Hierarchical Non-Structured Pruning for Computing-In-Memory Accelerators with Reduced ADC Resolution Requirement

Wenlu Xue, Jinyu Bai, Sifan Sun and Wang Kang
School of Integrated Circuit Science and Engineering, Beihang University, Beijing, 100191, China
Email: wang.kang@buaa.edu.cn

***Abstract*—The crossbar architecture, which is comprised of novel nano-devices, enables high-speed and energy-efficient computing-in-memory (CIM) for neural networks. However, the overhead from analog-to-digital converters (ADCs) substantially degrades the energy efficiency of CIM accelerators. In this paper, we introduce a hierarchical non-structured pruning strategy where value-level and bit-level pruning are performed jointly on neural networks to reduce the resolution of ADCs by using the famous alternating direction method of multipliers (ADMM). To verify the effectiveness, we deployed the proposed method to a variety of state-of-the-art convolutional neural networks on two image classification benchmark datasets: CIFAR10, and ImageNet. The results show that our pruning method can reduce the required resolution of ADCs to 2 or 3 bits with only slight accuracy loss ($\sim$0.25%), and thus can improve the hardware efficiency by 180%.**

***Index Terms*—computing-in-memory, neural network, partial sum, pruning, alternative direction method of multipliers(ADMM)**

I. Introduction

Deep neural networks (DNNs) have become the key technology to realize artificial intelligence. Researchers have demonstrated the effectiveness of DNNs in various applications such as image classification [1], object detection [2], speech recognition [3], and natural language processing [4]. However, with the increasing number of parameters and the complexity of computation, the conventional von Neumann architecture fails to improve efficiency further because of the memory wall.

Recently, computing-in-memory (CIM) hardware is garnering attention as a new computing platform for DNNs [5]–[7]. CIM hardware uses memory arrays as both data storage and processing units, thus avoids frequent data transfer between memory and processing units. However, because of the limitations of the manufacturing process, it is difficult to fabricate large arrays. As a consequence, neural network's vector-matrix-multiplications (VMMs) must split the operands into several arrays to get partial sums and then sum them up. Because CIM performs computation in analog domain, high resolution digital-to-analog converters (DACs) and analog-to-digital converters (ADCs) are required for inputs and partial sums, respectively, which will significantly degrade the energy efficiency. Though the resolution of DACs can be lowered to 1 bit by processing input values in bit-serial manner [8], [9], high-resolution ADCs are still inevitable. Therefore, there is a great need for an efficient and accurate algorithm to reduce the resolution of ADCs.

This work was supported by the Natural Science Foundation of China (62274008), Beijing MSTC Nova Program (Z201100006820042 and Z211100002121014), Beijing Natural Science Foundation (L223004) and Joint Funds of the National Natural Science Foundation of China (U20A20204).

In order to reduce the resolution of ADCs, an intuitive method is to reduce the distribution range of partial sums. An intuitional implement method is to set more weights to zero by pruning. Thus, we proposed a hierarchical non-structured pruning method with both value-level and bit-level pruning to reduce the requirement in ADC's resolution. The main contributions of this paper are summarized as follows:

- We propose AMP, an adaptive magnitude-based value-level pruning method with a newly defined array-wise pruning granularity, to set the majority of weights to zero. The pruning threshold is determined adaptively for different arrays according to the resolution of ADCs so that partial sums can be reduced with less pruning error.
- In order to further improve the sparsity of weights, we introduce the summation of powers-of-two (PoT) terms based bit-level pruning (TBP). We determine a sparsity threshold for each array based on how sensitive the array is to pruning.
- Both of the value-level pruning and bit-level pruning can be formulated as discretely constrained non-convex optimization problems with indicator function. To solve the hierarchical pruning problem, we adopt ADMM to find the global optimal value.
- We evaluate the effectiveness of the proposed method in terms of accuracy, area and energy efficiency. We show that our method attains better performance than other works.

II. Related Work

A. CIM-Based Neural Network Computing

As shown in Fig. 1, because of the limitations of the manufacturing process, operands of neural networks' VMMs have to be split into several arrays. In neural networks, input X is a $3-D$ tensor with the shape of M×N×C, and the shape of weight W is P×C×E×F. To complete the calculation of VMMs which are divided into G groups on the input channel (i.e. each array has E×F×C/G rows), the input feature X_i is applied to each row of the array in the form of a voltage V_i converted by DACs, and P weight matrices are stored on the P column nano-devices in the form of conductance. According to Ohm's Law and Kirchhoff Circuit Law, the output of a filter is

$$Y_j = \sum_{g=0}^{G-1} ADC\Big(\sum_{i=0}^{E\times F\times C/G-1} W_{i,j,g} V_{i,g} \Big) \tag{1}$$

978-3-9819263-7-8/DATE23/© 2023 EDAA

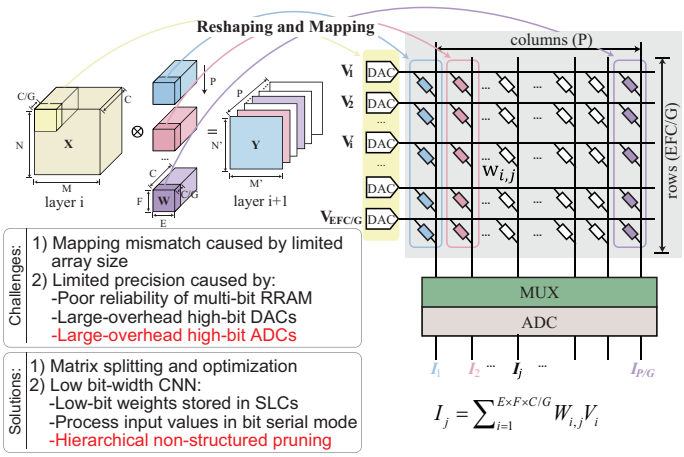

Fig. 1. The typical architecture of CIM-based VMMs computation. A VMM operation is firstly partitioned. Then inputs and weights are reshaped to 2-D matrices and mapped to the array. The accumulated currents are converted by ADCs as partial sums.

which will be converted by ADCs, and $ADC(\cdot)$ function is the transfer function of the ADC.

In addition, we use single-level cells (SLCs) to store the weights, which means that each cell will only save one bit, 0 or 1. A b_w-bit weight using SLCs is supposed to be

$$w = \sum_{i=0}^{b_w-1} 2^i w^{(i)} \tag{2}$$

where $w^{(i)}$ denotes the i^{th} bit in the binary representation. Compared with multi-level cells (MLCs), SLCs are not restricted to the type of memory devices. Various types of memory devices, such as volatile SRAM and nonvolatile ReRAM and MRAM, can be adopted to implement SLCs. Moreover, SLCs become more reliable against process variation because only two states are presented, namely, the high-conductance state (1) and the low-conductance state (0).

B. Non-Structured Neural Network Pruning

Pruning methods are widely used for compressing trained DNNs. They usually use some metrics to measure the importance of parameters and then set thresholds to abandon parameters with low importance. There are different ways to prune a neural network: non-structured pruning [10] where arbitrary weight can be pruned, and structured pruning [11] which maintains certain regularity.

Non-structured pruning can obtain neural networks with higher sparsity than structured pruning. However, practical acceleration can only be achieved by structured pruning when there is no additional hardware design for sparse matrix multiplication [12]. Thus, most of researchers concentrate on structured pruning. In this work, we focus on non-structured pruning from different perspectives which are value-level and bit-level. Non-structured sparsity can be used to reduce the distribution range of partial sums effectively.

C. Basics of ADMM

Deep learning heavily relies on optimization algorithms to solve its learning models. Constrained problems constitute a major type of optimization problem, and the ADMM method [13] is a commonly used algorithm to solve constrained problems, especially linearly constrained ones. It uses augmented Lagrangian to decompose the original problem into two subproblems and alternately updates the parameters until the optimal solution is obtained. In this way, we can get rid of the combinatorial constraints and solve the problem that is difficult to solve directly.

Consider a non-convex optimization problem as:

$$\begin{aligned} &\min_x f(x) + g(z) \\ &s.t. \quad x - z = 0 \end{aligned} \tag{3}$$

The augmented Lagrangian of Eq. (3) can be formed as:

$$\iota_\rho(x, z, u) = f(x) + g(z) + \frac{\rho}{2}\left\|x - z + u\right\|_2^2 - \frac{\rho}{2}\left\|u\right\|_2^2 \tag{4}$$

where u is the Lagrangian multipliers and $\rho > 0$ is a hyper-parameter. This problem can be decomposed into two subproblems on x and z, each of which is then easier to handle. Then ADMM consists of three-step iterations:

$$\begin{aligned} x^{k+1} &:= \arg\min_x (f(x) + \frac{\rho}{2}\left\|x - z^k + u^k\right\|_2^2) \\ z^{k+1} &:= \arg\min_z (g(z) + \frac{\rho}{2}\left\|x^{k+1} - z + u^k\right\|_2^2) \\ u^{k+1} &:= u^k + x^{k+1} - z^{k+1} \end{aligned} \tag{5}$$

Pruning problem can be viewed as a constrained problem where weights are constrained to a specific set. ADMM has already been adopted to neural network pruning [14].

III. Our Approach

A. Overview

In order to reduce the hardware overhead caused by high-resolution ADCs, we propose hierarchical non-structured pruning, which can narrow the distribution range of partial sums by improving the sparsity ratio of weights. As shown in Fig. 2, we first divide the weight matrices according to the size of the array, then perform value-level pruning designed according to the characteristic of the CIM array (Fig. 2(b)). Bit-level pruning is performed on the binary code of weights (Fig. 2(c)). Both value-level and bit-level pruning are decomposed into two subproblems through the application of ADMM. After these two pruning steps, we can obtain a highly sparse weight array, and then conduct VMMs to obtain smaller partial sums which can be converted by low-resolution ADCs.

B. Array-Adaptive Magnitude-Based Value-Level Pruning

Global network-wise pruning threshold is firstly used to prune neural networks [15]. Then layer-wise magnitude-based pruning is proposed to prune connections with absolute weight values lower than layer-specific thresholds [16]. However, as mentioned in Section I, VMMs in neural networks are computed in array-wise manner for CIM accelerators. We analyze

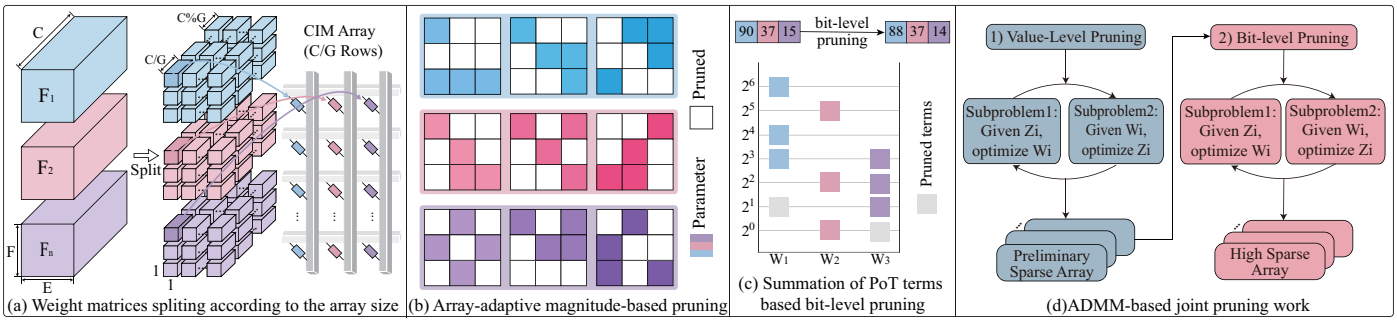

Fig. 2. Overview of our proposed hierarchical non-structured pruning method. The pruning problem is decomposed into two subproblems: the first one can be solved using standard stochastic gradient descent (SGD); the second one can be optimally and analytically solved.

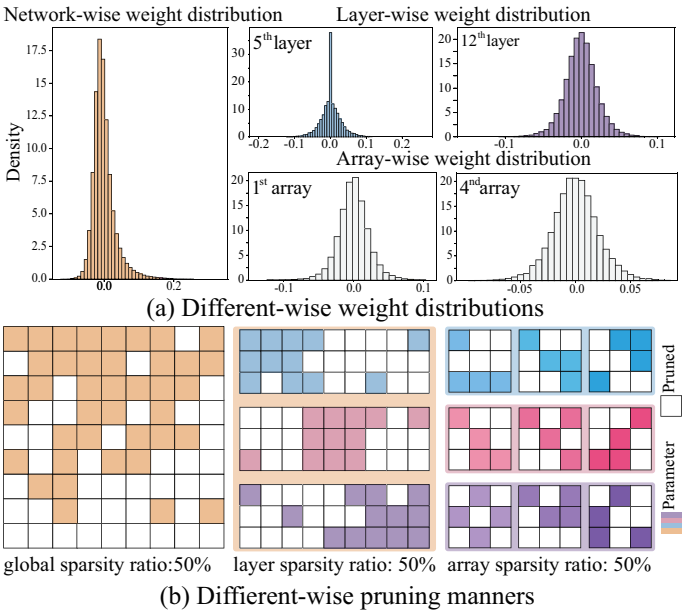

Fig. 3. (a) Weights distributions using different granularities (ResNet-50). (b) Different kinds of pruning granularities in terms of global-wise (left), layer-wise (middle), and array-wise (right).

the distribution of pruned weights in network-wise, layer-wise, and array-wise manners respectively (Fig. 3(a)). The weight distribution of network-wise shows the Gaussian-like distribution with a mean of zero. However, the variance and shape of the weights distribution for each layer become very different. Further, we can find that the distributions of weights for arrays vary substantially, even if they come from the same layer. Therefore, there is a possibility for further improvement if weight pruning is implemented in an array-wise manner.

Assuming that the required resolution of ADCs is b_{ADC}, the number of non-zero weights n_g on a bit line in the array g can be computed as

$$n_g = 2^{b_{ADC}-(b_a+b_w)} \tag{6}$$

where b_a and b_w represents the bit number of inputs and weights, respectively. Therefore, the number of weights on each bit line in an array should be less than or equal to n_g.

By this way, the generation of the pruned model is decided by the resolution of ADCs in each array. After sorting the weights

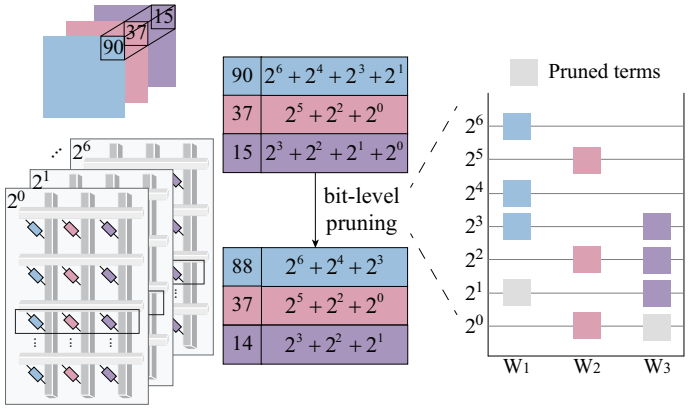

Fig. 4. Weights are represented as the summation of PoT terms. Then keep $s_g = 3$ terms in each weight.

according to the absolute vale, the pruning strategy of array g in each column can be described as

$$AMP(W, n_g) = \begin{cases} W_i & 1 \le i \le n_g \\ 0 & \text{others} \end{cases} \tag{7}$$

It should be noted that memory cells in a CIM array cannot directly represent negative weights, so we use two arrays to store the positive and negative elements in a weight matrix respectively as

$$Y = W \otimes X = (W_p - W_n) \otimes X = W_p \otimes X - W_n \otimes X \tag{8}$$

Here W_p contains all the positive weights and W_n includes the absolute value of all negative weights. Since we use array-wise granularity, the positive and negative matrices have separate thresholds. Compared with a matrix that contains positive and negative elements, the divided matrices have finer granularity, which is helpful to decrease the pruning error.

C. Summation of PoT Terms Based Bit-Level Pruning

As denoted in Eq. (2), we use SLCs to store the weights. In other words, a single value of weight has to be represented as the summation of PoT terms. For instance, the 7-bit value 67 (1000011) is the summation of 3 PoT terms: $2^6 + 2^1 + 2^0$.

According to this property, we can increase the sparsity ratio of binary codes in weights by limiting the number of terms in each weight to s_g, and map weights to the closest

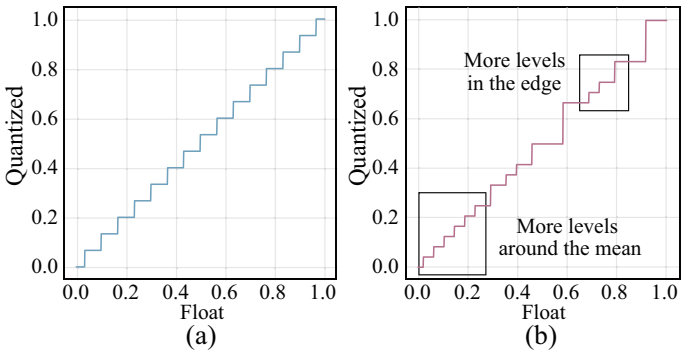

Fig. 5. Quantization curves when weights are quantized to 4-bit with uniform quantization (a) and TBP (b). TBP has a more reasonable resolution assignment.

value. The distribution of partial sums can be narrowed without introducing complicated peripheral circuits. Fig. 4 illustrates how TBP is applied to an array in a 7-bit weight matrix (only unsigned numbers need to be considered), and we retain up to $s_g = 3$ terms. These weights are formulated as the form of summation of PoT terms and decomposed into individual terms. Because there are only 3 terms in 37, we do not need to consider it. Weight with value of 90, which has 4 terms, is mapped to the closest value 88, so we pruned the term 2^1. For 15, we only need to prune the term 2^1 and map it to 14.

The most important parameter in this method is s_g. Research shows that the network can have significantly different sensitivity to pruning of each layer, and the distribution of weights in each layer is various [16]. Therefore, it is reasonable to speculate that each array is also different in sensitivity to pruning. Thus, we determine s_g according to the sensitivity of each array. It has been proved that the average number of the top eigenvalue of the Hessian matrix can be used to measure sensitivity [17] as

$$SEN = \frac{\lambda}{n} \tag{9}$$

where λ is the top eigenvalue of the Hessian matrix and n is the number of weights of an array. Smaller SEN indicates lower sensitivity, and a smaller s_g is required.

Note that it is not possible to explicitly form the Hessian matrix since a large amount of calculation and storage are needed. Hutchinson algorithm [17] can be used to reduce the computational complexity. Assume that H is the Hessian matrix of the weights W, and the Jacobian matrix of W is defined as J, we can compute λ with a random vector v as

$$\lambda = \frac{\partial (J^T v)}{\partial W} = \frac{\partial J_T}{\partial W} v + J^T \frac{\partial v}{\partial W} = \frac{\partial J^T}{\partial W} v = Hv \tag{10}$$

Since the random vector is independent of the weights, we have $\frac{\partial v}{\partial W} = 0$.

TBP can be regarded as an efficient nonuniform quantization. It can be seen from Fig. 5 that TBP achieves to assign more levels around zero compared to uniform quantization, and there are also more levels at the edge region. The distribution of the quantization levels matches weights with Gaussian-like distribution better than uniform quantization.

D. ADMM-Based Hierarchical Non-Structured Pruning

A trained DNN model can be represented as $\omega = \{W^{(i)}\}_{i=1}^{G}$, where G is the number of arrays. Let $f(\omega)$ denote the loss function which measures the accuracy of the model. To learn a compressed DNN model, we have the constrained problem:

$$\begin{aligned} &\min_{\{W^{(i)}\}} f(\left\{W^{(i)}\right\}_{i=1}^{G}) \\ &s.t. \quad W^{(i)} \in S^{(i)} \end{aligned} \tag{11}$$

For the AMP problem, the constraint set $S^{(i)} = \{$ n_g weights with the largest absolute values of each column in an array $\}$, and for the TBP problem, the constraint set $S^{(i)} = \{$the weights in array g are mapped to the quantization values $\}$, where the quantization values are a set of numbers that have s_g '1's in the binary code. Then we convert the constraints to an indication function as

$$I_s(W^{(i)}) = \begin{cases} 0 & \text{if } W^{(i)} \in S^{(i)} \\ +\infty & \text{otherwise} \end{cases} \tag{12}$$

By introducing an auxiliary variable Z^i, we can rewrite Eq. (11) as Eq. (13) with an extra equality constraint so that weights are constrained to be equal to the discrete variable.

$$\begin{aligned} &\min_{\{W^{(i)}\}} f(\left\{W^{(i)}\right\}_{i=1}^{G}) + \sum_{i=1}^{G} I_s(Z^{(i)}) \\ &s.t. \quad W^{(i)} = Z^{(i)} \end{aligned} \tag{13}$$

This non-convex optimization with convex linear constraints can be solved with ADMM. Eq. (13) can be decomposed into two subproblems through the application of the augmented Lagrange and can be formulated as,

$$\begin{aligned} &\min_{\{W^{(i)}\}} f(\left\{W^{(i)}\right\}_{i=1}^{G}) + \sum_{i=1}^{G} \frac{\rho_i}{2} \left\| W^{(i)} - Z_k^{(i)} + U_k^{(i)} \right\|_2^2 \\ &\min_{\{Z^{(i)}\}} \sum_{i=1}^{G} I_s^{(i)}(Z^{(i)}) + \sum_{i=1}^{G} \frac{\rho_i}{2} \left\| W_{k+1}^{(i)} - Z^{(i)} + U_k^{(i)} \right\|_2^2 \end{aligned}$$

where W, Z, U are simply updated as Eq. (5). The first subproblem can be solved by stochastic gradient descent, which has the same complexity as training the original DNN. The analytical solution to the second subproblem is

$$Z_i^{k+1} = \prod_{S_i} (W_{k+1}^{(i)} + U_k^{(i)}) \tag{14}$$

Here $\prod_{S_i}(\cdot)$ is a projection function of $W_{k+1}^{(i)} + U_k^{(i)}$ onto the set S_i. For the AMP, Eq. (14) is to keep n_g elements in $W_{k+1}^{(i)} + U_k^{(i)}$ with largest magnitude and set the rest to be zero. For the TBP, Eq. (14) is to set every elements in $W_{k+1}^{(i)} + U_k^{(i)}$ to be the quantization values. The whole procedure of the ADMM framework is summarized in Fig. 2(d).

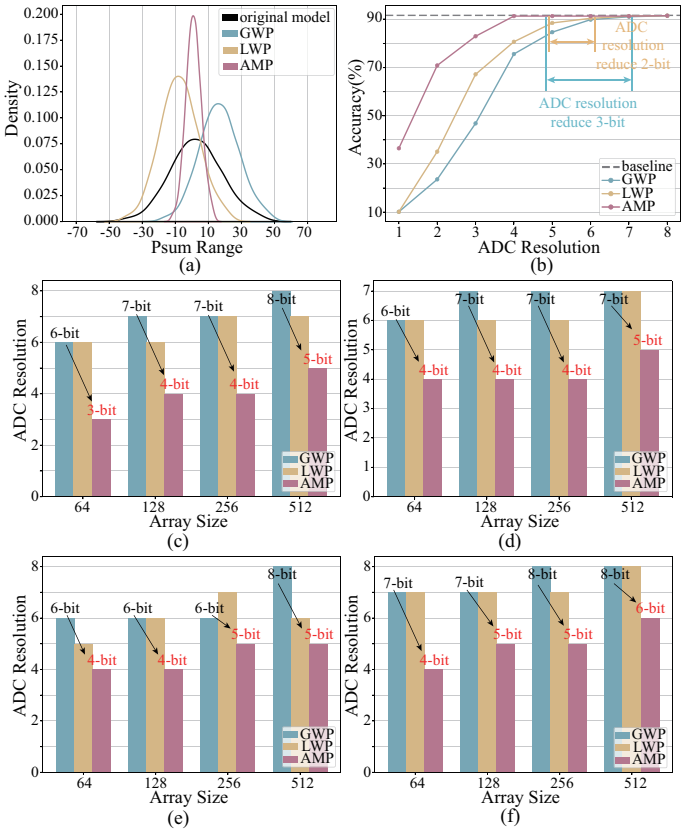

Fig. 6. (a) The distributions of partial sums obtained by the original model, GWP, LWP, and AMP (VGG-8 on CIFAR10), (b) accuracy comparison with same ADC resolution (VGG-8 on CIFAR10), and ADC resolution comparisons with the same accuracy: (c) VGG-8 on CIFAR10, (d) ResNet-18 on CIFAR10, (e) ResNet-18 on ImageNet, (f) ResNet-50 on ImageNet.

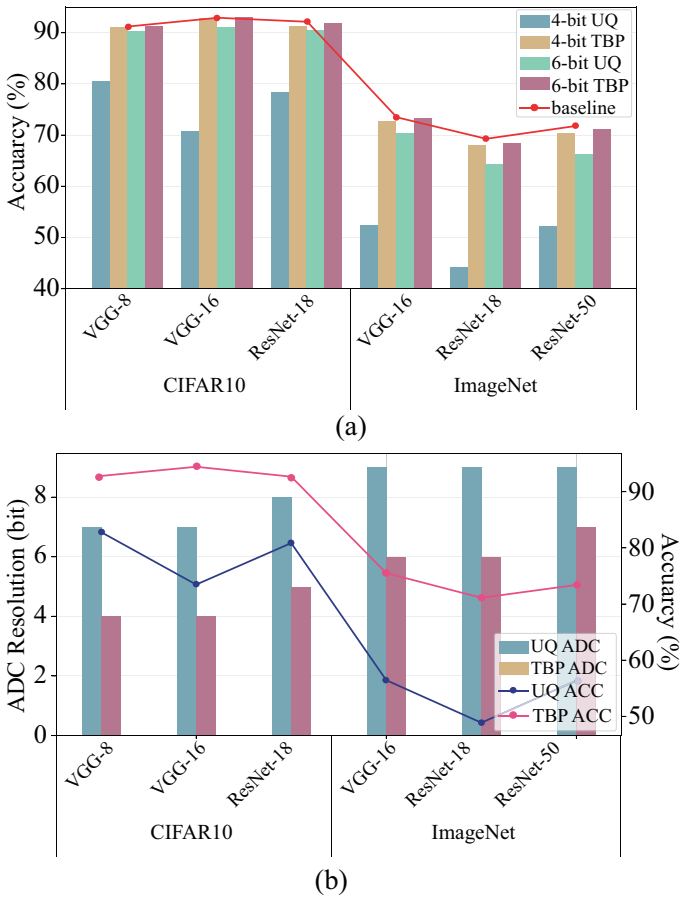

Fig. 7. (a) The comparison of accuracy when using 4-bit and 6-bit ADCs. (b) The required ADC resolution, and the accuracy when using 4-bit ADCs.

IV. Experiments

A. Experimental Setup

The goal of our method is to reduce the resolution of ADCs without sacrificing accuracy. We apply our algorithm framework to a set of benchmarks, VGG-8, VGG-16, and ResNet-18 on CIFAR10, VGG-16, ResNet-18, and ResNet-50 on ImageNet. We compare the ADC resolution and accuracy of the previous pruning works with that of our AMP method and TBP method respectively, and then perform comparisons with representative works on CIM-based partial sums quantization to demonstrate the significant improvement of our ADMM-based joint pruning framework.

In addition, we use NeuroSim [18] to examine the hardware performance, which is an end-to-end benchmarking framework for CIM-based accelerators. The comprehensive experiments indicate that our hierarchical pruning algorithm outperforms prior state-of-the-art designs in area and energy efficiency.

B. Effectiveness of Finer Granularity for Pruning

In this subsection, we demonstrate the effect of pruning granularity on ADC resolution and model accuracy. We perform the pruning work in three different ways: global-wise pruning (GWP) [15], layer-wise pruning (LWP) [16], and our AMP. We use VGG-8 on CIFAR10 with array size set to 128×128 as case study. Fig. 6(a) shows that the range of partial sums is $[-64, 63]$ when using the original model, and it is narrowed to 25% of the original after using AMP. We also record the range of partial sums obtained by using GWP and LWP, and our method performs better obviously.

Using this benchmark, we also compare the model accuracy with the same resolution of ADCs. It can be seen from Fig. 6(b) that when the resolution of ADCs goes down, array-wise pruning granularity can get better accuracy than others.

Furthermore, we perform different value-level pruning methods based on GWP, LWP and AMP to compare the resolution of ADCs. During the experiment, we control the accuracy of various methods to be almost the same as the baseline. As can be seen from Fig. 6 (c)-(f), our AMP always achieves the lowest resolution. According to the experimental results, we can also find that when the array size decreases, the resolution of ADCs becomes lower. Because the number of parameters decreases, resulting in a small range of partial sums. Our method only needs 3-bit ADC in an array with the size of 64×64, in particular, when implemented on VGG-8 on CIFAR10.

C. Effectiveness of TBP based Bit-Level Pruning

In order to verify the effectiveness of our TBP method, We assume that the neural network is implemented on the CIM accelerator with array row count 128. As the proposed TBP can

TABLE I
ACCURACY COMPARISON BETWEEN DIFFERENT DATASETS, NETWORKS, METHODS, AND THE ADC RESOLUTION.

Method	ADC Resolution	Acc(%)	Acc Drop(%)	Efficiency (Tops/W)	Efficiency Improve
VGG-8 on CIFAR10					
[19]	5-bit	88.38	3.16	66.5	1.04×
[20]	3-bit	90.25	1.29	84.4	1.32×
[21]	3-bit	91.3	0.24	82.5	1.29×
Ours	**2-bit**	**91.19**	**0.35**	**87.7**	**1.37×**
ResNet-18 on CIFAR10					
[19]	6-bit	91.2	1.25	112.37	1.22×
[20]	4-bit	91.4	1.02	120.66	1.31×
[21]	3-bit	91.9	0.53	133.56	1.45×
Ours	**3-bit**	**91.82**	**0.64**	**152.9**	**1.66×**
ResNet-18 on ImageNet					
[19]	7-bit	68.3	1.3	120.4	1.23×
[20]	4-bit	69.44	0.16	155.6	1.59×
[21]	3-bit	69.31	0.29	174.2	1.78×
Ours	**3-bit**	**69.39**	**0.21**	**168.3**	**1.72×**
ResNet-50 on ImageNet					
[19]	7-bit	71.87	0.43	99.9	1.36×
[20]	4-bit	71.92	0.38	130.1	1.77×
[21]	4-bit	71.99	0.31	128.7	1.75×
Our work	**3-bit**	**72.05**	**0.25**	**132.3**	**1.80×**

be viewed as a non-uniform quantization method, we compare the accuracy of TBP with uniform quantization (UQ). As can be seen from Fig.7 (a), the accuracy of TBP is always higher than that of the UQ. When using 4-bit ADCs, the accuracy gap becomes more obvious. We also report the lowest ADC resolution required for two methods and compare the accuracy when using 4-bit ADCs. Fig. 7(b) demonstrated that TBP can even reduce the required resolution of ADC to 4-bit when UQ needs to use 7-bit ADCs on VGG-8 and CIFAR10. Moreover, when using 4-bit ADCs, the accuracy of TBP is far better than UQ in various network models. It should be noted that 1-bit resolution reduction halves the area and energy consumption of ADCs, so 2-3 bits reduction of ADC's resolution is meaningful.

D. Performance of Hierarchical Non-Structured Pruning

Table 1 presents the results on VGG-8, ResNet-18, and ResNet-50. For comparison, we take the results reported in [19]–[21]. We can observe that our work can achieve the lowest requirement of ADCs' resolution with slight accuracy loss. Particularly, our work uses only 3-bit ADCs with the minimal accuracu loss (only ∼0.25%), and improve the hardware efficiency by 180% when performed on ResNet-50, which is 1-4 bits less than that of other methods.

Fig. 8 reports the ADC area and energy for VGG-8 with array size set to 128×128. This result implies that the overhead of ADC is significant in CIM-based DNN accelerators, so lowering ADC overhead can improve overall efficiency of the CIM accelerators. Specifically, our framework saves 26.28% of the area and 9.55% of the energy compared to [21].

V. CONCLUSION

CIM based accelerator has shown a great prospect in neural network acceleration. However, the high-resolution ADCs for partial sums lead to the limited efficiency. In this paper, we propose hierarchical non-structured pruning to achieve lower requirement of ADCs' resolution while maintaining the accuracy to boost the efficiency of DNN inference. Our evaluation shows that the proposed hierarchical non-structured pruning outperforms other similar solutions in area, energy-efficiency, and accuracy. We can reduce the ADC resolution even to 2-3 bits and improve the hardware efficiency by 180% with slight accuracy loss (∼0.25%).

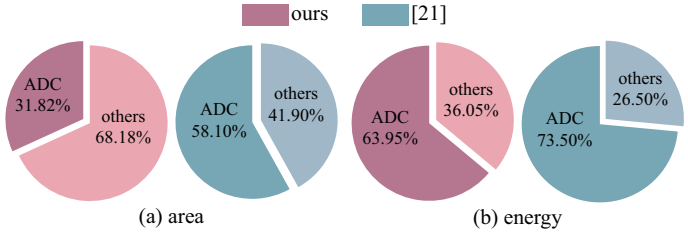

Fig. 8. (a) Area comparison on VGG-8, CIFAR10. (b) Energy comparison on VGG-8, CIFAR10.

REFERENCES

[1] A. Krizhevsky et al., "Imagenet classification with deep convolutional neural networks," Communications of the ACM, vol.60, pp. 84-90, 2012.
[2] W. Liu et al., "Ssd: Single shot multibox detector," ECCV, pp. 21-37, October 2016.
[3] W. Chan et al., "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," ICASSP, pp. 4960-4964, March 2016.
[4] Y. Goldberg, "Neural network methods for natural language processing," Synth, vol. 10, No. 1, pp. 1-309, 2017.
[5] P. Chi et al., "PRIME: A novel processing-in-memory architecture for neural network computation in ReRAM-based main memory," ISCA, pp. 27-39, 2016.
[6] A. Shaiee et al., "ISAAC: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars," ISCA, pp. 18–22, 2016.
[7] Z. Zhu et al., "Configurable multi-precision CNN computing framework based on single bit RRAM," DAC, pp. 1–6, 2019.
[8] H Jia et al., "A programmable neural-network inference accelerator based on scalable in-memory computing," ISSCC'21, pp. 236–238, 2021.
[9] A Shaiee et al., "A convolutional neural network accelerator with in-situ analog arithmetic in crossbars," ISCA, pp. 18–22, 2016.
[10] J. H. Luo et al., "An entropy-based pruning method for cnn compression," arXiv preprint arXiv: 1706. 05791, 2017.
[11] J. H. Luo et al., " Thinet: A filter level pruning method for deep neural network compression," ICCV, pp. 5058–5066, 2017.
[12] Z. Liu et al., " Learning efficient convolutional networks through network slimming," ICCV, pp. 2736–2744, 2017.
[13] D. Gabay et al., "A dual algorithm for the solution of nonlinear variational problems via finite element approximation," COMPUT MATH APPL, vol. 2(1), pp. 17–40, 1976.
[14] S. Ye 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] B. Hassibi et al., "Optimal brain surgeon and general network pruning," IJCNN, pp. 293–299, 1993.
[16] J. Lee et al., "Layer-adaptive sparsity for the magnitude-based pruning," arXiv preprint arXiv:2010.07611, 2020.
[17] Z. Dong et al., "Hawq: Hessian aware quantization of neural networks with mixed-precision," ICCV, pp. 293-302, 2019.
[18] X. Peng et al., "DNN+ NeuroSim: An end-to-end benchmarking framework for compute-in-memory accelerators with versatile device technologies," IEDM, pp. 32-5, 2019.
[19] Q. Wu et al., "Software-hardware co-optimization on partial-sum problem for PIM-based neural network accelerator," HPEC, pp. 1-7, 2021.
[20] Y. Kim et al., "Extreme partial-sum quantization for analog computing-in-memory neural network accelerators," JETC, 2022.
[21] A. Azamat et al., "Quarry: Quantization-based ADC reduction for ReRAM-based deep neural network accelerators," ICCAD, pp. 1-7, 2021.