MATIC: Learning Around Errors for Efficient Low-Voltage Neural Network Accelerators

Sung Kim†, Patrick Howe†, Thierry Moreau‡, Armin Alaghi‡, Luis Ceze‡, Visvesh Sathe†

Department of Electrical Engineering†, Paul G. Allen School of Computer Science and Engineering‡, University of Washington, USA

{sungk9, pdh4}@ee.washington.edu, {moreau, armin, ceze}@cs.washington.edu, sathe@ee.washington.edu

***Abstract*—As a result of the increasing demand for deep neural network (DNN)-based services, efforts to develop dedicated hardware accelerators for DNNs are growing rapidly. However, while accelerators with high performance and efficiency on convolutional deep neural networks (Conv-DNNs) have been developed, less progress has been made with regards to fully-connected DNNs (FC-DNNs). In this paper, we propose MATIC (Memory Adaptive Training with In-situ Canaries), a methodology that enables aggressive voltage scaling of accelerator weight memories to improve the energy-efficiency of DNN accelerators. To enable accurate operation with voltage overscaling, MATIC combines the characteristics of destructive SRAM reads with the error resilience of neural networks in a memory-adaptive training process. Furthermore, PVT-related voltage margins are eliminated using bit-cells from synaptic weights as in-situ canaries to track runtime environmental variation. Demonstrated on a low-power DNN accelerator that we fabricate in 65 nm CMOS, MATIC enables up to 60-80 mV of voltage overscaling (3.3× total energy reduction versus the nominal voltage), or 18.6× application error reduction.**

***Keywords*—Deep neural networks, voltage scaling, SRAM, machine learning acceleration.**

I. Introduction

DNNs have demonstrated state-of-the-art performance on a variety of signal processing tasks, and there is growing interest in DNN hardware accelerators for application domains ranging from IoT to the datacenter. However, much of the recent success with DNN-based approaches has been attributed to the use of larger model architectures (i.e., models with more layers), and state-of-the-art model architectures can have millions or billions of trainable weights. As a result of weight storage requirements, neural network algorithms are particularly demanding on memory systems; for instance, the authors of [1] found that main memory accesses dominated the total power consumption for their accelerator design. Subsequently, [2] developed an accelerator that leveraged large quantities of on-chip SRAM, such that expensive off-chip DRAM accesses could be eliminated. To a similar end, [3] used pruning and compression techniques to yield sparse weight matrices, and [4] leveraged data-reuse techniques and run-length compression to reduce off-chip memory access. Nevertheless, once off-chip memory accesses are largely eliminated, on-chip SRAM dedicated to synaptic weights can account for greater than 50% of total system power [3]. The on-chip memory problem is particularly acute in DNNs with dense classifier layers, since classifier weights are typically unique and constitute greater than 90% of total weight parameters [5]. As a result, convolutional data-reuse techniques like those proposed in [4] and [6] lose effectiveness.

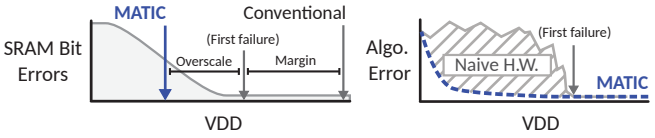


Fig. 1. MATIC enables margin reduction with in-situ canaries, and recovers from errors caused by voltage overscaling with memory-adaptive training.

Voltage scaling can enable significant reduction in static and dynamic power dissipation, however read and write stability constraints have historically prevented aggressive voltage-scaling on SRAM. While alternative bit-cell topologies can be used, they typically trade-off bit-cell stability for non-trivial overheads in terms of area, power, or speed. Hence, designs that employ SRAM either place on-chip memories on dedicated supply-rails, allowing them operate hundreds of millivolts higher than the rest of the design, or the system shares a unified voltage domain. In either case, significant energy savings from voltage scaling remain unrealized due to SRAM operating voltage constraints; this translates to either shorter operating lifetime for battery-powered devices, or higher operating costs. Furthermore, to account for PVT-variation, designers either apply additional static voltage margin, or add dummy logic circuits that detect imminent failure (*canaries*).

In this paper we present Memory Adaptive Training and In-situ Canaries (MATIC, Figure 1), a hardware/algorithm co-design methodology that enables aggressive voltage scaling of weight SRAMs with tuneable accuracy-energy tradeoffs. To achieve this end, MATIC jointly exploits the inherent error tolerance of DNNs with the specific characteristics of SRAM read-stability failures. To evaluate the effectiveness of MATIC we also design and implement SNNAC, a low-power DNN accelerator for mobile devices fabricated in 65 nm CMOS, and demonstrate state-of-the-art energy-efficiency on classification and regression tasks.

The remaining sections are organized as follows. Section II provides background on DNNs and failure modes in 6T SRAM. Section III gives the algorithmic details and design of MATIC, and Section IV describes the prototype chip. Results and comparison to prior works are discussed in Sections V and VI, respectively.

978-3-9819263-0-9/DATE18/©2018 EDAA

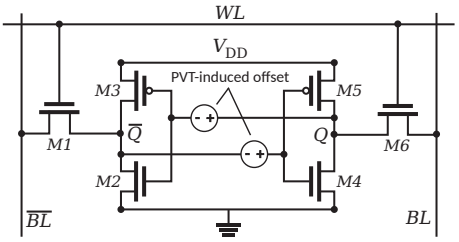


Fig. 2. An SRAM 6T bit-cell with mismatch-induced, input-referred static offsets. During a read, the active pull-down device (either M2 or M4) may be overcome by the current sourced from the pre-charged bit-line (via the access device) if there is insufficient static noise margin.

II. Background

This section briefly reviews relevant background on DNN operation, and SRAM read-stability failure.

A. Deep Neural Networks

DNNs are a class of bio-inspired machine learning models that are represented as a directed graph of neurons [7]. During inference, a neuron k in layer j implements the composite function:

$$z_k^{(j)} = f\left(\sum_{i=1}^{N^{(j-1)}} w_{k,i}^{(j)}\ z_i^{(j-1)}\right),$$

where $z_i^{(j-1)}$ denotes the output from neuron i in the previous layer, and $w_{k,i}^{(j)}$ denotes the weight in layer j from neuron i in the previous layer to neuron k. $f(x)$ is a non-linear function, typically a sigmoidal function or rectified linear unit (ReLU). Since the computation of a DNN layer can be represented as a matrix-vector dot product (with $f(x)$ computed element-wise), DNN execution is especially amenable to dataflow hardware architectures designed for linear algebra.

Training involves iteratively solving for weight parameters using a variant of gradient descent. Given a weight $w_{i,k}^{(j)}$, its value at training iteration $n+1$ is given by:

$$w_{k,i}^{(j)}[n+1] = w_{k,i}^{(j)}[n] - \alpha \frac{\partial J}{\partial w_{k,i}^{(j)}[n]},$$

where α is the step size and J is a suitable loss function (e.g., cross-entropy) [7]. The partial derivatives of the loss function with respect to the weights are computed by propagating error backwards via partial differentiation (*backprop*). MATIC relies on the observation that backprop makes error caused by artificial weight perturbations observable, and subject to compensation via weight adaptation.

B. SRAM Read Failures

Figure 2 shows a 6T SRAM bit-cell that is composed of two cross-coupled inverters (M2/M3 and M4/M5) and access transistors (M1 and M6). Variation-induced mismatch between devices in the bit-cell creates an inherent, state-independent offset [8]. This offset results in each bit-cell having a "preferred state." For instance, the bit-cell depicted in Figure 2 favors driving Q and $\overline{Q}$ to logic '1' and '0', respectively. Due to statistical variation, larger memories are likely to see greater numbers of cells with significant offset error.

As supply voltage scales, the diminished noise margin allows the bit-cell to be flipped to its preferred state during a read [8]. This type of read-disturbance failure, which occurs at the voltage $V_{min,read}$, is a result of insufficient pull-down strength in either M2 or M4 (whichever side of the cell stores 0) relative to the pass-gate and pre-charged bit-line.

Once flipped, the bit-cell retains state in subsequent repeated reads, favouring its (now incorrect) bit value due to the persistence of the built-in offset. Consequently, the occurrence of bit-cell read-stability failure at low supply voltages is random in space, but essentially provides *stable* read outputs consistent with its preferred state [8]. Notably, the read failures described above are in distinct from bit-line access-time failures, which can be corrected with ample timing margin.

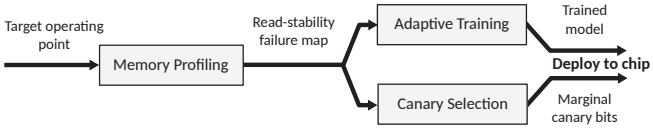


Fig. 3. Overview of the MATIC compilation and deployment flow.

III. Voltage Scaling for DNN Accelerators

We now present MATIC, a voltage scaling methodology that leverages memory-adaptive training (MAT) to operate SRAMs past their point of failure, and in-situ synaptic canaries to remove static voltage margins for accurate operation across PVT variation. In conjunction, the two techniques enable system-wide voltage scaling for energy-efficient operation, with tuneable accuracy-energy tradeoffs. Figure 3 shows an overview of the processing and deployment flow, which is detailed below.

A. SRAM Profiling

SRAM read failures are profiled post-silicon to be used later during the memory-adaptive training process and in-situ canary selection. The SRAM profiling procedure takes place once at compile time, and consists of a read-after-write and read-after-read operation on each SRAM address, at the target DNN accuracy level (bit-error proportion). The word address, bit index, and error polarity of each bit-cell failure are then collected to form a complete failure map for each voltage-scalable weight memory in the hardware design. The entire profiling process and failure-map generation is automated with a host PC that controls on-chip debug software and external digitally-programmable voltage regulators.

B. Memory-Adaptive Training

MAT augments the vanilla backprop algorithm by injecting profiled SRAM bit-errors into the training process, enabling the neural network to compensate via adaptation. As described in Section II, random mismatch results in bit-cells that are statically biased towards a particular storage state. If a bit-cell stores the complement of its "preferred" state, performing a read at a sufficiently low voltage flips the cell and subsequent

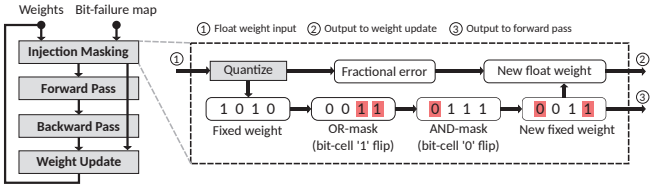


Fig. 4. The modified DNN training algorithm and injection masking process. OR and AND bit masks annotate information on SRAM bit-errors that occur due to voltage overscaling, and quantization convergence issues are avoided by maintaining both float and fixed-point weights.

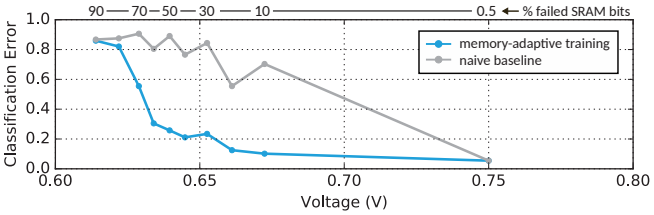


Fig. 5. Simulated performance of memory-adaptive training on MNIST.

reads will be incorrect, but stable. MAT leverages this stability during training with an *injection masking* process (Figure 4). The injection mask applies bit-masks corresponding to profiled bit-errors to each DNN weight before the forward training pass. As a result, the network error propagated in the backward pass reflects the impact of the bit-errors, leading to compensatory weight updates in the entire network. Since the injection masking process operates on weights that correspond to real hardware, weights are necessarily quantized during training to match the SRAM word length. However, previous work has shown that unmitigated quantization during training can lead to significant accuracy degradation [9]. MAT counteracts the effects of quantization during training by preserving the fractional quantization error, in effect performing floating point training to enable gradual weight-updates that occur over multiple backprop iterations. The augmented weight update rule for MAT is given by

$$w_{k,i}^{(j)}[n+1] = m_{k,i}^{(j)}[n] - \alpha \frac{\partial J}{\partial m_{k,i}^{(j)}[n]} + \varepsilon_q, \text{ and}$$

$$m_{k,i}^{(j)}[n] = B_{or} B_{and} Q\left(w_{k,i}^{(j)}[n]\right),$$

in which $m_{k,i}^{(j)}$ is the quantized weight that corresponds to $w_{k,i}^{(j)}$, B_{or} and B_{and} are the bit-masks corresponding to bit-cell faults in the physical SRAM word, Q is the quantization function, and ε_q is the fractional quantization error.

To evaluate the feasibility of memory-adaptive training, we first examine the memory-adaptive training flow with simulated SRAM failure patterns. We implement the training modifications described above in the open-source FANN [10] and Caffe [11] frameworks. A proportion of randomly selected weight bits are statically flipped at each voltage, where the proportion of faulty bits is determined from SPICE Monte Carlo simulations of a 6T bit-cell. Figure 5 shows that a significant fraction of bit errors can be tolerated, and that MATIC provides a reasonably smooth energy-error tradeoff curve.

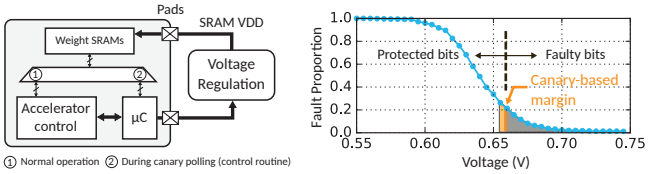


Fig. 6. Overview of the runtime SRAM-voltage control scheme, using bit cells from synaptic weights as in-situ canaries. Between inferences, the integrated μC polls canary bits to determine if voltage modifications should be applied.

Algorithm 1 In-Situ Canary-based Voltage Control

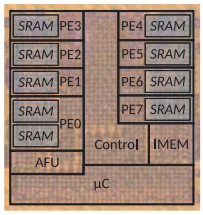
```
C := Set of marginal canary bits
v_0 := Safe initial voltage
▷ ... controller wakes from sleep
SetSRAMVoltage(v_0)
repeat
    SetSRAMVoltage(v − Δv)
    any_failed ← CheckStates(C)
    if any_failed then
        SetSRAMVoltage(v + Δv)
        RestoreStates(C)
    else
        v ← v − Δv
until any_failed
▷ ... controller returns to sleep
```

C. In-Situ Synaptic Canaries

The in-situ canary circuits are bit-cells selected directly from weight SRAMs that facilitate SRAM supply-voltage control at runtime (Figure 6). Traditional canary circuits replicate critical circuits to detect imminent failure, but require added margin and are vulnerable to PVT-induced mismatch. Instead, MATIC uses weight bit-cells directly as in-situ canary circuits, leveraging a select number of bit-cells that are on the margin of read-failure. This maintains a target bit-cell fault *pattern*, and in turn maintains the level of classification accuracy. In contrast to a static voltage margin, the in-situ canary-based margin provides reliability tailored to the specific failure patterns of the test chip. The in-situ canary technique relies on two key observations:

1. Since the most marginal, failure prone bit-cells are chosen as canaries, canaries fail before other performance-critical bit-cells and protect their storage states.
2. Neural networks are inherently robust to a limited number of *uncompensated* errors [12]. As a result, network accuracy is not dependent on the failure states of canary bit-cells, and the actual operating voltage can be brought directly to the $V_{min,read}$ boundary of the canaries.

At runtime, in-situ canary bits are polled by a runtime controller to determine whether supply voltage modifications should be applied. While we use an integrated microcontroller in the test chip described in Section V, the runtime controller can be implemented with faster or more efficient circuits. For canary selection and voltage adjustment, we conservatively select eight distributed, marginal canary bit-cells from each weight-storage SRAM. The in-situ canary-based voltage control routine is summarized in Algorithm 1.



(a)

Technology	TSMC GP 65 nm
Core Area	1.15×1.2 mm
SRAM	9 KB
Voltage	0.9 V
Frequency	250 MHz
Power	16.8 mW
Energy	67.1 pJ/cycle

(b)

Fig. 7. **(a)** Microphoto of a fabricated SNNAC test chip, and **(b)** nominal performance characteristics.

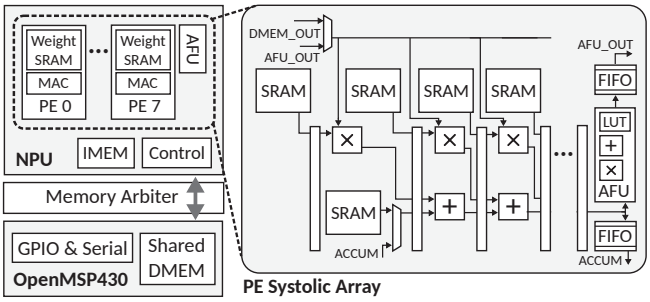


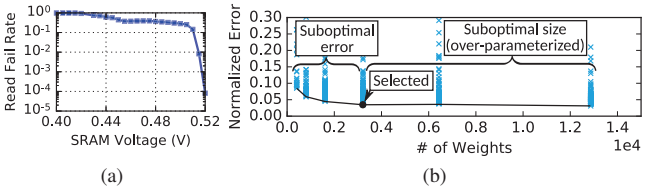
Fig. 8. Architecture of the SNNAC DNN accelerator.

IV. DNN Accelerator Architecture

To demonstrate the effectiveness of MATIC on real hardware, we implement SNNAC (Systolic Neural Network AsiC) in 65 nm CMOS technology (Figure 7). The SNNAC architecture (Figure 8) is based on the open-source systolic dataflow design from SNNAP [13], optimized for integration with a light-weight SoC.

The SNNAC core consists of a fully-programmable Neural Processing Unit (NPU) that contains eight multiply-accumulate (MAC)-based Processing Elements (PEs). The PEs are arranged in a 1D systolic ring that maintains high compute utilization during inner-product operations. Energy-efficient arithmetic in the PEs is achieved with 8-22 bit fixed-point operands, and each PE includes a dedicated voltage-scalable SRAM bank to enable local storage of synaptic weights. The systolic ring is attached to an activation function unit (AFU), which minimizes energy and area footprint with piecewise-linear approximation of activation functions (e.g., sigmoid or ReLU).

The operation of the PEs is coordinated by a lightweight control core that executes statically compiled microcode. To achieve programmability and support for a wide range of layer configurations, the computation of wide DNN layers is time-multiplexed onto the PEs in the systolic ring. When the layer width exceeds the number of physical PEs, PE results are buffered to an accumulator that computes the sum of all atomic MAC operations in the layer. SNNAC also includes a sleep-enabled OpenMSP430-based microcontroller (μC) [14] to handle runtime control, debugging functions, and off-chip communication with a UART serial interface. To minimize data movement, NPU input and output data buffers are memory-mapped directly to the μC data address space.



(a) (b)

Fig. 9. **(a)** Measured SRAM read-failure rate at 25°C. **(b)** Topology selection to avoid biased overparameterization - each point is a unique DNN topology.

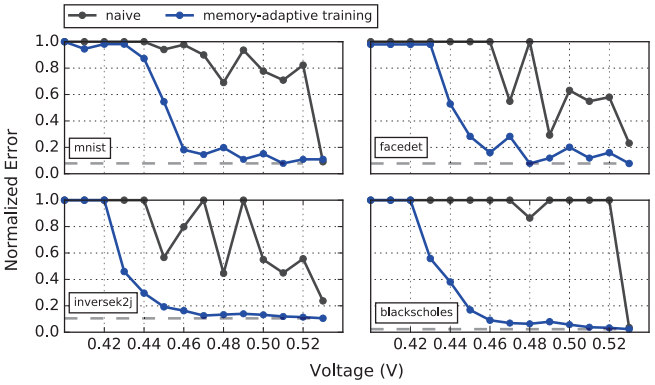


Fig. 10. Error performance of SNNAC, with and without MATIC deployed.

V. Experimental Setup and Hardware Results

At 25°C and 0.9 V, a nominal SNNAC implementation operates at 250 MHz and dissipates 16.8 mW, achieving a 90.6% classification rate on MNIST handwritten character recognition [15]. In addition to MNIST, we evaluate face detection on the MIT CBCL face database [16], and two approximate computing benchmarks from [17]. For all of the benchmark tasks, we divide the datasets into training and test subsets with either a 7-to-1 or 10-to-1 train/test split.

We find that the compiled SRAMs (rated at 0.9 V) exhibit bit-errors starting from 0.53 V at room temperature, with all reads failing at ~0.4 V (Figure 9a). Since the point of first failure is dictated by the tails of the $V_{min,read}$ statistical distribution, we expect voltage savings to increase in more advanced process nodes, and with larger memories. For instance, the SRAM variability study from [8] exhibits $V_{min,read}$ failures starting at ~0.66 V with a 45 nm, 64 kb array.

A. Application Error

Figure 10 shows how MATIC recovers application error after the point of first failure. Compared to a voltage-scaled naive system (the SNNAC accelerator operating with baseline DNN models), MATIC demonstrates much lower application error. The baseline and memory-adaptive models use the same DNN model topologies (e.g., layer depth and width configurations), but memory-adaptive training modifications are disabled for the naive case. To avoid unfair bias in the application error analysis, all benchmarks use *compact* DNN topologies that minimize intrinsic over-parameterization (Figure 9b). Table I lists the benchmarks along with model descriptions, and application error for the baseline (naive) and memory-adaptive evaluations. We list the application error at the nominal

TABLE I. DNN BENCHMARKS AND APPLICATION ERROR MEASUREMENTS.

Benchmark	Description	Error Metric	DNN Topology	Error@0.9 V (nominal)	Error@0.50 V (naive)	Error@0.50 V (adaptive)	Error@0.46 V (naive)	Error@0.46 V (adaptive)	AEI (naive)	AEI (adapt.)	Reduction
mnist [15]	Digit recognition	Classif. rate	100-32-10	9.4%	70.7%	13.0%	84.0%	15.6%	62.5%	5.0%	**12.5**
facedet [16]	Face detection	Classif. rate	400-8-1	12.5%	33.6%	15.6%	47.7%	15.8%	37.5%	5.6%	**6.7**
inversek2j [17]	Inverse kinematics	Mean sq. error	2-16-2	0.032	0.169	0.040	0.245	0.050	50.7%	1.9%	**26.7**
bscholes [17]	Option pricing	Mean sq. error	6-16-1	0.021	0.094	0.023	0.094	0.026	65.3%	2.3%	**28.4**
Average	-	-	-	-	-	-	-	-	-	-	**18.6**

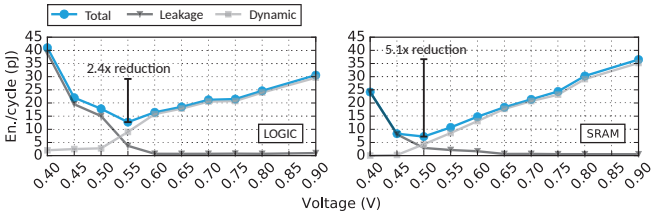


Fig. 11. Energy-per-cycle measurements for SNNAC, obtained from test chip current measurements.

TABLE II. ENERGY-EFFICIENCY WITH MATIC-ENABLED SCALING.

Param/Config.	*HighPerf*	Base	*EnOpt_split*	Base	*EnOpt_joint*	Base
Logic Voltage (V)	0.9	0.9	0.55	0.55	0.55	0.9
SRAM Voltage (V)	0.65	0.9	0.5	0.9	0.55	0.9
Frequency (MHz)	250	-	17.8	-	17.8	-
Total Energy (pJ/cycle)	**48.96**	**67.08**	**19.98**	**49.23**	**20.60**	**67.08**
Logic	30.58	30.58	12.73	12.73	12.73	30.58
SRAM	18.37	36.50	7.24	36.50	7.86	36.50
Energy Reduction	**1.4**×	-	**2.5**×	-	**3.3**×	-

voltage (0.9 V), in addition to the energy-optimal voltage (0.5 V), and 0.46 V, which is the voltage where application error increases significantly. Between 0.46 V and 0.53 V, the use of MATIC results in 6.7× to 28.4× application error reduction versus naive hardware. When averaged across both voltage and all benchmarks, the average error-increase (AEI) is reduced by 18.6×.

B. Energy-Efficiency

For energy-efficiency we consider the operation of SNNAC in three feasible operating scenarios, *HighPerf* (high performance, maximum frequency), *EnOpt_split* (energy optimal, disjoint logic and SRAM voltages), and *EnOpt_joint* (energy optimal, unified voltage domains). Figure 11 shows the energy-per-cycle measurements on SNNAC for both logic and weight SRAMs, derived from test chip leakage and dynamic current measurements. In *HighPerf*, operating frequency determines voltage settings, while frequency settings for *EnOpt_split* and *EnOpt_joint* are determined by the minimum-energy point (MEP) subject to voltage domain configurations. The baselines for each operating scenario use the same clock frequencies and logic voltages as the optimized cases, but with SRAM operating at the nominal voltage.

In *HighPerf*, we observe that timing limitations prevent voltage scaling for the logic and memory in both the baseline and optimized configurations. However, while the baseline is unable to scale SRAM voltage due to stability margins, the optimized case (with MATIC) is able to scale SRAM down to 0.65 V, resulting in 1.4× energy savings; timing requirements in the SRAM periphery prevent further scaling.

In *EnOpt_split*, where SRAM and logic power rails are separated, the baseline is able to scale logic to the MEP but

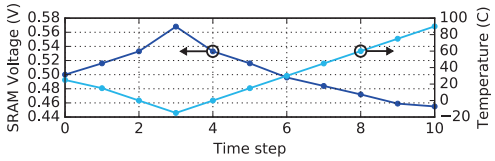


Fig. 12. Runtime closed-loop SRAM voltage control enabled by the in-situ canary system, in response to ambient temperature variation. The actual distance between time steps varies from 2-10 minutes due to the variable heating/cooling rate of the test chamber.

TABLE III. COMPARISON WITH STATE-OF-THE-ART DNN ACCELERATORS

	Low-power Fully-connected			High-perf. Conv.	
	This Work*	ISSCC'17* [18]	ISCA'16 [3]	DATE'17 [6]	ISSCC'16* [4]
Process	**65 nm**	40 nm	45 nm	28 nm	65 nm
Area (mm sq.)	**1.4**	7.1	0.64	-	12.2
DNN Type	**Fully-conn.**	Fully-conn.	Fully-conn.	Conv.	Conv.
Power (mW)	**0.37**	0.29	9.2	33	567.5
Frequency (MHz)	**17.8**	3.9	800	204	700
Voltage (V)	**0.44-0.9**	0.63-0.9	1.0	0.9	0.82-1.17
Energy (GOPS/W)	**119.2 / 400.5**†	374	174	1421	243

*Performance metrics from a fabricated chip,
† Nominal energy efficiency / efficiency with MATIC

SRAM remains at the nominal voltage. While the baseline is unable to voltage-scale its weight memories, with MATIC, we are able to scale both logic and SRAM to the MEP, leading to 2.5× energy savings. Furthermore, SRAM energy is minimized at 0.5 V with a 28% SRAM bit-cell failure rate, which corresponds to an 87% classification rate on MNIST (versus 29.3% for naive hardware).

Finally, in *EnOpt_joint*, where voltage domains are unified to emulate a system with stringent power grid requirements, the baseline is unable to scale *both* SRAM and logic voltages. While logic voltage in the *HighPerf* scenario was limited due to timing requirements, logic in the baseline case for *EnOpt_joint* is limited by SRAM $V_{min,read}$ since the power rails are shared; in this case, *SRAM PVT and read stability margins prevent system-wide voltage scaling.* The MATIC-SNNAC combination, on the other hand, is able to scale both logic and SRAM voltages to the unified energy-optimal voltage, 0.55 V, which results in 3.3× energy savings. The baseline design in *EnOpt_split* is more efficient than the baseline design in *EnOpt_joint*. Although the relative savings-versus-baseline shows better results for *EnOpt_joint*, the *EnOpt_split* configuration provides the highest efficiency. The energy-per-cycle measurements for the scenarios described above are summarized in Table II.

C. Temperature Variation

To demonstrate system stability over temperature, we execute the application benchmarks in a chamber with ambient temperature control, and sweep temperature from -15°C to

90°C for a given nominal voltage. After initialization at the nominal voltage and temperature, we sweep the temperature down to -15°C, and then up from -15°C to 90°C in steps of 15°C, letting the chamber stabilize at each temperature point. Figure 12 shows the SRAM voltage settings dictated by the in-situ canary system for an initial setting at 0.5 V on *inversek2j*. The results illustrate how the in-situ canary technique adjusts SRAM voltage to track temperature variation, while conventional systems would require static voltage margins. We note that the operating voltages for the temperature chamber experiments are below the temperature inversion point for the 65 nm process; this is illustrated by the inverse relationship between temperature and SRAM voltage.

D. Performance Comparison

A comparison with recent DNN accelerator designs is listed in Table III. The performance comparison shows that the MATIC-SNNAC combination is comparable to state-of-the-art accelerators despite modest nominal performance, and enables a comparatively wider operating voltage range. While the algorithmic characteristics (and hardware requirements) of networks containing convolutional layers are vastly different from FC-oriented DNNs [19], we include two recent Conv accelerators to show that SNNAC is competitive despite the lack of convolution-oriented optimization techniques.

VI. Related Work

There has been a vast body of work on DNN hardware accelerators [1]–[4], [6], [13], [18] (see Section I), as well as work addressing DNN fault tolerance. Temam [12] explores the impact of defects in logic gates, and is among the first to develop fault-tolerant hardware for neural networks. Srinivasan et al. [20] exploit DNN resilience with a mixed 8T-6T SRAM where weight MSBs are stored in 8T cells. However, this approach has no adaptation mechanism. Xin et al. [21] and Liu et al. [22] design variability-tolerant training schemes, but for simulated resistive-RAM-based crossbars. Yang and Murmann [23] develop a noisy training scheme for Conv-DNNs, however noise is only added to input images. In contrast, our work focuses on increasing the energy-efficiency of DNN accelerators in standard CMOS technologies by targeting a primary bottleneck in power dissipation, namely weight storage, and presents results from a fabricated DNN accelerator.

VII. Conclusions

This paper presents a methodology and algorithms that enable energy-efficient DNN accelerators to gracefully tolerate bit-errors from memory supply-voltage overscaling. Our approach uses (1) memory-adaptive training - a technique that leverages the adaptability of neural networks to train around errors resulting from SRAM voltage scaling, and (2) in-situ synaptic canaries - the use of bit-cells directly from weight SRAMs for voltage control and variation-tolerance. To validate the effectiveness of MATIC, we designed and implemented SNNAC, a low-power DNN accelerator fabricated in 65 nm CMOS. As demonstrated on SNNAC, the application of MATIC results in either 3.3× total energy reduction and 5.1× energy reduction in SRAM, or 18.6× reduction in application error. Thus, MATIC enables accurate inference on aggressively voltage-scaled DNN accelerators, and enables robust and efficient operation for a general class of DNN accelerators.

Acknowledgment

The authors would like to thank Fahim Rahman, Rajesh Pamula, and John Euhlin for design support. This work was supported in part by the National Science Foundation Grant CCF-1518703, generous gifts from Oracle Labs, Nvidia, and by C-FAR, one of the six SRC STARnet Centers, sponsored by MARCO and DARPA.

References

[1] T. Chen *et al.*, "DianNao: A Small-footprint High-throughput Accelerator for Ubiquitous Machine-learning," in *ASPLOS*, pp. 269–284, 2014.

[2] Z. Du *et al.*, "ShiDianNao: Shifting Vision Processing Closer to the Sensor," in *ISCA*, pp. 92–104, 2015.

[3] S. Han *et al.*, "EIE: Efficient Inference Engine on Compressed Deep Neural Network," in *ISCA*, pp. 243–254, 2016.

[4] Y.-H. Chen *et al.*, "Eyeriss: An Energy-Efficient Reconfigurable Accelerator for Deep Convolutional Neural Networks," in *ISSCC*, 2016.

[5] A. Krizhevsky *et al.*, "Imagenet classification with deep convolutional neural networks," in *NIPS*, pp. 1097–1105, 2012.

[6] S. Wang *et al.*, "Chain-NN: An energy-efficient 1D chain architecture for accelerating deep convolutional neural networks," in *DATE*, pp. 1032–1037, 2017.

[7] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., 2006.

[8] Z. Guo *et al.*, "Large-Scale SRAM Variability Characterization in 45 nm CMOS," *JSSC*, vol. 44, no. 11, pp. 3174–3192, 2009.

[9] S. Gupta *et al.*, "Deep Learning with Limited Numerical Precision," *CoRR*, vol. abs/1502.02551, 2015.

[10] S. Nissen, "Implementation of a Fast Artificial Neural Network Library (fann)," University of Copenhagen, Tech. Rep., 2003, http://fann.sf.net.

[11] Y. Jia *et al.*, "Caffe: Convolutional Architecture for Fast Feature Embedding," in *ACM Multimedia*, pp. 675–678, 2014.

[12] O. Temam, "A defect-tolerant accelerator for emerging high-performance applications," in *ISCA*, pp. 356–367, 2012.

[13] T. Moreau *et al.*, "SNNAP: Approximate computing on programmable SoCs via neural acceleration," in *HPCA*, pp. 603–614, 2015.

[14] O. Girard, "OpenMSP430," http://opencores.org, 2009-2015.

[15] Y. LeCun and C. Cortes, "MNIST handwritten digit database," 2010.

[16] M. Alvira and R. Rifkin, "An Empirical Comparison of SNoW and SVMs for Face Detection," MIT, Cambridge, MA, Tech. Rep., 2001.

[17] H. Esmaeilzadeh *et al.*, "Neural acceleration for general-purpose approximate programs," in *MICRO*, pp. 449–460, 2012.

[18] S. Bang *et al.*, "A 288 μW programmable deep-learning processor with 270KB on-chip weight storage using non-uniform memory hierarchy for mobile intelligence," in *ISSCC*, pp. 250–251, 2017.

[19] N. P. Jouppi *et al.*, "In-Datacenter Performance Analysis of a Tensor Processing Unit," in *ISCA*, pp. 1–12, 2017.

[20] G. Srinivasan *et al.*, "Significance driven hybrid 8T-6T SRAM for energy-efficient synaptic storage in artificial neural networks," in *DATE*, pp. 151–156, 2016.

[21] L. Xia *et al.*, "Fault-Tolerant Training with On-Line Fault Detection for RRAM-Based Neural Computing Systems," in *DAC*, pp. 33:1–33:6, 2017.

[22] C. Liu *et al.*, "Rescuing Memristor-based Neuromorphic Design with High Defects," in *DAC*, pp. 87:1–87:6, 2017.

[23] L. Yang and B. Murmann, "SRAM voltage scaling for energy-efficient convolutional neural networks," in *ISQED*, pp. 7–12, 2017.