Energy-Efficient Machine Learning Acceleration: From Technologies to Circuits and Systems

Chukwufumnanya Ogbogu[1], Madeleine Abernot[2], Corentin Delacour[2], Aida Todri-Sanial[2,3], Sudeep Pasricha[4], Partha Pratim Pande[1].
[1]School of EECS Washington State University, Pullman WA, USA. [2]LIRMM, University of Montpellier, CNRS, France. [3]Eindhoven University of Technology, Netherlands. [4]Dept. of Electrical and Computer Engg., Colorado State University, Fort Collins, Colorado, USA
[1]{c.ogbogu, pande}@wsu.edu, [2]{madeleine.abernot, corentin.delacour}@lirmm.fr, [3]a.todri.sanial@tue.nl, [4]sudeep@colostate.edu.

2023 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED) | 979-8-3503-1175-4/23/$31.00 ©2023 IEEE | DOI: 10.1109/ISLPED58423.2023.10244360

***Abstract*—Advanced computing systems have long been enablers for breakthroughs in Machine Learning (ML) algorithms either through sheer computational power or form-factor miniaturization. However, as ML algorithms become more complex and the size of datasets increase, existing computing platforms are no longer sufficient to bridge the gap between algorithmic innovation and hardware design. With the rising needs of advanced algorithms for large-scale data analysis and data-driven discovery, and significant growth in emerging applications from the edge to the cloud, we need energy-efficient, low-cost, high- performance, and reliable computing systems targeted for these applications. This paper presents the latest developments in oscillatory neural networks, optical computing, and memristive processing-in-memory (PIM) to address the various challenges in designing efficient computing systems specifically targeting ML applications.**

Keywords—optical neural networks, silicon photonics, oscillatory neural networks, ReRAM, GNNs

I. Introduction

Deep machine learning (ML) algorithms are employed in a wide variety of real-world applications, e.g., self-driving cars, medical diagnosis, network security, and industrial automation. Both training and inferencing of these deep ML models are computationally demanding tasks and are typically accomplished on the cloud. However, there is a growing necessity to implement deep learning on edge platforms due to privacy and security concerns, the need for user-specific customization, low latency, and real-time requirements (such as in augmented/virtual reality applications). However, implementing these applications on edge devices is challenging due to area and energy constraints. Addressing this necessitates suitable high-performance and energy efficient hardware support. In this paper, we present the salient features and design challenges with three emerging hardware paradigms, viz., oscillatory analog computation, optical computing, and memristive processing-in-memory (PIM).

Computing with phase dynamics of coupled oscillators enables not only signal voltage amplitude reduction but brings massive parallelism allowing for fast computation with energy efficiency. We believe that analog computing based on coupled oscillatory neural networks might be what is needed for certain ML tasks running on edge devices that have latency and power constraints. Using phase dynamics allows for solving associative memory and combinatorial optimization types of problems with simple circuits that can enable more versatile edge AI functions.

Optical computing has become an attractive substrate for accelerating emerging ML workloads due to the continued miniaturization of silicon photonics devices that also possess compatibility with CMOS fabrication. This paradigm combines light speed latencies for communication with light-speed computational operations, such as matrix vector multiplications and summations, which can significantly improve performance and energy-efficiency when executing various types of ML algorithms.

Resistive random-access memory (ReRAM) based processing-in-memory (PIM) architectures have been used to accelerate deep ML applications such as CNNs, RNNs, GNNs, transformers, etc. ReRAM-based systems are more area-efficient compared to their GPU counterparts and do not require expensive off-chip memory access due to the "in-memory" nature of the ReRAM-based computation. The crossbar structure of ReRAM-based architectures enables efficient Matrix-Vector Multiplication (MVM) operations, which are ubiquitous in modern ML tasks.

II. Computation with Oscillatory Neural Networks

In response to the challenges introduced by cloud computing, current research focuses on enabling ML at the edge, bringing ML capabilities closer to the data source, to reduce latency and minimize energy requirements.

A solution takes inspiration from biology to design neuromorphic computing techniques, like Spiking Neural Networks [1]. In this work, we focus on another promising neuromorphic paradigm with Oscillatory Neural Networks (ONN) [2] inspired by brain oscillations. ONNs are networks of coupled oscillators computing with inherent parallel phase synchronisation of coupled oscillators. Phase computing encodes information in the phase relationship among oscillators. It allows to reduce power consumption by limiting the voltage amplitude. Low power and fast parallel ONN computing makes it attractive for edge AI [3].

Using phase-computing ONNs, information is encoded in the phase relationship among oscillators. For example, for binary information, a logic '0' is encoded with a 0° phase, while a logic '1' is encoded with a 180° phase. ONN computation starts by initializing phases of each oscillator in the network from input data. Then, phases evolve depending on the coupling between oscillators, and stabilizes to a final phase state. The evolution of phases corresponds to the minimization of an intrinsic energy parameter, like in attractor networks [4]. The final phase state gives the network output information. Thus, the coupling among oscillators,

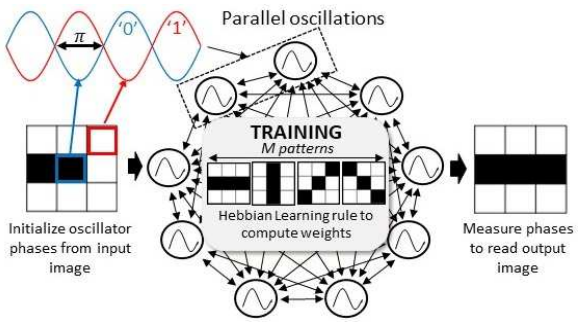


Figure 1. Phase computing with coupled oscillatory neural networks.

979-8-3503-1175-4/23/$31.00 ©2023 IEEE

defined by learning, is the main parameter to solve specific tasks efficiently.

In the current state-of-the-art, ONNs are typically implemented using a fully connected (FC) architecture, utilizing unsupervised learning for auto-associative memory tasks similar to Hopfield networks, as illustrated in Fig. 1. This configuration allows the network to memorize patterns based on its connections and converge to a learned pattern when initialized with a corrupted input. FC-ONNs configured for pattern recognition have found applications in various domains. However, the FC-ONN architecture necessitates a large number of coupling elements, which increases quadratically with the number of neurons. For a network of N neurons, the number of synaptic elements required is given by N(N-1)/2, making large-scale implementation challenging. Additionally, while associative memory tasks are intriguing, the capacity of the network is often limited by unsupervised learning, restricting the number of patterns that can be efficiently learned and retrieved. Therefore, there is a pressing need to explore alternative architectures and applications that can be applied to ONNs.

Recently, researchers have introduced two-layer ONN architectures that incorporate bidirectional and feedforward connections between layers while excluding connections among oscillators within the same layer [5]. These two-layer ONNs have been successfully utilized for image edge detection in the respective studies. Notably, the implementations in these works were digitally executed on FPGA platforms. However, it is important to highlight that achieving a feedforward ONN architecture using analog implementation is challenging, as the default nature of coupling among oscillators is bidirectional.

A. *Analog ONNs and combinatorial optimization*

As ONNs are dynamical systems, continuous-time analog architectures that follow the "let physics compute" principle, they are well suited for efficient ONN implementation. Analog ONNs naturally perform gradient descent of their energy function through time, without any algorithm and clock. This intrinsic minimization mechanism is appealing for many optimization tasks, especially for combinatorial optimization problems that are often Nondeterministic Polynomial time hard (NP-hard) and require the exploration of a solution space that scales exponentially. Thus, accelerating the search is crucial for large instances. ONN high parallelism is particularly promising for this task and can provide up to four orders of magnitude runtime improvement compared to CPUs [6]. As a matter of fact, all NP-complete problems can be reduced to finding the ground states of a corresponding Ising Hamiltonian, which in principle can be mapped to the ONN energy by relaxing the initial discrete variables to continuous phases [7].

In practice, various CMOS ONN solvers have been demonstrated [8] [9], the largest being a 1968-node ONN based on ring oscillators oscillating at 1 GHz [10]. Other approaches focus on reducing the oscillator and synapse footprint using novel devices such as magnetic tunnel junctions (MTJ), memristors, or transition metal oxide devices (TMO). Vanadium dioxide (VO2) is an example of TMO device that has a hysteresis switching behaviour and produces oscillations at room temperature [11]. When scaled down to submicron dimensions, VO2-based oscillators are expected to reach very low energy consumptions down to 10 fJ at 100 MHz [3], but yet remains to be demonstrated.

Just like any other neural network, implementing densely connected ONNs is challenging. Most of current hardware limit the synaptic connections to nearest neighbours which overall necessitate more physical neurons than the initial network after mapping [10]. Moreover, finding the optimal network mapping to the hardware can cause an important computational overhead compared to the initial problem to solve [12]. Another interesting ONN capability is to emulate all-to-all connectivity using the injection of a modulated external signal [13]. This approach replaces the connectivity complexity by a modulation scheme which is also challenging to design, but could advantageously be performed off-chip.

B. *Digital ONN implementation for edge applications*

With all challenges related to analog ONN implementations, researchers also developed digital-based ONN implementations. [8] introduced a mixed signal implementation using digital oscillators with additional analog components, and more recently, a fully-digital ONN was implemented on FPGA to demonstrate ONN computing paradigm for AI edge applications [14].

The ONN on FPGA first demonstrated interesting properties to solve pattern recognition tasks based on the fully-connected recurrent architecture from Hopfield. In particular, a 10x6 fully-connected ONN was trained with images of digits and implemented inside the FPGA performing real-time pattern recognition from a camera stream [14]. An additional system architecture has also been proposed to allow on-chip unsupervised learning with the digital ONN design, being able to solve pattern recognition with on-chip Hebbian or Storkey learning implementation. Then, two cascaded fully-connected ONNs were used to perform obstacle avoidance on mobile robots. A first ONN detects obstacles from proximity sensor data, and a second ONN uses the detected obstacles to define a novel direction allowing real-time obstacle avoidance from proximity sensors dataflow [15].

Later, the digital ONN implementation has been adapted to fit with layered architectures, considering bidirectional or feedforward synaptic connections, and was applied to the image edge detection application. Finally, recently, the ONN for image edge detection was proposed as an accelerator of the SIFT feature detection algorithm [16].

Comparable to analog ONN, dense digital ONN implementation is challenging. The recently introduced layered architectures permits the implementation of larger scale ONNs, the investigation of novel general learning methods, and the exploration of novel edge applications.

III. Optical Computing for ML Acceleration

The gradual plateauing of improvement in performance-per-watt with electronic ML accelerators in recent years has led to a widespread and urgent search for alternative computing technologies. One very promising technology that has emerged from this search is silicon photonics (SiPh), which involves the use of light to transmit data [17] [18]. Increasingly, SiPh components are being used to perform computations with data. Optical computing systems built with SiPh components are not constrained by the limits of

electron movement that hamper today's electronic platforms. Instead, they rely on speedy photons for information transfer and manipulation. Such systems have the potential to revolutionize data-hungry ML application acceleration by providing ultra-fast data transfers and processing (e.g., multiply and accumulate, or Fast Fourier Transform operations) in the analog optical domain while consuming much less energy than traditional electronic computing [19].

A. Photonic Devices and Circuits

Non-coherent optical computing involves the use of multiple wavelengths that can perform parallel data transfers as well as operations such as matrix-vector operations concurrently [20]. Both data transfers and computation require the use of various optoelectronic components that can be fabricated in CMOS-compatible foundries.

Lasers (off-chip or on-chip) are used to generate optical signals necessary for computation and communication in optical circuits. *SiPh* waveguides made of a core (Si) and a cladding (SiO_2) material provide high-refractive-index contrast and allow for total internal reflection and hence optical signal confinement and transmission. Microring Resonators (MRs) involve a ring-shaped waveguide that is designed to be sensitive to a particular wavelength, referred to as the MR's resonant wavelength. MRs are used for modulation and filtering data carrying wavelengths from optical interconnects. These same devices can also be used to perform computation via amplitude modulation on different wavelengths using a tuning circuit (which can be electro-optic (EO) or thermo-optic (TO) [20]) that modifies the operational characteristics of the MRs. As an example, two parameters in an ML model that need to be multiplied can be deployed on MRs with the same resonant wavelength, along the same input waveguide. The first MR modulates the signal to represent the first parameter. The parameter specific amplitude modulation with the second MR results in a multiplication operation [21]. Lastly, photodetectors are used to detect optical signals and convert them to electrical signals.

B. Low Power Optical Computing Design Challenges

To achieve energy-efficient and low-power non-coherent optical computing, several challenges must be addressed. Fabrication-process variations (FPVs) and thermal variations create unintended changes in fabricated optoelectronic components, requiring power-hungry tuning circuits to mitigate their effects [22]. Tuning circuits also have high energy overheads due to their high latencies of operation. Thermal crosstalk arising due to such tuning circuits further reduces MR stability and increases energy consumption. Crosstalk between multiple wavelengths requires increasing laser power to overcome degradation in signal-to-noise ratio (SNR) [23]. Lastly, electro-optic conversions require power hungry analog-to-digital converters (ADCs) and digital-to-analog converters (DACs). All of these factors must be addressed during the design of an optical computing accelerator for ML.

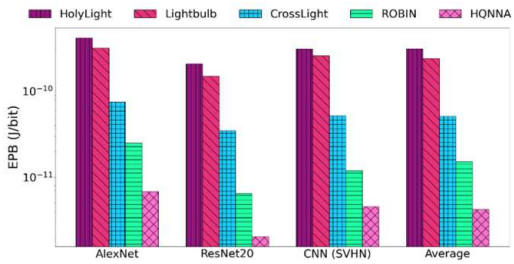


Figure 2: EPB for CNN models, across optical-domain accelerators

C. Cross-Layer Design for Optical Computing

In [24], we proposed *CrossLight*, the first cross-layer optimized optical computing accelerator for ML workloads that included CNNs and DNNs.

At the device level, we fabricated a 1.5×0.6 mm^2 chip with high-resolution Electron Beam (EBeam) lithography and performed a comprehensive design-space exploration of MRs to compensate for FPVs while improving MR device insertion loss and Q-factor. We found that in an MR design of any radii and gap, when the input waveguide is 400 nm wide and the ring waveguide is 800 nm wide at room temperature (300 K), the undesired resonant wavelength shift due to FPVs can be reduced from 7.1 to 2.1 nm (70% reduction). This is a significant result, as these engineered MRs require less compensation for FPV-induced resonant wavelength shifts, which reduces the power consumption of architectures using such MRs.

At the circuit level, we attempted to reduce the reliance on thermo-optic (TO) tuning circuits that create thermal crosstalk. We developed a hybrid tuning circuit where both thermo-optic (TO) and electro-optic (EO) tuning are used to compensate for resonant wavelength shifts due to FPVs and thermal variations. The hybrid tuning approach supports faster operation of MRs with fast EO tuning to compensate for small wavelength shifts and, using TO tuning when large wavelength shifts need to be compensated. This significantly reduces the energy overheads associated with MR operation. We further used a commercial 3D heat transport simulation EDA tool for *SiPh* devices (Lumerical HEAT) to determine phase crosstalk ratio to determine optimal inter-MR distance to minimize power consumption in TO microheaters.

At the architecture level, we designed vector dot product (VDP) with banks of MRs for optical matrix multiplication. The VDP units were clustered into two groups: one to support convolution (CONV) layer acceleration and the other to support fully connected (FC) layer acceleration. We focused on these two types of layers as they are the most widely used and consume the most latency and power in computational platforms that execute CNNs and DNNs. We also optimized laser reuse across VDP units, optimized the dimensions and cluster sizes of VDPs, and proposed dataflow mechanisms to map computations to VDP units.

Table 1 shows the energy-per-bit (EPB) and performance-per-watt (in terms of thousands of image frames per second processed, per watt) for *CrossLight* compared to other *SiPh*-based accelerators (HolyLight, DEAP_CNN) and electronic platforms. The cross-layer optimizations allow *CrossLight* to outperform the state-of-the-art ML accelerators, highlighting the promise of optical computing.

D. Sparse and Quantized Optical Computing

To limit the DAC and ADC power consumption, while sustaining model accuracy, in [25] we proposed the *ROBIN* optical ML accelerator for binarized neural networks (BNNs). BNNs use 1-bit weights, but activations use more bits (e.g., 4 or 8) to preserve accuracy. As BNNs only require simple switching circuits for weight parameter representation and

Table 1: Average EPB and kiloFPS/Watt values across accelerators

Accelerator	Avg. EPB (pJ/bit)	Avg. kiloFPS/watt
P100	971.31	24.9
IXP 9282	5099.68	2.39
AMD-TR	5831.18	2.09
DaDianNao	58.33	0.65
Edge TPU	697.37	17.53
Null Hop	2727.43	4.48
DEAP_CNN	44453.88	0.07
Holylight	274.13	3.3
CrossLight	28.78	52.59

simpler DAC circuitry for activations, they lead to better energy efficiency. In [26], we proposed the *HQNNA* optical accelerator that supported heterogeneous quantization, where each layer in CNN/DNN models could be quantized uniquely between 2 to 8 bits. By allocating more bits to layers that are critical for preserving model accuracy, heterogeneous layer-wise quantization can better balance model accuracy with energy costs. We improved computation orchestration via time division multiplexing and bit-slicing, and optimized laser power to aggressively reduce energy overheads in HQNNA. Fig. 2 shows how these optimizations allow HQNNA to achieve lower EPB than ROBIN as well as *CrossLight*.

Another approach to reduce power involves leveraging sparsification. Sparsity in a DNN model refers to removal of weights (i.e., replacing them with 0s) which do not contribute to the overall accuracy of the model. This technique is often used to reduce memory footprint for ML models. But sparse models incur unwanted operations which lead to a 0 output because of the 0 valued weights involved. This leads to wasted energy consumption. Our work in [27] tackled this challenge by combining a dataflow mechanism for compressing sparse matrices to dense matrices along with better VDU design to handle any residual sparsity. These hardware/software codesign optimization helped achieve 27.6× lower EPB than state-of-the-art electronic and optical-domain accelerators.

E. Emerging Use Cases for Optical Computing

There are many opportunities to utilize optical computing to accelerate ML models beyond CNNs and DNNs.

In [28], we proposed RecLight, the first optical accelerator for sequence learning tasks (e.g., network anomaly detection [29]) that involve Recurrent Neural Networks (RNNs). RNNs, which can include Gated Recurrent Units (GRUs) and Long Short-Term Memory (LSTM) cells, are challenging to accelerate due to the recursive nature of these models and the compute-intensive operations required for large-dimensional sequence data. We developed custom optical computing units that supported accelerating GRU and LSTM cell operations, as well as efficient optical-domain implementations of non-linear activation functions such as sigmoid and tanh.

Large Language Models (LLMs), such as those used in OpenAI's ChatGPT and Google's Bard rely on transformer neural networks. These unique neural networks combine attention layers, feed-forward layers, and normalization layers to learn context and meaning by tracking relationships in data sequences. However, the complex structure of these models creates challenges for accelerating their execution. In [30] we proposed the first optical computing platform to accelerate a broad family of language-based and vision-based transformer models such as BERT and Vision Transformers.

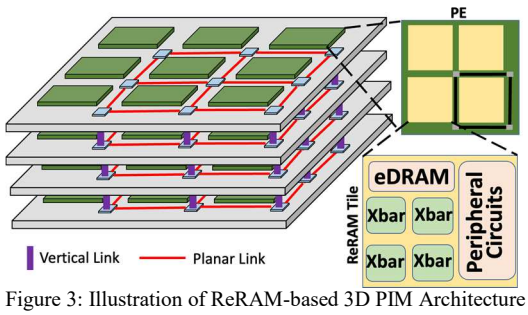


Figure 3: Illustration of ReRAM-based 3D PIM Architecture [41].

We adapted many of the cross-layer optimizations described earlier to the unique requirements of transformer models.

IV. GNN Training on ReRAM-Based PIM Architecture

Training machine learning (ML) models at the edge (training on-chip or on embedded systems) can address many pressing challenges, including data privacy/security, increase the accessibility of ML applications to different parts of the world by reducing the dependence on the communication fabric and the cloud infrastructure, and meet the real-time requirements of emerging applications like augmented/virtual reality (AR/VR) applications. However, existing edge platforms do not have sufficient capabilities to support on-device training of ML models such as Convolutional Neural Networks (CNNs), Graph Neural Networks (GNNs) etc. Moreover, it is estimated that training a single unpruned neural network on conventional compute platforms, such as GPUs, can cost over $10,000 and emit as much carbon as five cars over their lifetimes [31]. Resistive random-access memory (ReRAM) based processing-in-memory (PIM) architectures are a promising solution to address this problem. ReRAM-based PIM systems have been proposed to accelerate both CNN and GNN computation [32]. The crossbar structure of ReRAM enables efficient Matrix-Vector Multiplication (MVM), which is ubiquitous in modern ML tasks including CNN/GNN training and inferencing. In this section, we principally focus on discussing the advantages and challenges of designing ReRAM-based accelerators for GNN training. However, by considering area, power and storage, the overall architecture needs to be divided into multiple ReRAM tiles with bounded crossbar size. Hence, despite the PIM capability, when we design a ReRAM-based manycore architecture for large-scale CNN/GNN computation, it gives rise to a substantial amount of on-chip traffic that creates performance bottlenecks if not addressed appropriately. Moreover, ReRAM crossbar arrays suffer from low write endurance [33] [34]. By incorporating model and graph pruning, we can reduce the on-chip traffic and storage requirements and improve the endurance of the ReRAM-based architectures significantly. ReRAM-based PIM architecture for GNN training.

Recently, we have proposed a ReRAM-based 3D PIM architecture called ReMaGN, tailored for on-chip training of GNNs [32]. We adopt a 3D architecture to enable high degree of integration [35]. This architecture consists of multiple PEs stacked vertically across four layers as shown in Fig. 3. Each PE contains multiple ReRAM crossbar arrays for executing MVM operations. The salient features of the ReMaGN architecture are:

1. To effectively utilize the high-throughput computation provided by ReRAM-based PEs, the overall architecture needs to be supported with a high-performance and efficient communication backbone. In this architecture, we utilize a 3D mesh network-on-chip (NoC) as the interconnection backbone for communication between PEs during GNN training. We have reduced the NoC traffic by incorporating DropEdge and Dropout.
2. ReMaGN employs a pipelined training methodology. Training GNNs on one big monolithic graph is often impractical due to memory concerns. In addition, training on large graphs does not exploit the benefits of ReRAM-based architectures, which rely on a pipelined implementation [32]. Pipelining reduces the number of ReRAM writes (which are slow), leading to higher overall throughput. However, this strategy is not amenable to GNNs if the entire graph needs to be processed altogether. Graph clustering/partitioning is used in ReMaGN to address this problem.
3. Typically, ReRAMs compute using 16-bit fixed-point, which has significantly less representation capability than 32-bit floating point used by traditional GPUs. We have incorporated stochastic rounding to successfully address the accuracy loss due to reduced precision.
4. ReMaGN outperforms conventional GPUs by up to 9.5X (on average 7.1X) in terms of execution time, while being up to 42X (on average 33.5X) more energy efficient without sacrificing accuracy.

A. Model and Data Pruning for GNN Training

To improve the performance and energy efficiency of ReRAM-based architectures without loss of accuracy, we proposed crossbar-aware pruning techniques to simultaneously prune the GNN model weights and the input data graphs. Model pruning for neural networks helps reduce redundant computations [36]. Several crossbar-aware model pruning techniques have been proposed to exploit the ReRAM crossbar structure to reduce area and improve energy efficiency without compromising accuracy [37]. However, all these methods prune pre-trained DNN models for inferencing purposes, hence they are not suited for training. Moreover, existing crossbar-aware methods prune weights along rows/columns only, which leads to marginal energy and area savings. Here, we generally refer to these crossbar row/column-based pruning methods as 'RCP'. Fig. 4(a) illustrates how the weights when pruned using RCP are mapped to ReRAM crossbars. However, despite the structured row-/column-wise pruning in RCP, this only yields marginal energy savings as an entire crossbar is still activated for computation [38].

Recently, an iterative model pruning method known as the Lottery Ticket Hypothesis (LTH) was proposed to obtain highly sparse DNN models for training [39]. An LTH-inspired unified graph sparsification (UGS) has also been used to prune GNN models and graph adjacency matrices [40]. UGS jointly prunes GNN weights and graph adjacency matrices using trainable masks to reduce the number of matrix-vector-multiplication (MVM) operations associated with GNN training. However, the graph pruning in UGS only removes the edges from the graph adjacency matrix and thus the overall size of the input (i.e., the number of input subgraphs) still remains unchanged. Additionally, UGS introduces huge storage overhead for mask parameters, which are proportional to the GNN weights and number of adjacency matrix entries. Moreover, existing LTH-based methods don't take the ReRAM crossbar structure into consideration. As a result, there is no significant area or power savings. As an illustration, Fig. 4(b) shows how the GNN weights pruned using the standard LTH-based methods are mapped to ReRAM crossbars. The unstructured pruning nature of the LTH method only prunes individual weights and yields no significant advantage from a hardware standpoint.

We have proposed an LTH-inspired crossbar-aware pruning method for GNN training on ReRAM-based architectures. Our method enhances existing RCP methods by taking the overall crossbar structure into consideration and implements the pruning in an iterative manner to ensure that the pruned GNN models can be trained from scratch without loss in accuracy. Fig 4(c) illustrates how weights are mapped to the ReRAM crossbars after they are pruned using our proposed crossbar-aware method. Unlike RCP, we can see that entire (c×c) crossbars are pruned out, which leads to significant area and energy savings. We complement the crossbar-aware weight pruning with an optimized non-zero-storage mechanism for the graph adjacency matrix to further reduce the on-chip crossbar storage overhead as shown in Fig. 4(d) [41]. The optimized graph storage is achieved by dividing the adjacency matrix into non-overlapping segments based on the crossbar size. The size of the sliding window is determined by the crossbar size (c × c) to decompose the N×N graph adjacency matrix into "valid" and "invalid" segments (as shown in green and red boxes in Fig. 4(d) respectively) for storing on ReRAM crossbar arrays. A valid segment contains at least a non-zero element and is stored on the crossbar. However, an invalid segment consists of all-zeros and is discarded, as an MVM operation with zeros

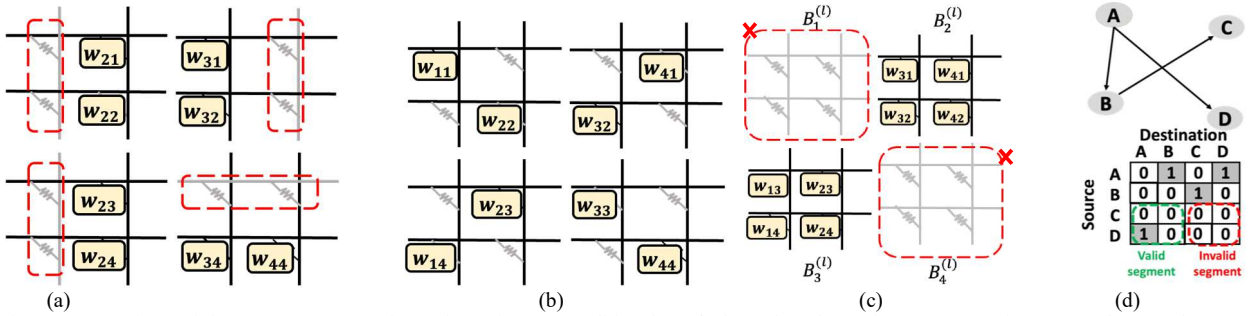


Figure 4. Mapping weights to ReRAM crossbars after using (a) Traditional row/column-based crossbar-aware pruning (RCP) (b) Crossbar-unaware pruning (LTH and UGS) (c) DietGNN pruning. (d) optimized graph adjacency matrix storage technique in DietGNN [41].

mapped to ReRAM crossbars is redundant and only leads to zeros. As a result, this non-zero-storage optimization mechanism reduces the number of crossbars required for the adjacency matrix while maintaining the graph connectivity. We refer to our crossbar aware model pruning method and the optimized graph storage mechanism together as 'DietGNN' henceforth (as proposed in [41]). Overall DietGNN enables energy-efficient GNN training and inferencing on ReRAM-based PIM architectures.

In addition to DietGNN, we can further prune the input data graphs for GNN training on ReRAM-based architectures. Large training datasets pose a challenge for resource-constrained platforms (such as edge devices) as they require high memory and processing power [42]. Recent work has proposed methods to reduce the amount of data required for training, generally referred to as data pruning (DP). An importance score-based data pruning methodology for CNNs was proposed to greatly reduce the amount of input data [43]. Graph Early Bird (GEB) prunes the edges of the input graphs to reduce MVM operations for GNN training very early during the training process [44]. Thus, the size of the input feature vector remains unchanged as only graph edges are pruned. In contrast to existing data pruning techniques for GNNs, our proposed method intelligently prunes large portions of an input graph (both nodes and edges) to reduce the end-to-end pipeline depth of GNN training on ReRAM-based architectures.

It is well known that ReRAM crossbar arrays suffer from low write endurance [33] [34]. ReRAM cells become highly prone to permanent faults after a limited number of writes (typically 106 - 109 writes), thus limiting their lifetime. As graph pruning reduces the number of input subgraphs, it also helps to reduce the number of weight updates occurring during training. Hence, incorporating data pruning helps to preserve the lifetime of the ReRAM-based architecture and is complementary to existing reliability enhancement techniques such as gradient sparsification [34].

For a comprehensive evaluation of DietGNN+DP, we have considered three diverse GNN models, namely: Graph Convolution Networks (GCN), Graph Attention Networks (GAT), and Graph Sample and Aggregate (GSA); and six benchmark real-world graph datasets: PPI, Reddit (RDT), Amazon2M (A2M), Flickr (FKR), Yelp (YLP), and Open Graph Benchmarks-Proteins (OGP) [45]. In Fig. 5(a) and (b), we show the accuracy and achievable sparsity respectively of DietGNN+DP method relative to other pruning techniques (LTH, UGS and RCP). As an example, in Fig 5(a) and (b), we consider the GCN model pruned using each method, and mapped to the ReRAM-based PIM architecture for training. We note that we observe similar accuracy and sparsity trends with the other GNN models (GAT and GSA) across multiple datasets. As shown in Fig. 5(a), UGS is unable to prune a lot of weights without experiencing accuracy loss due to its deletion of graph edges hence it achieves lower sparsity. Overall, Fig. 5 shows that except for UGS, all the pruned models are very sparse (~90%), and they can be trained from scratch with very minimal accuracy (as shown in Fig. 5(b)) loss compared to their unpruned counterparts.

Finally, we carry out a performance evaluation of the DietGNN+DP-enabled GNN training on the ReRAM-based PIM architecture. Fig 6(a) and (b) compare the DietGNN+DP method with; the unpruned counterpart, and other pruning approaches (UGS, RCP, and DietGNN alone) in terms of execution time and energy consumption. Here, we assume an iso-area scenario for this analysis, i.e., the number of ReRAM crossbars available is the same for all the cases. As shown in Fig. 6(a) and 6(b), the DietGNN+DP training improves execution time and energy consumption by ~57% and ~73% respectively on average compared to the unpruned model running on an iso-area ReRAM-based PIM architecture. Overall, the DietGNN+DP-enabled training achieves low energy- and storage-efficient GNN computation. The key highlights of the DietGNN+DP framework are summarized as follows:

1.) DietGNN demonstrates that it is possible to prune more than 90% of GNN weights for diverse GNNs and real-world graph datasets.
2.) The pruned GNNs enable significant energy savings (that is, crossbar diet) and performance improvements without sacrificing accuracy.
3.) The experimental results demonstrate that DietGNN+DP accelerates GNN training by up to $4.5\times$ while using $6.6\times$ less energy on average when compared to its unpruned counterpart on ReRAM-based 3D PIM architectures.

B. Limitations of Existing ReRAM-based PIM Architectures

Reliability of ReRAMs: The ReRAM fabrication process is not as mature as conventional CMOS fabrication [33]. As a result, ReRAMs are prone to many types of hardware faults and noise. For instance, hard faults prevent the resistance of a ReRAM cell from being updated, resulting in write failures. Moreover, the limited endurance of ReRAM cells makes them suffer from short programming cycles before they suffer permanent write failures. The write endurance of ReRAM chips typically ranges from 106 to 1012 writes before they fail [33]. However, training of state-of-the-art large-scale deep neural networks (DNN) usually demands numerous weight updates which result in multiple ReRAM cell programming cycles [46]. As a result, this limits the lifetime of ReRAM devices. Hence, enhancing the lifetime of ReRAM crossbars is important to facilitate their widespread

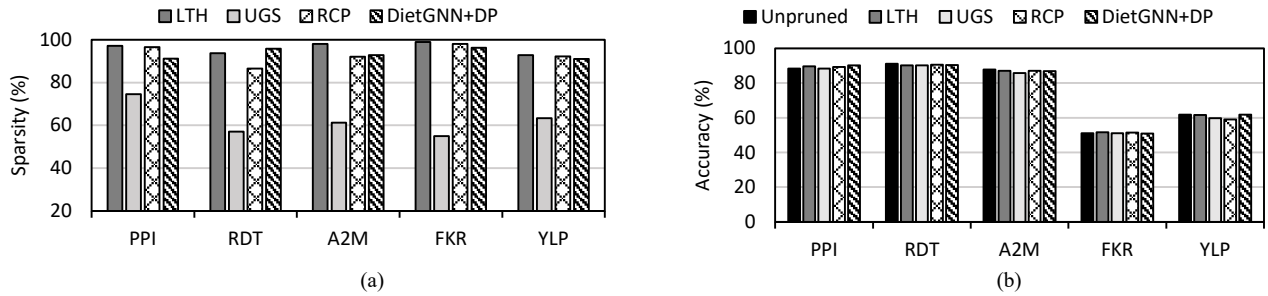
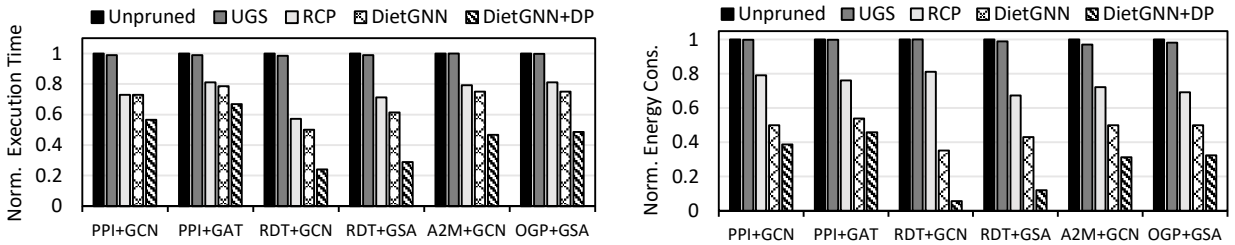


Figure 5: (a) Sparsity and (b) Accuracy of pruned GNN models (winning tickets) obtained using different methods. All models are trained on ReRAM crossbars.



(a) (b)

Figure 6: Normalized (a) execution time and (b) energy consumption for the unpruned, UGS, RCP, DietGNN, and DietGNN+DP-enabled GNN (normalized with respect to the execution of the unpruned model on the ReRAM-based PIM architecture).

adoption as hardware accelerators for DNN training. Recent efforts have tried to address the write endurance issue of ReRAM devices some of which include device-level optimizations and software-level approaches. However, most device-level methods aim at reducing only the per-write energy and do not reduce the actual number of writes. Meanwhile, software techniques focus solely on reducing the number of write operations due to weight updates; and do not consider ReRAM cell writes due to activations or intermediate products. Overall, these methods often introduce area, power, and performance overheads.

Write Latency and Energy of ReRAMs: In addition to the write endurance challenge of ReRAM cells, they also suffer from high write energy (~2 nJ) and latency (~100 ns) compared to other Non-volatile Memory (NVM) technologies for PIM architectures. As a result, most ReRAM-based PIM platforms require traditional memory hierarchies and a pipelined implementation to efficiently hide the memory latency due to slow ReRAM writes. Researchers have explored parallel writes via bit-slicing as a technique to mitigate the expensive write operations during ML training, thus enabling ReRAM-based training architectures.

Accelerating Large Language Models (LLMs): LLMs have achieved significant success in a wide variety of natural language processing (NLP) tasks. However, language tasks are increasingly becoming complex, and are fast outpacing the computing capabilities of traditional platforms such as GPUs and CPUs. This is due to their numerous parameters, high storage requirements, and significant off-chip data movement costs [47]. Hence, ReRAM-based PIMs have been proposed as an alternative for accelerating LLMs due to their high-density storage, in-memory processing capability, and energy-efficient computing capability. However, simple LLM tasks such as inferencing requires a significant number of write operations, and ReRAM-based PIMs are still plagued by the write endurance problem. Moreover, LLM inference tasks are usually time-critical applications. Thus, the high write latency of ReRAM cells can potentially limit its adaptability for LLM inferencing.

V. Conclusion

Advancement in novel ML algorithms and computing systems design is tightly coupled and advancement in one cannot be achieved without the other. However, the slowing down of Moore's law has impacted the development of new computing platforms, which is detrimental to future developments and applications of ML. In this paper, we have discussed the benefits and challenges associated with three emerging hardware paradigms as enablers for designing energy-efficient and high-performance hardware architectures for accelerating various types of ML applications.

Acknowledgments

This work was funded in part by grants from the National Science Foundation CCF-1813370 and CCF-2006788 and funding from EU Commission Horizon EU research and innovation program in the framework of PHASTRAC (https://phastrac.eu) with grant no. 101092096.

References

[1] C. Schuman, S. Kulkarni and M. Parsa et al., "Opportunities for neuromorphic computing algorithms and applications," *Nat Comput Sci* , vol. 2, 2022.

[2] G. Csaba and W. Porod, "Coupled oscillators for computing: A review and perspective," *Applied Physics Reviews,* vol. 7, 2020.

[3] C. Delacour et al., "Energy-Performance Assessment of Oscillatory Neural Networks Based on VO2 Devices for Future Edge AI Computing.," *IEEE Transactions on Neural Networks and Learning Systems,* 2023.

[4] J. Buhmann, R. Divko and K. Schulten, "Associative memory with high information content.," *Physical Review* , 1989.

[5] M. Abernot et al., "Two-Layered Oscillatory Neural Networks with Analog Feedforward Majority Gate for Image Edge Detection Application.," in *IEEE International Symposium on Circuits and Systems.*, 2023.

[6] C. Delacour, et. al., "A Mixed-Signal Oscillatory Neural Network for Scalable Analog Computations in Phase Domain," *preprint at (hal-03961010),* 2023.

[7] T. Wang, et. al., "Solving combinatorial optimisation problems using oscillator based Ising machines," *Nat Comput,* vol. 20, 2021.

[8] T. Jackson, S. Pagliarini and L. Pileggi, "An Oscillatory Neural Network with Programmable Resistive Synapses in 28 Nm CMOS," *2018 IEEE International Conference on Rebooting Computing (ICRC),* pp. 1-7, 2018.

[9] M. Graber, et. al., "A Versatile & Adjustable 400 Node CMOS Oscillator Based Ising Machine to Investigate and Optimize the Internal Computing Principle," *2022 IEEE 35th International System-on-Chip Conference (SOCC),* 2022.

[10] W. Moy, et. al., "A 1,968-node coupled ring oscillator circuit for combinatorial optimization problem solving," *Nat. Electron.,* vol. 5, 2022.

[11] S. Carapezzi, et. al., "Role of ambient temperature in modulation of behavior of vanadium dioxide volatile memristors and oscillators for neuromorphic applications," *Sci. Rep.,* vol. 12, 2022.

[12] M. Graber, et. al., "A Fast Graph Minor Embedding Heuristic for Oscillator Based Ising Machines," *2022 Austrochip Workshop on Microelectronics (Austrochip),* 2022.

[13] D. Albertsson, et. al., "Highly reconfigurable oscillator-based Ising Machine through quasiperiodic modulation of coupling strength," *Sci. Rep.,* vol. 13, 2023.

[14] M. Abernot, T. Gil, M. Jimenez, J. Nunez, M. J. Avedillo, B. Linares-Barranco, T. Gonos, T. Hardelon and A. Todri-Sanial, "Digital Implementation of Oscillatory Neural Network for Image Recognition Applications," *Frontiers in Neuroscience,* vol. 15, 2021.

[15] M. Abernot, et al., "Oscillatory Neural Networks for Obstacle Avoidance on Mobile Surveillance Robot E4," *2022 International Joint Conference on Neural Networks (IJCNN),* pp. 1-8, 2022.

[16] M. Abernot et al., "SIFT-ONN: SIFT Feature Detection Algorithm Employing ONNs for Edge Detection.," in *Proceedings of the 2023 Annual Neuro-Inspired Computational Elements Conference*, 2023.

[17] S. Bahirat and S. Pasricha, "UC-PHOTON: A Novel Hybrid Photonic Network-on-Chip for Multiple Use-Case Applications," in *IEEE ISQED*, 2010.

[18] Y. Xu and S. Pasricha, "Silicon Nanophotonics for Future Multicore Architectures: Opportunities and Challenges," *IEEE Design & Test,* 2014.

[19] F. Sunny, E. Taheri, M. Nikdast and S. Pasricha, "A survey on silicon photonics for deep learning," *ACM JETC,* vol. 17, no. 4, 2022.

[20] S. Pasricha and M. Nikdast, "A Survey of Silicon Photonics for Energy Efficient Manycore Computing," *IEEE Design and Test,* 2020.

[21] A. N. Tait and et al, "Broadcast and Weight: An Integrated Network For Scalable Photonic Spike Processing," *IEEE JLT,* 2014.

[22] S. V. R. Chittamuru, I. Thakkar and S. Pasricha, "LIBRA: Thermal and Process Variation Aware Reliability Management in Photonic Networks-on-Chip," *IEEE TMSCS,* no. Oct-Dec 2018.

[23] S. V. R. Chittamuru, I. Thakkar and S. Pasricha, "PICO: Mitigating Heterodyne Crosstalk Due to Process Variations and Intermodulation Effects in Photonic NoCs," in *IEEE/ACM Design Automation Conference (DAC)*, 2016.

[24] F. Sunny, A. Mirza, M. Nikdast and S. Pasricha, "CrossLight: A Cross-Layer Optimized Silicon Photonic Neural Network Accelerator," in *IEEE/ACM Design Automation Conference (DAC)*, 2021.

[25] F. Sunny, A. Mirza, M. Nikdast and S. Pasricha, "ROBIN: A Robust Optical Binary Neural Network Accelerator," *ACM TECS,* Oct 2021.

[26] F. Sunny, M. Nikdast and S. Pasricha, "A Silicon Photonic Accelerator for Convolutional Neural Networks with Heterogeneous Quantization," in *ACM GLSVLSI*, 2022.

[27] F. Sunny, M. Nikdast and S. Pasricha, "SONIC: A Sparse Neural Network Inference Accelerator with Silicon Photonics for Energy-Efficient Deep Learning," in *IEEE/ACM ASPDAC*, 2022.

[28] F. Sunny, M. Nikdast and S. Pasricha, "RecLight: A Recurrent Neural Network Accelerator With Integrated Silicon Photonics," in *IEEE ISVLSI*, 2022.

[29] V. K. Kukkala, S. V. Thiruloga and S. Pasricha, "LATTE: LSTM Self-Attention based Anomaly Detection in Embedded Automotive Platforms," *ACM TECS,* 2021.

[30] S. Afifi, F. Sunny, M. Nikdast and S. Pasricha, "TRON: Transformer Neural Network Acceleration with Non-Coherent Silicon Photonics," in *ACM GLSVLSI*, 2023.

[31] E. Strubell, A. Ganesh and A. McCallum, "Energy and policy considerations for modern deep learning research," in *AAAI 2020 – 34th AAAI Conference on Artificial Intelligence*, 2020.

[32] A. I. Arka, B. K. Joardar, J. R. Doppa, P. P. Pande and K. Chakrabarty, "Performance and Accuracy Tradeoffs for Training Graph Neural Networks on ReRAM-Based Architectures," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems,* vol. 29, no. 10, pp. 1743-1756, 2021.

[33] W. Wen, Y. Zhang and J. Yang, "ReNEW: Enhancing Lifetime for ReRAM Crossbar based Neural Network Accelerators," in *IEEE International Conference on Computer Design (ICCD)*, 2019.

[34] Y. Cai et al., "Long live TIME: Improving lifetime for training-in-memory engines by structured gradient sparsification," in *Proceedings - Design Automation Conference (DAC)*, 2018.

[35] S. Das, J. R. Doppa, P. P. Pande and K. Chakrabarty, "Design-Space Exploration and Optimization of an Energy-Efficient and Reliable 3D Small-world Network-on-Chip," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, (TCAD),* vol. 36, 2017.

[36] S. Han, J. Pool, J. Tran and W. Dally, "Learning both weights and connections for efficient neural networks," *Advances in Neural Information Processing Systems (NeurIPS),* pp. 1135-1143, 2015.

[37] L. Liang, L. Deng, Y. Zeng, X. Hu, Y. Ji, X. Ma, G. Li and Y. Xia, "Crossbar-Aware Neural Network Pruning," *IEEE Access,* 2018.

[38] G. Yuan et. al, "TinyADC: Peripheral Circuit-aware Weight Pruning Framework for Mixed-signal DNN Accelerators," in *DATE*, 2021.

[39] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," in *International Conference on Learning Representations (ICLR)*, 2019.

[40] T. Chen et al., "A Unified Lottery Ticket Hypothesis for Graph Neural Networks," in *International Conference on Machine Learning (ICML)*, 2021.

[41] C. Ogbogu, A. I. Arka, B. K. Joardar, J. R. Doppa, H. Li, K. Chakrabarty and P. P. Pande, "Accelerating Large-Scale Graph Neural Network Training on Crossbar Diet," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* 2022.

[42] W. Chiang et al., "Cluster-GCN: An efficient algorithm for training deep and large graph convolutional networks," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

[43] M. Paul, S. Ganguli and G. K. Dziugaite, "Deep Learning on a Data Diet: Finding Important Examples Early in Training," in *Advances in Neural Information Processing Systems 34 (NeurIPS 2021)*, 2021.

[44] H. You, Z. Lu, Z. Zhou, Y. Fu and Y. Lin, "Early-Bird GCNs: Graph-Network Co-Optimization Towards More Efficient GCN Training and Inference via Drawing Early-Bird Lottery Tickets," in *AAAI Conference on Artificial Intelligence*, 2022.

[45] W. Hu, M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta and J. Leskovec, "Open Graph Benchmark: Datasets for Machine Learning on Graphs," in *34th Conference on Neural Information Processing Systems (NeurIPS)*, Vancouver, Canada., 2020.

[46] K. Roy, I. Chakraborty, M. Ali, A. Ankit and A. Agrawal, "In-Memory Computing in Emerging Memory Technologies for Machine Learning: An Overview," in *IEEE Design Automation Conference (DAC)*, 2020.

[47] M. Kang, H. Shin and L.-S. Kim, "A Framework for Accelerating Transformer-Based Language Model on ReRAM-Based Architecture," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* vol. 42, no. 9, 2022.

[48] G. Plastiras, M. Terzi, C. Kyrkou and T. Theocharidcs, "Edge Intelligence: Challenges and Opportunities of Near-Sensor Machine Learning Applications," in *IEEE 29th International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, 2018.