In-Memory Computation of a Machine-Learning Classifier in a Standard 6T SRAM Array

Jintao Zhang, *Student Member, IEEE*, Zhuo Wang, *Member, IEEE*, and Naveen Verma *Member, IEEE*,

***Abstract*— This paper presents a machine-learning classifier where computations are performed in a standard 6T SRAM array, which stores the machine-learning model. Peripheral circuits implement mixed-signal weak classifiers via columns of the SRAM, and a training algorithm enables a strong classifier through boosting and also overcomes circuit nonidealities, by combining multiple columns. A prototype 128 × 128 SRAM array, implemented in a 130-nm CMOS process, demonstrates ten-way classification of MNIST images (using image-pixel features downsampled from 28 × 28 = 784 to 9 × 9 = 81, which yields a baseline accuracy of 90%). In SRAM mode (bit-cell read/write), the prototype operates up to 300 MHz, and in classify mode, it operates at 50 MHz, generating a classification every cycle. With accuracy equivalent to a discrete SRAM/digital-MAC system, the system achieves ten-way classification at an energy of 630 pJ per decision, 113× lower than a discrete system with standard training algorithm and 13× lower than a discrete system with the proposed training algorithm.**

***Index Terms*— Analog computation, classification, image detection, in-memory computation, machine learning.**

I. Introduction

MACHINE-LEARNING algorithms enable data-driven models for inference, and are thus playing an important role in sensing applications where correlations between embedded signals and inferences of interest are complex and difficult to model analytically. In many such applications, there is the need for always-on sensing and inference, so that systems can respond as specific events of interest occur. The challenge is that the state-of-the art machine-learning models can be complex, requiring several millijoules of energy per decision [1]–[3]. An alternative is the approach in Fig. 1. Here, an ultralow-energy detector, still employing machine-learning models to address the complex correlations, provides somewhat coarser but continuous detection, to selectively activate a full-functioned node (note, the sensor energy shown [4] may also be reduced, for instance by accessing only a subset of the imager pixels required for coarse detection).

Looking at the energy for such detection shows that memory accessing can dominate. The reason is that data-driven models often do not have compact parametric representations, and their access from even modest-sized memories poses orders-of-magnitude higher energy than computation (e.g., 20–100 pJ per access of 16-b word from 32 kB to 1 MB memory versus 1 pJ per multiply, in 45-nm CMOS [5]). An underlying limitation emerges in current architectures for digital accelerators, which separate data storage from computation. Storing the data fundamentally associated with a computation requires area, and thus, its communication to the location of computation incurs energy and throughput cost, which can dominate. This has motivated thinking about architectures that integrate some forms of memory and compute [6]–[10]. This paper presents a machine-learning classifier where data storage and computation are combined in a standard 6T SRAM [10], overcoming this limitation.

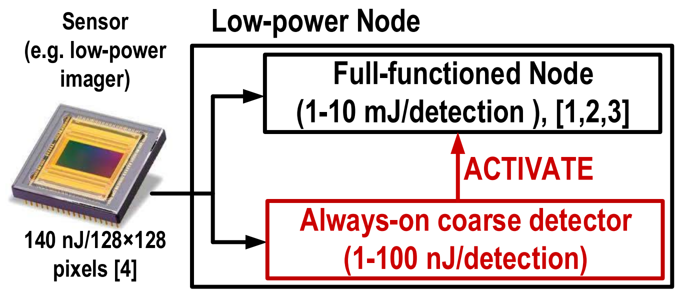


Fig. 1. Architecture for always-on sensing and inference, based on a low-energy coarse detector to trigger a full-functioned node.

This paper is organized as follows. Section II provides an overview of the in-memory-classifier architecture. Section III presents the algorithm for training the classifier, particularly to overcome circuit limitations. Section IV presents circuit-level design details, and Section V presents prototype-measurement and application-demonstration results. Finally, Section VI analyzes the proposed architecture with respect to the fundamental limitation identified above for traditional digital accelerators, and Section VII concludes this paper.

II. System Overview

Fig. 2 shows the architecture of the proposed in-memory classifier. It consists of a standard 6T bit-cell array, and periphery for two modes of opertation. In the *SRAM mode*, the operation is typical read/write of digital data. This is how machine-learning models derived from training are stored in bit cells. In the *Classify Mode*, all wordlines (WLs) are driven at once to analog voltages. Thus, *parallel operation of all bit cells is involved* (by comparison, and in the SRAM mode, only one WL is driven at a time). Each analog WL voltage corresponds to a feature in a feature vector we wish to classify. The features are provided as digital data, loaded through the

Manuscript received August 10, 2016; revised October 14, 2016; accepted December 7, 2016. This paper was approved by Guest Editor Makoto Ikeda.

The authors are with Princeton University, the Department of Electrical Engineering, Princeton, NJ 08544 USA (e-mail: jintao@princeton.edu; zhuow@princeton.edu; nverma@princeton.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JSSC.2016.2642198

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

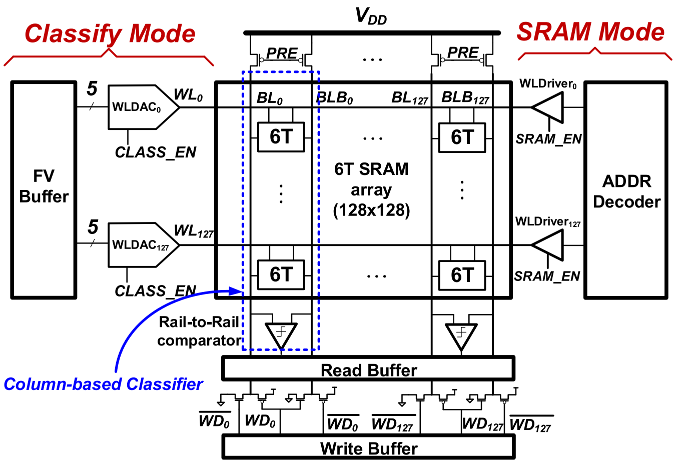


Fig. 2. Architecture of in-memory classifier, showing periphery for two modes of operation.

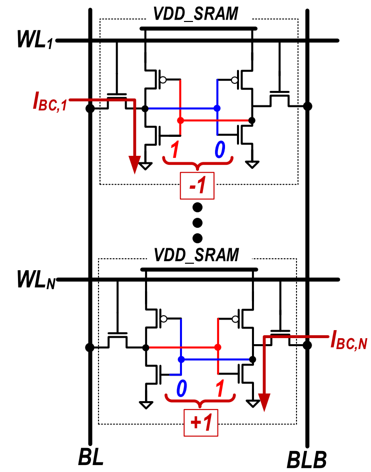


Fig. 3. Column-based weak classifier within the in-memory classifier architecture.

feature-vector buffer, and analog voltages are generated using a WLDAC in each row.

In the Classify Mode, each SRAM column forms a *weak classifier*. In machine learning, a *weak classifier* is one that cannot be trained to fit arbitrary data distributions, and a *strong classifier* is one that can be. Due to fitting errors, a weak classifier typically has low performance in real-world application. For example, a linear classifier is a weak classifier, because it only implements linear decisions boundaries in a feature space, where generally feature data may take on complex distributions. On the other hand, a strong classifier (e.g., support-vector machine with radial-basis-function kernel) can form arbitrary decision boundaries for separating such data distributions. Thus, a strong classifier is ultimately required; but, below, the operation of the column-based weak classifier is first described.

A. Column-Based Weak Classifier

In a standard linear classifier, computation of a decision d is shown in 1, where x_i corresponds to elements from a feature vector $\vec{x}$, and w_i corresponds to weights in a weight vector $\vec{w}$, derived from training

$$d = sgn\left(\sum_{i=1}^{N} w_i \times x_i\right). \quad (1)$$

As shown in Fig. 3, in the Classify Mode, each column of the SRAM performs a similar computation. First, the bit-line pair (BL/BLB) is precharged. Then, the WLs are driven with analog voltages representing the feature values x_i, leading to corresponding bit-cell currents $I_{BC,i}$. Each $I_{BC,i}$ is applied to either BL or BLB depending on the data stored in the bit cell. Thus, treating BL/BLB as a differential voltage signal, the stored data can be thought of as multiplying a feature value (represented as $I_{BC,i}$), by a weight of -1 or $+1$, respectively. Finally, currents from all bit cells are summed together on BL/BLB resulting in aggregated discharge, and a comparator provides sign thresholding.

Thus, the structure operates as a classifier, but one that is even *weaker* than a linear classifier, because the bit-cell currents are nonideal (due to variation and nonlinearity) and the weights are restricted to $+/-1$. Section III presents a machine-learning algorithm for achieving a strong classifier, specifically addressing these points.

III. Classifier Training

A specialized training algorithm is required to address the nonidealities of the column-based weak classifiers. Boosting [11] is an approach from machine learning for constructing a strong classifier from multiple base weak classifiers. In addition to the typical problem that boosting addresses, namely overcoming errors due to inadequate fitting of the weak classifiers, the column-based classifiers raise two additional challenges. First, their substantial circuit nonidealities (bit-cell variations and nonlinearities) cause classification outputs to deviate from those expected of even a nominal weak classifier. Second, while a column-based classifier is similar to a linear classifier with weights restricted to 1-b (Section II-A), standard linear-classifier training algorithms followed by such extreme quantization would lead to inadequate performance. Sections III-A and III-B describe the training algorithm, starting with a boosting algorithm to overcome circuit nonidealities, followed by a base-classifier training algorithm to overcome the very low weight precision.

A. Error-Adaptive Classifier Boosting

Error-adaptive classifier boosting (EACB) [12] is an approach that extends from Adaptive Boosting (AdaBoost) [11]. In AdaBoost, base weak classifiers are trained iteratively, at each stage biased to emphasize and correct fitting errors from the previous iterations (which arise due to a decision boundary not adequately separating training data from different classes). EACB performs training using the specific nonideal instances of implemented weak classifiers to bias each stage of training. Thus, errors due to

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

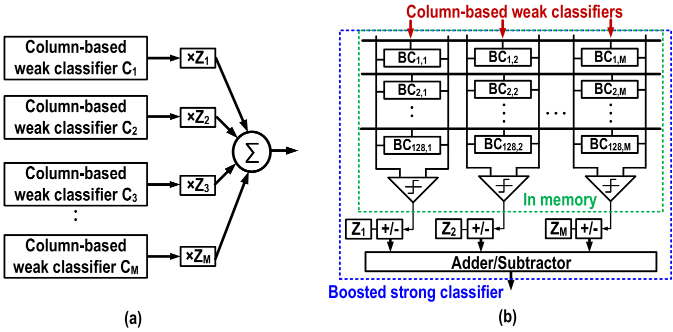


Fig. 4. Illustration of EACB. (a) Logical structure of a resulting strong classifier and (b) its implementation within the in-memory classifier architecture.

the nonideal implementation are adaptively corrected along with fitting errors.

Once all base weak classifiers are thus trained, the strong classifier is constructed via weighted voting over weak-classifier decisions. Fig. 4(a) shows the logical structure, and Fig. 4(b) shows its implementation, using multiple column-based weak classifiers for boosting and weighted voting over these outside the SRAM. It is expected that circuit nonidealities will increase the number of column-based weak classifiers required; this will be shown in the measurement results in Section V-B. Note that all weak classifiers take the same feature vector, readily inputted in the SRAM structure via the shared WLs.

We point out that if errors due to die-to-die circuit variations are large, we would like devices to self-construct their own EACB models on a die-to-die basis. This raises two concerns, which have been previously explored in EACB [12]: 1) access to a training set (data and labels) and 2) computational resources for training. During operation, training-set data are readily available, including feature vectors and associated weak-classifier outputs for biasing the training iterations. While training-set labels (ground truths) are not available, in an architecture such as Fig. 1, a high-performance (energy-intensive) classifier can be implemented in the full-functioned node, whose class declarations provide *estimated* labels. This is shown to enable performance at a level whose limit is set by the high-performance classifier and a nonideal classifier trained with true labels [12]. Since training typically occurs infrequently, computational resources (e.g., energy) can be largely amortized. An exception is the embedded memory needed to store the training set, which must typically be large for low generalization error. This has been addressed in EACB by exploiting the relative immunity of weak classifiers against overfitting. This enables a smaller training set to be used for each iteration, while a new training set can be selectively acquired at each iteration to enhance diversity across the strong ensemble. Thus, the instantaneous training set is reduced, substantially mitigating the embedded memory required [12].

B. Base Classifier Training for Binary Weights

The problem with quantizing weights to 1-b after standard linear-classifier training (e.g., linear regression) is that the resulting model is significantly perturbed, no longer optimally fitting the training set. Instead, we integrate weight quantization *in* the optimization cost function of the base-classifier learning rule. The resulting model has weaker fitting than with high-precision weights, but is optimal at the 1-b level. Equation (2) specifies the optimization, where 1-b quantization is represented by constraining the weight vector $\vec{w}$ to $+/-1$, and also introducing an additional postive scaling variable α to be optimized ($\vec{x}_s/y_s$ is the training-set feature-vectors/labels)

$$\underset{\alpha,\vec{w}}{\text{minimize}} \sum_s (y_s - \alpha \cdot \vec{w} \cdot \vec{x}_s)^2$$
$$\text{subject to } \alpha > 0, \quad \vec{w} \in \{-1, 1\}^N. \tag{2}$$

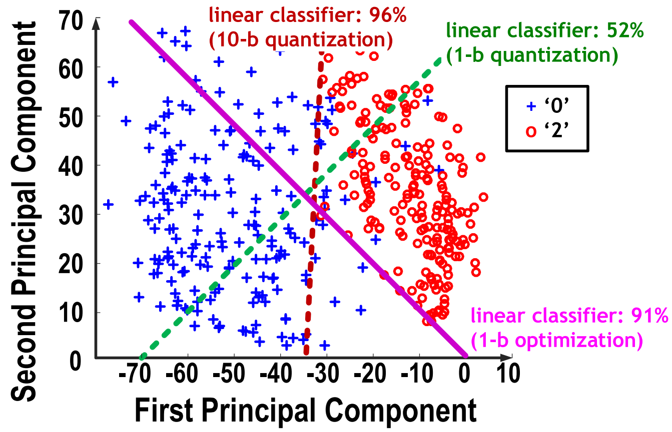


Fig. 5. Demonstration of discrete optimization in hand written digit recognition of digit 0 versus 2.

Unlike conventional linear regression, this optimization is discretized and not convex, with complexity scaling exponentially in the vector dimensionality N. However, as shown in (5), by pulling α scaling into the constraints (such that $\vec{v} = \alpha \cdot \vec{w}$), as well as introducing binary variables b_i (to be optimized) and a constant c [simply chosen to be larger than $\alpha + max(|v_i|)$], reformulation to a mixed-integer program is possible (i.e., quadratic objective and linear constraints). For this, fast solvers such as [13] are available

$$\underset{\vec{v},\alpha,\vec{b}}{\text{minimize}} \sum_s (y_s - \vec{v} \cdot \vec{x}_s)^2 \tag{3}$$
$$\text{subject to } -\alpha \le v_i \le \alpha, \quad v_i + c \cdot b_i \ge \alpha,$$
$$v_i + c \cdot (b_i - 1) \le -\alpha \tag{4}$$
$$\text{where } b_i \in \{0, 1\}, \quad c > \alpha + max(|v_i|), \; i = 1, \ldots, N. \tag{5}$$

Fig. 5 illustrates this approach in an MNIST [14] digit-recognition application, for a 0-versus-2 classifier[1] using 81 image-pixel features (downsampled from 784, and projected to two dimensions using PCA for visualization). The decision boundary for a linear classifier with weights quantized to 10-b retains high accuracy of 96%. Decision boundaries from 1-b weights correspond to 45° lines in the 2-D feature space, with simple quantization leading to a low accuracy

[1]This represents a relatively difficult classification due to similar shape of the hand-written digits.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

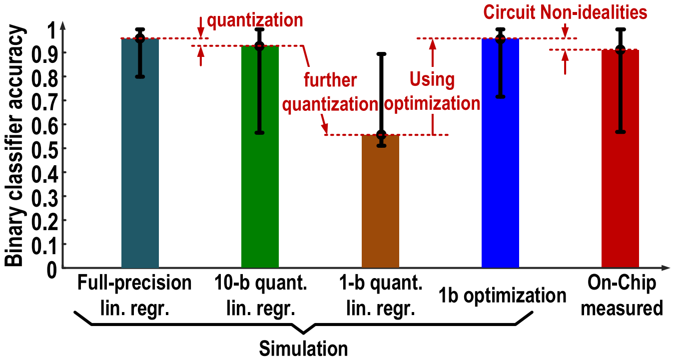


Fig. 6. Comparison of base-classifier training accuracy using standard and proposed approaches (error bars represent minimum/maximum performance across 45 binary classifiers required for ten-way classification of MNIST images [14]).

of 52%. On the other hand, 1-b quantization using the optimization above substantially restores accuracy to 91% (we point out that the separability in an 81-D feature space is better than that actually seen in the 2-D projection).

Fig. 6 overviews the training approach, using training accuracy for various base classifiers in an MNIST [14] 0–9 digit-recognition application (error bars show min/max accuracy over 45 required binary classifiers for all pairs of digits). First, it is the performance of a linear classifier trained via linear regression, using 64-b floating-point weights. Next, it is a classifier with weights quantized to 10-b (found to be the limit before accuracy substantially degrades). Then, it is a classifier with further weight quantization to 1-b, showing poor accuracy. After this, it is the performance of a classifier using 1-b weights from the optimization above, showing significantly higher accuracy. All classifiers thus far represent ideal implementation, simulated in MATLAB. The last bar shows the measured accuracy of column-based classifiers from the implemented prototype. We see degradation due to circuit nonidealities, showing that EACB is critical for restoring accuracy to the level of the ideal system.

IV. Circuit Design

This section presents the details of the circuit design and its analysis.

A. Wordline DAC

Fig. 7 shows the WLDAC circuit. The 5-b digital feature values $X[4:0]$ are inputted to a current DAC, formed from binary-weighted pMOS current sources. The resulting current I_{DAC} is converted into an output WL voltage by running it through an upsized replica of a bit cell. The driver-transistor replica $M_{D,R}$ receives a V_{DD} gate bias when Classify Mode is enabled (via $CLASS\_EN$), thus representing a pull-down condition in a bit cell. The access-transistor replica $M_{A,R}$ is self-biased to generate a WL voltage corresponding to the DAC current. Consequently, I_{DAC} is mirrored by the bit cell, giving roughly linear I_{BC} with the inputted digital feature value (scaled by the upsizing ratio R), and tracking with voltage, temperature, and process skews is achieved.

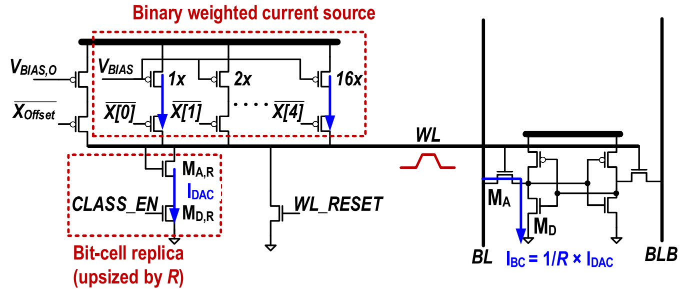


Fig. 7. WLDAC circuit, consisting of binary-weighted current sources and a bit-cell replica, to generate analog voltage on WL corresponding to inputted digital feature values.

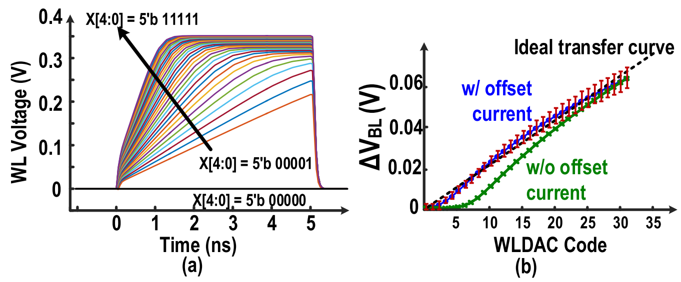


Fig. 8. Simulation of WLDAC, showing (a) transient WL pulses with variable settling time for different input codes and (b) bit-line discharge transfer function versus WLDAC input code, showing standard deviation (due to variations) and nonlinearity.

Fig. 8(a) shows a transient simulation (capacitances extracted) of the WL voltage (at 100 MHz operation), for different $X[4:0]$ codes. The biasing of pMOS current sources and the upsizing ratio R sets the WL amplitudes. The maximum amplitude is designed taking the aggregate bit-cell current and BL/BLB capacitance into account, to ensure BL/BLB discharge does not excessively saturate. We note that the design is roughly independent as the number of bit cells in the column increases, since nominally the ratio of aggregate bit-cell current to BL/BLB capacitance remains constant. Thus, approaches for creating an SRAM compiler may be considered. This also implies that the discharge-phase duration (set by the start of the WL pulse and the instant of sense-amplifier strobing) remains roughly constant (usually set in SRAM compilers by designing a controlled timing path [15], [16]).

With regard to error sources, we see from Fig. 8(a) that the WL settling time is variable. This is due to self-biasing of the bit-cell replica (i.e., DAC output impedance $\sim 1/g_{mA,R}$ depends I_{DAC}). Thus, at low currents, driving the large WL capacitance exhibits slewing behavior. This leads to a source of nonlinearity, as shown in the BL-discharge transfer function of Fig. 8(b). To mitigate this, an offset current source is included in the DAC (i.e., pMOS devices on the far right in Fig. 7) to reduce its output impedance, and the current source is enabled irrespective of the digital input code. As seen, this notably improves the transfer-function linearity.

Some remaining nonlinearity is observed, but is addressed by EACB. Strictly speaking, nonlinearity poses a systematic

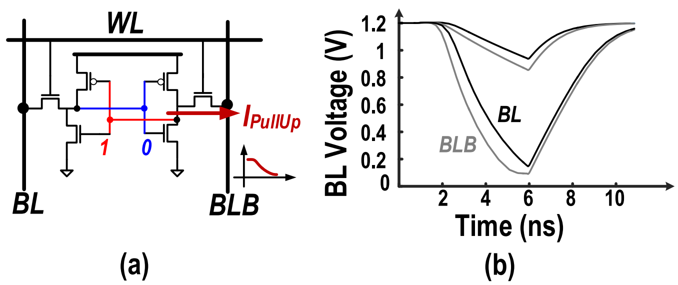


Fig. 9. Bit-line discharge nonlinearity. (a) Source arising from possible pull-up condition in bit cells. (b) Simulation showing variable compression of nominally equivalent BL/BLB differential voltage due to different common-mode levels.

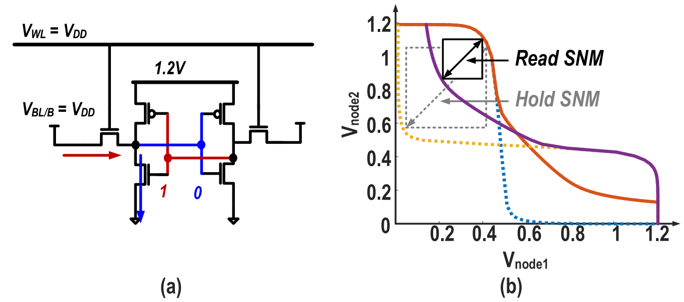


Fig. 10. Possible cell upset in SRAM Mode (a) arising due to read condition and (b) measured by read SNM.

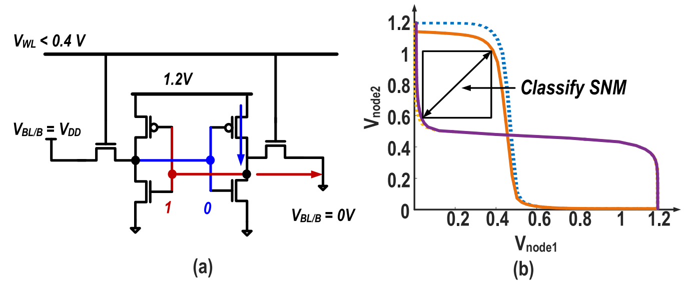


Fig. 11. Possible cell upset in Classify Mode (a) arising due to large-swing BL/BLB discharge condition and (b) measured by "Classify SNM."

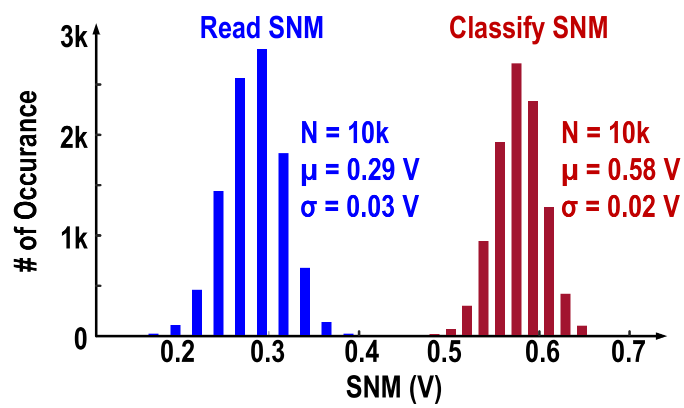


Fig. 12. Comparison between SRAM Mode (read) SNM and Classify Mode SNM using 10k-point Monte Carlo simulation, showing that Classify Mode has much larger margin and lower standard deviation.

source of error across all weak classifiers, not accounted for in the learning rule. Thus, even if the model derived at each iteration of training is appropriately adapted, a consistent nonlinearity in applying the model at every iteration will prevent the errors from being corrected (and a change in the learning rule to account for the nonlinearity will be required [17]). However, if variations dominate the nonlinearity, the errors will not be consistent across iterations, and will be corrected over the iterations [12]. Error bars in Fig. 8(b), representing the standard deviation (from Monte Carlo simulations), show that this is the case.

Yet another source of error arises due to bit-line discharge. The transfer function in Fig. 8(b) assumes that BL/BLB remains near their precharge voltage. However, unlike the SRAM mode where low-swing BL/BLB is preferred, in the Classify Mode, large BL/BLB swings are allowed to accommodate the dynamic range needed for summation of all bit-cell pull-down currents. This introduces two sources of error: 1) the currents from all cells pulling down are reduced due to decreasing V_{DS} across their access and driver transistors and 2) cells not pulling down begin to pull up, albeit through a weakly biased nMOS. As shown in Fig. 9(a), this occurs because the access transistors experience reducing source voltage and begin to turn on. Consequently, as shown in the simulations of Fig. 9(b), a lower common-mode voltage causes the BL/BLB differential voltage to be compressed. However, we note that in absence of comparator offset, this does not change the sign and thus the result of classification. Furthermore, because comparator offset is expected to be uncorrelated between column-based weak classifiers, EACB can overcome such errors. Additionally, Section IV-C describes a method by which comparator offsets are compensated.

B. Bit Cells

The bit-cell array is identical to a standard 6T SRAM, including sizing. We note that in the Classify Mode, the cells face the potential for a new upset condition. However, this exhibits much greater margin than the standard SRAM Mode. Fig. 10(a) shows the read condition during the SRAM Mode. An upset could occur because the stored data are exposed to precharged BL/BLB, which can pull up the internal nodes. To ensure the stored data are retained, the butterfly curves shown in Fig. 10(b) must exhibit bistable lobes; the read static noise margin (SNM) measures the size of the largest embedded square in these lobes [18]. On the other hand, Fig. 11(a) shows a condition, similar to SRAM write, which could occur during the Classify Mode. Namely, BL/BLB can be pulled low, potentially causing an internal node at Logic 1 to be pulled low. However, because the WL is restricted to <0.4V [Fig. 8(a)], the internal nodes are minimally affected, and as seen in the butterfly curves of Fig. 11(b), large "Classify SNM" is maintained. Fig. 12 shows distributions from Monte Carlo simulations of the SRAM Mode read SNM and the Classify Mode SNM, showing that with standard cell sizing, the Classify SNM is much larger and has smaller standard deviation.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

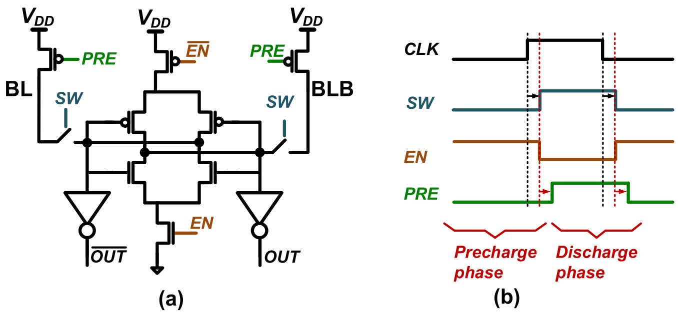


Fig. 13. Sense amplifier (a) circuit and (b) signal timing for charge equalization.

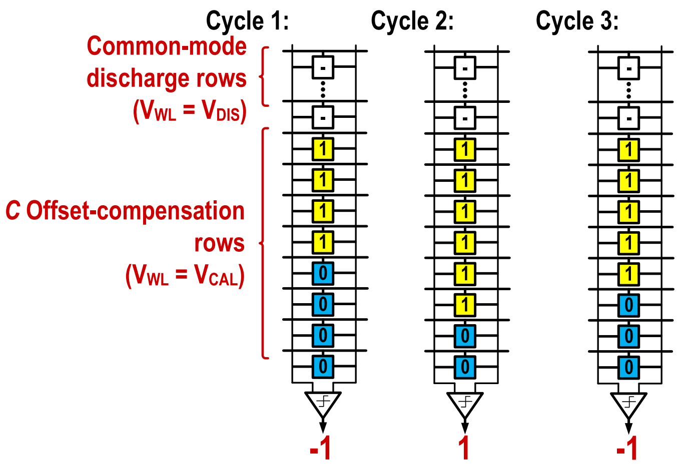
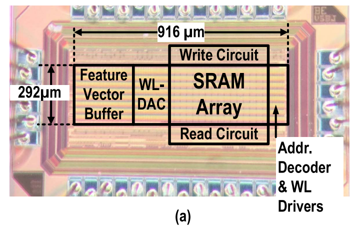


Fig. 14. Comparator offset-compensation approach via binary search, in parallel for all columns (eight offset compensation rows are shown, but tests in Section V employ 32 rows).

C. Sense Amplifiers

Fig. 13(a) shows the sense-amplifier comparator used for each column. Unlike in a standard SRAM, which employs low-swing BL/BLB, the sense amplifier must accommodate rail-to-rail inputs that can occur in the Classify Mode. Thus, a topology is used based on back-to-back inverters for regeneration, where the coupled nodes are driven by BL/BLB. As shown in Fig. 13(b), the timing is designed, such that the input switches are turned on just before the precharge phase ends. This way, precharging BL/BLB also equalizes charge on the regenerative nodes before the discharge phase begins.

Comparator offset, like many other circuit nonidealities, can be overcome via EACB. But, the SRAM structure readily affords self-compensation to substantially reduce comparator offset, thus reducing the EACB iterations required (as demonstrated in Section V-B). This is shown in Fig. 14. C rows are designated for offset compensation, with a WL bias V_{CAL} set to achieve a desired granularity of compensation (in the figure, $C = 8$ but tests in Section V-A employ $C = 32$). Since comparator offset can be dependent on the input common-mode level, other rows are designated for discharging BL/BLB to nominally equivalent values corresponding to the average discharge level expected during operation. Initially, half of the offset-compensation rows are set to store a Logic 1, and half are set to store a Logic 0. Then, BL/BLB discharge is performed and a comparison is made. For columns that return a -1 decision, half of the Logic 0 cells are written with Logic 1 data, and for columns return a $+1$ decision, half of the Logic 1 cells are written with Logic 0 data. Progressing in this way via binary search, the offset of all columns can be compensated in parallel with $\log_2 C$ cycles. In reality, equivalent discharge of BL/BLB to the expected common-mode level will be subject to bit-cell variations. To mitigate this, each binary-search decision can be made by averaging the comparison output over various configurations of discharge rows; in the tests of Section V-A, averaging over ten different discharge-row configurations is employed. The overhead of offset compensation is the number of rows required (adding area and bit-line capacitance), which sets the ratio of the offset-compensation range and granularity (i.e., small V_{CAL} gives finer granularity but reduces range, requiring more rows). However, because all rows are physically identical, the number of rows employed for offset compensation can be configured to optimize for different applications.



CHIP SUMMARY	
Technology	130 nm
SRAM Size	128 × 128 bits
Bit cell Size	1.26 µm × 3.44 µm
Energy / 10-way Class.	633.4 pJ
Speed (Classify)	50 MHz
Speed (SRAM)	300 MHz
Energy/cycle (Classify)	46.6 pJ
Energy/cycle (SRAM)	14.7 pJ

(b)

Fig. 15. Prototype (a) die photo and (b) measurement-summary table.

V. Prototype Measurements

A prototype is developed in 130-nm CMOS with a 128 × 128 6T bit-cell array supporting both SRAM and Classify Mode operation. For testing, write data, read data, and feature vectors are provided via buffers, implemented as scan chains. The die photo and measurement results are summarized in Fig. 15.

A. IC Characterization

SRAM Mode operates at 300 MHz while Classify Mode operates at 50 MHz, with an energy per clock cycle of 14.7 and 46.6 pJ, respectively. The bit cells employ standard sizing of transistors, but are laid out with logic rules, thus yielding a cell size somewhat larger than high-density 130-nm 6T cells. The periphery layout is optimized for testability, but the added periphery circuits (most notably WLDACs) occupy roughly the same area of the standard SRAM periphery (address decoder and WL drivers).

Comparator offset before and after compensation is measured by sweeping the differential BL/BLB voltage (around the expected common-mode level). This is done by sweeping the WLDAC codes of rows configured to discharge BL/BLB. As in the case of the common-mode level, the differential BL/BLB voltage attained this way is subject to bit-cell variations. Thus, averaging is performed over various configurations of rows. Fig. 16 shows the measured offset across 128 columns from

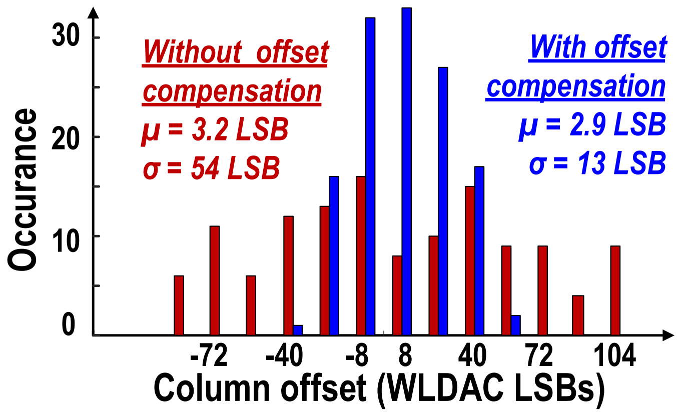


Fig. 16. Result of offset compensation, showing the standard deviation of comparator offset can be reduced.

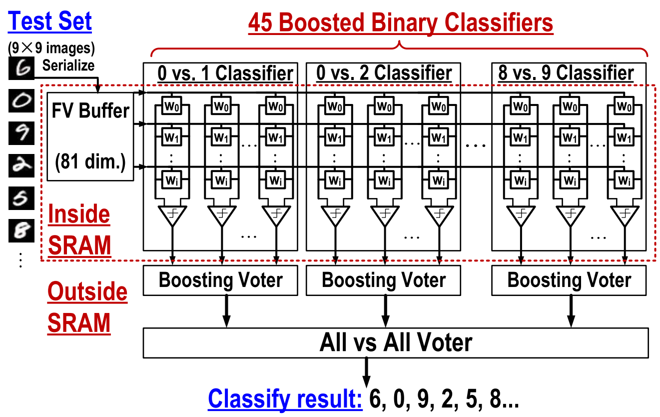


Fig. 17. Implementation of demonstration system for classifying 0–9 handwritten numerical data from MNIST data set.

one chip, where 32 offset-compensation rows are employed with a granularity V_{CAL} set by a WLDAC code of 5'b01000. The measured offset standard deviation is 54 LSB before compensation and 13 LSB after compensation. As shown next, this significantly reduces the number of EACB iterations required.

B. System Demonstration

For application demonstration, image recognition of handwritten numerical digits (0–9) from the MNIST data set [14] is performed, with features corresponding to raw pixel values. However, MNIST images have $28 \times 28 = 784$ pixels. Since the prototype implements only 128 rows, we low-pass filter and downsample the images to $9 \times 9 = 81$ pixels. MATLAB simulation of an ideal system based on boosted linear classifiers shows that the detection accuracy reduces from 96% to 90%. This now becomes the target for the system measurements, where 81 of the rows are used for feature inputs, 32 of the rows are used for offset compensation, and 15 rows are disabled (by setting the corresponding WLDAC inputs X_{offset} and $X[4:0]$ to 0).

For ten-way digit classification, 45 binary classifiers are implemented for all pairs of digits, and all-versus-all (AVA) voting is performed over these. As shown in Fig. 17, the binary

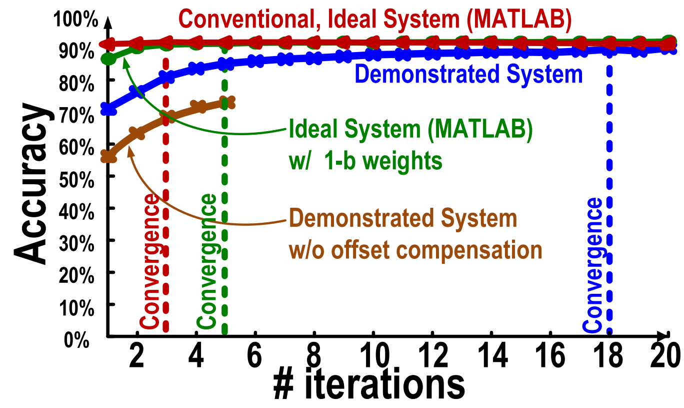


Fig. 18. Accuracy comparison of demonstrated system for MNIST-image classification. Increased EACB iterations required for restoring performance in prototyped system show that demonstration is not a trivial example inherently tolerant to circuit nonidealities.

classifiers (each including multiple iterations for EACB) are implemented using SRAM columns, while the adders required for boosting (7-b) and AVA voting (16-b) are outside the array.

Fig. 18 shows the measured accuracy versus the number of EACB iterations. The conventional, ideal system (implemented in MATLAB), corresponding to boosted linear classifiers with 10-b weights (determined to be the limit before performance degrades), achieves convergence in three iterations. The ideal system with 1-b weights (implemented in MATLAB), corresponding to boosted linear classifiers trained using the proposed approach in Section III-B, is somewhat weaker and achieves convergence in five iterations. For both ideal systems, EACB is only correcting weak-classifier fitting errors, not circuit nonidealities. The prototyped system achieves the performance of the ideal systems with 18 EACB iterations, required to overcome circuit nonidealities. Thus, we see that this application is not a trivial example in the sense of being inherently tolerant to circuit nonidealities that impact the in-memory architecture. Finally, we also show the performance, over the first few iterations, of the prototyped system without comparator-offset compensation; boosting is still achieved, but more EACB iterations would be required for performance convergence. We note that the 18 iterations required correspond to more than 128 columns. Thus, the input feature data are replayed in multiple runs to obtain data from all required weak classifiers, 128 at a time (energy reported below is for all runs). Testing to several iterations was performed on multiple chips, all demonstrating very similar performance (e.g., testing to 18 iterations done for two chips showed accuracy >90%).

Fig. 19(a) shows the energy analysis. On the left, it is the estimated energy of a conventional, discrete SRAM/digital-MAC system, using boosted linear classifiers with 10-b weights, requiring 71 nJ per ten-way classification. Next, it is the estimated energy of a discrete SRAM/digital-adder system, with $+/-1$ weights from the proposed approach; it requires more iterations, but total energy is reduced to 7.9 nJ thanks to fewer SRAM accesses and simpler computations following. Next, it is the measured energy of the prototype; it requires more iterations still, but total energy is further reduced

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

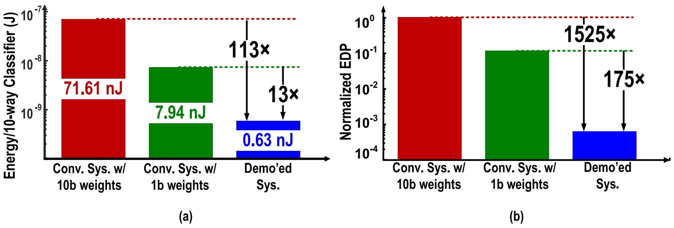
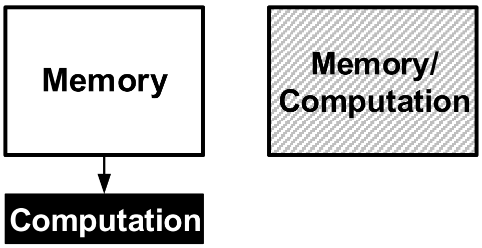


Fig. 19. Comparison of demonstrated system with discrete SRAM/digital-MAC systems showing (a) 113× and 13× energy reduction and (b) 1525× and 175× EDP reduction.



Metric*	Traditional Architecture	In-Memory Architecture
Bandwidth	$1/D^{1/2}$	1
Latency	D	1
Energy	$D^{3/2}$	D
Metric	1	$1/D^{1/2}$

*Over entire computation with full D data.

Fig. 20. Energy and delay scaling in traditional and in-memory architectures.

to just 0.63 nJ thanks to no explicit SRAM accesses and much fewer simple computations following (only additions for boosting and AVA voting). This corresponds to 113× and 13× energy savings, respectively. In all cases, multiplier and adder energies are estimated from postlayout simulations in the 130-nm technology (5-b × 10-b multiplication takes 4.7 pJ, 22-b addition takes 0.6 pJ, 16-b addition takes 0.4 pJ), and SRAM energy is measured from the prototype (SRAM Mode). Finally, Fig. 19(b) shows the energy-delay-product (EDP) analysis for the three cases (normalized to the first case). The frequency of the discrete SRAM/digital-MAC systems is assumed to be 300 MHz (i.e., SRAM access speed), and a ten times wider SRAM interface is assumed for the first system (though more typically the SRAM would operate slower and without such large interface). The prototype is estimated to achieve 1525× and 175× EDP reduction, respectively, enhanced by parallel operation over 81 bit-cells per column every clock cycle.

VI. Analysis of In-Memory Architecture

This section examines how the in-memory architecture alters the tradeoffs in traditional 2-D architectures, where separation of data storage and computation raises a communication cost, scaling with the amount (area) of data to be stored. Fig. 20 shows the two architectures (we point out that various other architectures, such as stacked 3-D, and computational models, such as neuromorphic computing, will raise alternate tradeoffs in terms of wire capacitance and bit-cell accessing patterns, impacting the tradeoffs analyzed). We assume that the amount of data required for a computation is D bits, arranged in a $D^{1/2} \times D^{1/2}$ array, as is typically preferred. We identify the following metrics of interest: bandwidth, latency, energy, and SNR (though roughly fixed in traditional SRAMs, SNR plays prominently in the in-memory architecture). These are analyzed *for the entire computation* (i.e., over the entire data) as D increases.[2] Fig. 20 summarizes the analysis detailed as follows.

1) *Bandwidth:* Traditional architectures are limited to providing data to computational hardware at each cycle through a single memory port. While the port size can increase as $D^{1/2}$, to access the full data, $D^{1/2}$ cycles are required (i.e., number of rows), causing the total bandwidth to decrease accordingly. On the other hand, combining storage and computational hardware in the in-memory architecture affords massive parallelism, where all data are inherently available to the computational hardware.
2) *Latency:* In traditional architectures, the BL/BLB capacitance increases as $D^{1/2}$ (causing proportional increase in discharge time), and the number of accesses increases as $D^{1/2}$, resulting in latency scaling as D. In the in-memory architecture, the BL/BLB capacitance increases similarly, but the number of bit cells pulling down in parallel increases proportionally, resulting in roughly constant absolute BL/BLB swing for given discharge time (BL/BLB differential voltage impacts SNR, as discussed in the following).
3) *Energy:* In traditional architectures, we assume the energy of BL/BLB discharge dominates. The increasing BL/BLB capacitance (as $D^{1/2}$), the increasing number of bit lines (as $D^{1/2}$), and the increasing number of accesses (as $D^{1/2}$) cause the energy to scale as $D^{3/2}$. In the in-memory architecture, BL/BLB discharge also dominates, and further involves large swing. However, the number of accesses is constant, causing the energy to scale only as D (which is much lower than $D^{3/2}$ for practical values of D). We point out that the in-memory architecture also drives all WLs at once, making the WL energy higher than in traditional architectures. However, the WL voltages are low ($<$0.4V, as seen in Fig. 8a), making this energy small, measured to be $<$20%.
4) *SNR:* Traditional SRAMs employ low-swing BL/BLB signaling set by sensing requirements, regardless of D. On the other hand, for the in-memory architecture, the SNR for each weak-classifier decision varies, with the worst case set by a condition where an equal number bit cells pull down BL/BLB and only one additional bit cell pulls down either BL or BLB, setting the differential voltage. While the absolute discharge on BL/BLB is roughly constant with D, the differential discharge decreases with the BL/BLB capacitance as $D^{1/2}$, thereby reducing the effective signal for the classification decision. Though error sources due to device noise

[2] We note that for each column-based weak classifier, the fundamental amount of data is $D^{1/2}$; this results in similar tradeoffs, but we focus on the ensemble classifier as a more relevant computation.

(e.g., thermal/shot noise) also reduce with capacitance (roughly as $D^{1/4}$), in practice, SNR is limited by other error sources (e.g., nonlinearities, offsets, and variations).

We see that the in-memory architecture alters the above tradeoffs, benefiting bandwidth, latency, and energy at the cost of SNR. This is preferred given the algorithmic approaches employed to address SNR (Section III). We note that taken to an extreme, this can eventually degrade efficiency (e.g., requiring many EACB iterations). But, an important and promising direction that emerges with ensemble classifiers is the ability to segment feature vectors, for instance by partitioning [19] or low-dimensional embedding [20], [21] of feature subspaces. This allows the data fundamentally associated with a particular weak classifier to be reduced, so D can be segmented into subarrays (avoiding excessive SNR degradation due to increasing BL/BLB capacitance). Thus, algorithmic approaches can enable optimization of an in-memory implementation across all metrics for a broad range of data size and applications.

VII. Conclusion

This paper presents a strong machine-learning classifier implemented in a standard 6T SRAM array. Throughput is enhanced because operation is performed over *all* bit cells in a column at once, and energy is enhanced because this corresponds to only a *single* BL/BLB discharge (as compared with standard read/write, where operation is performed over only one bit cell in a column at once). The primary impact is reduced SNR of the computation, limited by circuit nonidealities and weaker classifiers (i.e., lower precision weights). Training algorithms are introduced (for boosting and base-classifier training) to address this. A measured prototype demonstrating MNIST image classification shows that the algorithms are successful in restoring performance to the level of an ideal, discrete SRAM/digital-MAC system while lowering energy by 113× when a standard training algorithm is employed and by 13× when the proposed training algorithm is employed in the ideal, discrete systems.

References

[1] Y.-H. Chen, T. Krishna, J. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Sep. 2016, pp. 262–263.

[2] Y. Kim, D. Shin, J. Lee, Y. Lee, and H.-J. Yoo, "A 0.55V 1.1mW artificial-intelligence processor with PVT compensation for micro robots," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Sep. 2016, pp. 258–259.

[3] S. Park, S. Choi, J. Lee, M. Kim, J. Park, and H.-J. Yoo, "A 126.1mW real-time natural UI/UX processor with embedded deep-learning core for low-power smart glasses," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Oct. 2016, pp. 254–255.

[4] S. Hanson and D. Sylvester, "A 0.45–0.7V sub-microwatt CMOS image sensor for ultra-low power applications," in *Proc. Symp. VLSI Circuits*. 2009, pp. 176–177.

[5] M. Horowitz, "Computing's energy problem (and what we can do about it)," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Oct. 2014, pp. 10–14.

[6] R. Genov and G. Cauwenberghs, "Kerneltron: support vector 'machine' in silicon," *IEEE Trans. Neural Netw.*, vol. 14, no. 5, pp. 1426–1434, Sep. 2003.

[7] S. Jeloka, N. B. Akesh, D. Sylvester, and D. Blaauw, "A 28 nm configurable memory (TCAM/BCAM/SRAM) using push-rule 6T bit cell enabling logic-in-memory," *IEEE J. Solid-State Circuits*, vol. 51, no. 4, pp. 1009–1021, Apr. 2016.

[8] M. Kang, M. S. Keel, N. R. Shanbhag, S. Eilert, and K. Curewitz, "An energy-efficient vlsi architecture for pattern recognition via deep embedding of computation in SRAM," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2014, pp. 8326–8330.

[9] M. Kang, S. Gonugondla, A. Patil, and N. Shanbhag. (2016). "A 481pJ/decision 3.4M decision/s multifunctional deep in-memory inference processor using standard 6T SRAM array." [Online]. Available: https://arxiv.org/abs/1610.07501.

[10] J. Zhang, Z. Wang, and N. Verma, "A machine-learning classifier implemented in a standard 6T SRAM array," in *IEEE Symposium on VLSI Circuits (VLSI-Circuits)*, Jun. 2016, pp. C252–C253.

[11] R. E. Schapire and Y. Freund, *Boosting: Foundations and Algorithms*. Cambridge, MA, USA: MIT Press, 2012.

[12] Z. Wang, R. E. Schapire, and N. Verma, "Error adaptive classifier boosting (EACB): Leveraging data-driven training towards hardware resilience for signal inference," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 62, no. 4, pp. 1136–1145, Apr. 2015.

[13] Gurobi Optimization, Inc. (2015). *Gurobi Optimizer Reference Manual*. [Online]. Available: http://www.gurobi.com

[14] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[15] K. Osada *et al.*, "Universal-Vdd 0.65-2.0V 32 kB cache using voltage-adapted timing-generation scheme and a lithographical-symmetric cell," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Sep. 2001, pp. 168–169.

[16] K. Sohn *et al.*, "An autonomous SRAM with on-chip sensors in an 80nm double stacked cell technology," in *Proc. Symp. VLSI Circuits*, Jun. 2005, pp. 232–235.

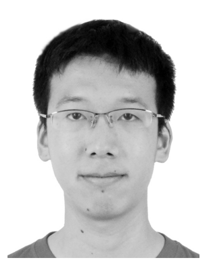
[17] Z. Wang and N. Verma, "Enabling hardware relaxations through statistical learning," in *Proc. Allerton Conf. Commun., Control Comput.*, Oct. 2014, pp. 319–326.

[18] E. Seevinck, F. J. List, and J. Lohstroh, "Static-noise margin analysis of MOS SRAM cells," *IEEE J. Solid-State Circuits*, vol. 22, no. 5, pp. 748–754, Oct. 1987.

[19] Y. Piao *et al.*, "A new ensemble method with feature space partitioning for high-dimensional data classification," *Math. Problems Eng.*, vol. 2015, Jan. 2015, Art. no. 590678.

[20] N. C. Oza, "Boosting with averaged weight vectors," in *Proc. Int. Workshop Multiple Classifier Syst., (LNCS)*, vol. 2709. 2003, pp. 15–24.

[21] K. M. Ting, J. R. Wells, S. C. Tan, S. W. Teng, and G. I. Webb, "Feature-subspace aggregating: Ensembles for stable and unstable learners," *Mach. Learn.*, vol. 82, no. 3, pp. 375–397, 2011.



Jintao Zhang (S'12) received the B.S. degree in electrical engineering from Purdue University, West Lafayette, IN, USA, in 2012, and the M.A. degree in electrical engineering from Princeton University, Princeton, NJ, USA, in 2014, where he is currently pursuing the Ph.D. degree in the Department of Electrical Engineering.

His current research interests include realization of machine learning algorithms in mixed-signal/digital hardware, low-energy mixed-signal ASIC design for machine learning applications, and algorithms designed for such embedded system.

Zhuo Wang (S'12–M'16) received the B.S. degree in microelectronics from Peking University, Beijing, China, in 2011, the M.S. degree from the Department of Electrical Engineering, Princeton University, Princeton, NJ, USA, in 2013, where he is currently pursuing the Ph.D. degree.

His current research interests include leveraging statistical approaches, such as machine learning, for achieving hardware relaxation in an algorithmic and architectural level in resource-constrained platforms, such as embedded sensing systems.

Mr. Wang was a recipient of the 2011 Peking University Best Undergraduate Dissertation Award, the 2014 ICASSP conference Best Paper Award nomination, and the 2015 Princeton University Honorific Fellowship.

Naveen Verma (M'05) received the B.A.Sc. degree in electrical and computer engineering from the University of British Columbia, Vancouver, BC, Canada, in 2003, and the M.S. and Ph.D. degrees in electrical engineering from the Massachusetts Institute of Technology, Cambridge, MA, USA, in 2005 and 2009, respectively.

Since 2009, he has been with the Department of Electrical Engineering, Princeton University, Princeton, NJ, USA, where he is currently an Associate Professor. His current research interests include advanced sensing systems, including low-voltage digital logic and SRAMs, low-noise analog instrumentation and data-conversion, large-area sensing systems based on flexible electronics, and low-energy algorithms for embedded inference, especially for medical applications.

Dr. Verma was a recipient or co-recipient of the 2006 DAC/ISSCC Student Design Contest Award, the 2008 ISSCC Jack Kilby Paper Award, the 2012 Alfred Rheinstein Junior Faculty Award, the 2013 NSF CAREER Award, the 2013 Intel Early Career Award, the 2013 Walter C. Johnson Prize for Teaching Excellence, the 2013 VLSI Symposium Best Student Paper Award, the 2014 AFOSR Young Investigator Award, the 2015 Princeton Engineering Council Excellence in Teaching Award, and the 2015 IEEE TRANSACTIONS ON COMPONENTS PACKAGING, AND MANUFACTURING TECHNOLOGY Best Paper Award. He is a Distinguished Lecturer of the IEEE Solid-State Circuits Society and serves on the technical program committees of ISSCC, VLSI Symposium, DATE, and the IEEE Signal-Processing Society.