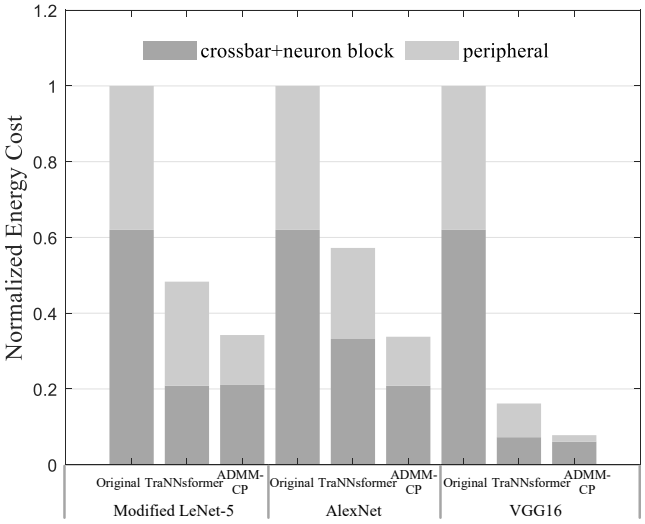

Fig. 7. The normalized energy cost of Original, TraNNsformer and Proposed on Modified LeNet-5, AlexNet and VGG16.

TABLE III
PRUNING PERFORMANCE OF RELATED ALGORITHMS ON MLP TOPOLOGY

Experimental results on 3-Layer MLP for MNIST dataset								
Layer	1st layer		2st layer		3st layer		4st layer	
Indicator	Pruned	MCAs Required	Pruned	MCAs Required	Pruned	MCAs Required	-	-
Original	-	58800	-	90000	-	900	-	-
TraNNsformer	71%	30376	30.60%	30325	2.37%	1938	-	-
ADMM-CP	64%	21215	66.70%	31179	65.40%	603	-	-
Experimental results on 4-Layer MLP for SVHN dataset								
Layer	1st layer		2st layer		3st layer		4st layer	
Indicator	Pruned	MCAs Required	Pruned	MCAs Required	Pruned	MCAs Required	Pruned	MCAs Required
Original	-	76800	-	90000	-	90000	-	900
TraNNsformer	24%	38382	27.0%	46385	31%	76521	87.10%	901
ADMM-CP	60%	31905	52%	52618	67.20%	30443	66.90%	397
Experimental results on 4-Layer MLP for FASHION-MNIST dataset								
Layer	1st layer		2nd layer		3rd layer		4th layer	
Indicator	Pruned	MCAs Required	Pruned	MCAs Required	Pruned	MCAs Required	Pruned	MCAs Required
Original	-	117600	-	360000	-	180000	-	900
TraNNsformer	33%	112905	36.42%	241333	37.77%	60094	80.56%	821
ADMM-CP	64%	42521	65.43%	124820	66.61%	60415	45.53%	486

TABLE IV
PRUNING PERFORMANCE OF RELATED ALGORITHMS ON CNN TOPOLOGY

Experimental results on MNIST dataset for Modified LeNet-5						
Layer	1st FC layer		2st FC layer		3st FC layer	
Indicator	Pruned	MCAs Required	Pruned	MCAs Required	Pruned	MCAs Required
Original	-	76800	-	90000	-	900
TraNNsformer	58%	57104	73.86%	62170	2.52%	1978
ADMM-CP	64%	27578	67.38%	30265	66.48%	302
Experimental results on Cifar10 dataset for AlexNet						
Layer	1st FC layer		2st FC layer		3st FC layer	
Indicator	Pruned	MCAs Required	Pruned	MCAs Required	Pruned	MCAs Required
Original	-	131072	-	262144	-	1536
TraNNsformer	19%	116213	21.87%	131512	25.77%	2027
ADMM-CP	67%	44203	66.10%	89240	66.00%	699
Experimental results on VGG16 for CIFAR-100 dataset						
Layers	1st FC layer		2nd FC layer		3rd layer	
Indicators	Pruned	MCAs Required	Pruned	MCAs Required	Pruned	MCAs Required
Original	-	65536	-	262144	-	12800
TraNNsformer	80.89%	25798	91.89%	42494	55.27%	11452
ADMM-CP	90.00%	4519	90.30%	22057	89.60%	1112