CoDG-ReRAM: An Algorithm-Hardware Co-design to Accelerate Semi-Structured GNNs on ReRAM

Yixuan Luo†*, Payman Behnam‡*, Kiran Thorat§, Zhuo Liu†, Hongwu Peng§, Shaoyi Huang§, Shu Zhou†, Omer Khan§, Alexey Tumanov‡, Caiwen Ding§, Tong Geng†

† The University of Rochester, NY ‡ Georgia Institute of Technology, GA § The University of Connecticut, CT

†{yixuan.luo, zhuo.liu, szhou34, tong.geng}@rochester.edu, ‡{payman.behnam, atumanov}@gatech.edu, §{kiran_gautam.thorat, hongwu.peng, shaoyi.huang, khan, caiwen.ding}@uconn.edu

***Abstract*—Graph Neural Networks (GCNs) have attracted wide attention and are applied to the real world. However, due to the ever-growing graph data with significant irregularities, off-chip communication with poor data locality has become the major bottleneck hurdling the development of GCNs. Fortunately, recent works demonstrate Resistive Random Access Memory (ReRAM) has the potential to perform inherently parallel in-situ computation of Matrix-Vector Multiplication (MVM) in the analog regime fundamentally breaking the communication bottleneck.**

Inspired by this observation, we propose a novel ReRAM-based GCN acceleration co-design (i.e. algorithm-hardware) framework, CoDG-ReRAM, that can deliver real-time GCN inference with high accuracy. On the algorithm side, we propose a novel model optimization pipeline that simultaneously and efficiently sparsifies and regularizes both graph and parameter matrices in GCNs and creates ReRAM-friendly models. On the hardware side, we take advantage of the software optimization results to provide a more systematic mapping scheme and increase computation efficiency to have an energy-efficient ReRAM-based GCN acceleration with low latency. Experimental results show that the proposed work improves performance and energy efficiency by 4× and 5.1 × respectively over SOTA ReRAM-based accelerators of GCNs with a negligible accuracy loss.

***Index Terms*—Graph Neural Network, Processing In Memory, Computer Architecture, Resistive Random Access Memory**

I. INTRODUCTION

Due to the high accuracy and excellent information acquisition capability, GCN has become an extremely vital and fundamental method in graph applications including recommendation systems, power grids, and biomedical research [54]. Different from the traditional Deep Neural Networks (DNNs) whose data have to be structured, GCNs directly work with non-Euclidean data, which drastically extends their applications. However, due to the ever-growing graphic data size and its irregularity, data communication has been the key bottleneck [1], [32], hurdling the development of GCNs and seriously restricting their industrial boarding.

As a rising and rapidly developing technology, ReRAM devices are considered to be a promising approach to perform the inherent parallel in-situ MVM in the analog regime with O(1) complexity [33], [37], [41], [48], which can greatly reduce the data movement and save computation resources. Thanks to those distinctive properties, researchers have enforced DNN accelerators on ReRAM devices [30], [33], [47], [48], which present excellent performance including low energy consumption and low inference latency. Although there exist several high-performance accelerators [16], [18], [44], [46] and even ReRAM-based solutions to accelerate GCNs [22], few of them leverage the potentials of algorithm-hardware co-design of ReRAM-based mixed-signal accelerators for GCNs.

In this paper, we propose a novel ReRAM-based GCN acceleration co-design framework to fundamentally eliminate the communication bottleneck in GCN computation while maintaining extraordinary accuracy. Figure 1 illustrates the overview of our co-design framework. There are two essential efforts that contribute to this work to achieve superior performance: software algorithm and hardware design. In the algorithm part, we propose and enforce three optimizations: a novel semi-structural graph topology optimization, column balanced block-wise weight pruning, and ReRAM-customized weight quantization. Starting with graph topology optimization, to reduce the irregularity and enhance the structural sparsity of the graphs, we reconstruct the whole graph via subgraph classification and group partitioning. Besides, to maintain high accuracy, we also detect and prune edges that make negative contributions to classification. Then, to reduce redundant computation, we apply ADMM-based column balanced block-wise weight pruning [31], which can ensure that the weight matrices share a similar pruning pattern with the optimized graph and furthermore make the entire model more hardware-friendly. Finally, to better deploy GCNs on ReRAM, we apply ADMM-based weight quantization [52] to reduce the precision of weight matrices from single/double precision floating point to 2/4/8 bits while maintaining high accuracy. On the hardware level, we propose an efficient ReRAM-based accelerator that utilizes the optimizations from the algorithm side to more systematically map the weight and feature matrices into crossbars. our contributions are summarized as follows:

* Equal contribution

2576-6996/22/$31.00 ©2022 IEEE
DOI 10.1109/ICCD56317.2022.00049

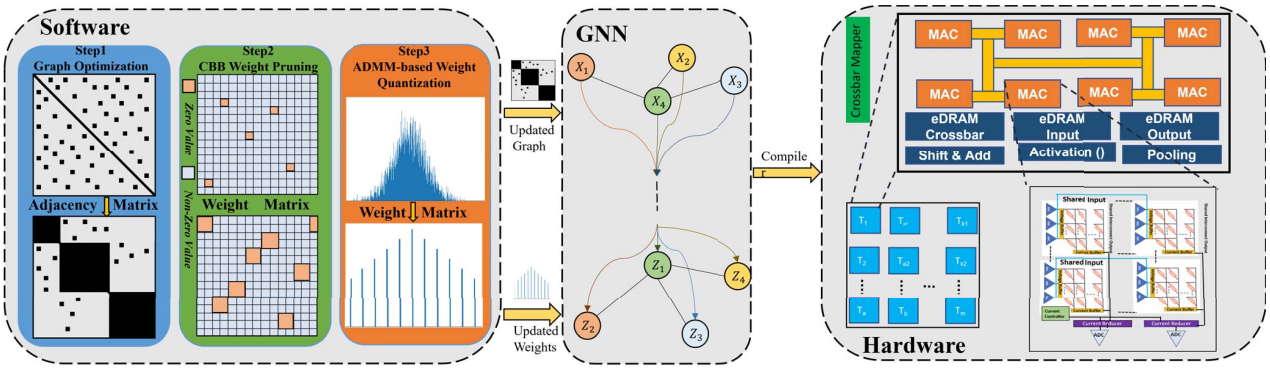

Fig. 1. Overview of the proposed CoDG-ReRAM codesign framework for GNN acceleration.

- We propose the ReRAM-based GNN co-design acceleration framework, CoDG-ReRAM, to fundamentally address the inherent communication problems in GNNs, delivering both low latency and high accuracy.
- We propose a novel algorithmic optimization pipeline for GCNs, which creates highly sparse and regular GCN models that are friendly to ReRAM.
- We leverage the algorithm optimizations into hardware design and come up with a more systematic approach to map weights and feature matrices into ReRAM crossbars to increase computational efficiency and reduce latency.
- Experimental results demonstrate that CoDG-ReRAM reduces inference latency and energy consumption by 4× and 5.1× over SOTA ReRAM-based accelerators of GCNs with a negligible accuracy loss.

To the best of our knowledge, this work is the first attempt on algorithm-hardware co-design of ReRAM-based GCN acceleration framework.

II. BACKGROUND AND RELATED WORKS

A. Resistive Random Access Memory

ReRAM memory technologies are non-volatile, with high density, almost free of leakage power, and more immunity to transient faults compared with traditional volatile DRAM memories. They also have shown high, excellent speed switching and low power consumption [13], [35]. Compared to other types of nonvolatile memories, ReRAM-based devices show high scalability, superb multilevel cell storage capability, and the possibility of low-cost 3D fabrication [13]. In the last decade, there are extensive works that illustrate the results of fabricated ReRAM memory cells [55], memory arrays [39], and also fabricated ReRAM-based accelerators [6], [9], [43]. Moreover, constant progress has been made towards pushing non-volatile memories to commercial products. For instance, Micron and Intel jointly produced 3D Xpoint [19], [21].

B. Graph Convolutional Networks

As one of the most distinguished Graph Neural Network (GNN) models, GCN [26] starts the trend of using neural network methods in graph information extraction. There are many other modalities based on message passing methods of GraphCONV used in GNNs, e.g. GraphSage [20] and Graph Isomorphism Network (GIN) [42].

As stated in [18], the forward computation flow of these models generally contains two common phases: *Aggregation* and *Combination*. During the aggregation phase, all nodes gather and aggregate the features of their neighbor nodes to update their own feature vectors. During the combination phase, the GraphCONV layer builds up a local Multi-Layer Perceptron (MLP) network to merge the updated feature-vectors, which helps the model extract high-level abstraction information.

Let define a graph $\mathscr{G} = (\mathscr{V}, \mathscr{E})$ with N nodes $v_i \in \mathscr{V}$, edges $(v_i, v_j) \in \mathscr{E}$, the adjacency matrix $A \in R^{N \times N}$, the degree matrix $D_{ii} = \sum_j A_{ij}$ and the feature matrix $X = \{x_1, x_1, ..., x_N\}$. We can assume $\widehat{A}$ as the normalized A: $\widehat{A} = D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$ and W_i as the weight matrix of the i^{th} layer. Using given parameters, the forward procedure of a two-layer GCN model can be simply formulated as follow:

$$Z = f(A, X) = \widehat{A}\, ReLU(\widehat{A} X W_0) W_1 \tag{1}$$

C. GCNs Accelerator

Researchers have proposed a series of high-performance computer architecture for GNN acceleration focusing different computational and communication problems in GNN training and inference [1]–[3], [12], [16]–[18], [28], [44], [46], [49], [50], [53].

In [3], Auten et al., first introduces the concept of GNN hardware accelerator which can realize high performance in tackling irregular data movement and intensive computation for GNN inference by designing four specialized modules for graph traversals, dense matrix operations, data scheduling, and graph aggregations, respectively. *HyGCN* [44] is one of the earliest GNN accelerators. Since the inference procedure of GCNs contains two phases with different computation patterns, HyGCN proposes a hybrid architecture with dedicated modules for aggregation and combination, respectively. *AWB-GCN* [16] is another early study of GCN acceleration. It observes that the power-law distribution of the non-zeros in the adjacency matrix results in workload imbalance issues. To

solve this problem, the authors propose a workload autotuning technique. Inspired by AWB-GCN, researchers discover another basic problem in GNN acceleration is the poor data locality, especially in graph aggregation.

Recently, *I-GCN* [18] is proposed to solve this problem. I-GCN uses a new graph reordering algorithm named islandization and realized it with Intel FPGA. With islandization algorithm, I-GCN can greatly improve the data locality of graph aggregation at runtime to the point that almost all data of the graph adjacency matrix, feature matrices, and weight matrices are accessed from off-chip only once. *EnGN* [28] introduces a unified architecture to accelerate GNNs and enforces a ring-based network to perform aggregation. The results produced by PEs are sent to the network where they are aggregated. Researchers have also developed novel hardware designs for training. *Rubik* [12] proposes an offline graph reordering method to improve data locality. *GraphACT* [49] uses heterogeneous platforms with CPUs and FPGAs and uses pre-processing to find and skip redundant operations among two-node shared neighbors.

G-CoS [53] is the first GNN co-search framework for network structure and accelerator architecture. G-CoS can automatically search for the matched GNN structures and accelerators to maximize both task accuracy and acceleration efficiency.

In addition to the efforts on designing hardware architecture for GNN acceleration, *GCoD* [46] first proposes a co-design framework targeting traditional devices which can alleviate the aforementioned workload imbalance and poor data locality problems and accelerate GNNs' inference by effectively enhancing graph regularity.

Besides the aforementioned digital accelerators for GNNs, researchers have also investigated the potentials of using mixed-signal designs to accelerate GCNs. REFLIP [22] builds an in-situ ReRAM-based PIM acceleration engine for both combination and aggregation kernels of GCNs. It presents a novel flexible mapping scheme for crossbar architectures by exploiting intra- and inter- vertex parallelism of GCNs. PIM-GCN [10] proposes a ReRAM-based accelerator. To take full advantage of the intra-vertex parallelism, it employs dense data mapping as well as a search-execute architecture. It also proposes two scheduling strategies to improve inter-vertex parallelism and pipeline. PASGCN [45] optimizes the PIMGCN further by proposing edge selection strategies that are obtained in the training phase by learning downstream feedback signals for each GCN layer separately and adaptively. The selected edges are used in the inference time.

III. SOFTWARE FRAMEWORK

In this section, we present our newly proposed model optimization pipeline that generates ReRAM-friendly GCN models with superior regularity and sparsity with the focus on the three optimizations including graph topology optimization, ReRAM Customized model quantization, and Block-wise model sparsification.

With the extensive use of model compression techniques in all kinds of neural networks, semi-structured pruning and quantization have been demonstrated that they can efficiently accelerate deep learning inference while maintaining high accuracy. On the other hand, recent works indicate the inference speeds of GNN models can also be greatly and uniquely improved via graph topology optimization. Therefore, to take the advantages of both methods at the same time, we first propose a new GCN model optimization pipeline. The key novelties in our work are graph topology simplification/optimization and model compression. To enforce the graph topology optimization, we perform non-structural sparsification and polarization with the adjacency matrices after partitioning the raw graph into several subgraphs. Then, using the optimized graph, the weight matrices of GCN models are further optimized with semi-structural pruning and quantization based on ADMM, which effectively makes GCN models ReRAM-friendly. In the following subsections, we describe these three optimization methods and explain how to implement and deploy them in general GCN training process.

A. Graph Topology Optimization

The purpose of graph topology optimization is to reconstruct the graph adjacency matrices to have higher sparsity with regular non-zero distributions, which drastically enhances the data locality of graph processing and further makes GCN models more hardware-friendly. The proposed optimization algorithm is inspired by the ones used in GCoD which demonstrates that graph reconstructing can significantly improve the performance of GNN inference. However, the algorithms used in GCoD unfortunately lead to non-negligible accuracy degradation which can be further enlarged after weight sparsification and quantization are applied. To this end, on the top of the algorithm designs proposed in GCOD, we propose a new graph topology optimization algorithm to obtain adjacency matrices with the same level of regularity but higher sparsity and higher accuracy.

Same as GCoD, to improve the sparsity and regularity of adjacency matrices, CoDG-ReRAM also adopts the subgraph classification algorithm and the group partitioning algorithm proposed in the graph topology optimization stage as shown in Figure 2(a). We briefly introduce these algorithms below for the convenience of readers. More details are omitted due to space limitations and can be found in [46]. The subgraph classification algorithm is mainly in charge of improving the regularity of the graph adjacency matrix by clustering nodes with homologous degrees into the same class. After alleviating the irregularity of the adjacency matrix, the group partitioning algorithm is used to decrease the boundary connections, which helps reduce inter-group communication.

As mentioned above, the method used in GCoD leads to severe accuracy degradation on GCN models. To maintain GCN models' inference accuracy, we further augment the method described above by adding a novel inferred-class-based negative edge pruning technique which is able to partially filters out the edges that make negative contributions to

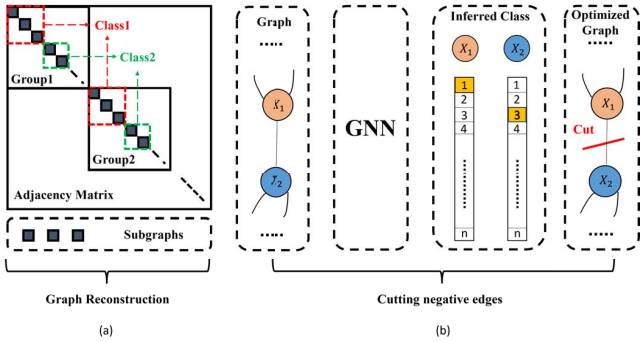

Fig. 2. Illustrating (a) GCoD's algorithm for graph reconstruction and (b) the workflow of negative edge pruning

the classification results. By so doing, the models' accuracy and sparsity can be both improved. Besides, as negative edges are normally the ones connecting different graph components, we also observe that this optimization also helps create more regular graph typologies, more clustered non-zero elements in adjacency matrices, and clearer connections among graph components.

Figure 2 (b) shows the workflow of the proposed inferred-class-based negative edge pruning. From the last process of previous graph reconstruction, we have trained a GCN model, with which we can obtain inferred class of each sample X_i. The inferred class can be formulated as:

$$Y_{Inferred\_Class|X_i} = index_{MAX}(Y_{X_i|class1}, Y_{X_i|class2}, ..., Y_{X_i|classM}), \tag{2}$$

where $Y_{i|classM}$ denotes the output probability of the i^{th} sample for class M. For two nodes with a connection, if their inferred classes are different, we consider their connection as a negative edge and will wrongly contribute to the task and hence prune these negative edges from the graph. With this optimization, we can not only avoid accuracy degradation but also even obtain accuracy improvement for some datasets. Section V.A will evaluate the benefits of this proposed method.

B. Column Balanced Block-wise Weight Pruning

Given a GCN model with N layers with W_i as the weight matrix at the i^{th} layer and b_i as the bias at the i^{th} layer, the loss function for the N-layers GCN model can be formulated as: $\mathscr{L}(\{W_i\}_{i=1}^N, \{b_i\}_{i=1}^N)$. Then the problem of weight pruning can be written as an optimization problem:

$$\begin{aligned} \underset{\{W_i,b_i\}}{minimize} \quad & \mathscr{L}(\{W_i\}_{i=1}^N, \{b_i\}_{i=1}^N) \\ subject \quad to \quad & W_i \in C_i, i = 1, ..., N, \end{aligned} \tag{3}$$

where the C_i is the constraint set for the i^{th} weight matrix W_i. Specifically, the constraint set C_i restricts the number of non-zero values in weight matrix W_i to be less than or equal to $\mathscr{T}_i$, where $\mathscr{T}_i$ is the desired number of non-zero value after pruning in the i^{th} layer.

However, obviously, the optimization mentioned above is non-convex with combinatorial constraints, which means that we cannot reach the global minimization via traditional gradient descent algorithms (e.g. SGD and Adam). Thanks to the rapid development of the non-convex optimization methods, we can efficiently solve it by using the Alternating Direction Method of Multipliers (ADMM) framework which can get rid of the combinatorial constraints. To help apply the ADMM method to weight pruning, we define an auxiliary indicator function as:

$$\mathscr{I}_i(W_i) = \begin{cases} 0 & if \quad W_i \in C_i \\ +\infty & otherwise \end{cases} \tag{4}$$

Then we can use an auxiliary variable V_i instead of W_i in the non-differentiable terms and rewrite our optimization problem to be:

$$\begin{aligned} \underset{\{W_i,b_i\}}{minimize} \quad & \mathscr{L}(\{W_i\}_{i=1}^N, \{b_i\}_{i=1}^N) + \sum_{i=1}^N \mathscr{I}_i(V_i) \\ subject \quad to \quad & W_i \in V_i, i = 1, ..., N. \end{aligned} \tag{5}$$

Applying the augmented Lagrangian, the rewritten problem can be divided into two subproblems by ADMM. We solve each subproblem respectively and iteratively until reaching the global solution. The first subproblem can be written as:

$$\begin{aligned} \underset{\{W_i,b_i\}}{minimize} \quad & \mathscr{L}(\{W_i\}_{i=1}^N, \{b_i\}_{i=1}^N) \\ & + \sum_{i=1}^N \frac{p_i}{2} \|W_i^{k+1} - V_i + D_i^k\|_F^2, \end{aligned} \tag{6}$$

where $D_i^k = D_i^{k-1} + W_i^k - V_i^k$ and D_i^k is the dual variable which will be updated in each epoch. In the first subproblem function, the left term stands for the differentiable loss function of the GCN model and the right term is a quadratic function that is differentiable and convex. Therefore, without the combinatorial constraints in it, the first subproblem is similar to the original GCN optimization problem and can be solved efficiently via gradient descent algorithms (e.g. Adam). The second subproblem can be formulated as:

$$\begin{aligned} \underset{\{C_i\}}{minimize} \quad & \sum_{i=1}^N \mathscr{I}_i(C_i) \\ & + \sum_{i=1}^N \frac{p_i}{2} \|W_i^{k+1} - V_i + D_i^k\|_F^2 \end{aligned} \tag{7}$$

Since $g_i(\cdot)$ denotes the indicator function of S_i, we can conclude the solution to the second problem is

$$C_i^{k+1} = \prod_{C_i} (W_i^{k+1} + D_i^k), \tag{8}$$

where $\prod_{S_i}$ denotes the Euclidean projection of the term $W_i^{k+1} + D_i^k$. Then the updated V_i^{k+1} will be used instead of V_i^k in the first subproblem in the next epoch.

Furthermore, to enforce the weight matrices that can simply be mapped to ReRAM, as Figure 3 shows, we adopt a column balanced block-wise weight pruning algorithm which can enhance GCN inference performance on ReRam devices and make the resulting weight matrices have a similar sparse pattern to the one in optimized adjacency matrices hence

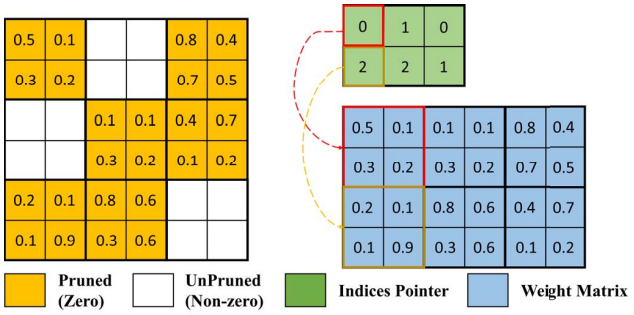

Fig. 3. Illustrating the detailed pruning workflow of Column Balanced Block-wise Weight Pruning with 0.33 pruning rate

simplifying the ReRAM hardware design. Assuming that the weight matrix W has the shape $n \times m$ and setting the pruning block size to be $a \times c$, the weight matrix can be efficiently decomposed into k sub-matrices where $k = n/c$. For each sub-matrices W_k, we can divide it into l sub-matrices where $l = m/a$. Finally the weight matrix W is separated into $k \times l$ blocks. For each W_k, we set the values in the block with the lowest l_2 norm to be zeros. After pruning each W_k, the sub-matrices $W_1, W_2, W_3, ..., W_k$ can be concatenated horizontally to compose the pruned matrix W_{pruned}. Compared with other structural or nonstructural pruning algorithms, column balanced block-wise weight pruning algorithm greatly improves the regularity of the weight matrix and provides high sparsity.

C. ReRAM Customized Weight Quantization

After enforcing the column balanced block-wise weight pruning algorithm, we have obtained semi-structural pruned weight matrices with 32-bit (or 64-bit) precision. Compared to the models with original weight matrices, the current models have the potential to provide a much superior performance of GCN inference due to structural and clustered zeros. Until now, there is only one problem that needs to be solved before the optimized GCN models can be efficiently mapped onto ReRAM crossbars – the models need to be quantized with an acceptable loss of accuracy, otherwise, they cannot be mapped and implemented on ReRAM. Fortunately, researchers have proposed many efficient quantization methods for DNNs. Among them, ADMM-based quantization has been demonstrated to be highly efficient. Hence, in this work, we select ADMM-based model quantization methods that are normally used in DNNs, map them to GNN acceleration, and demonstrate their efficiency with GNNs.

Similar to the weight pruning process, we build up a GCN model with N layers and assume that the weight matrix for the i^{th} layer is W_i and the bias for the i^{th} layer is b_i. The loss function for an N-layer GCN model can be written in the following form: $f(\{W_i\}_{i=1}^N, \{b_i\}_{i=1}^N)$. Then the problem of weight pruning can be written as an optimization problem:

$$\begin{aligned} &\underset{\{W_i,b_i\}}{minimize} \quad \mathscr{L}(\{W_i\}_{i=1}^N, \{b_i\}_{i=1}^N) \\ &subject \quad to \quad W_i \in C_i, i = 1, ..., N, \end{aligned} \tag{9}$$

where the C_i is the constraint set for the weight matrix W_i at the i^{th} layer. In our work, we apply fixed $\mathscr{K}$ bit pruning algorithms which means the values in the quantized matrices should satisfy the requirement $W_{i,j} \in Q$, where $Q = \{-2^{k-1}+1, -2^{k-1}+2, ..., 2^{k-1}-1\}$. Therefore, different from the previous section, the constraint set S_i here is that all the values in weight matrices should be mapped to the possible value set Q. Using the same algorithm for decomposing non-differential problems, the optimization problem (i.g. formula 3) in the weight quantization procedure can also be divided into 2 subproblems (e.g. formula 6 and formula 7) using different constraint sets mentioned above. Then, we can conclude the optimal solution to subproblem 2 is to set every element's value to the closest possible quantized value and then feed them into subproblem 1, where we can efficiently use the gradient descent algorithm(e.g. Adam) to reach the global optimization iteratively. Finally, the weight matrices' values will be mapped to the desired bit width while preventing the accuracy from dropping significantly.

IV. HARDWARE FRAMEWORK

In this section, we introduce the architecture, mapping mechanism, and the dataflow of the ReRAM-based GCN mixed-signal accelerator. We explain how CODG-ReRAM leverages the proposed software optimizations to build a more efficient hardware design along with a more systematic mapping solution.

A. Hardware Architecture

Due to irregular non-zero distribution in the weight and feature matrices, various demeanour of unbalanced workloads, different characteristics of the combination and aggregation phases, massive vector-matrix multiplication operations, and especially the huge and irregular off-chip data movement, conventional DNN [33], [47], [48] and GCN accelerators [16], [44], [46] fall short of providing satisfactory speedups. Recently a few PIM designs such as PIM-GCN [10], PAS-GCN [45], and REFLIP [22] address the aforementioned issues; however, there is still vast space for improvement. The reason is that the current PIM-based GNN works are almost oblivious to algorithmic optimizations and try to address the current issues with hardware-only procedures, which incurs considerable hardware overheads.

As Figure 1 shows, the mixed-signal GCN accelerator entails several tiles, which are connected to each other with an on-chip mesh network. Each analog tile contains multiple Multiply-Accumulate Units (MACs), three different eDRAMs that hold values as crossbar inputs, crossbar weights, and intermediate results as crossbar outputs. It also entails shift-and-add units, activation, and pooling functions that are connected with a shared bus. MAC units are made up of DACs, ReRAM crossbars, ADCs, and shift-and-add units. The CODG-ReRAM utilizes voltage/buffer and the current reducer techniques recommended in [22] to save more power and area due to the reduced overheads of DAC and ADCs.

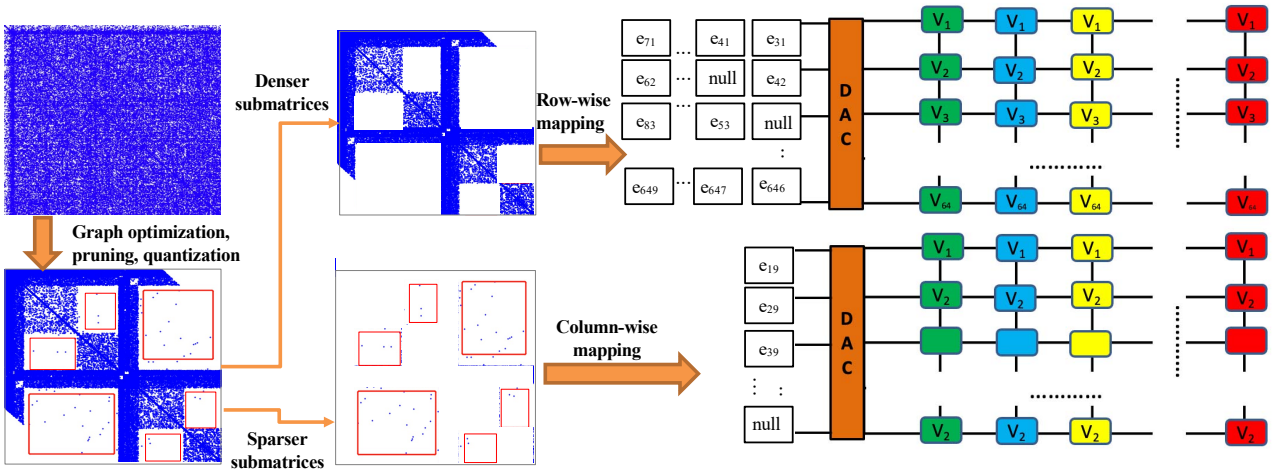

Fig. 4. The Hybrid mapping schemes in CoDG-ReRAM.

B. Mapping Procedure

As combination and aggregation phases in GCNs have different demeanors, they have to be differently treated to achieve high performance. To do so, we employ the same MVM engine of crossbars but utilize different mapping mechanisms to handle combination and aggregation procedures. Note that the proposed method is different from the one used in GCOD that utilizes two-pronged engines for sparse and dense workloads in combination and aggregation. Like REFLIP [22], the *eDRAM crossbar* entails weight parameters and vertex features in combination and aggregation phases, respectively. Similarly, *eDRAM Input* holds vertex features and edge data during combination and aggregation episodes. By so doing, it is ensured that the hardware utilization is increased considerably without imposing different engines with significant overheads to handle these two procedures separately.

Inspired by REFLIP [22], we employ a SIMT model and parallelize workloads of a GraphCONV layer among different tiles to enjoy inter- and intra-vertex data parallelism per layer. The reason is that, unlike DNN engines, GCN accelerators need to store both weights and graph data; however, they have limited layers where the load intensity per layer is usually high.

To map the parameters into crossbars, there are two distinct ways: row-wise and column-wise. In the row-wise solution, each node is loaded into crossbars only once and the edge data are fed into crossbars in a row-wise manner. Therefore, there is a chance that input edges become null. As a result, this method suffers from feeding zero bits that do not contribute to the final results [22], [48]. This happens especially frequently for the sparser parts of the adjacency matrices. On the other hand, in the column-wise mapping procedure, each node is normally processed multiple times in crossbars and null edges are usually abstained from being fed into crossbars. The column-wise solution removes those zero-valued edges considerably, however, it incurs significant parameter overheads [22].

To overcome the problems of both row-wise and column-wise approaches, REFLIP recommends a hybrid approach where row-wise schemes are adopted for vertices with a high degree and column-wise ones are employed for vertices with lower degrees. However, REFLIP employs an ad-hoc approach to find a threshold and identifies high-degree and low-degree vertices, which may lead to low hardware utilization especially when processing vertices that are close to the threshold.

Unlike REFLIP, CoDG-ReRAM employs sparsification and polarization of adjacency matrices in the algorithm optimization to identify the proper mapping mechanisms. Our results indicate that 27%, 32%, and 25% of Cora, Pubmed, and, CiteSeer adjacency matrices lead to denser sub-matrices, which can be handled with the row-wise mapping scheme; while the rest are sparser and are steered by the column-wise approach. By so doing, in a more systematic way, CODG-ReRAM increases resource utilization and computational efficiency of ReRAM in accelerating GCNs.

The *Crossbar Mapper* takes the responsibility to control this procedure during transferring parameters in combination and aggregation stages. The mapper also decides which partial results should be added up to produce the correct results. Graph optimization techniques lead to different classes/subgraphs with different sizes as shown in Figure 4. *Crossbar Mapper* estimates the workload sizes of dense subgraphs and tries to distribute them among the row-wise crossbars evenly to create better load balancing. This makes the mapping mechanism more complex but brings better load balancing and higher system utilization. It is noteworthy to mention that graph optimization, weight pruning, and quantization mechanisms reduce to workloads considerably and hence less data needs to be mapped in crossbars.

Figure 4 illustrates the concept of CoDG-ReRAM mapping procedure. The left-side visualizes the adjacency matrices after applying the proposed algorithmic optimizations, while the

right-side manifests the hybrid mapping mechanism.

Like previous works [10], [22], [51], [51], we use Compress Sparse Row (CSR) and Compress Sparse Column (CSC) formats for high- and low-degree vertices to maintain the edges [10], [22]. We can then ensure sequential off-chip access to edge data and reduce irregular and redundant accesses. Due to the structure of the matrices in GCNs, we prioritize combination over aggregation as it demands less computation and converts both multiplications in $A(X \times W)$ to MVMs that can be handled by the analog engine of the crossbar very efficiently.

V. Experiments

In this section, we first evaluate the proposed software optimization algorithms in terms of model accuracy and data sparsity achieved after graph optimization, column balanced block-wise weight pruning, and ReRAM customized weight quantization are applied on different real-world and widely-evaluated graph datasets. We then evaluate the efficiency of the proposed ReRAM hardware by comparing the latency and energy consumption of the CoDG-ReRAM and SOTA GNN accelerators.

In particular, our evaluation uses a 2-layer GCN model consisting of 16 hidden units based on three citation graph datasets (i.e. Cora, CiteSeer, and Pubmed). The algorithm evaluation compares GNN models with different combinations of optimizations with the raw baseline model. The hardware evaluation results are based on the optimized GNN models after all optimizations are applied. The graph topology optimization includes both structural optimization that is already used in GCoD and negative-edge pruning that is newly added in this work. For Column Balanced Block-wise Weight Pruning, the pruning ratio is set to be 0.3. For ReRAM Customized Weight Quantization, targeting bit widths are set to be 2, 4, and 8. All models are trained on a 4 $\times$ NVIDIA 1080ti GPU server using Pytorch-Geometric (PyG) framework.

We implement an in-house simulator to get the overhead of the proposed ReRAM architecture and the baselines. The tool employs Cacti [5], NVSIM [15], and PIM primitives library [40]. The results of software optimizations are back annotated to the hardware simulator. We are able to run the tool in the mode of the design space exploration to get the area and power of buffer, ReRAM cells, read and write latency and energy, and size, power, and area of the crossbars. We can run the tool in simulation mode to get execution time, energy and throughput. To have a fair comparison with REFLIP, we use the same parameters as REFLIP [22].

A. Algorithm Evaluation

As table I shows, our novel optimization algorithms can greatly maintain high accuracy while applying structural graph topology optimization and model compression algorithms to the GCN models. Compared to the original model, we discover that with our graph topology optimization algorithm, even though the graph has been reconstructed to match the ReRAM property, it can still achieve superior accuracy which is even higher than the raw graphs with no pruning applied for some datasets. The accuracy improvement is from the proposed negative-edge pruning. Table II further demonstrates this phenomenon that cutting down the negative edges can effectively enhance the accuracy by preventing the negative edges from wrongly contributing to the classification tasks by comparing our final models with the ones optimized by GCoD and the raw model. Besides the high accuracy, the proposed graph topology optimization technique also significantly reduces the number of edges compared to both GCoD and raw models, which benefits the computation of GCNs with not only ReRAM but also other devices. Moreover, from the table I we can also observe that the Column Balanced Block-Wise weight pruning leads to accuracy degradation due to the constraints from block pruning pattern which greatly improves the regularity of the weight matrices. Fortunately, the ADMM-based weight quantization algorithm not only helps us reduce the data width but also effectively rescues the accuracy from the unavoidable dilemma.

TABLE I
THE EXPERIMENTAL RESULTS OF EACH STEP

	Accuracy (%)		
Methods	Cora	Pubmed	CiteSeer
Vanilla	81.1	79.1	70.2
(+) Graph Optimization	80.0	78.7	71.5
(+) CBB Weight Pruning	76.6	78.1	60.5
(+) Weight Quantization			
2-bit	77.8	79.5	67.8
4-bit	79.8	78.6	70.5
8-bit	79.5	78.6	69.9

TABLE II
COMPARISON AMONG VANILLA, GRAPH OPTIMIZATIONS WITH AND WITHOUT NEGATIVE EDGE PRUNING

	Vanilla	Without	With
Dataset		Cora	
Number of edges	10556	9500(-10.0%)	8178(-22.5%)
Accuracy(%)	—	79.6	80.0(+0.4)
Dataset		Pubmed	
Number of edges	88648	79782(-10.0%)	69090(-22.0%)
Accuracy(%)	—	79.4	78.7(-0.7)
Dataset		CiteSeer	
Number of edges	9104	8192(-10.0%)	7476(-17.8%)
Accuracy(%)	—	70.3	71.5(+1.2)

B. Hardware Evaluation

Table III compares the power and area results of CoDG-ReRAM with REFLIP [22] and HyGCN [44]. Due to graph optimization, pruning, and quantization (8bits vs 32bits), the number of required crossbars and also the total number of required tiles are significantly reduced. We utilize a 64 $\times$ 64 crossbar array. Each MAC unit comprises 16 ReRAM

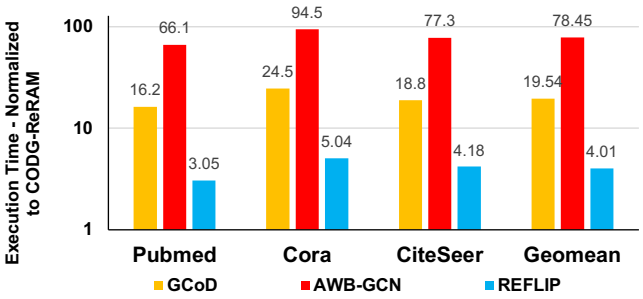

Fig. 5. Comparing execution time of CoDG-ReRAM with other baselines

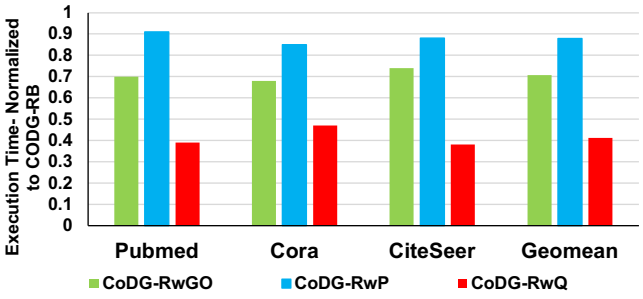

Fig. 6. The impact of each algorithmic optimization on execution time

crossbar arrays, while we use 8 MAC units per tile and 36 tiles. Accordingly, compared to REFLIP (HyGCN), the area and power of the proposed accelerator are reduced by 35% (39%) and 28% (37%), respectively.

Figure 5 compares the execution time of the CoDG-ReRAM with GCOD [46], AWB-GCN [16], and REFLIP [22]. The figure shows normalized results with Y axis using a logarithmic scale and CoDG-ReRAM as 1 (hence not illustrated in the figure). As the results demonstrate, CODG-ReRAM outperforms these baselines by 19.5×, 78.4×, and 4×, respectively. Since CoDG-ReRAM performs in-situ computation, it reaches superior performance compared with GCoD and AWB-GCN. Due to proposed algorithm optimizations and more efficient mapping solutions, we are able to outclass the REFLIF baseline as well. Figure 6 shows the impact of assorted algorithm optimizations on the execution time. As it can be observed, due to aggressive high-quality quantization, it has the most impact on reducing execution time. Moreover, the graph topology optimization plays a more important role than pruning in the execution time reduction while applying the pruning rate of 0.3. Those algorithm/hardware optimizations also contribute to reducing energy consumption as Figure 7 illustrates. Compared with REFLIP, AWB-GCN, and GCOD, the energy improvements are 5×, 196.6×, and 91.6×, respectively.

TABLE III
COMPARING POWER AND AREA OF CODG-RERAM WITH REFLIP

REFLIP [22]		HyGCN [44]		CoDG-ReRAM	
Power(W)	Area(mm^2)	Power(mW)	Area(mm^2)	Power(mW)	Area(mm^2)
47.38	43.63	54.66	46.2	34.22	28.17

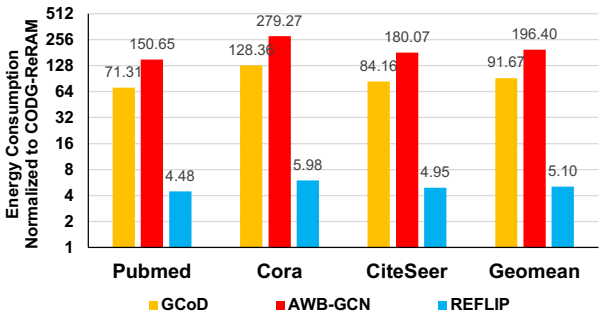

Fig. 7. Comparing energy consumption of CoDG-ReRAM with baselines

VI. RERAM-BASED GNN ACCELERATOR CHALLENGES

In this section, we talk about the challenges that need to be addressed to have efficient ReRAM-based GNN accelerators. Several of them will be also valid for other machine learning accelerators.

A. Algorithm Perspective

Efficient full-stack software is a key that enables us to get the maximum benefits of underline hardware. This is a missing component in many GNN accelerator designs. This is helpful as instead of addressing challenges at the hardware-level with a significant cost, we can address them at the algorithm-level [46]–[48] with orders of magnitude lower cost as we showed in this paper.

The main components in a software stack are the programming language and its compiler [11], [23], [27], [34]. We need a programming language that allows designers to define different requirements of the accelerators such as various types of memories, the graph model, optimizations that need to be employed, the location and the order in which these optimizations need to be applied, and etc [23]. Having an efficient compiler that automatically maps the proposed GCN dataflow into underline architecture and fills the gap between high-level accelerator description and low-level execution engine is also highly desired.

B. Hardware Perspective

Although ReRAM-based GCN accelerators show promising results and capabilities, there are still problems that prevent them to be extensively used in the commercial products. Endurance is a fundamental problem of ReRAM memories. It causes to have small crossbar array sizes (i.e., the maximum size is 512×512 [43]), which imposes crossbar to crossbar communication cost. However, in recent years, there is a good progress in improving ReRAM endurance [36]. In addition, memory cells can only be written a few number of times before they stop working. This can be mitigated by utilizing ReRAM devices that endure more writes or employing architecture techniques such as wear leveling [7], [38] that distribute writes across all memory locations evenly. Another problem is the ReRAM sneak currents, which can be alleviated using techniques such as using access transistors (e.g., two-terminal selector) and self-rectifying RRAM [24], [25].

Besides the aforementioned problems, we also summarize other three worth mentioning challenges in designing and deploying ReRAM solutions for GCN acceleration. The first is the size of ADC-size that contributes significantly to the area and power consumption (58% of tile power and 31% of tile area [33]). Recently researchers propose various solutions to reduce the ADC size by leveraging hardware and software techniques [4], [47], [48]. The second problem is that the engine of these accelerators works in analog regime, which is sensitive to noise and imperfections. Techniques such as noise-aware training [29] and hybrid acceleration where important weights are handled by digital cores are proposed to mitigate this problem [8], [14]. One more challenge is the writing cost in non-volatile memories. Unlike volatile memories, the energy and latency associated with the non-volatile ones are high. This is also why many architectures unfold all the tiles and write all the weight parameters of all layers into ReRAM crossbars at the beginning. Although it helps increase the parallelism and throughput, it incurs hardware overhead.

We hope by having a full software stack along with addressing challenges associated with the hardware counterpart, mixed-signal accelerators can be commercialized and employed extensively.

VII. Conclusion and Future Work

This paper proposes, CoDG-ReRAM, a ReRAM-based GCN acceleration framework with algorithm-hardware co-design. On the algorithm side, CoDG-ReRAM is equipped with three optimizations as follows: (1) a graph topology optimization method with negative edge pruning to regularize graph topology, improve the model accuracy, and reduce the computation demand and the number of edges of graphs; (2) Column Balanced Block-wise Weight Pruning to sparsify weight matrices with the semi-structural sparse pattern; and (3) ReRAM customized Weight Quantization to provide high-quality quantization so that the optimized model can be mapped onto ReRAM devices. Overall, the newly proposed model optimization pipeline delivers more regular sparse matrices with higher sparsity and lower bit-widths while maintaining high accuracy. On the hardware side, we take advantage of the algorithm optimization results to provide a more systematic mapping scheme, reduce latency and increase the computation efficiency of the ReRAM-based GCN accelerator. Conclusively, CoDG-ReRAM provides 4× speedups over SOTA ReRAM-based GNN accelerators with negligible loss of accuracy.

In future work, we will further explore the scalability of CoDG-ReRAM by examining it with larger graphs and sizes of crossbars. Besides, the proposed algorithm optimizations especially the graph topology optimization insert extra workload imbalance to the graphs which limits the performance of our work. Therefore, it is worthy to study how to fundamentally address the graph-level workload imbalance issue within ReRAM crossbars. Last, CoDG-ReRAM focuses on static graphs. In the future, we will try to extend its support to evolving graphs and spatio-temporal graph models.

References

[1] A. I. Arka, J. R. Doppa, P. P. Pande, B. K. Joardar, and K. Chakrabarty, "Regraphx: Noc-enabled 3d heterogeneous reram architecture for training graph neural networks," in *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2021, pp. 1667–1672.

[2] A. I. Arka, B. K. Joardar, J. R. Doppa, P. P. Pande, and K. Chakrabarty, "Dare: Droplayer-aware manycore reram architecture for training graph neural networks," in *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*. IEEE, 2021, pp. 1–9.

[3] A. Auten, M. Tomei, and R. Kumar, "Hardware acceleration of graph neural networks," in *Proceedings of the 57th ACM/EDAC/IEEE Design Automation Conference*, ser. DAC '20. IEEE Press, 2020.

[4] A. Azamat, F. Asim, and J. Lee, "Quarry: Quantization-based adc reduction for reram-based deep neural network accelerators," in *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*. IEEE, 2021, pp. 1–7.

[5] R. Balasubramonian, A. B. Kahng, N. Muralimanohar, A. Shafiee, and V. Srinivas, "Cacti 7: New tools for interconnect exploration in innovative off-chip memories," *ACM Transactions on Architecture and Code Optimization (TACO)*, vol. 14, no. 2, pp. 1–25, 2017.

[6] F. M. Bayat, M. Prezioso, B. Chakrabarti, H. Nili, I. Kataeva, and D. Strukov, "Implementation of multilayer perceptron network with highly uniform passive memristive crossbar circuits," *Nature communications*, vol. 9, no. 1, pp. 1–7, 2018.

[7] P. Behnam, A. P. Chowdhury, and M. N. Bojnordi, "R-cache: A highly set-associative in-package cache using memristive arrays," in *2018 IEEE 36th International Conference on Computer Design (ICCD)*. IEEE, 2018, pp. 423–430.

[8] P. Behnam, U. Kamal, and S. Mukhopadhyay, "An algorithm-hardware co-design framework to overcome imperfections of mixed-signal dnn accelerators," *arXiv preprint arXiv:2208.13896*, 2022.

[9] F. Cai, J. M. Correll, S. H. Lee, Y. Lim, V. Bothra, Z. Zhang, M. P. Flynn, and W. D. Lu, "A fully integrated reprogrammable memristor–cmos system for efficient multiply–accumulate operations," *Nature Electronics*, vol. 2, no. 7, pp. 290–299, 2019.

[10] N. Challapalle, K. Swaminathan, N. Chandramoorthy, and V. Narayanan, "Crossbar based processing in memory accelerator architecture for graph convolutional networks," in *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, 2021, pp. 1–9.

[11] P. Chatarasi, H. Kwon, A. Parashar, M. Pellauer, T. Krishna, and V. Sarkar, "Marvel: a data-centric approach for mapping deep learning operators on spatial accelerators," *ACM Transactions on Architecture and Code Optimization (TACO)*, vol. 19, no. 1, pp. 1–26, 2021.

[12] X. Chen, Y. Wang, X. Xie, X. Hu, A. Basak, L. Liang, M. Yan, L. Deng, Y. Ding, Z. Du, and Y. Xie, "Rubik: A hierarchical architecture for efficient graph neural network training," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 4, pp. 936–949, 2022.

[13] Y. Chen, "Reram: History, status, and future," *IEEE Transactions on Electron Devices*, vol. 67, no. 4, pp. 1420–1433, 2020.

[14] S. Dash and S. Mukhopadhyay, "Hessian-driven unequal protection of dnn parameters for robust inference," in *2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*. IEEE, 2020, pp. 1–9.

[15] X. Dong, C. Xu, Y. Xie, and N. P. Jouppi, "Nvsim: A circuit-level performance, energy, and area model for emerging nonvolatile memory," *IEEE T-CAD*, 2012.

[16] T. Geng, A. Li, R. Shi, C. Wu, T. Wang, Y. Li, P. Haghi, A. Tumeo, S. Che, S. Reinhardt, and M. C. Herbordt, "Awb-gcn: A graph convolutional network accelerator with runtime workload rebalancing," in *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2020, pp. 922–936.

[17] T. Geng, C. Wu, C. Tan, C. Xie, A. Guo, P. Haghi, S. Y. He, J. Li, M. Herbordt, and A. Li, "A survey: Handling irregularities in neural network acceleration with fpgas," in *2021 IEEE High Performance Extreme Computing Conference (HPEC)*. IEEE, 2021, pp. 1–8.

[18] T. Geng, C. Wu, Y. Zhang, C. Tan, C. Xie, H. You, M. Herbordt, Y. Lin, and A. Li, "I-gcn: A graph convolutional network accelerator with runtime locality enhancement through islandization," in *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 1051–1063.

[19] F. T. Hady, A. Foong, B. Veal, and D. Williams, "Platform storage performance with 3d xpoint technology," *Proceedings of the IEEE*, vol. 105, no. 9, pp. 1822–1833, 2017.

[20] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17. Red Hook, NY, USA: Curran Associates Inc., 2017, p. 1025–1035.

[21] J. Handy, "Understanding the intel/micron 3d xpoint memory," *Proc. SDC*, vol. 68, 2015.

[22] Y. Huang, L. Zheng, P. Yao, Q. Wang, X. Liao, H. Jin, and J. Xue, "Accelerating graph convolutional networks using crossbar-based processing-in-memory architectures," in *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2022, pp. 1029–1042.

[23] Y. Ikarashi, G. L. Bernstein, A. Reinking, H. Genc, and J. Ragan-Kelley, "Exocompilation for productive programming of hardware accelerators," in *Proceedings of the 43rd ACM SIGPLAN International Conference on Programming Language Design and Implementation*, 2022, pp. 703–718.

[24] S. H. Jo, T. Kumar, S. Narayanan, and H. Nazarian, "Cross-point resistive ram based on field-assisted superlinear threshold selector," *IEEE Transactions on Electron Devices*, vol. 62, no. 11, pp. 3477–3481, 2015.

[25] T.-H. Kim, B. Song, I.-J. Jung, and S.-O. Jung, "A sneak current compensation scheme with offset cancellation sensing circuit for reram-based cross-point memory array," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 69, no. 4, pp. 1583–1594, 2021.

[26] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *International Conference on Learning Representations*, 2017.

[27] D. Koeplinger, M. Feldman, R. Prabhakar, Y. Zhang, S. Hadjis, R. Fiszel, T. Zhao, L. Nardi, A. Pedram, C. Kozyrakis *et al.*, "Spatial: A language and compiler for application accelerators," in *Proceedings of the 39th ACM SIGPLAN Conference on Programming Language Design and Implementation*, 2018, pp. 296–311.

[28] S. Liang, Y. Wang, C. Liu, L. He, H. Li, D. Xu, and X. Li, "Engn: A high-throughput and energy-efficient accelerator for large graph neural networks," *IEEE Transactions on Computers*, vol. 70, pp. 1511–1525, 2021.

[29] Y. Long, X. She, and S. Mukhopadhyay, "Design of reliable dnn accelerator with un-reliable reram," in *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2019, pp. 1769–1774.

[30] H. Peng, S. Huang, S. Chen, B. Li, T. Geng, A. Li, W. Jiang, W. Wen, J. Bi, H. Liu, and C. Ding, "A length adaptive algorithm-hardware co-design of transformer on fpga through sparse attention and dynamic pipelining," in *2022 59th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 2022.

[31] H. Peng, S. Huang, T. Geng, A. Li, W. Jiang, H. Liu, S. Wang, and C. Ding, "Accelerating transformer-based deep learning models on fpgas using column balanced block pruning," in *2021 22nd International Symposium on Quality Electronic Design (ISQED)*, 2021, pp. 142–148.

[32] S. Rashidi, M. Denton, S. Sridharan, S. Srinivasan, A. Suresh, J. Nie, and T. Krishna, "Enabling compute-communication overlap in distributed deep learning training platforms," in *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2021, pp. 540–553.

[33] A. Shafiee, A. Nag, N. Muralimanohar, R. Balasubramonian, J. P. Strachan, M. Hu, R. S. Williams, and V. Srikumar, "Isaac: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars," *ACM SIGARCH Computer Architecture News*, vol. 44, no. 3, pp. 14–26, 2016.

[34] P. Srivastava, M. Kang, S. K. Gonugondla, S. Lim, J. Choi, V. Adve, N. S. Kim, and N. Shanbhag, "Promise: An end-to-end design of a programmable mixed-signal accelerator for machine-learning algorithms," in *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2018, pp. 43–56.

[35] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, "The missing memristor found," *nature*, vol. 453, no. 7191, pp. 80–83, 2008.

[36] Z. Swaidan, R. Kanj, J. El Hajj, E. Saad, and F. Kurdahi, "Rram endurance and retention: Challenges, opportunities and implications on reliable design," in *2019 26th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*. IEEE, 2019, pp. 402–405.

[37] A. Thomas, "Memristor-based neural networks," *Journal of Physics D: Applied Physics*, vol. 46, no. 9, p. 093001, feb 2013.

[38] S. E. Wells, "Method for wear leveling in a flash eeprom memory," Aug. 23 1994, uS Patent 5,341,339.

[39] H.-S. P. Wong, H.-Y. Lee, S. Yu, Y.-S. Chen, Y. Wu, P.-S. Chen, B. Lee, F. T. Chen, and M.-J. Tsai, "Metal–oxide rram," *Proceedings of the IEEE*, vol. 100, no. 6, pp. 1951–1970, 2012.

[40] Y. N. Wu, V. Sze, and J. S. Emer, "An architecture-level energy and area estimator for processing-in-memory accelerator designs," in *2020 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. IEEE, 2020, pp. 116–118.

[41] L. Xia, B. Li, T. Tang, P. Gu, X. Yin, W. Huangfu, P.-Y. Chen, S. Yu, Y. Cao, Y. Wang, Y. Xie, and H. Yang, "Mnsim: Simulation platform for memristor-based neuromorphic computing system," in *2016 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2016, pp. 469–474.

[42] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" in *International Conference on Learning Representations*, 2019.

[43] C.-X. Xue, T.-Y. Huang, J.-S. Liu, T.-W. Chang, H.-Y. Kao, J.-H. Wang, T.-W. Liu, S.-Y. Wei, S.-P. Huang, W.-C. Wei *et al.*, "15.4 a 22nm 2mb reram compute-in-memory macro with 121-28tops/w for multibit mac computing for tiny ai edge devices," in *2020 IEEE International Solid-State Circuits Conference-(ISSCC)*. IEEE, 2020, pp. 244–246.

[44] M. Yan, L. Deng, X. Hu, L. Liang, Y. Feng, X. Ye, Z. Zhang, D. Fan, and Y. Xie, "Hygcn: A gcn accelerator with hybrid architecture," in *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 2020, pp. 15–29.

[45] T. Yang, D. Li, F. Ma, Z. Song, Y. Zhao, J. Zhang, F. Liu, and L. Jiang, "Pasgcn: An reram-based pim design for gcn with adaptively sparsified graphs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 1–1, 2022.

[46] H. You, T. Geng, Y. Zhang, A. Li, and Y. Lin, "Gcod: Graph convolutional network acceleration via dedicated algorithm and accelerator co-design," in *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2022, pp. 460–474.

[47] G. Yuan, P. Behnam, Y. Cai, A. Shafiee, J. Fu, Z. Liao, Z. Li, X. Ma, J. Deng, J. Wang *et al.*, "Tinyadc: Peripheral circuit-aware weight pruning framework for mixed-signal dnn accelerators," in *(Best Paper Nomination) (DATE'21) Design Automation & Test in Europe Conference & Exhibition*, 2021.

[48] G. Yuan, P. Behnam, Z. Li, A. Shafiee, S. Lin, X. Ma, H. Liu, X. Qian, M. N. Bojnordi, Y. Wang, and C. Ding, "Forms: Fine-grained polarized reram-based in-situ computation for mixed-signal dnn accelerator," in *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*, 2021, pp. 265–278.

[49] H. Zeng and V. Prasanna, "Graphact: Accelerating gcn training on cpu-fpga heterogeneous platforms," in *Proceedings of the 2020 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, ser. FPGA '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 255–265.

[50] C. Zhang, T. Geng, A. Guo, J. Tian, M. Herbordt, A. Li, and D. Tao, "H-gcn: A graph convolutional network accelerator on versal acap architecture," *arXiv preprint arXiv:2206.13734*, 2022.

[51] M. Zhang, Y. Wu, Y. Zhuo, X. Qian, C. Huan, and K. Chen, "Wonderland: A novel abstraction-based out-of-core graph processing system," *ACM SIGPLAN Notices*, vol. 53, no. 2, pp. 608–621, 2018.

[52] T. Zhang, S. Ye, K. Zhang, J. Tang, W. Wen, M. Fardad, and Y. Wang, "A systematic dnn weight pruning framework using alternating direction method of multipliers," in *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[53] Y. Zhang, H. You, Y. Fu, T. Geng, A. Li, and Y. Lin, "G-cos: Gnn-accelerator co-search towards both better accuracy and efficiency," in *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*. IEEE, 2021, pp. 1–9.

[54] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph neural networks: A review of methods and applications," *AI Open*, vol. 1, pp. 57–81, 2020.

[55] W. Zhuang, W. Pan, B. Ulrich, J. Lee, L. Stecker, A. Burmaster, D. Evans, S. Hsu, M. Tajiri, A. Shimaoka *et al.*, "Novel colossal magnetoresistive thin film nonvolatile resistance random access memory (rram)," in *Digest. International Electron Devices Meeting,*. IEEE, 2002, pp. 193–196.