MaxNVM: Maximizing DNN Storage Density and Inference Efficiency with Sparse Encoding and Error Mitigation

Lillian Pentecost
lillian_pentecost@g.harvard.edu
Harvard University
Cambridge, Massachusetts

Marco Donato
Harvard University
Cambridge, Massachusetts

Brandon Reagen
New York University
New York, New York

Udit Gupta
Harvard University
Cambridge, Massachusetts

Siming Ma
Harvard University
Cambridge, Massachusetts

Gu-Yeon Wei
Harvard University
Cambridge, Massachusetts

David Brooks
Harvard University
Cambridge, Massachusetts

ABSTRACT

Deeply embedded applications require low-power, low-cost hardware that fits within stringent area constraints. Deep learning has many potential uses in these domains, but introduces significant inefficiencies stemming from off-chip DRAM accesses of model weights. Ideally, models would fit entirely on-chip. However, even with compression, memory requirements for state-of-the-art models make on-chip inference impractical. Due to increased density, emerging eNVMs are one promising solution.

We present MaxNVM, a principled co-design of sparse encodings, protective logic, and fault-prone MLC eNVM technologies (i.e., RRAM and CTT) to enable highly-efficient DNN inference. We find bit reduction techniques (e.g., clustering and sparse compression) *increase* weight vulnerability to faults. This limits the capabilities of MLC eNVM. To circumvent this limitation, we improve storage density (i.e., bits-per-cell) with minimal overhead using protective logic. Tradeoffs between density and reliability result in a rich design space. We show that by balancing these techniques, the weights of large networks are able to reasonably fit on-chip. Compared to a naive, single-level-cell eNVM solution, our highly-optimized MLC memory systems reduce weight area by up to 29×. We compare our technique against NVDLA, a state-of-the-art industry-grade CNN accelerator, and demonstrate up to 3.2× reduced power and up to 3.5× reduced energy per ResNet50 inference.

CCS CONCEPTS

• **Hardware** → *Memory and dense storage.*

KEYWORDS

eNVM, memory systems, neural networks, RRAM, CTT

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
MICRO-52, October 12–16, 2019, Columbus, OH, USA
© 2019 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-6938-1/19/10.
https://doi.org/10.1145/3352460.3358258

ACM Reference Format:
Lillian Pentecost, Marco Donato, Brandon Reagen, Udit Gupta, Siming Ma, Gu-Yeon Wei, and David Brooks. 2019. MaxNVM: Maximizing DNN Storage Density and Inference Efficiency with Sparse Encoding and Error Mitigation. In *The 52nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-52), October 12–16, 2019, Columbus, OH, USA.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3352460.3358258

1 INTRODUCTION

DNNs are in use everywhere from self-driving cars to wireless sensor nodes and implanted medical devices [16, 40, 43, 50, 52]. In these deeply-embedded environments, efficiency is paramount. For state-of-the-art DNN hardware accelerators, fetching weights from DRAM is a main performance and energy bottleneck, limiting inference efficiency. Ideally, DNNs weights would be stored entirely on-chip. However, even with aggressive weight compression, the capacity requirements are unrealistic for storage in on-chip SRAM.

Emerging embedded non-volatile memory (eNVM) technologies are one promising solution for eliminating DRAM inefficiencies. eNVMs provide high-capacity, low read-latency storage and can be significantly more dense than SRAM. Many eNVMs, including RRAM [68], PCM [13], and CTT [18], also have multi-level cell (MLC) capabilities, allowing multiple bits to be packed into a single device to further increase density. eNVMs are not perfect—two main drawbacks are decreased reliability and high write latency. Achieving the ultra-high densities necessary to eliminate DRAM for DNN inference requires aggressive MLC eNVM designs. However, MLC designs incur high memory access fault rates. Second, eNVMs offer fast read access, typically on the same order as SRAM. However, writes can be orders of magnitude slower, as they alter the physical property of the storage material. DNNs are robust to these issues because they often require infrequent updates and are implicitly fault tolerant, so we hypothesize that MLC eNVMs can significantly improve DNN inference efficiency by storing weights on-chip and eliminating DRAM accesses.

This paper demonstrates that MLC eNVMs can be used for highly-efficient DNN inference. To fit weights on-chip requires the consideration of many factors, resulting in a large design space. We present MaxNVM, a principled co-design method to consider algorithm-to-circuit effects and maximize benefit. As an efficient baseline, we start at the algorithmic level, applying clustering, pruning, and sparse encodings (CSR and BitMask) for DNN weights. To accommodate fault-prone MLC storage, we co-optimize the sparsity and encoding of DNNs with the storage density of MLC eNVMs, which directly impacts the reliability of the storage medium, to reduce memory requirements without sacrificing accuracy. We find sparsely encoded weights are more vulnerable than dense weights, which limits MLC eNVM density. Thus, we further push the storage density by applying protective mechanisms, ECC and IndexSync (proposed), when using more programmed levels per MLC.

To demonstrate the efficacy of our co-design approach on the evolving eNVM landscape, we evaluate two fundamentally different technologies: RRAM and CTT. Both are dense, MLC-capable, and can be made CMOS compatible. CTT has fast read latency and is very low-power, but incurs inordinately long write latency. Compared to CTT, RRAM has a faster write latency, but is less energy-efficient. eNVM models are built using NVSim [20]. RRAM cell parameters and fault models are derived from published work, and we construct models to represent projected [73] and demonstrated [8, 42] RRAM scaling. CTT parameters and fault distributions are derived from a measured chip prototype.

We demonstrate the benefits of MaxNVM by conducting memory system studies using NVDLA—a state-of-the-art, industrial-grade CNN accelerator. We demonstrate that even large models, e.g., VGG16 and ResNet50, can reasonable fit on-chip with our co-design approach. CTT results in the lowest energy-per-inference design point at 3.2× reduced energy per inference. If weights are to be updated more frequently, RRAM presents a compromise of writing weights orders of magnitude faster while giving up approximately 20% energy efficiency. In a case study, we constrain all on-chip memory to fit within 1mm^2 and sweep the percentage given to SRAM vs. eNVM. The study concludes a balance of eNVM and SRAM is ideal to maximize performance and efficiency.

This paper makes the following research contributions:

(1) We are the first to consider fault-prone MLC eNVMs and sparse encoded weights. Both reduce memory requirements, however we find a tension between the two: sparse encoding increases fault vulnerability, limiting the efficacy of MLCs. To reduce overall memory footprint, we first sparse encode weights to save raw bits then set the levels-per-cell to the highest configuration without accuracy loss.

(2) To further increase storage density, we improve DNN fault tolerance using protective logic. We consider IndexSynchronization, a proposed fault mitigation technique, and ECC. With judicious use, we show the total number of required memory cells to store DNN weights decreases by up to 22% with our proposed technique, and ECC overhead is never more than 1% of total DNN storage. Optimal MLC designs provide up to 29× area reduction relative to SLC eNVM.

(3) We propose new memory systems for NVDLA [62], an industry-grade CNN accelerator, based on RRAM and CTT. Using our co-design approach, weights for state-of-the-art CNNs can fit on-chip, eliminating the need for DRAM. Compared to the baseline NVDLA implementation, MLC eNVMs enable up to 3.5× lower energy per inference and 3.2× lower power, enabling entirely on-chip ResNet50 inference in about 2mm^2.

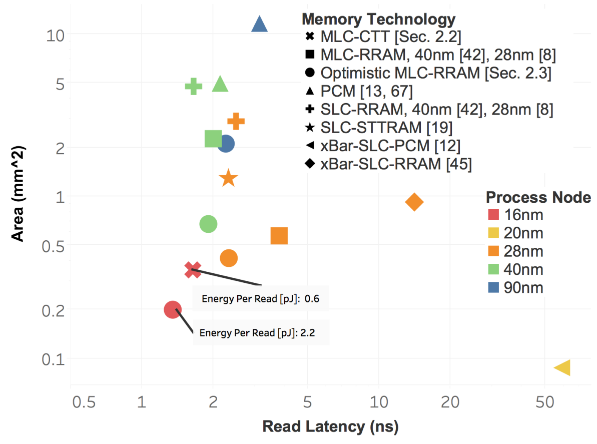


Figure 1: Characterization of published eNVM proposals comparing Area and Read Latency when extrapolated and characterized for a fixed capacity (4MB) using read-latency-optimized results from NVSim [20].

2 CHARACTERIZING AND MODELING eNVMS FROM FABRICATED EXAMPLES

In this section, we compare the characteristics of different eNVM technologies. This discussion lays the foundation for our modeling approach in terms of the specific memories evaluated and technology parameters extracted in order to develop fault models and area, energy, and performance estimates. We choose to model and evaluate MLCs which offer maximal storage density with minimal read latency and read energy, and we selectively evaluate corresponding SLC solutions as a competitive baseline.

2.1 Comparison and Modeling of eNVMs

In evaluating the varied landscape of non-volatile memory devices, we are interested in identifying implementations that achieve low read latency, high storage density, and proven ability to scale to advanced process nodes in order to support DNN inference in constrained computing contexts. We consider spin-transfer torque (STT) memory, phase-change memory (PCM), resistive RAM (RRAM), and charge trap transistor (CTT) memory. Table 1 shows a summary of implemented eNVMs with sufficient published data for the purposes of our modeling, spanning different technology nodes (90nm to 20nm) and memory array architectures (crossbar vs. CMOS-access). To better evaluate trade-offs between these memories, we have extrapolated their memory cell characteristics and generated an equivalent 4MB memory array optimized for read energy-delay product in NVSim (Figure 1) [20].

STT To push the boundaries of dense on-chip storage, we choose to evaluate memories with the capability of storing multiple bits per cell. For this reason, we have excluded spin-transfer torque (STT) memories from our evaluation: while they have been reported as state-of-the-art in terms of both write and read bandwidth [14, 19], MLC implementations require changing the structure of the memory device and are restricted to 2 bit/cell [4, 71, 72].

PCM arrays have been demonstrated in 90nm, and, more recently, in 40nm technologies, though maximal density is still higher

Table 1: Characterization of different non-volatile memory chips.

Reference	eNVM type	Technology	Access device	Cell Area (F^2)	Capacity	Macro area (mm^2)	Read latency	Write latency
[8]	RRAM	28nm	CMOS	39	1Mb	0.56	6.8ns	500ns - 100μs
[42]	RRAM	40nm	CMOS	53	1.4Mb	0.28	10ns	—
[45]	RRAM	24nm	diode	4	32Gb	130.7	40μs	230μs
[13]	MLC-PCM	90nm	CMOS	25	256Mb	120	320ns	—
[67]	PCM	40nm	CMOS	—	1Mb	—	—	120ns
[12]	PCM	20nm PRAM	diode	4	8Gb	59.4	120ns	150ns - 100μs
[19]	STT	28nm	CMOS	75	1Mb	0.214	2.8ns	20ns

for other eNVM proposals [13, 67]. While 20nm PCM has also been shown [12], this configuration has over 10× higher read latency than the design points we evaluate.

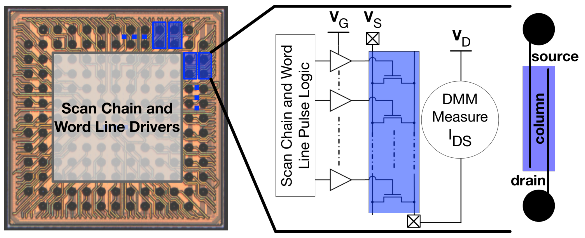
RRAM There are compelling RRAM solutions using either diode access (crossbar) or CMOS access (traditional memory array architecture). Crossbar arrays offer the best cell area ($4F^2$) [13, 45], but they are subject to higher access times (Figure 1). Though chips based on CMOS access devices show larger cell area, they provide more competitive read latencies. The larger cell area can be overcome by increasing storage density via MLC programming. In addition, RRAMs with CMOS access transistors have been demonstrated down to a 28nm process node [8]. We also evaluate an *optimistic* scaled RRAM memory based on a $10F^2$ cell size as a way of evaluating the maximum potential of promising technology advances, which is scaled to several process nodes in Figure 1.

CTT memories [35] use a standard CMOS transistor as a memory cell, and we are the first to present measurements of 3-bit-per-cell MLC programming with a fabricated test chip (Section 2.2.1). The absence of a dedicated access device and the ability to scale to advanced technology nodes gives CTT impressive density and read latency. However, the cost is long write latency (Section 2.2).

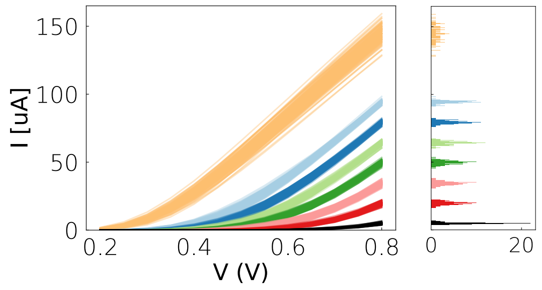
2.2 Multi-Level CTT Characterization

Previous work demonstrated that a single, standard-sized NMOS device can be used as a cost-effective embedded non-volatile memory cell [21, 35, 36, 49]. The resulting memories, often referred to as charge trap transistors (CTT), are made with standard high-*k* metal-gate CMOS transistors. Data is stored by trapping charge in the gate oxide using hot-carrier injection (HCI), which alters the threshold voltage of the device. As a result, the same transistor can be programmed to exhibit different saturation currents which are read out to determine the stored value. In this way, CTT-based NOR memory arrays can be fabricated in industrial-grade, cutting-edge standard CMOS technologies with *zero added manufacturing cost.* In addition to low cost, CTT has many other desirable properties: the memory arrays are analogous to programmable NOR ROM arrays and have similar-scale read latency; devices display very low leakage currents, as stored information is preserved in the transistor's threshold voltage with high retention independent of the applied voltage; the resulting density can be extremely high because CTTs rely on the transistor's gate terminal for access and do not require additional selector devices.

The benefits of CTTs come at the expense of two undesirable properties, limiting their deployment for many general-purpose compute tasks: (1) extremely long write-latencies and (2) potential for high cell fault rates. CTTs are programmed by iteratively


Scan Chain and Word Line Drivers
Scan Chain and Word Line Pulse Logic
V_G
V_S
V_D
DMM Measure I_{DS}
source
column
drain

(a) 16nm Chip and word line schematic



(b) IV Curves, MLC3 Programmed CTT

Figure 2: (a) Die photo of CTT test chip; (b) measured IV curves for 8-level (3-bit, MLC3) programmed CTT (left) and current histogram measured at 0.8V (right) [46].

injecting increments of charge and reading until a desired shift is achieved, which takes 100ms or more. Additionally, charge injection is a random process that inevitably leads to a spread in current distribution, and this spread translates to increased likelihood of misreading the stored value. Despite these limitations, we demonstrate CTT as a near-ideal solution for enabling efficient, embedded DNN inference with careful architectural co-design. This is possible because embedded DNN weights are updated infrequently and repeatedly used to make inferences, and DNNs are known to be fault tolerant [44, 58, 65]. We extend previous demonstrations of CTT with a multi-level-programmed fabricated test chip (MLC-CTT), and our careful algorithmic and architectural co-design overcomes the high fault rates that accompany MLC storage.

2.2.1 MLC-CTT test chip measurements. To prove the feasibility of the cell design, we fabricated MLC-CTT test structures in a commercial 16nm FinFET process. A die photo is shown in Figure 2a, which contains 36 columns of 128 cells each; internal scan chain and driver circuits mimic wordline drivers, with additional details reported in [46]. The bumps expose column bitlines to flexibly read and write individual cells via external test equipment. Figure 2b

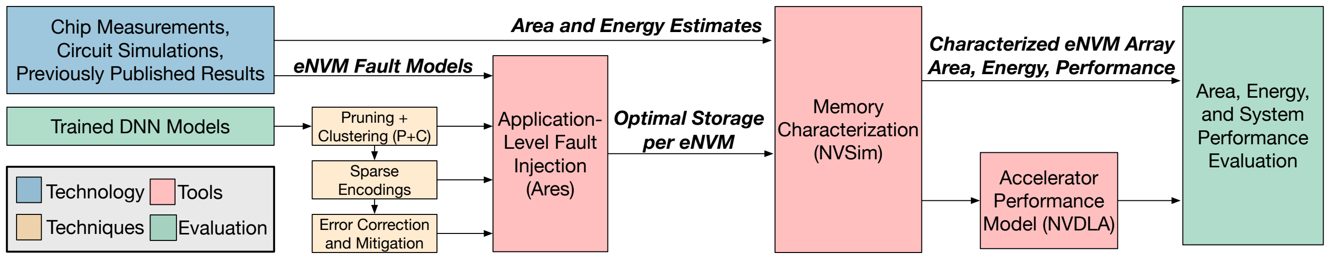

Chip Measurements, Circuit Simulations, Previously Published Results
Area and Energy Estimates
eNVM Fault Models
Trained DNN Models
Pruning + Clustering (P+C)
Sparse Encodings
Error Correction and Mitigation
Application-Level Fault Injection (Ares)
Optimal Storage per eNVM
Memory Characterization (NVSim)
Characterized eNVM Array Area, Energy, Performance
Accelerator Performance Model (NVDLA)
Area, Energy, and System Performance Evaluation
Technology
Tools
Techniques
Evaluation

Figure 3: MaxNVM summary of the tools, optimizations, and intermediate results used in final system evaluations.

shows the distribution of read currents at different wordline supply voltages for 8-level programmed CTTs (a 3-bit MLC). Different colors represent levels (i.e., programmed values), and each level is measured from 128 unique devices.

Figure 2b (right) shows the current distributions at nominal wordline voltage of 0.8V. The yellow cluster corresponds to unprogrammed cells with intrinsic V_{th} variations. Other program levels exhibit tighter distributions due to the iterative write-and-check process. Both cases are well approximated using a Gaussian distribution. As more levels are encoded per cell, these distributions tend to overlap, which increases the probability of misreading the programmed value. Given the wider distribution of intrinsic process variations, we separate the unprogrammed and first programmed state to minimize read errors of the unprogrammed state.

2.3 Fault Model

In developing a fault model for eNVM implementations, we focus on two primary sources of uncertainty: (1) intrinsic randomness associated with setting the value of different programmed levels in a memory cell, and (2) the effect of the sensing circuitry.

The intrinsic distributions for the stored levels in a RRAM implementations are extracted from published data for 3-bit MLC programming [74]. For MLC-CTT, inter-level fault rates are determined by directly measuring current distributions from our test chip (Figure 2b). For both technologies, current distributions can be modeled as gaussian distributions, and the overlap of level distributions determines the rate at which that value will be misread as an adjacent programmed value. In both cases, we use SPICE simulations to derive the distributions at the output of the current-to-voltage converter. The likelihood of misreading a MLC can be high in both CTT and RRAM (e.g., fault rates for MLC3 range from 10^{-3} to 10^{-5}). Errors typically result in reads to an adjacent level[1]. By arranging how data is encoded into MLCs and exploring the impact of varying number of bits per cell on DNN classification error, the effects of faults can be mitigated (Section 4).

We additionally study the effect of the sensing circuitry on MLC fault rate. For this work, we focus on a specific sense amplifier (SA) design, which is characterized by having low static power and input referred offset primarily determined by the input differential pair of transistors [38]. Therefore, we evaluate the input referred offset voltage by running Monte Carlo SPICE simulations for a variety of input transistor width values. Based on these simulations, we choose a SA size such that the overhead incurred for the overall array in our final results never exceeds 1%, and the inherent inter-level fault rates are altered by less than 2×. For the readout architecture, we consider a parallel sensing scheme similar to a flash ADC: the bitline is connected to a number of SAs equal to $N - 1$, where N is the number of levels that can be stored per cell, with each SA connected to the appropriate reference voltage. A parallel sensing scheme decodes the stored value in a single conversion step, but the number of required SAs increases exponentially with the number of stored bits. This overhead is mitigated by multiplexing memory array columns [68]. The resulting design is integrated with NVSim [20] to characterize the overall memory architecture.

3 EVALUATION METHODOLOGY

Our proposed, principled co-design incorporates optimizations and techniques at algorithmic and architectural levels. After developing a fault model based on technology-specific device characteristics and SPICE models of sensing circuitry, we use a previously-validated fault injection framework to quantify the impact of faults on DNN accuracy [57]. We leverage well-known tools to model the energy, performance, and area of the proposed systems [20, 51]. The interaction of these methods as they contribute to the final evaluation is summarized in Figure 3.

3.1 Model Optimizations

Proposed memory systems are evaluated against a competitive and realistic baseline by enforcing iso-accuracy and incorporating common DNN optimizations, as summarized in Table 2. We consider DNNs of different sizes: one CNN model for the small MNIST handwritten digits dataset, two larger, well-known CNNs (VGG16 and ResNet50) for the much larger ImageNet dataset, and another VGG-like topology for the mid-size CiFar10 dataset to span the gap between these cases [32, 39, 41, 59, 63].

Table 2: DNN models including baseline classification error and computed error bound (Section 3.1.1).

Model	LeNet5	VGG12	VGG16	ResNet50
Dataset	MNIST	CiFar10	ImageNet	ImageNet
Layers	4	12	16	54
Parameters	600810	7899840	138084352	24585472
Classification Error	0.83%	10.38%	35.07%	31.15%
Error Bound	0.05%	0.40%	0.57%	1.02%
Cluster Index Bits	4	4	6	7
Sparsity (% zero-valued)	89.9%	40.9%	81.1%	64.84%
16b Size	1.26MB	15.4MB	270MB	70MB
Pruned & Clustered (P+C)	316KB	3.86MB	101MB	30.6MB
CSR	84KB	3.78MB	30.2MB	25.1MB
BitMask	107KB	3.23MB	35.5MB	11.2MB

[1]Misread probability of non-adjacent level is 1.5×10^{-10} or below.

RIGHTSLINK

3.1.1 *Iso-Training Noise.* While previous works have considered trading accuracy for efficiency in deep learning systems, the most convincing demonstration for a practical system must address and preserve baseline model accuracy, as we guarantee via Iso-Training Noise (ITN) [17, 22, 56]. The intuition for this method is that accuracy varies for DNNs repeatedly trained with identical hyper-parameters. The resulting variance in the accuracy can be used as a bound for final classification error. As long as model alterations do not result in error exceeding this bound, they are said to be indistinguishable from ITN and therefore maintain iso-accuracy. All presented energy, performance, and area improvements maintain model accuracy within ITN bounds in Table 2.

3.1.2 *Simplification Techniques.* Magnitude-based weight pruning with retraining is used to sparsify DNN weights, as the models are widely known to be over-parameterized. The resulting proportion of zero-valued weights is in Table 2. One popular technique to reduce the number of bits required to store each DNN weight is to reduce precision using fixed-point quantization. Depending on the dynamic range of the DNN weight values, the number of integer and fractional bits can be drastically reduced, even to a few or a single bit per weight at some loss in model accuracy [28, 33].

Another way to reduce the number of bits required to represent each weight value is to use k-means clustering for each layer of the DNN [26]. For the models considered, we find that all the weight values within a given layer can be represented by 16 to 128 unique clustered values at no loss of accuracy. Thus, each weight can be encoded as a 4 to 7 bit cluster index value, with a simple look up table per layer to map indexes back to values. We find clustering uses strictly fewer bits per weight than fixed-point quantization without significant re-training for all DNNs (Table 2).

3.2 Sparse Encodings

Employing lossless sparse encodings to reduce required storage would be strictly beneficial for traditional memory technologies like SRAM. However, sparse-encoded values are no longer particularly fault tolerant due to the vulnerability of encoding structures compared to the resilience of weight values. Therefore, additional analysis is required to determine optimal MLC storage schemes with sparse encodings, as explored in Section 4.

3.2.1 *Compressed Sparse Row Storage (CSR).* CSR uses three data structures to encode a sparse matrix: an ordered list of all non-zero data elements (Weight Values), the column indexes of each non-zero element (Column Index), and counters of non-zero elements per row (Row Counter). The relative overhead of CSR varies proportionally with sparsity, so CSR is applied on a per-layer basis where worthwhile. Convolution layer weights are typically 3-D (filter width, height, and channels), and are mapped to 2-D to use CSR. NVDLA has specific requirements for data formatting into the convolutional core, which dictates a compatible 2-D mapping [51].

3.2.2 *BitMask Sparse Format.* The NVDLA framework natively supports sparse weight storage that utilizes an indicator bitmask for whether each weight is zero-valued and stores all non-zero data values in packed, 128-byte aligned groups. We refer to this bitmask-based encoding as **'BitM'**, and we maintain compatibility with the defined NVDLA sparse format [51]. The effective compression of

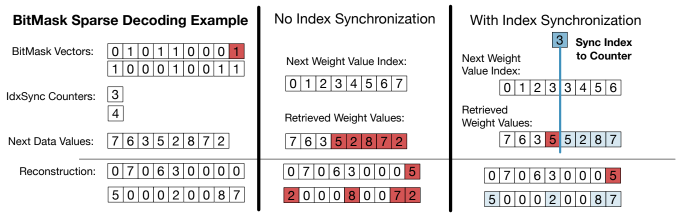


Figure 4: Index Synchronization (IdxSync) prevents errors in the bitmask from propagating by dynamically correcting the index into the non-zero weight values using a counter of the expected non-zero values in a given block of data during decoding. Note that IdxSync does not correct faults that have occurred in the current block, but rather corrects misalignment in subsequent blocks.

each encoding method is shown in Table 2, and the relative overhead of each method is determined by the sparsity of the weights.

3.3 Error Correction and Mitigation

Fault tolerance changes when weights are stored using sparse encodings, as explored in Section 4. A simple strategy to mitigate error is to store more vulnerable structures using fewer bits per cell (e.g., SLC or MLC2) [18]. We additionally explore when it is beneficial to incorporate **Hamming-style, parity-based ECC**. The configuration used is the lightest-weight ECC considered for NAND flash [60]. Note that if values are binary-encoded in a MLC, a level-to-level fault is not equivalent to a single bit flip, so Gray coding is used for ECC-protected values in MLCs to enable correction.

For even lower overhead error mitigation with BitM encoding, we propose storing a counter for the number of non-zero (or zero, depending on which is lower) weight values to be read in each 128 byte aligned block. When an error in the bitmask causes the wrong number of weights to be read during decoding, the counter is used to dynamically update the index to read from in the packed data array. We call this error mitigation strategy **Index Synchronization of the bitmask or 'IdxSync'**. IdxSync does not correct errors– rather, it prevents errors in previously read values from propagating during weight matrix reconstruction, as demonstrated with a simple decoding example in Figure 4.

3.4 Memory Modeling

We integrate new memory cell definitions in NVSim using the energy, performance, and area measurements from our test chip, published work, and SPICE simulations of sensing circuitry topologies [20]. NVSim evaluates all possible memory bank configurations for a given capacity, optimization target, and number of bits per cell (Table 3). Pareto-optimal points are chosen from NVSim output to fit within area or performance constraints in presented results.

Table 3: NVSim, baseline NVDLA parameters [7, 20, 62].

NVSim Parameter	Value
Data Width	8 - 128
Banks	(1 x 1) - (N x N)
Mats	(1 x 1) - (N x N)
Optimization Targets	Area
	Read Latency
	Read EDP
	Read Energy
	Leakage Power

NVDLA Baseline	NVDLA-64	NVDLA-1024
Convolutional Buffer Size	128KB	256KB
Number of MACs	64	1024
SRAM Capacity	512KB	2MB
Frequency	1 GHz	1GHz
Datapath Area	0.55mm^2	2.4mm^2
SRAM BW	6 GB/s	25 GB/s
DRAM Read BW		25 GB/s
LPDDR4 DRAM Power	100 mW	200 mW

RIGHTSLINK

3.5 Architecture Performance Model

NVIDIA deep learning accelerator (NVDLA) is an industry-grade, open source architecture solution to accelerate DNN inference. Baseline system parameters are given in Table 3, and a block diagram is shown in Figure 7(a) [51]. NVDLA supports CNN and FC layer execution, and all computation is performed using this datapath. By demonstrating our proposed memory integrated with this system, we highlight the opportunity for eNVM integration with existing systems for efficient DNN inference.

4 FAULT TOLERANCE OF SPARSE ENCODINGS FOR DNN INFERENCE

Model optimizations and sparse storage schemes significantly impact DNN fault tolerance. Thus, co-design with our carefully developed MLC eNVM fault models is required to enable dense, efficient storage. We use a previously validated application-level fault injection framework (Section 4.1) to quantify the impact of different encoding strategies on DNN classification error (Section 4.2). The results of these experiments guide us in incorporating error correction and mitigation techniques in order to maximize the effectiveness of MLC eNVM storage for DNN inference, as presented in Section 4.3.

4.1 Fault Injection Framework

Ares is a validated, application-level fault injection framework for quantifying DNN fault tolerance [57], which we modify to model MLC eNVM faults, accommodate sparse encoded values, and simulate error mitigation. Based on Sections 2.2.1 and 2.3, we generate an inter-level fault probability map for each MLC configuration and technology. This fault probability map is modified to include the effects of the sense amp by shifting the mean of each level distribution according to the characterized input referred offset.

Fault injection is performed by first converting the weight values into MLC representation. Then, for each eNVM cell, we sample a gaussian random variable from the appropriate level distribution and check if the result crosses the thresholds used to sense that level. If they do, a fault has occurred and that memory cell value is updated to the value represented by the adjacent level. The modified values are then used to perform inference on the entire test dataset. Experiments are repeated over many trials, and presented results are averaged over many unique generated fault maps.

Sparse encodings require separate fault injections on each structure (e.g., the bitmask and non-zero weight values), and we vary the number of bits per cell used to store each structure. When implementing support for ECC, we must ensure a level-to-level fault correlates to a correctable (single bit flip) error by using Gray coding to store binarized values in the MLCs. Dynamic error correction and mitigation strategies are integrated such that faults are detected and values are updated as appropriate prior to evaluating model accuracy. Ares enables the evaluation of DNN classification error over many randomly seeded trials for various MLC eNVM configurations and encoding strategies. From this analysis, we definitively determine the optimal number of bits stored per MLC and the minimal number of memory cells required for each encoding strategy for each DNN such that there is no loss in accuracy.

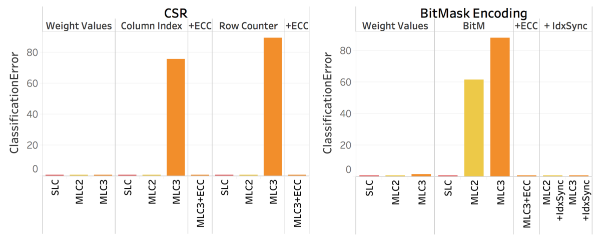


Figure 5: Impact of lightweight error correction (ECC) or mitigation (IdxSync) on the classification error for a sample DNN (MNIST-LeNet5). Each data structure is stored using CTT as SLC, MLC2, or MLC3, assuming perfect storage of other structures to isolate the impact of faults. ECC enables safe storage in MLC3 for CSR format. ECC and IdxSync both enable safe MLC3 storage for bitmask encoding, though IdxSync is a lighter-weight, lower-overhead mechanism.

4.2 Vulnerability of Encoding Strategies

Sparse encoding strategies allow fewer bits to be used when storing DNN weights, but experiments reveal that this compact data storage is in general much less tolerant of inter-level faults. For all DNNs considered, weights that could be safely stored by their cluster index values in MLC3 can no longer be safely stored in MLC3 with sparse encoding structures. For example, for several layers of ImageNet-VGG16, CSR-encoded CTT storage (fewer bits) must be stored in MLC2 to preserve accuracy, which does not offer area improvement compared to dense storage in MLC3.

These instances are due to the extreme vulnerability of sparse encoding data structures, as highlighted for MNIST-LeNet5 in Figure 5. This example demonstrates that the row counter and column index data structure of CSR exhibit exceptionally high vulnerability – MLC3 storage, which is roughly equivalent to one level-to-level fault per row counter structure, causes a pronounced degradation in DNN accuracy. This is because a single misread value may cause an offset when reading from the non-zero data array such that all remaining data values are incorrectly assigned during reconstruction. Similarly, column index values are stored as relative indexes to the previous non-zero value within each row. Thus, a misread value may offset the remaining data values, though the impact will be restricted to a particular row. The vulnerability of column indexes may be mitigated by using absolute indexes, but this requires strictly higher overhead than integrating lightweight ECC. Relative to CSR, the bitmask-style sparse encoding is even more vulnerable, as shown in the right portion of Figure 5. A single bit flip in the bitmask causes all remaining non-zero data values to be mis-assigned when reconstructing the matrix. Thus, the bitmask cannot safely be stored in MLCs without some protective technique.

4.3 Impact of Error Correction and Mitigation Techniques

The column index and row counter '+ECC' bars in Figure 5 show the error reduction when Hamming-style, parity-based ECC is employed to correct errors in the CSR structures. Different amounts of relative ECC overhead were tested per DNN based on the number of correctable errors encountered, and we consistently found that

RIGHTSLINK

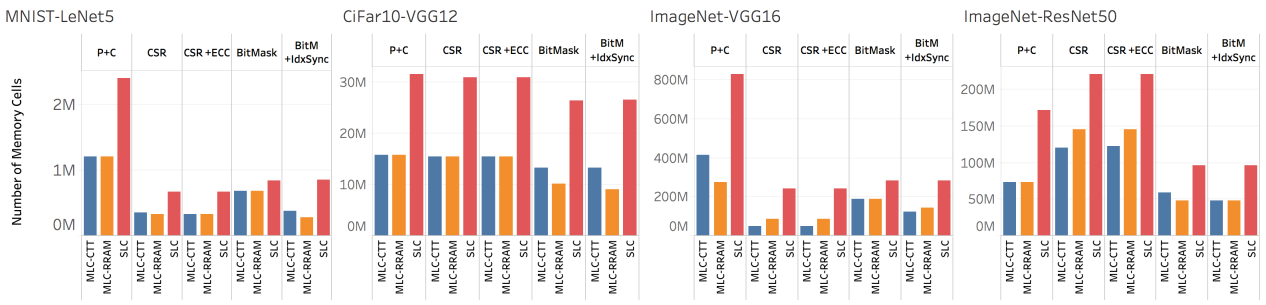


Figure 6: Minimal number of eNVM cells per DNN and per encoding strategy such that there is no loss in classification accuracy using MLC-CTT (blue), MLC-RRAM (orange), and SLC storage (red). These are the results of an exhaustive design space exploration of all possible combinations of bits per cell evaluated over many trials for eNVM technology.

the lowest overhead configuration (24 parity bits for each 4KB of row counter values) is sufficient to safely use MLC3 for CTT. The resulting ECC storage overhead is strictly less than 1% per layer for all DNNs. Additionally, ECC is single error correct, double error detect (SEC-DED), but the probability of a DED, even for the largest model considered (ImageNet-VGG16), is on the order of 10^{-18}. This probability is less than the standards for mass-produced standard memories [1], so our design accepts the risk of this case.

Looking at the classification error for the sample model in Figure 5, we see that both ECC and the IdxSync method enable the use of MLC3 to store the bitmask without degrading model accuracy. The bitmask can account for a significant portion of the total storage per DNN layer (up to 65%), so enabling denser storage of this structure can result in significant area benefits. For all DNNs considered, applying IdxSync is sufficient to enable the safe use of MLCs for the bitmask. This configuration requires less storage overhead compared to ECC and reduces the decoder complexity compared to Hamming-style ECC. However, the optimal storage strategy per DNN depends on model properties (per-layer sparsity, bits per weight value) and desired optimization target (lowest area dictating fewest cells, performance, energy) based on use case, so we continually leverage the results of these fault tolerance studies in determining proposed memory solutions in Sections 5 and 6.

4.4 Optimal Storage Configurations

By repeating the studies described in Section 4 for every DNN and style of sparse encoding (with and without error correction and mitigation) for the fault models of both MLC-CTT and MLC-RRAM, we determine the minimal number of memory cells required to store all DNN weights with a certain storage configuration (summarized in Figure 6). These results are a first-order reflection of the limitations of MLC storage due to the fault rates of each technology, which in turn determines the effective capacity, number of cells, and bits per cell when characterizing memory arrays in Section 5.

Savings stem from both the use of sparse encodings and from packing more bits per cell enabled by error correction and mitigation. There are several examples in which storing fewer bits does not equate to the minimal number of memory cells. These results validate the importance of exhaustive exploration in the design space of encodings and error protection schemes in order to determine optimal MLC eNVM storage solutions. For example, the bitmask sparse encoding with index synchronization (BitM+IdxSync) for ImageNet-VGG16 requires *fewer* memory cells than without error mitigation, despite the overhead of the counter values. This is because IdxSync enables MLC3 storage for the bitmask, rather than just the data, so the total number of cells used decreases by 22%, which in turn reduces the required on-chip memory area.

Although the storage capacity required for ResNet50 is smaller using CSR than with Pruned and Clustered weights (Table 2), it requires fewer memory cells to store *without* sparsely encoding using CSR because the pruned and clustered weights can be safely stored using 3 bits per cell (MLC3), while the vector of non-zero data values can only leverage MLC2 without loss of classification accuracy. However, in Figure 6, we see that the minimum number of memory cells required to store ResNet50 in either MLC-CTT or MLC-RRAM uses the BitMask sparse encoding with index synchronization to mitigate propagation of errors in the bitmask.

5 SINGLE CHIP DNN INFERENCE

We consider a completely self-contained inference accelerator that stores all of the weights in on-chip eNVM and does not require external DRAM, as indicated in Figure 7b. We envision this scenario being especially important for deeply embedded applications such as IoT devices and implanted medical devices where component cost, performance, and energy constraints are extreme [47, 52, 55].

To evaluate the area, energy, and performance of both CTT and RRAM per DNN, we use model optimization results to determine the number of bits required per encoding and perform fault tolerance analysis. This determines the appropriate number of bits per cell and minimum number of memory cells required without loss in classification accuracy (Section 4.4). Next, we use our extensions of NVSim to characterize memory arrays parameterized to accommodate the model size under the optimal encoding strategy. Finally, pareto-optimal points for characterized memories of four different eNVM proposals are identified per model to minimize the read energy-delay-product and area, shown in Figure 8. The resulting system performance is evaluated using the NVDLA performance model and compared to the performance of a baseline configuration relying on off-chip DRAM for weight storage in Figure 9. Table 4 gives a summary of area, energy, and performance for optimal storage for each eNVM evaluated, while Figure 9 gives an example of NVDLA system performance and energy for ResNet50 inference leveraging the various eNVM design points [51, 62].

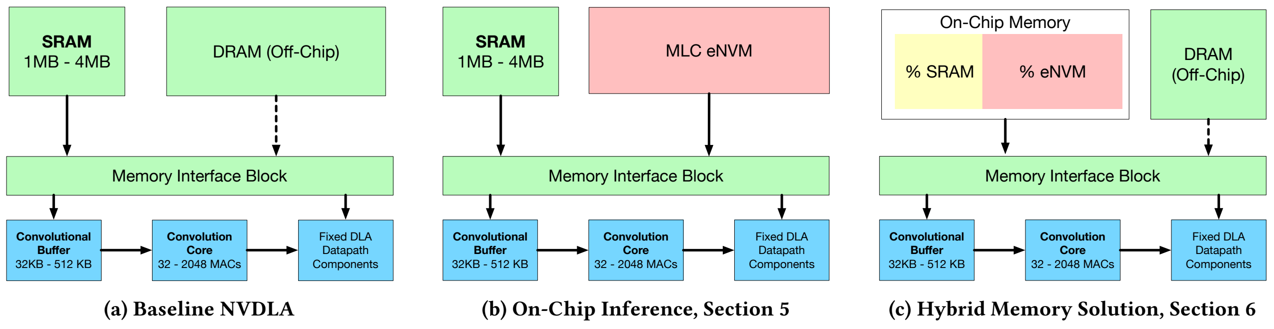

SRAM 1MB - 4MB
DRAM (Off-Chip)
Memory Interface Block
Convolutional Buffer 32KB - 512 KB
Convolution Core 32 - 2048 MACs
Fixed DLA Datapath Components
(a) Baseline NVDLA
MLC eNVM
(b) On-Chip Inference, Section 5
On-Chip Memory
% SRAM
% eNVM
(c) Hybrid Memory Solution, Section 6

Figure 7: Block-level schematic of NVDLA with and without support for an eNVM interface. (a) baseline system with SRAM (intermediate value storage) and off-chip DRAM (weight storage); (b) on-chip memory solution using eNVM for all weight storage (Section 5); (c) hybrid solution partitioning a fixed on-chip memory area and using DRAM for overflow (Section 6).

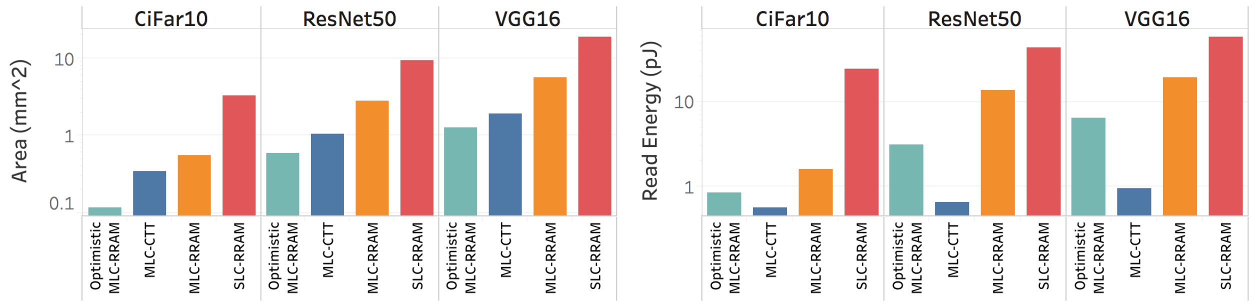


Figure 8: Optimal area and energy per read for memories characterized to store all weights on-chip in Optimistically scaled MLC RRAM (Optimistic MLC-RRAM), MLC CTT, an MLC extrapolation of [74], and SLC RRAM [42].

This study quantifies the increase in die area needed to enable entirely on-chip memory while maintaining competitive performance and achieving reduced power and reduced energy per inference. The number of cycles to execute an inference per input image (frame) is computed using the NVDLA performance model with the characterized read bandwidth and read latency determined by NVSim for each eNVM array. On-chip SRAM is used in the NVDLA system (512KB or 2MB in our designated NVDLA configurations, Table 3) to manage storage of intermediate values between layers of the DNNs, as the write latency of CTT is prohibitively high (on the order of ms [35]) and can still be orders of magnitude higher for MLC-RRAM than SRAM write latency (e.g., 160-640ns [68]), as explored in Section 7.1. We also quantify the relative impact of non-volatility on energy per inference in Section 5.3 and evaluate system performance under a fixed area budget in Section 6.

5.1 Area and Dynamic Read Energy

An interface to eNVM replaces NVDLA's interface to off-chip DRAM for DNN weights, and we demonstrate that aggressive MLC configurations of both CTT and optimistically scaled RRAM can store the sparsely-encoded, ECC protected weights of ImageNet-VGG16 (about 32MB Capacity) in 2mm^2 and 1.3mm^2, respectively. Thus, as a result of exhaustive design space exploration and a rigorous evaluation scheme, all DNN weights can reasonably fit in on-chip MLC eNVM in an area equivalent to 1-2MB of SRAM in modern process nodes. Even SLC RRAM fits all of ResNet50 (12MB capacity under sparse encoding) in under 10mm^2 (Figure 8). With additional model optimizations or more relaxed error bounds, MLC configurations could be leveraged even more aggressively at some loss in DNN classification accuracy, depending on use case. This study strictly avoids loss of accuracy. Even so, these models consume reasonable area for on-chip local memory storage using MLC eNVM.

Figure 8 summarizes the area and dynamic read energy per access of four distinct on-chip memory solutions. The area for each model is the read-energy-delay-product optimal point characterized by our extensions to NVSim that will hold all the DNN parameters in the most optimal encoding for each memory technology as determined in Section 4. Even relative to storing the same optimized and sparse-encoded weights in SLC-RRAM, the MLC-CTT array requires an average of 9.6× less area while maintaining competitive performance within 10% of the NVDLA baseline. Furthermore, our MLC extension of the SLC-RRAM achieves an average area benefit of over 3.2×, and the optimistically scaled MLC-RRAM (Optimistic MLC-RRAM) enables an average area benefit of 20×. The larger relative area benefit between MLC-CTT and the Optimistic MLC-RRAM for CiFar10-VGG12 (Figure 8, left) derives both from the inherent storage density of the Optimistic MLC-RRAM and the fact that more bits per cell can be safely used with the Optimistic MLC-RRAM (consistently 3 bits per cell) than with MLC-CTT (2 bits per cell) due to their respective fault characteristics and the impact on classification error, as exposed in Section 4.4.

The dynamic read energy for each of the memory proposals varies by orders of magnitude, even for equivalent-capacity characterized arrays (Figure 8, right). MLC-CTT is consistently lower

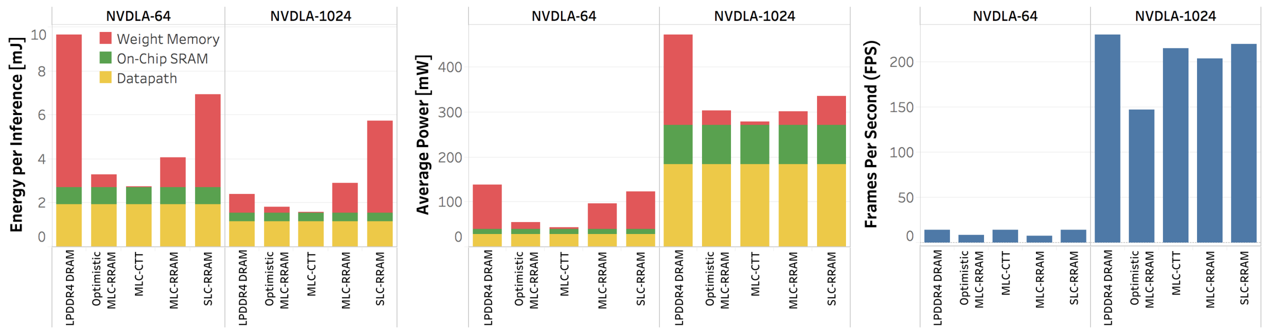


Figure 9: NVDLA energy per ResNet50 inference (left), average power (middle), and frames processed per second (right) for two configurations of NVDLA described in Table 3, considering replacing LPDDR4 DRAM (baseline) with proposed eNVM solutions and leveraging the specific memory arrays characterized for ResNet50 in Figure 8.

energy per access than even the Optimistic MLC-RRAM solution by over 4×, and also tends to maintain a lower read latency and higher read bandwidth (up to 9 GB/s).

5.2 System Performance

Though MLC-RRAM proposals do not surpass the maximal FPS possible with NVDLA (e.g., Figure 9, right), the best performance for each model consistently exceeds 60 frames per second with the NVDLA-1024 configuration, well above the image processing frame rate for standard motion pictures [48]. The latency overhead of MLC sensing tends to negate the effective bandwidth increase of MLC storage. This is seen in Figure 9; SLC-RRAM is competitive with MLC-CTT in terms of frames (i.e., inferences) processed per second. For the characterized eNVM arrays used in Figure 9, Optimistic MLC-RRAM exhibits lower read bandwidth and higher read latency than MLC-CTT, leading to lower FPS. We note that these performance estimates do not include the decoding overheads for sparse encodings and cluster index values. However, the overhead to reconstruct and decode weight values for both strategies will be minimal and consistent in the baseline and eNVM-based systems.

Figure 9 compares the average power consumption and total energy per inference for ResNet50 when operating at maximum performance. We consider two fixed NVDLA baseline datapath configurations given in Table 3, for which the power consumption is publicly available [62], and assume additional power consumption of LPDDR4 DRAM running at 1GHz is 200mW to form a conservative estimate of potential improvements via integration of on-chip eNVM [7]. NVDLA-64 is representative of a more extremely resource-constrained DNN accelerator, and the energy consumption of memory accesses for weight fetches using MaxNVM is reduced by over 100× (DRAM vs. 1mm^2 of on-chip MLC-CTT), resulting in overall average system power reduction of 3.2×. NVDLA-1024 provides considerably higher performance (Figure 9, right) and lower dynamic energy per inference, though the average power is higher. In this scenario, we see similar trends in MLC-CTT reducing energy consumption due to weight fetching by up to 13.6×. However, due to the higher baseline power of the larger convolutional core and buffer, the total relative system power reduction is up to 1.6×.

Of the three evaluated MLC proposals, MLC-CTT achieves the best system performance, lowest power, and lowest energy per inference. Compared to the optimistically scaled MLC-RRAM, higher read bandwidth and lower energy per read result in 20% overall lower energy per inference for MLC-CTT for NVDLA-64. These demonstrated benefits are relative to energy and performance characteristics of convolutional models using a highly-optimized CNN accelerator datapath that maximizes re-use of fetched values, so energy reduction due to memory fetches would be increasingly beneficial in other resource-constrained contexts that exhibit less re-use of fetched parameters (e.g., recurrent neural networks).

5.3 Benefits of Non-Volatility

Depending on how frequently inferences occur (i.e., required frame rate for an image processing task), eNVMs have an inherent relative benefit by virtue of not needing to reload weight values when powered on or, alternatively, keeping DRAM powered to avoid this cost. Figure 10 summarizes how the average energy per inference of the evaluated MLC eNVM solutions compares to the baseline system either remaining fully powered to retain values ("DRAM always on") vs. a conservative estimate of waking up the system for each inference ("DRAM wake up"). Each time the baseline system is woken up, there is an energy overhead to load all DNN weights from main memory into DRAM and increased execution time to load the first DNN layer before the inference task can begin computation, while these costs are not applicable to eNVMs that retain their values when powered off between inferences.

If the DRAM power stays constant while the frequency of inferences increases (i.e., FPS), the average energy per inference decreases as the system spends less time idle. Conversely, if the system is woken up for each inference, the average energy per inference is constant so long as the frequency (FPS) can be met by the NVDLA-1024 system, which Figure 9 (right) shows is clearly possible for the range of FPS considered. For frame rates below 22 FPS, such as would be relevant for object detection tasks for security camera systems [29], it is more energy efficient to wake up and the baseline system per inference, but our proposed MLC eNVM solutions maintain a relative 5.3× (MLC-RRAM) to 7.5× (MLC-CTT) reduced energy per inference. Energy savings are similar for the typical range of frame rates used for video conferencing and other standard image processing tasks (e.g., 30 FPS [70]). Though the average energy per inference of the always-on baseline system approaches the performance of MLC eNVM alternatives at higher frame rates, we note that even operating at the FPS mandated to

RIGHTSLINK

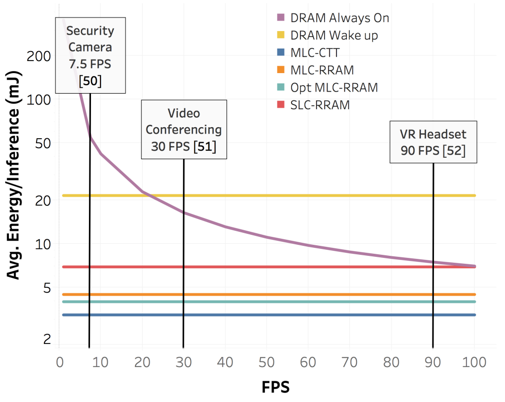


Figure 10: On-chip MLC eNVM in place of LPDDR4 DRAM results in 5.3×-7.5× lower average energy per ResNet50 Inference for varying frame rates.

support virtual reality headsets (at least 90 FPS [24]), MLC eNVM proposals achieve 1.7× (MLC-RRAM) to 2.5× (MLC-CTT) lower energy per inference. Thus, optimized MLC eNVM solutions are particularly compelling for applications with lower frame-rate requirements, and these benefits would be exaggerated for systems with less frequent wake-ups.

6 HYBRID MEMORY SOLUTION

New DNNs tend to improve accuracy by increasing model size. For the case when a DNN does not fit in on-chip eNVM, we consider a hybrid solution (Figure 7c). A fixed on-chip memory budget is split between SRAM and eNVM, and weights and activations that do not fit on-chip are stored and fetched from DRAM.

We choose an on-chip memory area budget of 1mm^2, enough to accommodate about 1MB of SRAM under various NVSim optimization targets. With eNVM, smaller DNNs fit within this area constraint. The results in Figure 11 are for the largest DNN, ImageNet-VGG16, and highlight the performance impact of incorporating varying amounts of on-chip space to eNVM. Note that the eNVM is not used as cache for DRAM; on-chip MLC eNVM and DRAM store mutually exclusive sets of model weights and both are fed directly into the accelerator's datapath. By partitioning fixed on-chip memory area between SRAM for intermediate storage and MLC eNVM

Table 4: Summary of optimal storage per eNVM proposal, characterized per DNN. 'BPC' is the max number of bits per cell used, 'MB' is the max capacity, 'FPS' is the max frames per second for NVDLA-1024, and 'Read' is the Read Latency of each eNVM array in ns.

Model	Memory Tech	Encoding	BPC	[MB]	Area [mm^2]	Read [ns]	FPS
CiFar10	Opt MLC-RRAM	BitM+IdxSync	3	4	0.12	5.1	132
	MLC-CTT	BitMask	2	4	0.35	1.6	2286
	MLC-RRAM	BitM+IdxSync	3	4	1.3	4.9	633
	SLC-RRAM	BitMask	1	4	3.4	1.7	2967
VGG16	Opt MLC-RRAM	CSR+ECC	3	32	1.3	4.2	102
	MLC-CTT	CSR+ECC	3	32	2.0	2.0	142
	MLC-RRAM	CSR+ECC	3	32	5.7	3.2	131
	SLC-RRAM	CSR	1	32	19.2	5.2	147
ResNet50	Opt MLC-RRAM	BitM+IdxSync	2	12	0.6	2.1	147
	MLC-CTT	BitM+IdxSync	2	12	1.0	1.9	215
	MLC-RRAM	BitM+IdxSync	2	12	2.8	1.4	203
	SLC-RRAM	BitMask	1	12	9.6	2.5	219

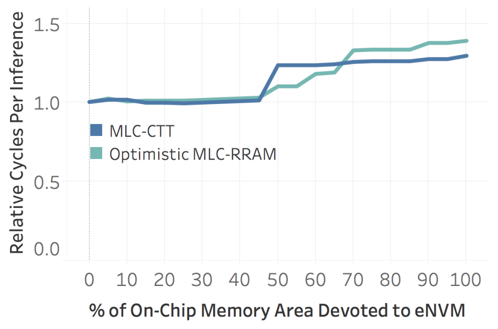


Figure 11: Relative performance of VGG16 when on-chip area is split between eNVM vs. SRAM (Fig. 7c).

for weight storage, we reduce costly DRAM accesses while drastically increasing total on-chip memory capacity. For each possible area partition, we use NVSim to characterize the maximal capacity and minimal read latency/energy within that area constraint and select pareto-optimal points, as in Section 5.

Depending on how the on-chip memory area is partitioned, inference execution could be bottlenecked by fetching weights from DRAM that did not fit in eNVM, or by read/write traffic of intermediates that do not fit in SRAM. We extend the NVDLA performance model to selectively read certain weights from eNVM rather than DRAM and maximize eNVM benefits by greedily selecting the weights of otherwise DRAM-bottlenecked layers to store first.

There is some initial benefit from alleviating the weight retrieval DRAM bottleneck when some amount of eNVM is allotted because the number of DRAM accesses reduces and the energy to access either eNVM is orders of magnitude lower than accessing DRAM, with lowest energy-per-inference occurring when about 45% of on-chip area devoted to MLC eNVM for both technologies. However, performance sharply degrades when on-chip SRAM storage can no longer hold the working set of intermediate values to feed the convolutional core and execution becomes bottlenecked on writing to and fetching activations from DRAM. The write characteristics of both MLC eNVMs are not sufficient to buffer intermediate values or to re-write weights during inference, as explored in Section 7.1.

7 DISCUSSION

Table 4 provides the energy, area, and performance per model from Section 5. The results emphasize the need for a comprehensive co-design methodology. Considering the storage density of MLC and energy advantages due to non-volatility, even unoptimized DNNs or other reasonably fault-tolerant applications may leverage these memories. However, choosing the optimal encoding strategy, MLC configuration, and whether to include error mitigation techniques varies between models and eNVM technologies. Thus, a rigorous evaluation of the design space per DNN, as our methodology effectively executes, maximizes efficiency gains.

7.1 Write Latency

Table 5 gives an approximate total time to write to the characterized memory array for each model's entire set of weights and each eNVM design point based on best-case-scenario write latency from previous work [35, 42, 74]. Depending on a given application space

RIGHTSLINK

Table 5: Optimistic total time to write all DNN weights.

Model	Memory Technology	Approximate Total Write Time
CiFar10	Opt MLC-RRAM	13ms
	MLC-CTT	2.6 **minutes**
	MLC-RRAM	33ms
	SLC-RRAM	3ms
ResNet50	Opt MLC-RRAM	117ms
	MLC-CTT	15.7 **minutes**
	MLC-RRAM	94ms
	SLC-RRAM	4.7ms
VGG16	Opt MLC-RRAM	254ms
	MLC-CTT	12.2 **minutes**
	MLC-RRAM	636ms
	SLC-RRAM	23ms

and resource constraints, waiting on the order of minutes may or may not be reasonable when re-writing of DNN weights is required, see Table 5. The desired frequency of rewriting weights may also be constrained by the endurance of the memory cells. However, in sensor nodes, mobile devices, and other constraint-driven devices in which DNN inference is a key workload, periodic down-time for synchronization and charging may be permissible [31, 50].

8 RELATED WORK

Research concerning specialized on-chip accelerator systems for DNN image classification is prevalent [2, 6, 9, 10, 22, 30, 34, 66, 69]. Across proposed solutions, off-chip DRAM access is a consistent performance and efficiency bottleneck that levies a high cost on lower-energy systems, even when considering optimizations for sparsity. The cost of external reads could be mitigated by pairing on-chip SRAM with dense embedded DRAM, but this solution still incurs the cost of frequent refresh cycles [3]. These limitations motivate work targeting alternative storage solutions for DNNs and related applications, including the use of eNVMs.

Abundant prior work exists related to the fault tolerance of DNNs [23, 25, 44, 44, 54, 57, 58, 65], and our presented fault tolerance studies (Section 4) confirm and expound upon previous results by additionally considering the resiliency of sparsely-encoded DNN weights. We demonstrate how and to what extent the fault tolerance of DNN weights changes when using different sparse encodings and the impact of a set of error correction and mitigation techniques.

8.1 eNVM technologies overview

The basic principle of many eNVMs consists of replacing charge-based storage, which is inherently volatile, with the ability to encode information by altering the electrical properties of the device material. Different implementations exist and new proposals are emerging, as discussed in Section 2. The MLC capability of CTT and applicability to DNN storage was first explored in [18], which is a preliminary proof-of-concept without system considerations or evaluation of power, performance, and area advantages for DNN inference. [18] compares storage of fixed-point and clustered weight values in MLCs, though it does not consider sparse encodings or error protection.

8.2 Analog eNVM Applications

A secondary effect of encoding information in the physical characteristics of the devices is that the range of programmable values does not have to be strictly binary. In this scenario, memristive devices can be used as *analog* memories, or even leveraged for reconfigurable interconnects [15]. An interesting consequence of eNVMs as analog memories is that computations can be performed directly in the analog domain. This is the basis for many proposed neuromorphic engines, in which weights are mapped directly to the resistive value of the devices connected in a crossbar array to perform in-memory matrix-vector multiplications. While examples of crossbar arrays implemented with RRAM and PCM have been shown, they are limited to small arrays [27, 37, 53]. More recently, larger neuromorphic arrays have been proposed, which promise better scalability for these architectures [67]. However, application to larger DNNs has yet to be demonstrated, though numerous neuromorphic proposals have leveraged eNVMs, particularly using RRAMs for processing-in-memory architectures [5, 11, 61, 64].

9 CONCLUSION

Our comprehensive co-design methodology, MaxNVM, revealed a collection of observations and trade-offs that can continue to guide the integration of eNVMs to enable efficient DNN inference. We are guided by published examples and direct measurements of compelling MLC-programmed eNVM solutions in order to maximize storage density, and MaxNVM offers a framework for evaluating co-design choices from the circuit to the algorithm level. This includes evaluation of different proposed eNVM technologies, model properties, encoding strategies, and error mitigation to rigorously explore the trade-offs between storage density, memory reliability, system performance, area, energy, and the resulting accuracy of DNN inference under various constraints. We plan to make components of MaxNVM publicly available to empower continued research into essential trade-offs when designing and evaluating proposed eNVMs in an application-driven context.

In this work, we demonstrate that sparse weight encoding strategies have implications for the fault tolerance of stored values for DNNs. Thus, we propose and integrate lightweight protection mechanisms in order to maximize the levels-per-cell of MLC eNVM storage without accuracy loss. We demonstrate that co-designing MLC encodings provides up to 29× area savings relative to a SLC eNVM solution, emphasizing the benefits of an exhaustive design space exploration. Finally, we evaluate a state-of-the-art DNN inference accelerator integrated with our proposed MLC eNVM solutions, and observe up to 3.5× and 3.2× lower energy per inference and power, respectively, at maximum frame rate for image classification tasks, or up to 7.5× lower energy per inference at lower frame rates.

ACKNOWLEDGMENTS

This work was partially supported by the Applications Driving Architectures (ADA) Research Center, a JUMP Center co-sponsored by SRC and DARPA.

REFERENCES

[1] 2017. Solid State Drive (SSD) Requirements and Endurance Test Method. https://www.jedec.org/standards-documents/focus/flash/solid-state-drives.

[2] Manoj Alwani, Han Chen, Michael Ferdman, and Peter Milder. 2016. Fused-layer CNN Accelerators. In *The 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-49)*. IEEE Press, Piscataway, NJ, USA, Article 22, 12 pages. http://dl.acm.org/citation.cfm?id=3195638.3195664

[3] I. Bhati, M. Chang, Z. Chishti, S. Lu, and B. Jacob. 2016. DRAM Refresh Mechanisms, Penalties, and Trade-Offs. *IEEE Trans. Comput.* 65, 1 (Jan 2016), 108–121. https://doi.org/10.1109/TC.2015.2417540

[4] X. Bi, M. Mao, D. Wang, and H. H. Li. 2017. Cross-Layer Optimization for Multilevel Cell STT-RAM Caches. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 25, 6 (June 2017), 1807–1820. https://doi.org/10.1109/TVLSI.2017.2665543

[5] M. N. Bojnordi and E. Ipek. 2016. Memristive Boltzmann machine: A hardware accelerator for combinatorial optimization and deep learning. In *2016 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. 1–13. https://doi.org/10.1109/HPCA.2016.7446049

[6] Mark Buckler, Suren Jayasuriya, and Adrian Sampson. 2017. Reconfiguring the Imaging Pipeline for Computer Vision. In *The IEEE International Conference on Computer Vision (ICCV)*.

[7] Karthik Chandrasekar, Christian Weis, Yonghui Li, Sven Goossens, Matthias Jung, Omar Naji, Benny Akesson, Norbert Wehn, , and Kees Goossens. [n. d.]. DRAMPower: Open-source DRAM Power and Energy Estimation Tool. http://www.drampower.info.

[8] M. Chang, J. Wu, T. Chien, Y. Liu, T. Yang, W. Shen, Y. King, C. Lin, K. Lin, Y. Chih, S. Natarajan, and J. Chang. 2014. 19.4 embedded 1Mb ReRAM in 28nm CMOS with 0.27-to-1V read using swing-sample-and-couple sense amplifier and self-boost-write-termination scheme. In *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*. 332–333. https://doi.org/10.1109/ISSCC.2014.6757457

[9] Tianshi Chen, Zidong Du, Ninghui Sun, Jia Wang, Chengyong Wu, Yunji Chen, and Olivier Temam. 2014. DianNao: A Small-footprint High-throughput Accelerator for Ubiquitous Machine-learning. In *ASPLOS*.

[10] Yu-Hsin Chen, Tushar Krishna, Joel Emer, and Vivienne Sze. 2016. Eyeriss: An Energy-Efficient Reconfigurable Accelerator for Deep Convolutional Neural Networks. In *ISSCC*.

[11] P. Chi, S. Li, C. Xu, T. Zhang, J. Zhao, Y. Liu, Y. Wang, and Y. Xie. 2016. PRIME: A Novel Processing-in-Memory Architecture for Neural Network Computation in ReRAM-Based Main Memory. In *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*. 27–39. https://doi.org/10.1109/ISCA.2016.13

[12] Y. Choi, I. Song, M. Park, H. Chung, S. Chang, B. Cho, J. Kim, Y. Oh, D. Kwon, J. Sunwoo, J. Shin, Y. Rho, C. Lee, M. G. Kang, J. Lee, Y. Kwon, S. Kim, J. Kim, Y. Lee, Q. Wang, S. Cha, S. Ahn, H. Horii, J. Lee, K. Kim, H. Joo, K. Lee, Y. Lee, J. Yoo, and G. Jeong. 2012. A 20nm 1.8V 8Gb PRAM with 40MB/s program bandwidth. In *2012 IEEE International Solid-State Circuits Conference*. 46–48. https://doi.org/10.1109/ISSCC.2012.6176872

[13] G. F. Close, U. Frey, J. Morrish, R. Jordan, S. C. Lewis, T. Maffitt, M. J. BrightSky, C. Hagleitner, C. H. Lam, and E. Eleftheriou. 2013. A 256-Mcell Phase-Change Memory Chip Operating at2+Bit/Cell. *IEEE Transactions on Circuits and Systems I: Regular Papers* 60, 6 (June 2013), 1521–1533. https://doi.org/10.1109/TCSI.2012.2220459

[14] D. C. Daly, L. C. Fujino, and K. C. Smith. 2018. Through the Looking Glass - The 2018 Edition: Trends in Solid-State Circuits from the 65th ISSCC. *IEEE Solid-State Circuits Magazine* 10, 1 (winter 2018), 30–46. https://doi.org/10.1109/MSSC.2017.2771103

[15] John Demme, Bipin Rajendran, Steven Nowick, and Simha Sethumadhavan. 2015. Increasing reconfigurability with memristive interconnects. 351–358. https://doi.org/10.1109/ICCD.2015.7357124

[16] G. Desoli, N. Chawla, T. Boesch, S. p. Singh, E. Guidetti, F. De Ambroggi, T. Majo, P. Zambotti, M. Ayodhyawasi, H. Singh, and N. Aggarwal. 2017. 14.1 A 2.9TOPS/W deep convolutional neural network SoC in FD-SOI 28nm for intelligent embedded systems. In *2017 IEEE International Solid-State Circuits Conference (ISSCC)*. 238–239. https://doi.org/10.1109/ISSCC.2017.7870349

[17] Caiwen Ding, Siyu Liao, Yanzhi Wang, Zhe Li, Ning Liu, Youwei Zhuo, Chao Wang, Xuehai Qian, Yu Bai, Geng Yuan, Xiaolong Ma, Yipeng Zhang, Jian Tang, Qinru Qiu, Xue Lin, and Bo Yuan. 2017. CirCNN: Accelerating and Compressing Deep Neural Networks Using Block-CirculantWeight Matrices. (aug 2017). https://doi.org/10.1145/3123939.3124552 arXiv:1708.08917

[18] Marco Donato, Brandon Reagen, Lillian Pentecost, Udit Gupta, David Brooks, and Gu-Yeon Wei. 2018. On-Chip Deep Neural Network Storage with Multi-Level eNVM. In *2018 The 55th Annual Design Automation Conference (DAC)*. https://doi.org/10.1145/3195970.3196083

[19] Q. Dong, Z. Wang, J. Lim, Y. Zhang, Y. Shih, Y. Chih, J. Chang, D. Blaauw, and D. Sylvester. 2018. A 1Mb 28nm STT-MRAM with 2.8ns read access time at 1.2V VDD using single-cap offset-cancelled sense amplifier and in-situ self-write-termination. In *2018 IEEE International Solid - State Circuits Conference - (ISSCC)*. 480–482. https://doi.org/10.1109/ISSCC.2018.8310393

[20] X. Dong, C. Xu, Y. Xie, and N. P. Jouppi. 2012. NVSim: A Circuit-Level Performance, Energy, and Area Model for Emerging Nonvolatile Memory. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 31, 7 (July 2012), 994–1007. https://doi.org/10.1109/TCAD.2012.2185930

[21] Yuan Du, Li Du, Xuefeng Gu, Xiao Wang, and Mau-Chung Frank Chang. 2017. A Memristive Neural Network Computing Engine using CMOS-Compatible Charge-Trap-Transistor (CTT). (2017).

[22] Z. Du, R. Fasthuber, T. Chen, P. Ienne, L. Li, T. Luo, X. Feng, Y. Chen, and O. Temam. 2015. ShiDianNao: Shifting vision processing closer to the sensor. In *2015 ACM/IEEE 42nd Annual International Symposium on Computer Architecture (ISCA)*. 92–104. https://doi.org/10.1145/2749469.2750389

[23] Z. Du, K. Palem, A. Lingamneni, O. Temam, Y. Chen, and C. Wu. 2014. Leveraging the error resilience of machine-learning applications for designing highly energy efficient accelerators. In *2014 19th Asia and South Pacific Design Automation Conference (ASP-DAC)*. 201–206. https://doi.org/10.1109/ASPDAC.2014.6742890

[24] Facebook Technologies. [n. d.]. Oculus guidelines for Virtual Reality Performance Optimization. https://developer.oculus.com/documentation/pcsdk/latest/concepts/dg-performance-guidelines/.

[25] B. Feinberg, S. Wang, and E. Ipek. 2018. Making Memristive Neural Network Accelerators Reliable. In *2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. 52–65. https://doi.org/10.1109/HPCA.2018.00015

[26] Yunchao Gong et al. 2014. Compressing Deep Convolutional Networks using Vector Quantization. *CoRR* (2014).

[27] X. Guo, F. M. Bayat, M. Bavandpour, M. Klachko, M. R. Mahmoodi, M. Prezioso, K. K. Likharev, and D. B. Strukov. 2017. Fast, energy-efficient, robust, and reproducible mixed-signal neuromorphic classifier based on embedded NOR flash memory technology. In *2017 IEEE International Electron Devices Meeting (IEDM)*. 6.5.1–6.5.4. https://doi.org/10.1109/IEDM.2017.8268341

[28] Suyog Gupta, Ankur Agrawal, Kailash Gopalakrishnan, and Pritish Narayanan. 2015. Deep Learning with Limited Numerical Precision. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37 (ICML'15)*. JMLR.org, 1737–1746. http://dl.acm.org/citation.cfm?id=3045118.3045303

[29] Mike Haldas. [n. d.]. CCTV Camera Recording Video Frame Rate Comparison. https://www.cctvcamerapros.com/CCTV-Video-Frame-Rate-Comparison-s/739.htm.

[30] Song Han, Xingyu Liu, Huizi Mao, Jing Pu, Ardavan Pedram, Mark A. Horowitz, and William J. Dally. 2016. EIE: Efficient Inference Engine on Compressed Deep Neural Network. *SIGARCH Comput. Archit. News* 44, 3 (June 2016), 243–254. https://doi.org/10.1145/3007787.3001163

[31] K. Hazelwood, S. Bird, D. Brooks, S. Chintala, U. Diril, D. Dzhulgakov, M. Fawzy, B. Jia, Y. Jia, A. Kalro, J. Law, K. Lee, J. Lu, P. Noordhuis, M. Smelyanskiy, L. Xiong, and X. Wang. 2018. Applied Machine Learning at Facebook: A Datacenter Infrastructure Perspective. In *2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. 620–629. https://doi.org/10.1109/HPCA.2018.00059

[32] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep Residual Learning for Image Recognition. *CoRR* abs/1512.03385 (2015). arXiv:1512.03385 http://arxiv.org/abs/1512.03385

[33] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. 2016. Binarized Neural Networks. In *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett (Eds.). Curran Associates, Inc., 4107–4115. http://papers.nips.cc/paper/6573-binarized-neural-networks.pdf

[34] Norman P. Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, Rick Boyle, Pierre-luc Cantin, Clifford Chao, Chris Clark, Jeremy Coriell, Mike Daley, Matt Dau, Jeffrey Dean, Ben Gelb, Tara Vazir Ghaemmaghami, Rajendra Gottipati, William Gulland, Robert Hagmann, C. Richard Ho, Doug Hogberg, John Hu, Robert Hundt, Dan Hurt, Julian Ibarz, Aaron Jaffey, Alek Jaworski, Alexander Kaplan, Harshit Khaitan, Daniel Killebrew, Andy Koch, Naveen Kumar, Steve Lacy, James Laudon, James Law, Diemthu Le, Chris Leary, Zhuyuan Liu, Kyle Lucke, Alan Lundin, Gordon MacKean, Adriana Maggiore, Maire Mahony, Kieran Miller, Rahul Nagarajan, Ravi Narayanaswami, Ray Ni, Kathy Nix, Thomas Norrie, Mark Omernick, Narayana Penukonda, Andy Phelps, Jonathan Ross, Matt Ross, Amir Salek, Emad Samadiani, Chris Severn, Gregory Sizikov, Matthew Snelham, Jed Souter, Dan Steinberg, Andy Swing, Mercedes Tan, Gregory Thorson, Bo Tian, Horia Toma, Erick Tuttle, Vijay Vasudevan, Richard Walter, Walter Wang, Eric Wilcox, and Doe Hyun Yoon. 2017. In-Datacenter Performance Analysis of a Tensor Processing Unit. In *Proceedings of the 44th Annual International Symposium on Computer Architecture (ISCA '17)*. ACM, New York, NY, USA, 1–12. https://doi.org/10.1145/3079856.3080246

[35] Faraz Khan, Eduard Cartier, Chandrasekara Kothandaraman, J. Campbell Scott, Jason C. S. Woo, and Subramanian S. Iyer. 2016. The Impact of Self-Heating on Charge Trapping in High-k -Metal-Gate nFETs. *IEEE Electron Device Lett.* 37

(2016), 88–91.
[36] Faraz Khan, Eduard Cartier, Jason C S Woo, and Subramanian Iyer. 2017. Charge Trap Transistor (CTT): An Embedded Fully Logic-Compatible Multiple-Time Programmable Non-Volatile Memory Element for high-k-metal-gate CMOS technologies. *IEEE Electron Device Letters* 38 (2017).
[37] S. Kim, M. Ishii, S. Lewis, T. Perri, M. BrightSky, W. Kim, R. Jordan, G. W. Burr, N. Sosa, A. Ray, J. . Han, C. Miller, K. Hosokawa, and C. Lam. 2015. NVM neuromorphic core with 64k-cell (256-by-256) phase change memory synaptic array with on-chip neuron circuits for continuous in-situ learning. In *2015 IEEE International Electron Devices Meeting (IEDM)*. 17.1.1–17.1.4. https://doi.org/10.1109/IEDM.2015.7409716
[38] T. Kobayashi, K. Nogami, T. Shirotori, Y. Fujimoto, and O. Watanabe. 1992. A current-mode latch sense amplifier and a static power saving input buffer for low-power architecture. In *1992 Symposium on VLSI Circuits Digest of Technical Papers*. 28–29. https://doi.org/10.1109/VLSIC.1992.229252
[39] Alex Krizhevsky. [n. d.]. Learning Multiple Layers of Features from Tiny Images. Technical Report https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.
[40] Nicholas D. Lane, Sourav Bhattacharya, Petko Georgiev, Claudio Forlivesi, and Fahim Kawsar. 2015. An Early Resource Characterization of Deep Learning on Wearables, Smartphones and Internet-of-Things Devices. In *Proceedings of the 2015 International Workshop on Internet of Things Towards Applications (IoT-App '15)*. ACM, New York, NY, USA, 7–12. https://doi.org/10.1145/2820975.2820980
[41] Yann LeCun and Corinna Cortes. [n. d.]. The MNIST database of handwritten digits. http://yann.lecun.com/exdb/mnist/.
[42] C. Lee, H. Lin, C. Lien, Y. Chih, and J. Chang. 2017. A 1.4Mb 40-nm embedded ReRAM macro with 0.07um lt;sup gt;2 lt;/sup gt; bit cell, 2.7mA/100MHz low-power read and hybrid write verify for high endurance application. In *2017 IEEE Asian Solid-State Circuits Conference (A-SSCC)*. 9–12. https://doi.org/10.1109/ASSCC.2017.8240203
[43] K. H. Lee, S. Y. Kung, and N. Verma. 2011. Improving kernel-energy trade-offs for machine learning in implantable and wearable biomedical applications. In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 1597–1600. https://doi.org/10.1109/ICASSP.2011.5946802
[44] Guanpeng Li, Siva Hari, Michael Sullivan, Timothy Tsai, Karthik Pattabiraman, Joel Emer, and Stephen W. Keckler. 2017. Understanding Error Propagation in Deep Learning Neural Network (DNN) Accelerators and Applications. In *SC*.
[45] T. Liu, T. H. Yan, R. Scheuerlein, Y. Chen, J. K. Lee, G. Balakrishnan, G. Yee, H. Zhang, A. Yap, J. Ouyang, T. Sasaki, S. Addepalli, A. Al-Shamma, C. Chen, M. Gupta, G. Hilton, S. Joshi, A. Kathuria, V. Lai, D. Masiwal, M. Matsumoto, A. Nigam, A. Pai, J. Pakhale, C. H. Siau, X. Wu, R. Yin, L. Peng, J. Y. Kang, S. Huynh, H. Wang, N. Nagel, Y. Tanaka, M. Higashitani, T. Minvielle, C. Gorla, T. Tsukamoto, T. Yamaguchi, M. Okajima, T. Okamura, S. Takase, T. Hara, H. Inoue, L. Fasoli, M. Mofidi, R. Shrivastava, and K. Quader. 2013. A 130.7mm lt;sup gt;2 lt;/sup gt; 2-layer 32Gb ReRAM memory device in 24nm technology. In *2013 IEEE International Solid-State Circuits Conference Digest of Technical Papers*. 210–211. https://doi.org/10.1109/ISSCC.2013.6487703
[46] S. Ma, M. Donato, S. K. Lee, D. Brooks, and G. Wei. 2019. Fully-CMOS Multi-Level Embedded Non-Volatile Memory Devices With Reliable Long-Term Retention for Efficient Storage of Neural Network Weights. *IEEE Electron Device Letters* 40, 9 (Sep. 2019), 1403–1406. https://doi.org/10.1109/LED.2019.2930212
[47] Wenjia Meng, Zonghua Gu, Ming Zhang, and Zhaohui Wu. 2017. Two-Bit Networks for Deep Learning on Resource-Constrained Embedded Devices. *CoRR* abs/1701.00485 (2017). arXiv:1701.00485 http://arxiv.org/abs/1701.00485
[48] Microsoft. [n. d.]. Understanding Frames per Second. ([n. d.]). https://support.microsoft.com/en-us/help/269068/understanding-frames-per-second-fps
[49] Kousuke Miyaji, Yasuhiro Shinozuka, and Ken Takeuchi. 2012. Zero Additional Process, Local Charge Trap, Embedded Flash Memory with Drain-Side Assisted Erase Scheme Using Minimum Channel LengthWidth Standard Complemental Metal-Oxide-Semiconductor Single Transistor Cell. *Jpn. J. Appl. Phys.* 51 (2012).
[50] NASA. 2016. Autonomous car facts 2016. Keynote: Autonomous Car A New Driver for Resilient Computing and Design-for-Test. https://nepp.nasa.gov/workshops/etw2016/talks/15WED/20160615-0930-Autonomous_Saxena-Nirmal-Saxena-Rec2016Jun16-nasaNEPP.pdf
[51] NVIDIA. 2017. NVIDIA Deep Learning Accelerator (NVDLA): a free and open architecture that promotes a standard way to design deep learning inference accelerators. nvdla.org
[52] S. Park, S. Choi, J. Lee, M. Kim, J. Park, and H. J. Yoo. 2016. 14.1 A 126.1mW real-time natural UI/UX processor with embedded deep-learning core for low-power smart glasses. In *2016 IEEE International Solid-State Circuits Conference (ISSCC)*. 254–255. https://doi.org/10.1109/ISSCC.2016.7418003
[53] S. Park, H. Kim, M. Choo, J. Noh, A. Sheri, S. Jung, K. Seo, J. Park, S. Kim, W. Lee, J. Shin, D. Lee, G. Choi, J. Woo, E. Cha, J. Jang, C. Park, M. Jeon, B. Lee, B. H. Lee, and H. Hwang. 2012. RRAM-based synapse for neuromorphic system with pattern recognition function. In *2012 International Electron Devices Meeting*. 10.2.1–10.2.4. https://doi.org/10.1109/IEDM.2012.6479016
[54] Vincenzo Piuri. 2001. Analysis of Fault Tolerance in Artificial Neural Networks. *J. Parallel and Distrib. Comput.* (2001). http://www.sciencedirect.com/science/article/pii/S0743731500916630
[55] M. Poggi and S. Mattoccia. 2016. A wearable mobility aid for the visually impaired based on embedded 3D vision and deep learning. In *2016 IEEE Symposium on Computers and Communication (ISCC)*. 208–213. https://doi.org/10.1109/ISCC.2016.7543741
[56] Brandon Reagen, Robert Adolf, Paul Whatmough, Gu-Yeon Wei, and David Brooks. 2017. Deep Learning for Computer Architects. In *Synthesis Lectures on Computer Architecture*.
[57] Brandon Reagen, Lillian Pentecost, Udit Gupta, Paul Whatmough, Sae Kyu Lee, Niamh Mulholland, David Brooks, and Gu-Yeon Wei. 2018. Ares: A framework for quantifying the resilience of deep neural networks.. In *2018 The 55th Annual Design Automation Conference (DAC)*. https://doi.org/10.1145/3195970.3196083
[58] Brandon Reagen, Paul Whatmough, Robert Adolf, Saketh Rama, Hyunkwang Lee, Sae Kyu Lee, Jose Miguel Hernandez-Lobato, Gu-Yeon Wei, and David Brooks. 2016. Minerva: Enabling Low-Power, Highly-Accurate Deep Neural Network Accelerators. In *ISCA*. http://vlsiarch.eecs.harvard.edu/wp-content/uploads/2016/05/reagen_isca16.pdf
[59] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. 2015. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)* 115, 3 (2015), 211–252. https://doi.org/10.1007/s11263-015-0816-y
[60] Cypress Semiconductor. 2017. What Types of ECC Should Be Used on Flash Memory? http://www.cypress.com/file/203021/download
[61] A. Shafiee, A. Nag, N. Muralimanohar, R. Balasubramonian, J. P. Strachan, M. Hu, R. S. Williams, and V. Srikumar. 2016. ISAAC: A Convolutional Neural Network Accelerator with In-Situ Analog Arithmetic in Crossbars. In *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*. 14–26. https://doi.org/10.1109/ISCA.2016.12
[62] F Sijstermans. 2018. The NVIDIA deep learning accelerator. In *Hot Chips*.
[63] Karen Simonyan and Andrew Zisserman. 2014. Very Deep Convolutional Networks for Large-Scale Image Recognition. (2014).
[64] L. Song, X. Qian, H. Li, and Y. Chen. 2017. PipeLayer: A Pipelined ReRAM-Based Accelerator for Deep Learning. In *2017 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. 541–552. https://doi.org/10.1109/HPCA.2017.55
[65] Olivier Temam. 2012. A Defect-tolerant Accelerator for Emerging High-performance Applications. In *Proceedings of the 39th Annual International Symposium on Computer Architecture (ISCA '12)*. IEEE Computer Society, Washington, DC, USA, 356–367. http://dl.acm.org/citation.cfm?id=2337159.2337200
[66] P. N. Whatmough, S. K. Lee, D. Brooks, and G. Wei. 2018. DNN Engine: A 28-nm Timing-Error Tolerant Sparse Deep Neural Network Processor for IoT Applications. *IEEE Journal of Solid-State Circuits* (2018), 1–10. https://doi.org/10.1109/JSSC.2018.2841824
[67] J.Y. Wu, Y.S. Chen, W.S. Khwa, S.M. Yu, T.Y. Wang, J.C. Tseng, Y.D. Chih, and Carlos H. Diaz. 2018. In *2018 International Electron Devices Meeting*.
[68] Cong Xu, Dimin Niu, N. Muralimanohar, N. P. Jouppi, and Yuan Xie. 2013. Understanding the trade-offs in multi-level cell ReRAM memory design. In *2013 50th ACM/EDAC/IEEE Design Automation Conference (DAC)*. 1–6.
[69] S. Zhang, Z. Du, L. Zhang, H. Lan, S. Liu, L. Li, Q. Guo, T. Chen, and Y. Chen. 2016. Cambricon-X: An accelerator for sparse neural networks. In *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. 1–12. https://doi.org/10.1109/MICRO.2016.7783723
[70] X. Zhang, Y. Xu, H. Hu, Y. Liu, Z. Guo, and Y. Wang. 2013. Modeling and Analysis of Skype Video Calls: Rate Control and Video Quality. *IEEE Transactions on Multimedia* 15, 6 (Oct 2013), 1446–1457. https://doi.org/10.1109/TMM.2013.2247988
[71] Y. Zhang, L. Zhang, W. Wen, G. Sun, and Y. Chen. 2012. Multi-level cell STT-RAM: Is it realistic or just a dream?. In *2012 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 526–532.
[72] H. Zhao, L. Xue, P. Chi, and J. Zhao. 2017. Approximate image storage with multi-level cell STT-MRAM main memory. In *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 268–275. https://doi.org/10.1109/ICCAD.2017.8203788
[73] Liang Zhao, Hong-Yu Chen, Shih-Chieh Wu, Zizhen Jiang, Shimeng Yu, Tuo-Hung Hou, H. . P. Wong, and Y. Nishi. 2014. Improved multi-level control of RRAM using pulse-train programming. In *Proceedings of Technical Program - 2014 International Symposium on VLSI Technology, Systems and Application (VLSI-TSA)*. 1–2. https://doi.org/10.1109/VLSI-TSA.2014.6839673
[74] L. Zhao, H.-Y. Chen, S.-C. Wu, Z. Jiang, S. Yu, T.-H. Hou, H.-S. Philip Wong, and Y. Nishi. 2014. Multi-level control of conductive nano-filament evolution in HfO2 ReRAM by pulse-train operations. *Nanoscale* 6 (2014), 5698–5702. Issue 11. https://doi.org/10.1039/C4NR00500G

RIGHTSLINK