HuNT: Exploiting Heterogeneous PIM Devices to Design a 3-D Manycore Architecture for DNN Training

Chukwufumnanya Ogbogu, *Graduate Student Member, IEEE*, Gaurav Narang, *Graduate Student Member, IEEE*, Biresh Kumar Joardar, *Member, IEEE*, Janardhan Rao Doppa, *Senior Member, IEEE*, Krishnendu Chakrabarty, *Fellow, IEEE*, and Partha Pratim Pande, *Fellow, IEEE*

***Abstract*—Processing-in-memory (PIM) architectures have emerged as an attractive computing paradigm for accelerating deep neural network (DNN) training and inferencing. However, a plethora of PIM devices, e.g., resistive random-access memory, ferroelectric field-effect transistor, phase change memory, MRAM, static random-access memory, exists and each of these devices offers advantages and drawbacks in terms of power, latency, area, and nonidealities. A heterogeneous architecture that combines the benefits of multiple devices in a single platform can enable energy-efficient and high-performance DNN training and inference. 3-D integration enables the design of such a heterogeneous architecture where multiple planar tiers consisting of different PIM devices can be integrated into a single platform. In this work, we propose the HuNT framework, which hunts for (finds) an optimal DNN neural layer mapping, and planar tier configurations for a 3-D heterogeneous architecture. Overall, our experimental results demonstrate that the HuNT-enabled 3-D heterogeneous architecture achieves up to 10× and 3.5× improvement with respect to the homogeneous and existing heterogeneous PIM-based architectures, respectively, in terms of energy-efficiency (TOPS/W). Similarly, the proposed HuNT-enabled architecture outperforms existing homogeneous and heterogeneous architectures by up to 8× and 2.4×, respectively, in terms of compute-efficiency (TOPS/mm^2) without compromising the final DNN accuracy.**

***Index Terms*—DNN, FeFET, PIM, ReRAM, SRAM.**

I. Introduction

DEEP neural networks (DNNs) are widely employed to solve complex problems in a variety of application domains, including computer vision, natural language processing (NLP), and time-series sensor data analytics [1]. However, DNNs have hundreds of millions of trainable parameters, which need to be tuned using large and complex datasets. The high latency and energy cost of data movement between the processing cores and memory units in traditional computing platforms based on the von-Neuman architecture (e.g., CPUs and GPUs) impose significant performance bottlenecks while executing DNN workloads, which is referred to as the *memory wall* challenge [2]. Consequently, there has been a growing demand for domain-specific computing platforms that seamlessly integrate both storage and computing, thereby enabling high-performance and energy-efficient acceleration of DNN workloads [3].

Processing-in-memory (PIM)-based computing platforms have emerged as a promising alternative for executing DNN workloads. This is due to their ability to perform energy-efficient computation within the memory to eliminate unnecessary data movement, thus addressing the memory-wall challenge. Specifically, the use of CMOS-based memory devices, such as static random-access memory (SRAM), and nonvolatile memory (NVM) devices, such as resistive random-access memory (ReRAM), phase change memory (PCM), ferroelectric field-effect transistors (FeFETs), and spintronic memory (MRAM), have been widely studied as suitable candidates for accelerating DNN training and inferencing [2], [3], [4], [5], [6]. However, each of these PIM devices offers specific advantages and drawbacks in terms of dynamic and leakage power, area, latency, retention, endurance, and nonidealities, when used as the PIM device in DNN accelerators [3]. For example, ReRAM devices have almost ∼30× higher write latency compared to FeFET devices. However, ReRAMs can have a write endurance of as high as ∼10^{12} programming cycles whereas FeFETs have an endurance of ∼10^5 cycles [7]. An ideal memory device suitable for energy-efficient and high-performance PIM-based DNN accelerators should have low read/write latency (< 1 ns), low dynamic and leakage energy (< 3 pJ), high write endurance (> 10^{17} cycles), small memory cell footprint (< 4F^2), and excellent scalability to lower technology nodes (< 10 nm) [8]. However, so far, no particular PIM device has all the ideal characteristics. At the same time, DNN workloads are composed of neural layers, which can differ significantly in terms of the number of layers, weight parameters, kernel size, input and output information across layers in the forward-propagation, and

Manuscript received 8 August 2024; accepted 9 August 2024. This work was supported in part by the U.S. National Science Foundation (NSF) under Grant CNS-1955353 and Grant CSR-2308530. This article was presented at the International Conference on Hardware/Software Codesign and System Synthesis (CODES + ISSS) 2024 and appeared as part of the ESWEEK-TCAD Special Issue. This article was recommended by Associate Editor S. Dailey. *(Chukwufumnanya Ogbogu and Gaurav Narang contributed equally to this work.) (Corresponding author: Partha Pratim Pande.)*

Chukwufumnanya Ogbogu, Gaurav Narang, Janardhan Rao Doppa, and Partha Pratim Pande are with Department of Electrical Engineering and Computer Science, Washington State University, Pullman, WA 99164 USA (e-mail: c.ogbogu@wsu.edu; gaurav.narang@wsu.edu; jana.doppa@wsu.edu; pande@wsu.edu).

Biresh Kumar Joardar is with the Department of Electrical and Computer Engineering, University of Houston, Houston, TX 77204 USA.

Krishnendu Chakrabarty is with the Department of Electrical and Computer Engineering, Arizona State University, Tempe, AZ 85287 USA.

Digital Object Identifier 10.1109/TCAD.2024.3444708

frequency of weight updates during the back-propagation step. These characteristics determine the suitability of each neural layer in the forward- and back-propagation phase of the DNN workload to be executed on a specific PIM-enabled processing element (PE) in terms of area, latency, power, and endurance. Hence, this PE-level heterogeneity in PIM-based architectures needs to be exploited to achieve the best tradeoff in terms of power, area, performance, and DNN accuracy while designing a suitable accelerator platform.

Integrating different memory devices in a single platform presents unique challenges. Specifically, manufacturing technologies of NVM devices vary and they are not always CMOS-compatible [9]. Hence, this hinders the feasibility of integrating such heterogeneous PEs into a single planar architecture. 3-D integration enables the mapping of disparate technologies to different planar tiers [9], [10]. However, existing implementations of 3-D heterogeneous architectures are not well optimized for PIM devices, as they do not consider the device-level characteristics in their design optimization flow. For example, 3-D architectures are known to give rise to thermal hotspots. PIM devices, such as ReRAM and FeFET, are susceptible to nonidealities due to thermal noise, which potentially degrades the accuracy of trained DNNs [11], [12]. As a result, critical DNN model layers mapped to PEs placed in planar tiers that are away from heat sinks can potentially degrade the test accuracy of the DNN due to thermal hotspots. Hence, in order to meet the high accuracy demand of DNN applications, suitable placement of the PEs on planar tiers in a 3-D system is important.

Furthermore, existing heterogeneous DNN accelerators do not consider the characteristics of DNN workloads and the properties of different PIM devices while mapping DNN neural layers to PEs in the overall architecture [2], [4]. For example, neural layers with large number of weights and activations mapped to a PE with high read/write energy would consume more power compared to a PE with less read/write energy. In addition, different neural layers have varying impact on DNN accuracy [13]. Hence, they need to be suitably mapped to appropriate PEs on a planar tier in the 3-D architecture without degrading the final predictive accuracy. Hence, the layer-to-PE and PE-to-tier mapping in a 3-D heterogeneous system impact the overall performance in terms of latency, area, power, and accuracy while executing DNN workloads.

In this article, we propose a design space exploration methodology called *HuNT* that undertakes neural layer-to-PE and PE-to-planar tier mapping to design an optimized 3-D heterogeneous manycore architecture for training DNN workloads. We consider SRAM, ReRAM, and FeFET PIM-enabled PEs for studying the efficacy of the HuNT framework. These heterogeneous PIM devices largely vary in terms of area, power, latency, and endurance. This variation provides HuNT with the scope of optimizing across multiple conflicting, yet crucial objectives, namely: *latency*, *accuracy*, *area*, and *power*. We capture these objectives using three performance evaluation metrics: 1) energy-efficiency (TOPS/W); 2) compute-efficiency (TOPS/mm^2); and 3) DNN predictive accuracy. Recent work has proposed optimization methodologies aimed at exploring device-level heterogeneity in PIM accelerators [14], [15], [16]. However, these techniques are focused on DNN inference scenarios, and cannot handle the more challenging scenario of DNN training. Specifically, the computation of the weight- and activation-gradients in the back-propagation phase requires multiple write operations and high-precision computation. However, NVM devices have limited write endurance, and store weights and activations in fixed-point representation [3], [17]. These critical drawbacks limit the applicability of existing NVM-based PIM accelerators to DNN training. In this work, in addition to the energy-efficient NVM devices (ReRAM and FeFET), we have also incorporated a CMOS-based memory device (SRAM) which can perform high-precision computation in the back-propagation phase, and has a high write endurance into the HuNT framework. This heterogeneity in PIM devices enables reliable, energy-efficient, and high-performance DNN training on 3-D heterogeneous PIM architectures. The key contributions of this work are as follows.

1) We propose the HuNT framework that determines the mappings of DNN layer to heterogeneous PEs and the corresponding PE to planar tier mapping to design a 3-D heterogeneous manycore architecture tailor-made for DNN training. The heterogeneity enables significant improvement in energy-efficiency, area-efficiency, and endurance compared to its homogeneous counterparts.
2) We demonstrate the transferability of the HuNT-enabled 3-D heterogeneous manycore architecture for diverse datasets. The hardware architecture optimized with CIFAR-10 dataset is equally effective for larger datasets, such as CIFAR-100 and TinyImageNet. Hence, this reduces the cost of repeated optimization as no extra training is required for complex datasets.
3) Our experimental results show that the HuNT-enabled 3-D heterogeneous PIM architecture outperforms state-of-the-art heterogeneous PIM architectures, namely, AccuReD and HyperX by up to 3.5× and 4.5×, respectively, in terms of energy-efficiency (TOPS/W), and 2.2× and 3.2× in terms of area-efficiency (TOPS/mm^2), respectively.

To the best of our knowledge, HuNT is the first-of-its kind framework that jointly incorporates DNN layer-to-PE and PE-to-planar tier mapping in a 3-D architecture to achieve high-performance, energy-efficient, and reliable DNN training.

II. Background and Related Prior Work

In this section, we discuss relevant prior work on PIM-based architectures for accelerating DNN workloads. Specifically, we focus on homogeneous PIM architectures solely based on either SRAM, ReRAM, or FeFET devices, as well as their advantages and limitations. Table I compares the characteristics of SRAM, ReRAM, and FeFET PIM devices. Next, we discuss heterogeneous architectures that combine two or more of these devices, and finally shed more light on 2.5-D and 3-D-based PIM accelerators for DNNs.

A. Homogeneous PIM Architectures

SRAM cells have been used as a crossbar-based PIM device for high accuracy DNN training and inference [18], [19]. This

TABLE I
COMPARISON OF VARIOUS PIM DEVICES

Property	**SRAM** [7]	**ReRAM** [33]	**FeFET** [7]
Multi-bit Cell	No	Yes	Yes
†Cell Area (F^2)	150F^2	4F^2	35F^2
Write Energy	3pJ	2nJ	5pJ
Write Latency	~1ns	~100ns	~3ns
Write Endurance	>10^{17} cycles	10^8 cycles	10^5 cycles
Leakage Energy	High	Low	Low

†F is the minimum feature size [14]

is due to their low device variability, high write endurance, low susceptibility to noise, and low write latency as shown in Table I [3]. However, the 6T-cell configuration of SRAMs with a cell size of 150F^2 (as shown in Table I) leads to the high area overhead of SRAM-based crossbar arrays [3], [14]. Additionally, SRAMs suffer from high leakage energy and have low density storage (i.e., can only store 1-bit per-cell) thereby making them less energy- and area-efficient compared to other PIM-devices. Hence, this makes SRAM-based PIM platforms infeasible for large DNN models with large number of weights and activations and many neural layers. Recent work has also leveraged DRAM technology for PIM-based architectures due to its small cell area [20]. However, DRAM suffers from high leakage power and refresh energy due to its volatile nature. Moreover, the 1T1C structure of the DRAM cell lacks in-situ compute capability, hence cannot enable parallel energy-efficient matrix–vector multiply (MVM) operations required for DNN training [20]. Consequently, this has led researchers to explore NVM devices, such as FeFET and ReRAMs.

ReRAM-based NVM device enables high-density storage due to its multibit cell storage capability [3], [4]. Additionally, ReRAM devices have relatively small cell area and low-leakage energy compared to SRAMs, as shown in Table I. However, despite these advantages, ReRAM cells suffer from low write endurance, high write energy, and latency compared to SRAMs. As a result, this limits the applicability of ReRAM-based PIM architectures for DNN training scenarios, as the back-propagation phase requires a significant number of write operations [3]. Additionally, ReRAM cells become less reliable as temperature increases over time, which can cause errors, thereby leading to a degradation in the DNN predictive accuracy. Also, despite the small cell area of ReRAMs ($\sim 4F^2$), the high-resolution ADCs required by the ReRAM crossbar array introduces significant area and energy overhead [4]. Hence, this potentially limits the benefits of using ReRAM-devices in PIM-based architectures.

FeFET devices have been explored as another possibility for PIM-based DNN accelerators. FeFET PIM devices are particularly attractive due to their relatively low cell area ($\sim$35 F^2) compared to SRAMs, high read and write speeds, low write energy, and low-leakage energy. Moreover, they exhibit relatively better temperature stability compared to ReRAM [7]. However, as shown in Table I, a key drawback of FeFET PIM devices is their low write endurance compared to other memory technologies, such as SRAMs and ReRAMs. This is due to the collapse of the separation between the ON and OFF states of the FeFET device (also known as the *memory window*) after repeated program/erase cycles [5]. Consequently, this can cause read errors during DNN training and inference.

Overall, homogeneous architectures built solely using either SRAM, ReRAM, or FeFET PIM devices have their unique advantages, as well as drawbacks that limit their applicability for DNN training and inference workloads. Therefore, exploring heterogeneous PIM architectures that combine one or more PIM devices is necessary to achieve better performance, power, area, and DNN predictive accuracy tradeoffs compared to the homogeneous ones. For the scope of this work, we have considered SRAM CMOS-based devices, FeFET and ReRAM NVM-based devices, as examples to demonstrate the viability of our proposed framework to design optimized heterogeneous PIM accelerators. Note however that other types of PIM devices, such as PCMs and MRAMs, can also be considered for heterogeneous systems.

B. Heterogeneous PIM Architectures

Prior work has proposed heterogeneous architectures that combine two or more PIM devices for accelerating DNN workloads. Various hybrid ReRAM/SRAM-based PIM architectures have been proposed to address the nonidealities in ReRAM devices, and reduce the high area overhead of SRAM. Some of these approaches involve encoding the MSBs using SRAMs, and RRAMs for the LSBs of multibit weights, while maintaining high energy-efficiency [21]. Other methods involve the use of ReRAM and SRAM to perform the DNN forward- and back-propagation operations, respectively, thereby mitigating the limited endurance challenge of ReRAM. In fact, a recent hybrid architecture incorporates SRAM macros to perform output compensation of the nonideal output of ReRAM crossbars, thereby enabling robust DNN inference [16]. However, these methods do not consider the layer-wise characteristics of DNN workloads (e.g., number of neural layers, weights, activations, size of kernels, etc.) while mapping neural layers to the heterogeneous PIM-based architectures. As a result, this can lead to suboptimal performance while executing DNN training and inference tasks.

A recent work called HyDe has proposed a design space exploration methodology for finding an optimal mapping of DNN layers to either SRAM, FeFET, or PCM devices in a hybrid platform [14]. This approach leverages the characteristics of each DNN layer to find its affinity toward a specific type of PIM device. However, this approach is aimed only at inferencing, and considered a scalarized single-objective optimization formulation. However, linear scalarization is known to perform poorly due to its inability to explore nonconvex regions of the Pareto front. Moreover, HyDe follows a differentiable optimization approach, which is not possible for all hardware design objectives and requires training DNN weights by considering the device characteristics. Hence, this is not practical for the DNN training task. Other works, such as HyperX, have proposed a hybrid SRAM/ReRAM architecture, where some DNN layer weights remain static, and are mapped to ReRAMs, while other layers are mapped to SRAMs for fine-tuning [22].

Despite the advantages of heterogeneity, previous solutions do not consider the challenges of integrating different PIM devices into a single platform. Moreover, they are mostly targeted at DNN inferencing/fine-tuning applications and cannot be used for end-to-end training of large DNNs. Hence, suitable heterogeneous PIM architectures for DNN training scenarios need to be explored.

C. 2.5-D/3-D-Based PIM Architectures

To address the challenges associated with integrating different PIM technologies in a single platform, various heterogeneous integration methods have been proposed. Specifically, chiplet-based (2.5-D) integration techniques have been proposed for DNN accelerators [14]. However, the long-range on-chip communication in planar 2.5-D systems presents a significant performance bottleneck in the execution of DNN workloads [23]. Hence, 3-D heterogeneous integration methods that stack planar tiers consisting of PEs connected to each other using through-silicon-via (TSV)-based vertical links have been proposed [23]. For example, a 3-D heterogeneous architecture for accelerating DNN training known as AccuReD was recently proposed. AccuReD leverages ReRAM-based PEs, and GPUs for accelerating all types of DNN layers to enable high accuracy DNN training [23].

Despite offering the advantages of 3-D heterogeneous integration, existing architectures do not consider the properties of the neural layers while determining the mapping for DNN workloads. Moreover, 3-D architectures inherently suffer from thermal issues, which have a varying impact on PEs with NVM devices (FeFET and ReRAM). Prior work does not adequately consider thermal issues while finding a suitable DNN layer-to-PE mapping in 3-D heterogeneous PIM architectures. As a result, this potentially leads to degradation of predictive accuracy, power, and latency when DNN workloads are executed. Hence, the properties of the DNN neural layers, PIM device characteristics of the PEs, as well as the PE to 3-D planar tier mapping should be jointly considered to enable high performance, energy-efficient, and reliable DNN training on heterogeneous PIM platforms.

III. HuNT Framework

This section presents the problem formulation and optimization methodology of the HuNT framework to find the optimal neural layer-to-PE and PE-to-tier mapping in a 3-D heterogeneous architecture for DNN training.

A. Problem Setup

We consider a manycore system with C PIM-based PEs distributed over Z planar tiers and stacked using TSV-based vertical links. We use a conventional mesh-based network on chip (NoC) as the communication backbone [23]. Each planar tier consists of PEs of one particular type of PIM device, i.e., either SRAM (S), ReRAM (R), or FeFET (F). Fig. 1 illustrates an example of a three-tier (i.e., $Z = 3$) 3-D heterogeneous manycore architecture. Given the characteristics of the DNN neural layers, and the physical properties of the PIM devices in the PEs, our goal is to find an optimized neural layer to PE mapping, and the corresponding PE to planar tier mapping that achieves a suitable tradeoff between the training accuracy, area, latency, and power.

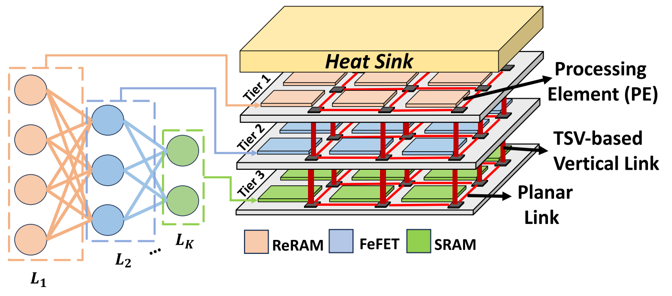

Fig. 1. Illustration of layer-to-PE and PE-to-tier mapping of DNN workload with K-layers on to a 3-D heterogeneous PIM-based architecture. Here, DNN layer L_1 is mapped to ReRAM-based PEs and placed on Tier 1 as an example.

Without loss of generality, Fig. 1 shows a DNN workload mapped on to a 3-D heterogeneous architecture. Here, each neural layer (L_i) of the DNN can be mapped onto either SRAM-/FeFET-/ReRAM-based PEs, which can be located either in tier-1, 2, or 3 as shown in Fig. 1. In addition, each neural layer is characterized by its corresponding kernel size, the number of input and output features, and the bit precision of weights/activations, and can be mapped to one or more PEs in a planar tier of the 3-D architecture. DNN training requires the high-precision computation of weight- and activation-gradients for each neural layer in the back-propagation phase. This process requires a significant number of write operations, which influences the choice of PIM device for the computation of the back-propagation phase.

Furthermore, the PIM devices in the PEs have their corresponding physical properties, such as write endurance limit, area, energy, latency, and temperature-dependent nonideal effects. Additionally, the distance of a planar tier from the heat sink in the 3-D architecture determines the degree of vulnerability of the PIM device to thermal noise, which can potentially lead to significant loss in DNN accuracy [23]. Consequently, this leads to a multiobjective optimization (MOO) problem of finding the suitable mapping of each neural layer to one of the C PIM-based PEs (i.e., either SRAM-/ReRAM-/FeFET-based PE), as well as its appropriate location in one of the Z planar tiers, that achieves the best latency, area, power, and accuracy tradeoff.

B. HuNT MOO Formulation

Fig. 2 shows the overview of the proposed HuNT framework. The inputs to the framework are the number of planar tiers (Z), total number of PEs (C), PIM device choices, and DNN workload characteristics (e.g., number of neural layers, their weights, activations, etc.). We define the mapping vector π to characterize the mapping of K neural layers on to PEs in the 3-D architecture and the corresponding PE to planar tier mapping $\alpha = [t_1, t_2, \ldots, t_Z]$, where t_i is the device type of the PEs in the ith planar tier. Subsequently, let $d = (\pi, \alpha)$ be a candidate design in the design space D which corresponds to a specific neural layer mapping on to the heterogeneous PEs (π) and PE-to-tier mapping (α). In each optimization iteration, one design d is evaluated using power, latency, area, and DNN accuracy estimation models. Our goal is to minimize the 1) loss in DNN accuracy (Err) due to various PIM device

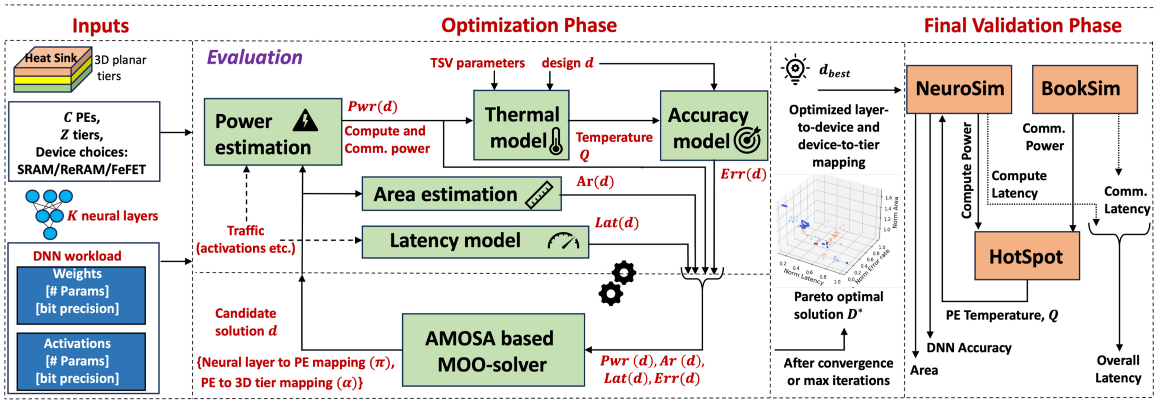

Fig. 2. Overall workflow of the HuNT framework, showing input stage, optimization phase, and the final validation phase.

nonidealities; 2) the area in terms of the number of PEs needed to map all the DNN layers (Ar); 3) the latency (Lat); and 4) power consumption (Pwr) while executing a given DNN training on the 3-D heterogeneous architecture. We represent the MOO formulation as

$$D^* = \text{MOO}(\text{OBJ} = \text{Pwr}(d),\ Ar(d), \text{Lat}(d), \text{Err}(d)) \quad (1)$$

where D^* is the set of Pareto-optimal designs. A design is called *Pareto optimal* if it cannot be improved in any of the design objectives without compromising some other objective. The goal is to first find the Pareto-optimal set $D^* \subseteq D$ using a MOO solver. Next, we select feasible designs from the Pareto set that meet the constraint (e.g., less than $\sim$1% accuracy loss compared to the ideal accuracy). Finally, we select the best design d_{best} from the feasible designs that achieves the best performance in-terms of either energy-efficiency (TOPS/W) or compute-efficiency (TOPS/mm^2).

Next, we discuss the key elements of our MOO formulation.

1) Inputs: The inputs to the HuNT framework are the number of planar tiers (Z), total number of PEs (C), PIM device choices (ReRAM, SRAM, and FeFET), and DNN workload characteristics (e.g., weights, activations, etc.).

2) Design Variables: There are two types of design variables for the optimization for a given DNN model. Each candidate solution represents 1) a neural layer mapping to PEs (π) and 2) a PE-to-tier mapping (α), i.e., $[t_1, t_2, \ldots, t_z]$, where the planar tiers t_1 and t_z are closest and farthest from the heat sink, respectively, resulting in higher temperature on planar tier t_z compared to t_1.

3) Design Objectives: Next, we explain the evaluation of the design objectives: latency, area, accuracy, and power. We can get accurate values for all these objectives for any candidate design by performing cycle-accurate simulations, which are very expensive. Since we need to evaluate many design choices to solve the MOO problem shown in (1), we consider surrogate design objectives elaborated below for tractable optimization.

Latency (Lat): We evaluate end-to-end latency incurred in DNN training for a candidate design ($Lat(d)$) considering a 3-D mesh-based NoC architecture. The latency for a candidate design is proportional to the sum of the computation and communication latency while executing the training task on the 3-D heterogeneous architecture given by (2). Computation during DNN training involves computing activations (Act), and gradients [activations gradients (ΔAG) and weight gradients (ΔWG)] in the forward- and back-propagation phases, respectively. Both phases have different precision requirements. In contrast to Act computation, ΔAG and ΔWG computation requires PEs with a PIM device that has high precision and high endurance due to large number of repeated write operations [24]. Hence, the neural layer computation in a training task is spread out on different 3-D planar tiers, where each tier consists of PEs constituting of a specific device type. This generates on-chip communication traffic, which depends on the layer-to-PE, and the PE-to-tier mapping. The end-to-end compute latency for a DNN workload depends on the compute latency incurred by the individual neural layers mapped to either SRAM-, ReRAM-, or FeFET-based PEs ($\text{Latency}_{S|R|F}$), as shown in (3). Similarly, the latency associated with sending Act, ΔAG, or ΔWG from PE_i to PE_j depends on the placement of PEs and contributes to the communication latency given by (4), where F_{ij} is either Act, ΔAG, or ΔWG as defined above. The parameter M_{ij} is the corresponding Manhattan distance between PE_i and PE_j

$$\text{Lat}(d \propto \mathcal{L})_{\text{compute}} + \mathcal{L}_{\text{comm}} \quad (2)$$

$$\mathcal{L}_{\text{compute}} \propto \sum_{i=1}^{K} \left[\text{Latency}_{S|R|F}(W_i + \text{Act}_i)\right] \quad (3)$$

$$\mathcal{L}_{\text{comm}}(i,j) \propto F_{ij} \cdot M_{ij}\ \forall F_{ij} \in \{\text{Act},\ \Delta AG, \Delta WG\}. \quad (4)$$

Area (Ar): It is desirable to execute a given DNN training task using less resources (PEs) to improve the compute efficiency (TOPS/mm^2). The number of PEs needed to map a given neural layer depends on its device type. For example, SRAMs have larger footprint (150F^2) compared to ReRAM cells (4F^2), where F is the minimum feature size as mentioned in Table I. Hence, a neural layer mapped to SRAM-based PEs would require a higher number of PEs than if it were otherwise mapped to ReRAM-based PEs, leading to comparatively lower TOPS/mm^2. The design objective Ar corresponds to the sum of computational resources needed to execute K layers of a DNN, where PEs needed for the ith neural layer (weights w_i and activations Act_i), depending on the PIM-device ($\text{Area}_{S|R|F}$)

$$Ar(d) \propto \sum_{i=1}^{K} \left[\text{Area}_{S|R|F}(w_i + \text{Act}_i)\right]. \tag{5}$$

Accuracy (Err): Prior work has shown that DNN models can be trained to be robust against conductance drift in NVM devices using techniques, such as adaptive noise injection, negative feedback training, etc. [25], [26]. For example, the injection of Gaussian noise is widely used to improve robustness of DNN training executed on NVM-based architectures [13]. In a 3-D manycore architecture, the thermal noise mainly depends on the placement of the PEs and their mutual interactions. Hence, the exact noise and specific layer-wise weight deviation (σ) is not known prior to neural layer-to-PE and PE-to-tier mapping on a given architecture. Thus, even with a model trained with conductance drift incorporated, the actual thermal noise depends on the neural layer-to-PE mapping and the location of the planar tier where the PE is placed. This necessitates the consideration of accuracy as one of the objectives in the MOO formulation. It should be noted that executing DNN training for each mapping candidate solution d in the optimization phase is costly. Hence, we model the loss in accuracy by capturing the deviation in stored weights and activations due to thermal noise.

3-D architectures with multiple stacked planar tiers are prone to thermal hotspots, which causes variations in stored DNN weights and activations especially in NVM devices (FeFET and ReRAM). This leads to a degradation in the DNN accuracy. However, SRAM is known to be more tolerant to the thermal noise compared to ReRAM and FeFET devices [14]. Hence, to achieve high DNN accuracy, it is desirable to execute high precision computations (involved in the back-propagation phase) on PEs with a PIM device that is more resilient to thermal noise. Thus, computations involved in a neural layer in different DNN training phases, i.e., forward- and back-propagation phases need to be mapped on different types of PEs to achieve high training accuracy. Furthermore, these different PEs can be mapped to planar tiers such that loss in DNN accuracy due to thermal noise is mitigated. For example, PEs with NVM devices should be placed closer to the heat sink, while SRAM-based PEs can be mapped to a planar tier farther from the heat sink.

In addition, a layer-to-PE and PE-to-tier mapping also needs to be considered for different NVM devices. This is crucial because thermal noise impacts variations in weights/activation of various NVM devices differently. For example, weights and activations of the neural layers are stored in ReRAM cells as conductance states. As the temperature increases, the OFF-state conductance of ReRAM cells increases exponentially, and the noise margin reduces [23]. On the other hand, the noise margin of FeFET devices, characterized by the memory window, reduces linearly with the increase in temperature [5]. For weight variation (Δw), we adopt a Gaussian distribution with $\Delta w \sim \mathcal{G}(0, \sigma^2)$, where σ represents standard deviation of weights, consistent with prior work [13]. The variation of weights/activations belonging to different DNN layers impact the model accuracy differently. The impact of weights/activations variations due to thermal noise is captured by loss in accuracy (Err) given by (6) and (7). Hence, Err depends on the neural layer mapping, PE_i temperature Q_i, and DNN layer weights w_i and activations Act_i

$$\text{Err}(d) = \sum_{i=1}^{C} (w_i + \text{Act}_i) \cdot \mathcal{N}(i) \tag{6}$$

$$\mathcal{N}(i) = \begin{cases} \exp[Q_i], & \text{ReRAM} \\ Q_i, & \text{FeFET} \\ \sim 0, & \text{SRAM} \end{cases} \tag{7}$$

To estimate the temperature Q_i of each PE, our framework utilizes the thermal model from prior work, which considers both vertical and horizontal heat flow, given by [27]

$$Q_{o,z} = \left\{ \sum_{u=1}^{z} \left(P_{o,u} \sum_{v=1}^{u} R_v \right) + R_b \sum_{u=1}^{z} P_{o,u} \right\} * \Phi_H \tag{8}$$

where $P_{o,u}$ is the power consumption of the PEs u tiers away from the sink in a vertical stack o and is a function of the neural layer to PE mapping, Φ_H represents the lateral heat flow, R_v is the thermal resistance in vertical direction, and R_b is the thermal resistance of the base layer on which the die is placed and z represents the zth tier where PEs are located. Values of R_v and R_b depend on the material characteristics and are calibrated using HotSpot [28].

Power (Pwr): The PE power consumption $P_{o,u}$ in (8) depends on the DNN training task, layer-to-PE, and PE-to-tier mapping. The total computation power corresponds to the power incurred while computing the individual neural layers mapped to either SRAM-, ReRAM-, or FeFET-based PEs ($\text{Power}_{S|R|F}$), as shown in (10). Further, routers and links associated with the PEs dissipate significant power due to high data exchange between the neural layers. If two subsequent neural layers exchanging large number of activations are mapped on to the PEs far apart, then such mapping creates traffic bottleneck due to frequent long distance data transfer. This creates unnecessary congestion resulting in increase in the communication power. The communication power required to transfer data from PE_i to PE_j is given by (11), where F_{ij} is either activations (Act) in forward phase, or activation gradients (ΔAG) and weight gradients (ΔWG) in back-propagation phase, communicated from PE_i to PE_j and M_{ij} is the corresponding Manhattan distance between PE_i and PE_j

$$Pwr(d) \propto P_{\text{compute}} + P_{\text{comm.}} \tag{9}$$

$$P_{\text{compute}} \propto \sum_{i=1}^{K} \left[\text{Power}_{S|R|F}(w_i + \text{Act}_i)\right] \tag{10}$$

$$P_{\text{comm.}}(i,j) \propto F_{ij} \cdot M_{ij} \quad \forall F_{ij} \in \{\text{Act},\ \Delta AG,\ \Delta WG\}. \tag{11}$$

AMOSA-Based MOO Approach: In this section, we discuss the algorithmic procedure to compute the Pareto-optimal set of designs (neural layer-to-PE and PE-to-tier mappings). Algorithm 1 shows a high-level pseudocode for our design optimization methodology based on the well-known AMOSA solver [29]. The goal is to distribute the computations of K DNN layers (forward- and back-propagation phases) across C PEs on Z planar tiers of different device types to obtain optimal

Algorithm 1: Neural Layer-to-PE and PE-to-3-D Planar Tier Mapping

```
Input: Target manycore system with $C$ PEs of PIM device types- SRAM, ReRAM or FeFET
$APP$ = DNN training task
Output: $D^*$, the Pareto optimal set of designs (optimized neural layer-to-PE mapping and PE-to-tier mappings)
1   Initialize: $D$ = non-dominated set of solutions; $A$ = Archive
2   Input variables ($\overrightarrow{x}$) = neural layer-to-PE mapping ($\pi$) and PE-to-tier mapping ($\alpha$)
3   Repeat
4       Select one $\overrightarrow{x}$ from $A$ and $Perturb \overrightarrow{x}$ to get a design $d$
5       $design\ d \leftarrow$ Candidate mapping of neural layer-to-PEs and PEs-to-3-D planar tiers
6       $Evaluate(design\ d,\ APP)/*$ using power, area, latency, and accuracy models [Section III-B]*/
7       Update non-dominated set of solutions $D$ via $Pwr(d),\ Ar(d),\ Lat(d),\ Err(d)$
8       Update Archive A
9       Until convergence or maximum iterations
10  Pareto optimal set of designs $D^* \leftarrow D$
11  return $D^*$, the Pareto optimal set of designs (optimized neural layer-to-PE mapping and PE-to-tier mappings)
```

tradeoffs between Pwr, Ar, Lat, and Err. The input variables $\overrightarrow{x}$ in our MOO approach are the neural layer-to-PE mapping (π) and PE-to-tier mapping (α). A candidate configuration of $\overrightarrow{x}$ corresponds to the design d which is a candidate mapping of neural layer-to-PEs and PEs-to-3-D planar tiers (Algorithm 1, line 5). First, we start with a randomly chosen mapping of DNN layers to PEs and PEs to planar tiers satisfying the mapping constraints: 1) a neural layer is mapped on to PEs of one device type and 2) a planar tier consists of PEs of one device type. It should be noted that it is possible for a neural layer to be mapped to different types of PEs in different tiers. However, this gives rise to synchronization issues as each type of PE has different latency and throughput. Computations involved in one neural layer need to be completed and the activations must then be sent to the next neural layer. If a layer is mapped on two different types of PEs with unequal timing characteristics, then the computation latency for a particular neural layer will be bottlenecked by the PE with the worst-case delay. This will lead to a degradation in the overall training performance. Hence, each neural layer is mapped on to PEs of one device type. Also, due to fabrication challenges, we refrain from integrating different types of NVM devices on the same tier.

Next, we perturb a candidate mapping solution to get a new layer-to-PE and PE-to-tier mapping (Algorithm 1, line 4). Here, a valid perturbation is defined as allocating a randomly chosen neural layer to a different PE such that the mapping constraints mentioned above, are satisfied. In each AMOSA iteration, the selected design is evaluated using the surrogate objectives for latency, area, power, and accuracy (Algorithm 1, line 6) and the nondominated set of designs and Archive are updated based on this new design evaluation. At convergence or after maximum iterations, we get the Pareto-optimal set of designs D^* from the MOO solver. We first select the feasible designs from D^* fulfilling the DNN accuracy constraint mentioned above (e.g., 1% accuracy loss with respect to ideal condition) by performing cycle-accurate simulations. Finally, we select the best design d_{best} from the feasible designs that achieves the best performance in-terms of either energy-efficiency (TOPS/W) or compute-efficiency (TOPS/mm^2). It should be noted that the HuNT framework optimizes layer-to-PE and PE-to-tier mapping at the design time for a given DNN workload and any other MOO solver can also be used to the same effect.

TABLE II
PIM ARCHITECTURE SPECIFICATIONS

64 PEs distributed over 4 tiers (16 PEs/tier), 4 tiles/PE	
ReRAM Tile	96 SAR ADCs (8-bits), 128×96 DACs (1-bit), 96 crossbars, 128×128 crossbar array, 2-bit/cell resolution, 0.40 mm^2
FeFET Tile	256×48 S/A (1-bit), 48 crossbars, 256×256 crossbar array, 1-bit/cell resolution, 0.40 mm^2
SRAM Tile	6T, 1-bit-cell, 256 S/A, 8KB SRAM array (256×256), 9 column/row-decoder, 9 SRAM arrays, 0.40 mm^2

IV. Experimental Results and Analysis

In this section, we present comprehensive experimental results for the HuNT-enabled 3-D heterogeneous PIM architecture for DNN training.

A. Experimental Setup

The HuNT optimization phase (described in Algorithm 1) is executed for 100 iterations, as this is sufficient to ensure the convergence of the AMOSA-based MOO. Algorithm 1 is executed at the design time; hence the time overhead is a one-time cost. The overall time complexity of the adapted AMOSA-based MOO solver is given by $O(T \times N \times (M + \log(N)))$, where T is the total iterations of the algorithm, N is the maximum number of nondominated solutions stored in the Archive, and M is the number of design objectives [29]. HuNT generates the optimized neural layer-to-PE mapping, and the corresponding PE-to-tier mapping, which is then mapped to the proposed 3-D heterogeneous PIM-based architecture.

3-D Heterogeneous PIM Architecture: The PIM architecture considered in this work consists of a total of 64 PEs distributed over four planar tiers and connected using TSV-based vertical links. Each PIM-based PE has its unique configuration, such as crossbar size, cell resolution, number of crossbars/6T cells, etc., as shown in Table II. We consider an *iso-PE area setting*, such that all PEs (irrespective of their device type) have the same area but different amount of storage and compute capability. Considering the storage capacity of each PIM-based PE, the HuNT-enabled architecture can have a storage capacity of up to ~75 MB. Each planar tier consists of 16 PEs of a particular PIM device type (SRAM/ReRAM/FeFET).

TABLE III
DNN Workloads With CIFAR-10 Dataset

	# Layers	Learning rate	Batch size	# Params
VGG11	11	0.01	64	1.5M
VGG16	16	0.01	64	2.2M
ResNet18	18	0.05	128	1.1M
ResNet34	34	0.05	128	2M
DenseNet40	40	0.01	128	900K

The area, energy, and latency of the SRAM, ReRAM, and FeFET devices and their associated peripheral circuits, such as ADC, sense-amps (S/A), DACs, buffers, column-/row-decoders, and nonlinear activation units (ReLU), were modeled via NeuroSim [24]. The connectivity between PEs follows the 3-D mesh topology and the workload-dependent inter-PE traffic is given as input to BookSim to estimate communication power and latency [30]. We employ HotSpot's default ambient temperature setting of 300 K to conduct thermal analysis with the power traces generated using NeuroSim and BookSim [28]. Finally, we model the thermal effects [shown in (6) and (7)] on the DNN accuracy using the PyTorch wrapper in NeuroSim for the different DNN models and datasets considered in this work. Following prior work, we use 16-bit fixed-point precision for the storage and computation of the DNN weights and activations in the forward pass, and 32-bit floating-point precision for the weight- and activation-gradient computation in the back-propagation phase [31]. The 3-D heterogeneous architecture utilizes a multicast-enabled 3-D mesh NoC as the interconnection backbone for communicating between the PEs during DNN training [23]. In our experimental evaluation, we consider energy-efficiency (TOPS/W) and compute-efficiency (TOPS/mm^2) as the two relevant performance metrics that capture the latency, area, and power objectives considered in Section III of this work.

DNN Models and Datasets: We evaluate the performance of the HuNT design optimization framework considering the CIFAR-10, CIFAR-100, and TinyImageNet datasets with five diverse DNN models, namely: VGG11, VGG16, ResNet18, ResNet34, and DenseNet40. Table III shows the characteristics and parameters of the DNN models executed on the HuNT-enabled 3-D heterogeneous PIM architecture. As shown in Table III, the largest network considered in this work (VGG16) has about 2.2M parameters which requires ~4.4 MB of storage, hence it can be easily stored on the HuNT-enabled 3-D heterogeneous architecture (with a storage capacity of up to ~75 MB) along with its activations and layer-wise gradients. However, for larger networks where the neural network size exceeds the total storage capacity of the PEs in the system, then we need to read/write weights and activations from/to main memory (DRAM). As a result, there will be an additional latency penalty corresponding to that. However, the layer-to-PE and PE-to-tier mapping obtained from the HuNT optimization framework is unimpacted by the off-chip memory accesses in the case of very large DNNs. In this work, we train the DNN models on the HuNT-enabled 3-D heterogeneous PIM architecture for 200 epochs using the Stochastic Gradient Descent method to ensure their training convergence without overfitting.

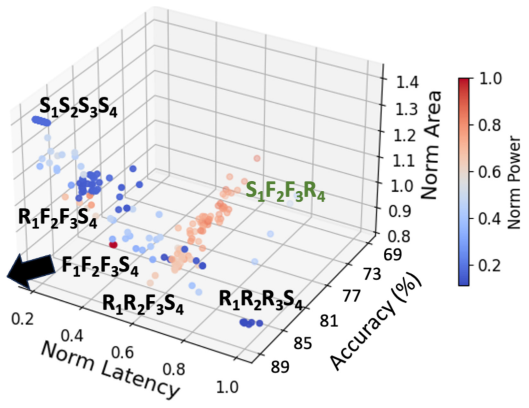

Fig. 3. Layer-to-PE and PE-to-tier mapping tradeoffs while running the DNN training task for ResNet34 model on the CIFAR-10 dataset.

B. Layer-to-PE and PE-to-Tier Mapping Tradeoffs

The neural layer-to-PE and PE-to-tier mapping affect the overall latency, power, area, and DNN accuracy. The aim of the HuNT framework is to determine the optimum configuration of the heterogeneous 3-D manycore architecture that achieves a suitable balance among all these metrics. Fig. 3 presents the Pareto front considering the above-mentioned design objectives, while executing the training task on ResNet34 model using the CIFAR-10 dataset as an example. Recall, the PE-to-tier mapping is represented by $\alpha = [t_1, t_2, \ldots, t_z]$, where a planar tier t_z has PEs of one device type—ReRAM (R), FeFET (F), and SRAM (S). Fig. 3 shows a representative Pareto-optimal set of designs D^* highlighted in black. It should be noted that all the Pareto-optimal configurations with heterogeneous PEs have the SRAM devices at the bottom tier, away from the heat sink, which is used for the gradient calculation during back-propagation. Further, to minimize the on-chip hardware resources for gradient computation, we do not need to process all the layers simultaneously, but just perform layer-by-layer weight gradient computation, following prior work [24]. Therefore, one tier of SRAM-based PEs is enough to support the layer with largest size of activation gradients for the DNN models considered here. Due to the necessity of the SRAM tier for the back-propagation, the homogeneous configurations where we have only one type of NVM PIM device like FeFET or ReRAM are: $[F_1, F_2, F_3, S_4]$ and $[R_1, R_2, R_3, S_4]$. Alternatively, the homogeneous configuration with only SRAM device is: $[S_1, S_2, S_3, S_4]$. All the design objectives shown in Fig. 3 are normalized with respect to a mapping corresponding to $\alpha = [S_1, F_2, F_3, R_4]$ (shown in green), since it has the worst DNN accuracy. As mentioned earlier, impact of thermal noise on ReRAM-based PEs is more severe compared to FeFET- or SRAM-based PEs. Thus, the mapping $[S_1, F_2, F_3, R_4]$ has the worst DNN accuracy because 1) the high power consuming FeFET-based PEs on two planar tiers lead to thermal hotspots and 2) ReRAM-based PEs are mapped to the planar tier farthest from the heat sink. On the other hand, candidate mappings with all thermal noise resilient SRAM-based PEs, i.e., $[S_1, S_2, S_3, S_4]$, achieve the highest DNN accuracy, but at the cost of extremely high area. The FeFET-based PEs contribute to high power density in the mapping corresponding to $[F_1, F_2, F_3, S_4]$, resulting in peak temperature of 380 K and

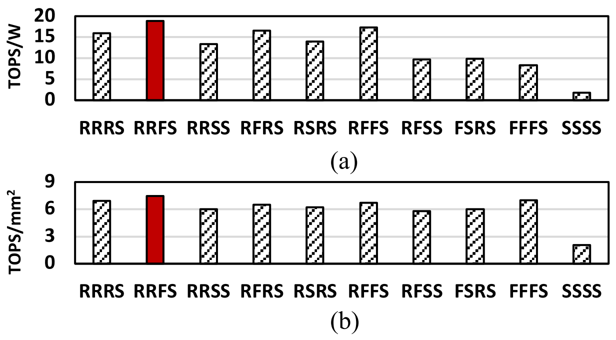

Fig. 4. Comparison of various Pareto-optimal layer-to-PE and PE-to-tier mappings in terms of (a) TOPS/W and (b) TOPS/mm^2, while running training task for VGG16 model on the CIFAR-10 dataset as an example.

lower DNN accuracy compared to $[R_1, R_2, R_3, S_4]$. However, the mapping $[R_1, R_2, R_3, S_4]$ incurs high write latency due to predominantly ReRAM-based PEs when compared to mappings on predominantly SRAM- or FeFET-based PEs. Thus, all homogeneous architectures score high in one specific design metric neglecting the others. On the other hand, heterogeneous 3-D architectures, such as $[R_1, F_2, F_3, S_4]$ and $[R_1, R_2, F_3, S_4]$, exploit device heterogeneity with optimal layer-to-PE and PE-to-tier mapping, and achieve suitable tradeoffs between power, latency, area, and DNN accuracy.

Next, we implement the HuNT-enabled Pareto-optimal set of designs D^* and evaluate their performance in realistic settings. Fig. 4(a) and (b) shows the comparative performance evaluation of the architectures in terms of energy-efficiency (TOPS/W) and compute-efficiency (TOPS/mm^2) for DNN training task on VGG16 model with CIFAR-10 dataset as an example. As shown in Fig. 4, neural layer mapping corresponding to a homogeneous SRAM-based PE configuration, i.e., $[S_1, S_2, S_3, S_4]$, leads to the lowest TOPS/W and TOPS/mm^2 due to higher power and area consumption when compared to FeFET- and ReRAM-based architectures. Similarly, $[F_1, F_2, F_3, S_4]$ achieves low TOPS/W due to high power FeFET-based PEs. As shown in Fig. 4, the layer-to-PE and PE-to-tier mapping corresponding to $[R_1, R_2, F_3, S_4]$ (highlighted in red) achieves highest TOPS/W and TOPS/mm^2 compared to rest of the Pareto-optimal candidate mappings. This mapping utilizes the ReRAM and FeFET-based PEs (on planar tiers 1–3) for low precision computation in the forward phase and SRAM-based PEs (on planar tier 4) for high precision gradients computation in the back-propagation phase. Further, the DNN layers processing high number of activations are mapped to dense ReRAM-based PEs, resulting in higher TOPS/mm^2 and closer to the SRAM tier, reducing the communication energy specifically during the back-propagation phase. Hence, this results in higher TOPS/W.

Next, we discuss the DNN layers' characteristics and their role in layer-to-PE and PE-to-tier mapping for the best-performing $[R_1, R_2, F_3, S_4]$ architecture. Fig. 5 shows layer-wise mapping on to PEs and 3-D planar tiers for training DenseNet40 and VGG16 models with CIFAR-10 dataset as an example. As discussed earlier, high precision gradients are calculated in the bottom tier (tier S4) and the forward phase computation is executed on tiers 1–3 of the $[R_1, R_2, F_3, S_4]$ architecture. As shown in Fig. 5(a), initial layers in DenseNet40 process higher number of activations than the latter layers and need more crossbars to store weights and activations. Therefore, these layers are mapped to dense, low power ReRAM-based PEs (R2) as well as closer to tier S4 for faster exchange of gradients. On the contrary, latter layers with comparatively fewer activations and smaller kernels, are mapped on tier F3 (layers 14–24) and R1 (layers 30–40). However, as shown in Fig. 5(b), the layer-wise characteristics of VGG16 are different than that of DenseNet40, i.e., initial layers process higher number of activations but have less crossbars requirement for storage and computation, due to the layers' input/output feature map and kernel size. Thus, the initial layers of VGG16 are mapped to FeFET-based PEs on tier F3 that have low latency but less dense when compared to ReRAMs. On the contrary, the middle layers consist of wider kernels and require more crossbars. Thus, these layers are mapped to dense, low power ReRAM-based PEs on tier R2. This highlights the importance of considering DNN layers' characteristics while finding optimal layer-to-PE and PE-to-tier mapping to achieve high compute- and energy-efficiency.

C. Overall Performance Evaluation

In this section, we present a thorough performance evaluation of the HuNT-enabled DNN layer-to-PE and PE-to-tier mapping for the proposed 3-D heterogeneous PIM architecture during DNN training. Fig. 6(a)–(c) compares the energy-, compute-efficiency, and accuracy of the HuNT-enabled 3-D heterogeneous architecture (*simply referred to as HuNT* here after) with the homogeneous and existing heterogeneous counterparts for all DNN workloads considered in this work with the CIFAR-10 dataset, respectively. For this comparison, the homogenous configurations are $[F_1, F_2, F_3, S_4]$, $[R_1, R_2, R_3, S_4]$, and $[S_1, S_2, S_3, S_4]$ as mentioned earlier. The existing heterogeneous counterparts considered in our comparative performance evaluation include the HyperX and AccuReD architectures [22], [23]. As discussed in the related work, HyperX leverages both ReRAM (R) and SRAM (S), while AccuReD leverages ReRAM- and GPU-based PEs to achieve high-performance DNN training. In our comparative performance evaluation with respect to HuNT, we use the two tiers of ReRAM and two GPU tiers $[R_1, R_2, \mathrm{GPU}_3, \mathrm{GPU}_4]$ configuration, and the $[R_1, S_2, S_3, S_4]$ configuration for the AccuReD and HyperX architectures, respectively [22], [23].

As shown in Fig. 6(a) and (b), HuNT achieves up to 20 TOPS/W and 10.73 TOPS/mm^2 on the CIFAR-10 dataset which corresponds to a ∼10× and ∼8× improvement in energy- and compute-efficiency, respectively, over the all-SRAM homogenous counterpart. As shown earlier in Table I, and corroborated in the literature, SRAM-based PIM architectures generally suffer from low energy- and compute-efficiency due to their high leakage power and significant area overhead, respectively [3]. Hence, they achieve a relatively low energy- and compute-efficiency of 2.2 TOPS/W and 1.1 TOPS/mm^2 on average across all DNN models as shown in Fig. 6(a) and (b), respectively. HuNT exploits device heterogeneity and DNN workload awareness to achieve up to a 1.2× and 1.3× improvement in TOPS/W and TOPS/mm^2, respectively, over

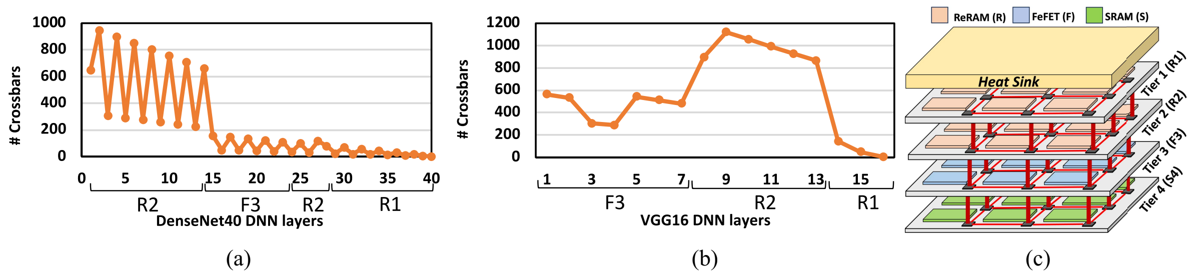

Fig. 5. Layer-to-PE and PE-to-tier mapping for DNN training task on (a) DenseNet40 and (b) VGG16 models with CIFAR-10 dataset on (c) optimized $[R_1, R_2, F_3, S_4]$ architecture. Here, R1 and R2 refer to Tiers 1 and 2 with ReRAM-based PEs, respectively, F3 refers to Tier 3 with FeFET-based PEs, and S4 refers to Tier 4 with SRAM-based PEs.

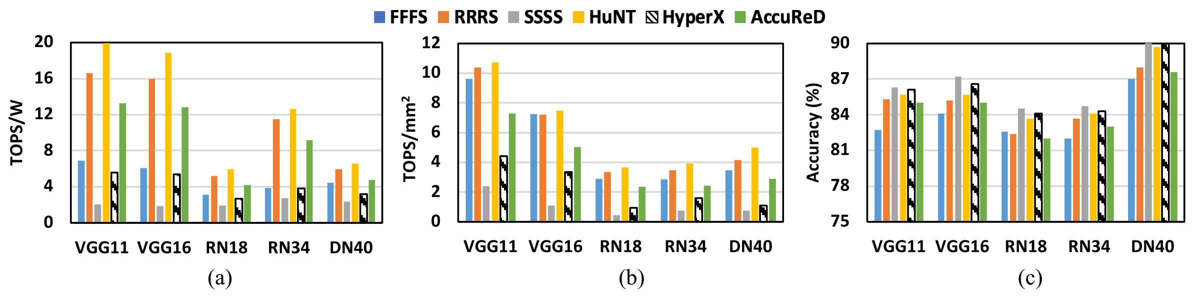

Fig. 6. Performance evaluation of the HuNT-enabled 3-D PIM architecture with state of the art in terms of (a) energy-efficiency (TOPS/W), (b) compute-efficiency (TOPS/mm^2), and (c) accuracy of DNN workloads executed on the CIFAR-10 dataset. Here, for brevity, we use FFFS, RRRS, and SSSS to refer to $[F_1, F_2, F_3, S_4]$, $[R_1, R_2, R_3, S_4]$, and $[S_1, S_2, S_3, S_4]$ homogeneous configurations, respectively.

the homogeneous ReRAM configuration ($[R_1, R_2, R_3, S_4]$). Similarly, HuNT achieves an improvement of up to 2.6× and 1.5× in TOPS/W and TOPS/mm^2, respectively, over the homogeneous FeFET configuration ($[F_1, F_2, F_3, S_4]$) on the CIFAR-10 dataset. Overall, HuNT outperforms HyperX and AccuReD by 3.1× and 1.4×, respectively, on average in terms of energy-efficiency, and by 2.7× and 1.5×, respectively, on average in terms of compute efficiency on the CIFAR-10 dataset. This is because the high power and area of the SRAM-based PEs and the GPU-based PEs in HyperX and AccuReD, respectively, make them less compute- and energy-efficient.

As shown in Fig. 6(c), we compare HuNT with the homogenous and heterogeneous counterparts in terms of the accuracy. Here, the all-SRAM configuration achieves the highest accuracy due its high reliability, and less vulnerability to thermal issues in the 3-D architecture [23]. However, the homogeneous FeFET and ReRAM counterparts suffer up to 4% and 2.5% accuracy loss. This is due to high power consumption of FeFET-based PEs and limited thermal endurance of ReRAM-based PEs when placed away from the heat sink. Overall, HuNT achieves less than 1% accuracy drop compared to the all-SRAM counterpart. In summary, our performance evaluation demonstrates that the HuNT-enabled 3-D heterogeneous PIM architecture achieves high energy- and compute-efficiency over the homogenous counterparts and the existing heterogeneous PIM-based architectures (AccuReD and HyperX). Overall, the HuNT-enabled 3-D PIM architecture achieves the highest TOPS/W and TOPS/mm^2 with negligible loss in DNN accuracy.

D. Transferability Across Datasets

In this section, we demonstrate that the HuNT-enabled optimized layer-to-PE and PE-to-tier mapping (d_{best}) obtained using the CIFAR-10 dataset can be transferred to another dataset for training on 3-D heterogeneous PIM architecture without compromising the DNN training accuracy, and overall performance. Here, the d_{best} for a given DNN workload is generated with a *source* dataset via the HuNT framework, and then mapped to the 3-D heterogeneous architecture for training using a *target* dataset. Figs. 7 and 8 demonstrate the transferability of d_{best} generated using the CIFAR-10 dataset (as the source dataset) to the CIFAR-100 and TinyImageNet datasets (as the target datasets), respectively. In Figs. 7 and 8, we consider d_{best} generated using CIFAR-100 and TinyImageNet, respectively, via the HuNT framework in each case as the *baseline*. In this work, we compare the performance of d_{best} obtained using the CIFAR-10 dataset with respect to the baselines in terms of energy-efficiency (TOPS/W), compute-efficiency (TOPS/mm^2), and the final DNN test accuracy as shown in Figs. 7(a) and 8(a), Figs. 7(b) and 8(b), and Figs. 7(c) and 8(c), respectively. Here, the configurations are denoted as $D_S \rightarrow D_T$, where D_S represents the "source" dataset and D_T represents the "target" dataset. In our analysis, we consider the CIFAR-10→CIFAR-100 and CIFAR-10→TinyImageNet configurations for the sake of brevity. However, it is worth noting that the HuNT framework is compatible with other image datasets, and the results shown here are reproducible for other configurations.

As shown in Fig. 7(a) and (b), we observe less than an average of 2% and 1.5% loss in energy- and compute-efficiency, respectively, compared to the baseline across all the DNN models for the CIFAR-10→CIFAR-100 configuration. Similarly, we also observe an average of 3.1% and 1.9% loss in energy- and compute-efficiency, respectively, compared to the baseline for the CIFAR-10→TinyImageNet configurations, respectively, as shown in Fig. 8(a) and (b). Overall, we

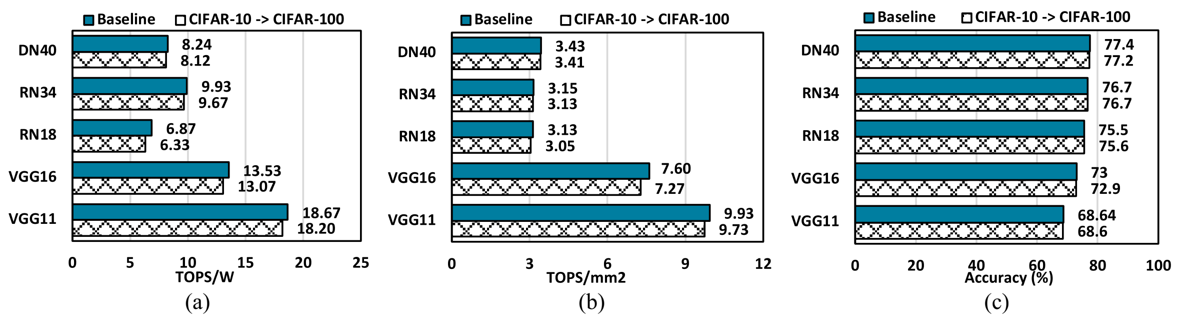

Fig. 7. Transferability from CIFAR-10 to CIFAR-100 dataset for (a) energy-efficiency (TOPS/W), (b) compute-efficiency (TOPS/mm^2), and (c) accuracy.

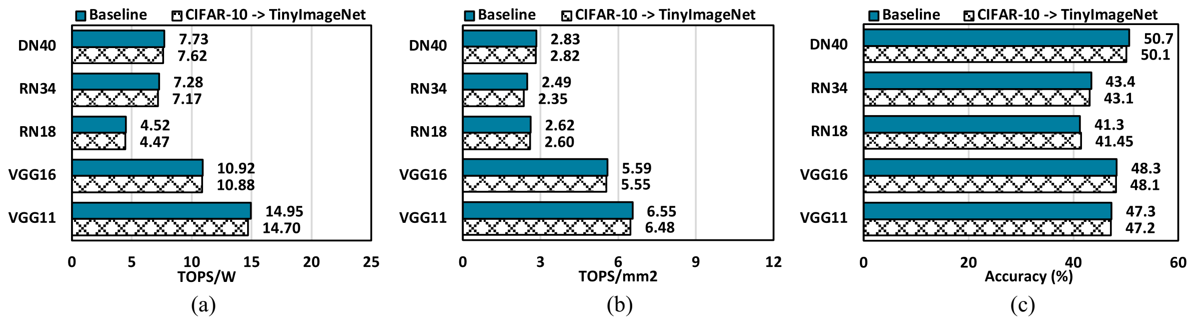

Fig. 8. Transferability from CIFAR-10 to TinyImageNet dataset for (a) energy-efficiency (TOPS/W), (b) compute-efficiency (TOPS/mm^2), and (c) accuracy.

observe a negligible accuracy loss of less than 1% across all DNN models considered in both the CIFAR-10→CIFAR-100 and CIFAR-10→TinyImageNet configurations as shown in Figs. 7(c) and 8(c), respectively. Here, the transferability of the optimal neural layer mapping of d_{best} across datasets is possible because the general DNN model behavior is often transferable between datasets. This idea is similar to transfer learning, where a model trained on one dataset can be reused with slight changes for another dataset [32]. Hence, the neural layer mapping of d_{best} for a given DNN model can also be used with other datasets, and achieve similar levels of performance (energy- and compute-efficiency) with negligible accuracy loss. However, it is worth noting that the absolute values of the achievable performance in terms of TOPS/W and TOPS/mm^2 vary across datasets due to their unique characteristics. For example, the TinyImageNet dataset generates more activations during training compared to the CIFAR-10 dataset. This requires more PEs. Hence, for the same system configuration, HuNT achieves lower compute- and area-efficiencies for TinyImageNet compared to both CIFAR-10 and CIFAR-100 datasets. The dataset characteristics influence the absolute achievable performance. However, the overall trend is agnostic to the dataset. In addition, the transferability of d_{best} across datasets eliminates the cost of implementing repeated MOO for more complex datasets. In essence, this further demonstrates the scalability and versatility of the HuNT-enabled optimized layer-to-PE and PE-to-tier mapping (d_{best}) to other datasets for DNN training on 3-D heterogeneous PIM accelerators.

E. Lifetime and Endurance Analysis

Lifetime and write endurance of NVM-based PIM devices are crucial for DNN training due to significant number of write operations required for the weight- and activation-gradient calculations as well as weight updates in the back-propagation phase. For our analysis, we consider realistic write endurance limit for the FeFET-, ReRAM-, and SRAM-based PEs reported in prior work, as shown in Table I. As discussed earlier, the HuNT-enabled layer-to-PE mapping maps the weights in the DNN layers to both ReRAM- and FeFET-based PEs, and the weight- and activation-gradient computation is performed on SRAM-based PEs (i.e., the $[R_1, R_2, F_3, S_4]$ configuration). Therefore, the weights mapped to the ReRAM- and FeFET-based PEs need to be reprogrammed during the weight update phase. However, ReRAM and FeFET devices suffer from low write endurance, which limits the number of times that they can be reprogrammed before they fail due to faults [17].

In Fig. 9, we present a comparative performance trade-off analysis between the energy-efficiency and endurance of the homogeneous architectures, and the HuNT-enabled heterogeneous architecture ($[R_1, R_2, F_3, S_4]$) executing the VGG-11 DNN workload with the CIFAR-10 dataset. We observe that beyond the endurance limit for each device, the achievable performance (TOPS/W) begins to reduce, as the number of resources (PEs) available to perform reliable computation reduces due to failures of the NVM devices. The $[S_1, S_2, S_3, S_4]$ configuration achieves the lowest TOPS/W due to its significant leakage power, however, it has the highest endurance. Overall, the HuNT-enabled heterogeneous architecture achieves an improvement of 10×, 3×, and 1.2× in terms of TOPS/W compared to the homogeneous SRAM-, FeFET-, and ReRAM-based architectures, respectively. At the same time, HuNT achieves similar write endurance as the homogeneous configurations with at least one type of NVM device ($[R_1, R_2, R_3, S_4]$ and $[F_1, F_2, F_3, S_4]$).

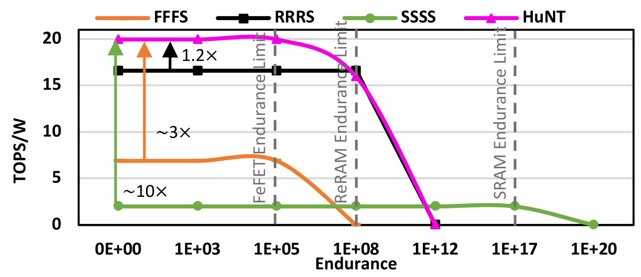

Fig. 9. Comparison of HuNT-enabled architecture with other homogeneous architectures in terms of energy-efficiency (TOPS/W) and endurance for training on VGG11 model with CIFAR-10 dataset as an example.

V. Conclusion

PIM-based architectures enable high-performance and energy-efficient hardware accelerators for DNN training. However, each PIM device has specific advantages and drawbacks. Hence, a heterogeneous architecture that combines multiple PIM devices in a single system is necessary to achieve the suitable balance between all the required design metrics. A 3-D architecture enables the design of such a heterogeneous platform where each planar tier consists of PEs designed with one type of device. This also avoids the fabrication challenges of integrating disparate technologies on a single tier. In this work, we propose the HuNT framework, which finds an optimal layer-to-PE and PE-to-tier mapping for 3-D PIM-based heterogeneous architectures. Overall, the HuNT-enabled 3-D heterogeneous architecture achieves up to a 10× and 8× improvement in energy- and compute-efficiency, respectively, over the homogenous counterparts and existing heterogeneous PIM-based architectures without compromising accuracy.

References

[1] W. Liu et al., "A survey of deep neural network architectures and their applications," *Neurocomputing*, vol. 234, pp. 11–26, Apr. 2017.

[2] L. Song, X. Qian, L. Hai, and Y. Chen, "PipeLayer: A pipelined ReRAM-based accelerator for deep learning," in *Proc. IEEE HPCA*, 2017, pp. 541–552.

[3] K. Roy, I. Chakraborty, M. Ali, A. Ankit, and A. Agrawal, "In-memory computing in emerging memory technologies for machine learning: An overview," in *Proc. IEEE DAC*, 2020, pp. 1–6.

[4] A. Shafiee et al., "ISAAC: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars ali," in *Proc. ISCA*, 2016, pp. 14–26.

[5] T. Soliman et al., "First demonstration of in-memory computing crossbar using multi-level cell FeFET," *Nat. Commun.*, vol. 14, no. 1, p. 6348, 2023.

[6] Y. Long et al., "A ferroelectric FET-based processing-in-memory architecture for DNN acceleration," *IEEE J. Explor. Solid-State Computat. Devices Circuits*, vol. 5, no. 2, pp. 113–122, Dec. 2019.

[7] A. Keshavarzi, K. Ni, W. Van Den Hoek, S. Datta, and A. Raychowdhury, "FerroElectronics for edge intelligence," *IEEE Micro*, vol. 40, no. 6, pp. 33–48, Nov./Dec. 2020.

[8] A. Yusuf, T. Adegbija, and D. Gajaria, "Domain-specific STT-MRAM-based in-memory computing: A survey," *IEEE Access*, vol. 12, 2024, pp. 28036–28056.

[9] G. Murali, X. Sun, S. Yu, and S. K. Lim, "Heterogeneous mixed-signal monolithic 3-D in-memory computing using resistive RAM," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 29, no. 2, pp. 386–396, Feb. 2021.

[10] X. Peng et al., "Benchmarking monolithic 3D integration for compute-in-memory accelerators: Overcoming ADC bottlenecks and maintaining scalability to 7nm or beyond," in *Proc. IEEE IEDM*, 2020, pp. 30.4.1–30.4.4.

[11] A. Kaul et al., "3-D heterogeneous integration of RRAM-based compute-in-memory: Impact of integration parameters on inference accuracy," *IEEE Trans. Electron Devices*, vol. 70, no. 2, pp. 485–492, Feb. 2023.

[12] M. Yayla et al., "FeFET-based binarized neural networks under temperature-dependent bit errors," *IEEE Trans. Comput.*, vol. 71, no. 7, pp. 1681–1695, Jul. 2022.

[13] X. Yang et al., "Multi-objective optimization of ReRAM crossbars for robust DNN inferencing under stochastic noise," in *Proc. IEEE/ACM ICCAD*, 2021, pp. 1–9.

[14] A. Bhattacharjee, A. Moitra, and P. Panda, "HyDe: A hybrid PCM/FeFET/SRAM device-search for optimizing area and energy-efficiencies in analog IMC platforms," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 13, no. 4, pp. 1073–1082, Dec. 2023.

[15] Y. Sun et al., "CREAM: Computing in ReRAM-assisted energy- and area-efficient SRAM for reliable neural network acceleration," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 70, no. 8, pp. 3198–3211, Aug. 2023.

[16] G. Krishnan et al., "Hybrid RRAM/SRAM in-memory computing for robust DNN acceleration," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 41, no. 11, pp. 4241–4252, Nov. 2022.

[17] W. Wen, Y. Zhang, and J. Yang, "ReNEW: Enhancing lifetime for ReRAM crossbar based neural network accelerators," in *Proc. IEEE ICCD*, 2019, pp. 487–496.

[18] C. Eckert et al., "Neural cache: Bit-serial in-cache acceleration of deep neural networks," in *Proc. 45th ISCA*, 2018, pp. 383–396.

[19] S. Spetalnick and A. Raychowdhury, "A practical design-space analysis of compute-in-memory with SRAM," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 69, no. 4, pp. 1466–1479, Apr. 2022.

[20] S. Roy, M. Ali, and A. Raghunathan, "PIM-DRAM: Accelerating machine learning workloads using processing in commodity DRAM," *IEEE J. Emerg. Sel. Topics Circuits Syste.*, vol. 11, no. 4, pp. 701–710, Dec. 2021.

[21] M. R. Haq Rashed, S. K. Jha, and R. Ewetz, "Hybrid analog-digital in-memory computing," in *Proc. IEEE/ACM ICCAD*, 2021, pp. 1–9.

[22] A. Kosta et al., "HyperX: A hybrid RRAM-SRAM partitioned system for error recovery in memristive Xbars," in *Proc. DATE*, 2022, pp. 88–91.

[23] B. K. Joardar, J. R. Doppa, P. P. Pande, H. Li, and K. Chakrabarty, "AccuReD: High accuracy training of CNNs on ReRAM/GPU heterogeneous 3D architecture," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 40, no. 5, pp. 971–984, May 2021.

[24] X. Peng et al., "DNN+NeuroSim V2.0: An end-to-end benchmarking framework for compute-in-memory accelerators for on-chip training," 2020, *arXiv:2003.06471*.

[25] N. Ye et al., "Improving the robustness of analog deep neural networks through a Bayes-optimized noise injection approach," *Commun. Eng.*, vol. 2, no. 1, p. 25, 2023.

[26] Y. Qin, Z. Yan, W. Wen, X. S. Hu, and Y. Shi, "Negative feedback training: A novel concept to improve robustness of NVCiM DNN accelerators," 2023, *arXiv:2305.14561*.

[27] J. Cong, J. Wei, and Y. Zhang, "A thermal-driven floorplanning algorithm for 3D ICs," in *Proc. IEEE/ACM ICCAD*, 2004, pp. 306–313.

[28] R. Zhang, M. Stan, and K. Skadron, "Hotspot 6.0: Validation, acceleration and extension," IBM T. J. Watson Res. Center, Univ. Virginia, Charlottesville, VA, USA, Rep. CS-2015-04, 2015.

[29] S. Bandyopadhyay, S. Saha, U. Maulik, and K. Deb, "A simulated annealing-based multiobjective optimization algorithm: AMOSA," *IEEE Trans. Evol. Comput.*, vol. 12, no. 3, pp. 269–283, Jun. 2008.

[30] N. Jiang et al., "A detailed and flexible cycle-accurate network-on-chip simulator," in *Proc. IEEE ISPASS*, 2013, pp. 86–96.

[31] H. Jin et al., "ReHy: A ReRAM-based digital/analog hybrid PIM architecture for accelerating CNN training," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 11, pp. 2872–2884, Nov. 2022.

[32] C. O. Ogbogu et al., "Accelerating graph neural network training on ReRAM-based PIM architectures via graph and model pruning," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 42, no. 8, pp. 2703–2716, Aug. 2023.

[33] D. Niu et al., "Design of cross-point metal-oxide ReRAM emphasizing reliability and cost," in *Proc. IEEE/ACM ICCAD*, 2013, pp. 17–23.