Analyzing the Impact of Memristor Variability on Crossbar Implementation of Regression Algorithms With Smart Weight Update Pulsing Techniques

Sahra Afshari, Mirembe Musisi-Nkambwe, *Senior Member, IEEE*, and Ivan Sanchez Esqueda, *Senior Member, IEEE*

***Abstract*— This paper presents an extensive study of linear and logistic regression algorithms implemented with 1T1R memristor crossbars arrays. Using a sophisticated simulation platform that wraps circuit-level simulations of 1T1R crossbars and physics-based models of RRAM (memristors), we elucidate the impact of device variability on algorithm accuracy, convergence rate and precision. Moreover, a *smart pulsing strategy* is proposed for practical implementation of synaptic weight updates that can accelerate training in real crossbar architectures. Stochastic multi-variable linear regression shows robustness to memristor variability in terms of prediction accuracy but reveals impact on convergence rate and precision. Similarly, the stochastic logistic regression crossbar implementation reveals immunity to memristor variability as determined by negligible effects on image classification accuracy but indicates an impact on training performance manifested as reduced convergence rate and degraded precision.**

***Index Terms*— RRAM, crossbar array, variability, machine learning, stochastic regression.**

I. Introduction

RESISTIVE random access memory (ReRAM or RRAM) technology is a great candidate for non-volatile memory (NVM) due to low power consumption, excellent scalability, high speed functionality, CMOS compatibility, and analog programmability (i.e., the ability to retain analog values) compared to conventional digital memory circuits [1], [2]. RRAM cells (also referred to as memristors) are generally two-terminal devices that consists of an insulating or switching layer (e.g., an oxide) inserted between two metal layers [3], [4]. RRAM operation for NVM applications typically involves programming (and reading) cells into two distinct (binary) states, a low resistance state (LRS) or high resistance state (HRS). Multistate storage has also been demonstrated using RRAM for NVM [5]. Additionally, RRAM analog-based implementations of in-memory computing and neuromorphic architectures rely on the ability to program a continuous range of states [6]. The programming of different resistive states is achieved via the formation and rupture of conductive filaments inside the oxide/switching layer of the cell. RRAM is considered a great candidate for training and inference applications [7], but the stochastic essence of conductive filament activity [8]–[10], introduces variability, programming abruptness, and non-linearity that may present significant challenge for the implementation of RRAM-based in-memory computing applications and machine-learning (ML).

In [19], reliability concerns for RRAM were identified and metrics were discussed based on the impact on distinguishability of states and computing accuracy. As presented in [19], the basic reliability metrics relate to endurance, retention, noise, and write/read disturbs. Other "functional" reliability metrics include non-linearity, variability, dynamic range, precision, variation, asymmetry, etc. These reliability metrics refer to functional properties of RRAM that can have a severe impact on computing accuracy when degraded. The paper summarizes results from [20]–[22] where degradation in dynamic range [20], non-linearity [21], and asymmetry [22] in the pulsed programming of RRAM can affect image classification accuracy on crossbar implementations. Another example [23], discussed how time-dependent drift and fluctuation in programmed conductance states impacts the reliability of neural network inference as determined by accuracy loss in classification on MNIST handwritten datasets. In this paper, we focus on another functional reliability concern, i.e., variability in RRAM characteristics, and its impact on neural network training (convergence rate, accuracy, precision) based on gradient descent algorithms.

Conductive bridging random access memory (CBRAM), another filamentary-based resistive switching memory, is also suitable for neuromorphic computing due to low power dissipation and low transmission consumption [24], [25]. Variation between different devices (device-to-device) and within individual devices (cycle-to-cycle) has been characterized and presented in various previous works [11]–[18]. Chen and Lin presented a collection of results on the variability of LRS and HRS in different RRAM and CBRAM technologies [26]. A similar collection of LRS and HRS variability from recent RRAM and CBRAM published results is shown in Figure 1 [11]–[18]. These indicate that large variation is prevalent for newer generation of RRAM devices as expected due to the stochastic nature of the resistive

Manuscript received August 12, 2021; revised November 16, 2021 and December 23, 2021; accepted January 7, 2022. This work was supported by the National Science Foundation under Grant CCF-2001107. This article was recommended by Associate Editor A. James. *(Corresponding author: Ivan Sanchez Esqueda.)*

The authors are with the Department of Electrical, Computer, and Energy Engineering, Arizona State University, Tempe, AZ 85287 USA (e-mail: isesqueda@asu.edu).

This article has supplementary material provided by the authors and color versions of one or more figures are available at https://doi.org/10.1109/TCSI.2022.3144240.

Digital Object Identifier 10.1109/TCSI.2022.3144240

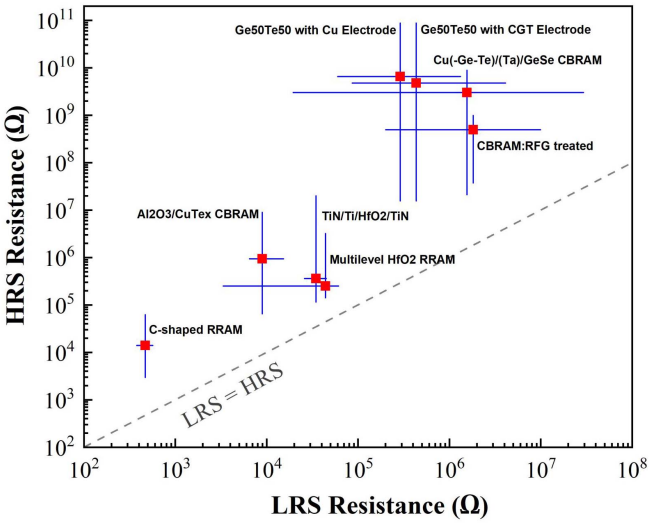

Fig. 1. Variability of resistive-switching characteristics in recent metal-oxide RRAM and CBRAM technologies discussed in previous research. Intersection for each device represents the mean value of HRS and LRS [11]–[18].

switching mechanisms. Thus, it is crucial to study the impact of device variability on in-memory computing circuits to gain insight on the viability of RRAM implementations. This paper analyzes the effects of RRAM device variability on accuracy and precision of gradient descent-based ML algorithms (linear and logistic regression) using crossbar architectures.

The algorithm-level analysis presented in this paper uses Spice (Synopsys HSpice) circuit-level simulations that incorporate a compact memristor model previously developed and verified with experimental data [27]. The primary goal of this paper is to investigate the impact of device variability on the performance of gradient-descent-based machine learning algorithms. Therefore, device-to-device and cycle-to-cycle variations are introduced into key model parameters. The approach involves randomly sampling the model parameters from an experimentally verified distribution. Additional details of the compact model and simulation approach are presented in Section II.

For the ML algorithm analysis, a modified gradient-descent approach is used to train the crossbar array, similar to that presented in previous work by Nair and Dudek [28]. In that previous work, a single programming voltage pulse of fixed amplitude and width is used to adjust the memristor conductance (i.e., the synaptic weights) independent of the magnitude of the required update. The polarity of the pulse (positive vs. negative amplitude) is selected based on the sign of the update as determined by the algorithm. In this work, we extend the approach by allowing a discrete number of programming pulses to update memristors in accordance with the necessary update. Based on this new approach, we study the convergence rate and performance of gradient-descent ML algorithms in the presence of large variation in memristor devices. Section III of this paper explores the improvements of the updated gradient descent approach on ML algorithm convergence rate and performance.

The results of our ML algorithm analysis provide insight on convergence rate, accuracy, and precision of pattern classification experiments on RRAM crossbars [29]–[33] and the effects of memristor variability. The paper is structured as follows: Section II identifies and analyzes the effects of variability on the resistive-switching characteristic of 1-transistor-1-resistor (1T1R) RRAM cells and describes the simulation approach for crossbar arrays. Section III presents the implementation of linear and logistic regression on memristor crossbars and establishes the impact of device variability on algorithm performance. Finally, Section IV provides conclusions and summarizes the main contributions of this work. Mainly, despite large RRAM cell variability, the crossbar implementation of regression algorithms achieves convergence (as indicated by clear improvements in accuracy with training), but with noticeable degradation on precision (fluctuation in the accuracy of trained arrays).

II. Modeling Approach

A. Modeling RRAM 1T1R Crossbars

Various works have presented memristor models for simulations of memory and neuromorphic computing applications [34]–[39]. In this work we use a compact model for HfOx-based RRAM devices [27]. The bipolar switching characteristics achieved in the model are based on fundamental physics related to filamentary operation and have been experimentally verified with HfOx devices [40]–[42]. A key parameter in the model that captures the internal state of the RRAM cell is the gap (g), specified as the distance between the top electrode and conductive filament as illustrated in Figure 2(a). The memristor conductance is directly related to this parameter. The dynamic process of resistive switching and current flow are modeled by the two general memristor equations shown in Figure 2(a). To model RRAM variation, the model fitting parameters I_0, υ_0 and γ_0 (related to filamentary formation/dissolution and conductance) are allowed a dispersion $3\sigma/\mu$ of 30%, 10% and 10%, respectively. These values were extracted to fit experimental HRS and LRS distributions in TiN/Hf/HfOx/TiN-based RRAM devices (cf. Figure 5 in [27]). As described in [27], dispersion in all three parameters should be included to account for the actual (experimental) variability in RRAM characteristics and measured distribution in LRS an HRS. The RRAM model is implemented in Verilog-A and circuit-level simulations are conducted using Synopsys HSpice. For simulating 1T1R cells we use a 65 nm n-type CMOS transistor model based on the Predictive Technology Model (PTM) from Arizona State University [43]. Figure 2(b) shows the schematic of the 1T1R cell, indicating the pulsing approach to increase or decrease the conductance of the memristor (i.e., set/reset the memristor). The n-MOS transistor acts as a selector device and the gate voltage is used to modulate or limit the amount of current that flows through the cell. The 1T1R configuration helps eliminate sneak path currents and improves analog programmability by reducing abrupt changes in conductance from set/reset pulses [27], [44]. Finally, Figure 2(c) is a schematic of the RRAM 1T1R crossbar arrays simulated in this work.

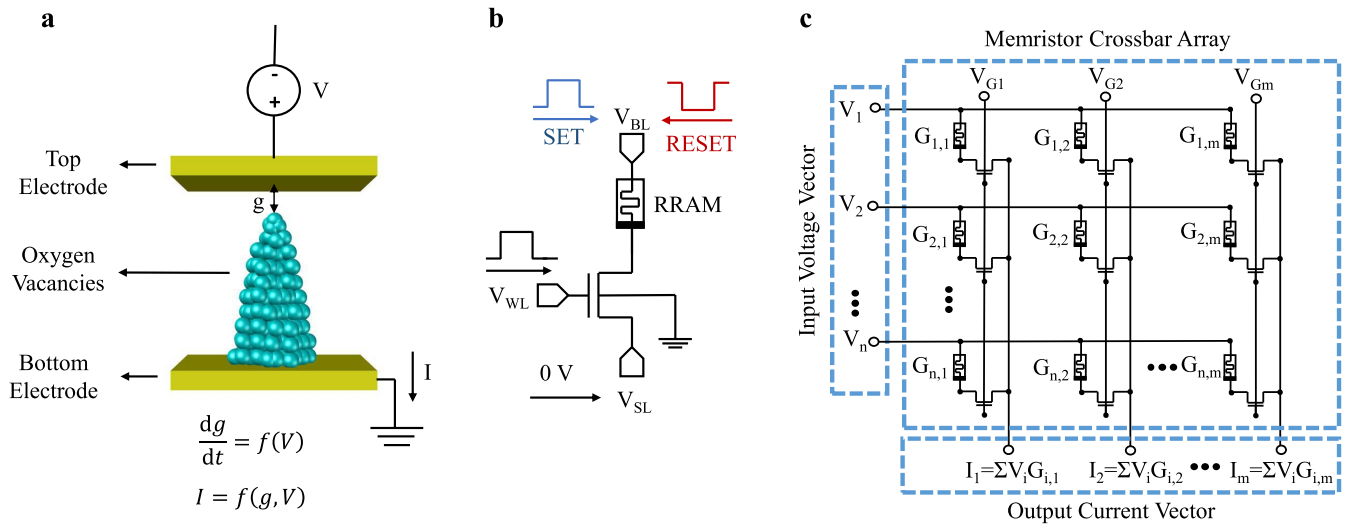

Fig. 2. (a) Filamentary operation and top-level mathematical representation of the physics-based RRAM model used in this work. (b), (c) Schematic of the 1T1R RRAM cell and crossbar array simulated with Synopsys HSpice.

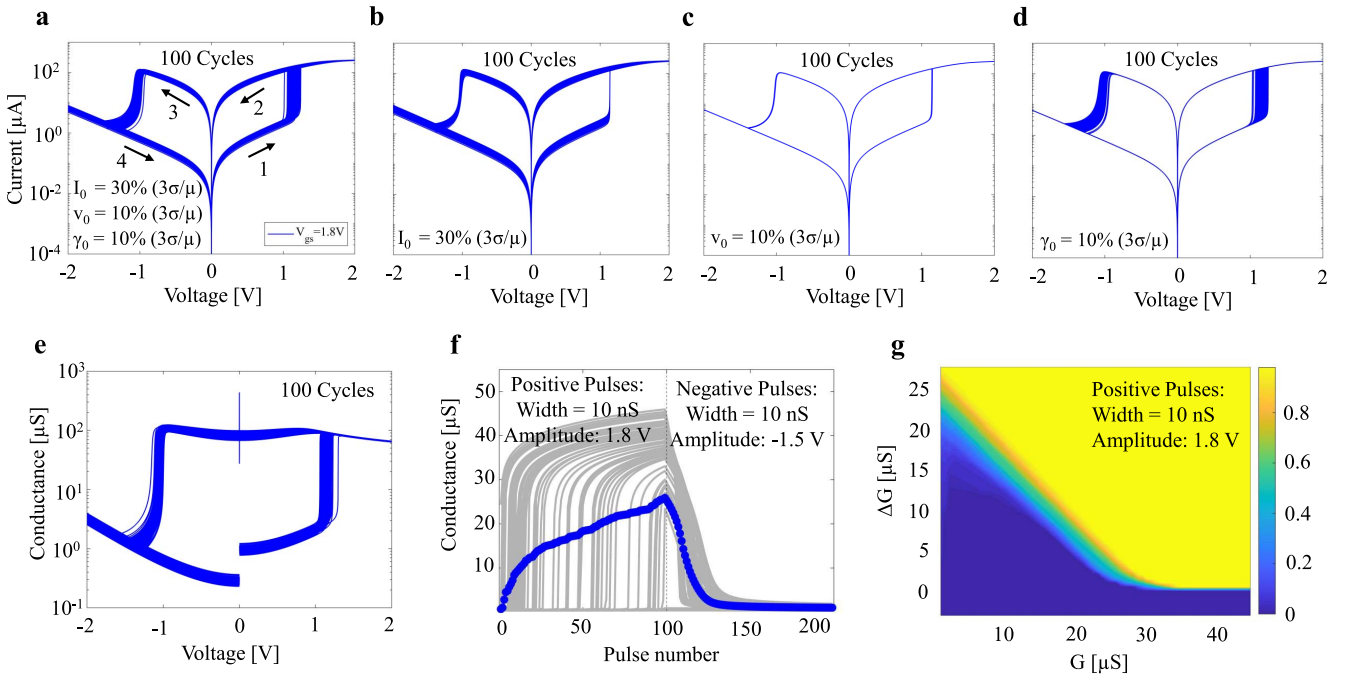

Fig. 3. (a) Simulation of DC resistive-switching current-voltage (I-V) characteristics of 1T1R RRAM cell considering joint effects of dispersion in model parameters, (b-d) Simulation of DC resistive-switching I-V characteristics of 1T1R RRAM cell considering individual effects of dispersion in model, (e) Plot of conductance vs. voltage (G-V) characteristics. (f) Pulse-programming of memristor conductance (multiple cycles and average), average is solid blue line with circles. (g) Contour plot of the cumulative distribution function (CDF) for change in conductance (ΔG) vs. conductance (G). CDF plot illustrates the non-linear and abrupt response of pulse ΔG programming.

B. Simulating the Effects of Variability on 1T1R Cells

The impact of RRAM variability on the 1T1R cell resistive switching properties is summarized in Figure 3. Figures 3(b-d) show the effects of model parameter dispersion individually (I_0, v_0 and γ_0), and Figure 3(a) shows the combined effects on the resistive switching current-voltage (I-V) characteristics. Figure 3(e) plots the conductance-voltage (G-V) characteristics including dispersion in all three model parameters. Figure 3(f) reveals the impact of variability on the pulsed characteristics (change in conductance with consecutive pulses). In Figure 3(f), 100 cycles are shown, each cycle consisting of 100 positive and 100 negative consecutive pulses. A different visualization for the impact of variability on the resistive-switching properties is provided in Figure 3(g). This plot shows contours for the cumulative distribution function (CDF) of change in conductance (ΔG) as a function of conductance (G). It provides a graphical representation of the non-linear and abrupt response of ΔG resulting from the programming pulses (only shown for positive pulses) [45], [46]. At low levels of G (starting with a weak filament),

the CDF shows that most pulses will result in large changes in conductance (abrupt). As G increases, the distribution shifts to smaller changes in conductance (less abrupt) and distribution is narrower (less variation for ΔG).

C. Simulation Approach for Regression Algorithms

The implementation and analysis of regression algorithms presented in this work uses MATLAB scripts that organize and execute Synopsys HSpice circuit-level simulations of RRAM 1T1R crossbars. In this simulation platform, the initialization of RRAM devices as well as the functions of the peripheral circuits (e.g., normalization of inputs and outputs, calculations of prediction/classification error, activation functions, etc.) are conducted in MATLAB software. However, crossbar functions including vector matrix multiplications (VMM) and pulsed programming of RRAM 1T1R cells are directly implemented with HSpice circuit simulations using the compact models described in section II-A. A detailed description of the simulation approach is provided in the supplementary material.

D. Smart Pulsing Strategy for Weight Updates

This paper presents a new weight update strategy for accelerated training in ML algorithms. The proposed strategy selects the number of programming pulses for each memristor at each training step not only based on the sign of the required update, but also on its magnitude. For practicality, the number of pulses is discretized to three different ranges of required weight update (see Figure 4(c)). For example, a large conductance update requirement leads to more consecutive pulses compared to a smaller update requirement. This leads to larger weight (conductance) changes during early training steps, and smaller changes in later steps to help fine tune and maximize accuracy as the training advances. We note that this technique does not affect the frequency of updates, as an update is still done at every training step. In section III, we demonstrate how this strategy results in higher convergence rate, as well as improved precision and accuracy for the crossbar implementation of multi-variable linear and logistic regression algorithms compared to existing techniques based on fixed update pulsing methods.

E. Discussion About Peripheral Circuits

Pulse updates can be generated by a simple CML (current-mode logic) driver circuit where the circuit is tuned to ensure enough drive voltage capability for loads presented in terms of crossbar size (crossbar interconnect resistance and memristors). The write voltages should be verified to have enough margin above the memristor write threshold to effectively drive the furthest memristor in the write path. More specifically, for our proposed smart pulse update strategy, a configurable ring oscillator can be used to ensure a specified number of similar spaced pulses as discussed in [47]. Read currents are accumulated at the end of crossbar and need to be sensed prior to digital conversion and further processing. The choices of voltage versus current mode sensing circuits are described in [48]. In this solution, a current mode sensing

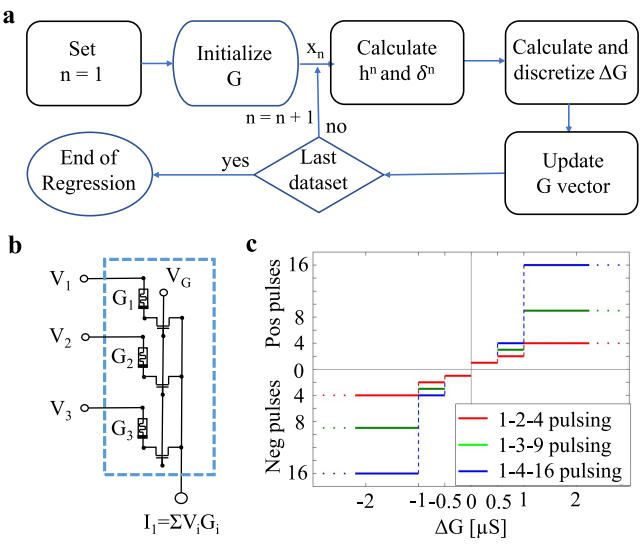

Fig. 4. (a) Flow chart of implementation of stochastic multi-variable linear regression algorithm on a memristor crossbar. (b) Schematic of the 3 × 1 1T1R crossbar array implemented in Synopsys HSpice for simulation of linear regression. (c) Translation of ΔG into number of positive or negative voltage pulses for realistic hardware implementation of the gradient-descent.

mechanism is preferred where a reference current is generated to compare against the accumulated output current. This choice, while more area intensive, allows for trackability of device variation mirrored in the reference crossbar array. A detailed scheme of the current-mode sensing circuit is described in [49].

III. Linear and Logistic Regression

A. Stochastic Multi-Variable Linear Regression

This section presents the implementation of stochastic multivariable linear regression on a 3 × 1 1T1R RRAM crossbar array. This is a type of regression algorithm with multiple independent variables (x_0, x_1, …x_n) combined into a linear prediction function of the dependent variable (y). The term stochastic comes from the stochastic gradient descent optimization approach where a single sample or subset of the data is randomly selected to update the model parameters during each training step. In practice, we present one data sample at a time to our crossbar array. The model prediction (h) is given by the dot product of the input variables (x_0, x_1, …x_n) and the model parameters which are stored as the memristor conductances (G_0, G_1, …G_n). Mathematically, the prediction h is given by:

$$h = x^T G, \quad x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix}, \quad G = \begin{bmatrix} G_0 \\ G_1 \\ G_2 \end{bmatrix} \tag{1}$$

Here, x is the normalized 3 × 1 input vector and G is the 3 × 1 vector of the memristor conductances.

Figure 4(a) is a flowchart for the crossbar implementation of stochastic multivariable linear regression. Figure 4(b) shows the schematic of the 3 × 1 1T1R RRAM crossbar array as implemented in the simulation. The *smart pulsing strategy* used in this demonstration is illustrated in Figure 4(c). This

discretized approach would be a practical implementation of gradient-descent on a real memristor crossbar. When training, the initial steps will typically require larger updates in conductance (ΔG) because the error (δ) is initially large, prompting a larger number of pulses. As the training advances and the error is reduced, the required update is also reduced leading to a smaller number of applied pulses. In Figure 4(c), three different versions of the pulsing strategy are shown. These correspond to different sets of programming pulses (positive or negative) used to update conductance based on the value of ΔG_{i}. For example, in the pulsing strategy labeled 1-2-4, one, two or four programming pulses are applied depending on whether the required update in conductance is between 0 and 0.5 μS, 0.5 and 1 μS, or above 1 μS.

Our demonstration of multivariable linear regression is based on an artificial data set for the price of a pizza as a function of two independent variables, x_1 and x_2, where x_1 represents the number of ingredients, and x_2 represents the size of the pizza. Note that the same approach can be easily extended to N independent variables on an (N + 1) crossbar array. In the hardware implementation, the input variables are presented as voltage signals ($x_{\mathrm{i}} \rightarrow v_{\mathrm{i}}$) on each row of the crossbar (see Figure 4(b)), and the prediction is represented by the current flowing on the crossbar array as given by Kirchhoff's law: $\mathrm{h}\rightarrow \mathrm{I} = \sum v_{\mathrm{i}} G_{\mathrm{i}}$. To ensure the accuracy of the prediction in this hardware implementation, the amplitude of the input voltage signals is normalized to a range between 0 and 0.25 V. This range results in good linearity (i.e., current is directly proportional to voltage, or equivalently conductance is independent of voltage) as shown in Figure 3(e). In the optimization process that occurs during training, a cost function J proportional to the mean square error is minimized through the update of the conductance values. The error is determined by the difference in the predicted and actual values as $\delta^{\mathrm{n}} = h^{\mathrm{n}} - y^{\mathrm{n}}$, where the superscript indicates the n^{th} data sample (also n^{th} training step). At each training step, each device requires a conductance update given by $\Delta G_{\mathrm{i}} = -\alpha \delta^{\mathrm{n}} v^{\mathrm{n}}{}_{\mathrm{i}}$. Here, α is a learning rate. In practice, it is not feasible (or required) to perfectly update the conductances by exactly ΔG_{i}. The goal is to minimize the error (or cost function). Therefore, the approach is to use a discrete number of programming pulses (positive or negative) to approximate the change in conductance state of each memristor according to the value of ΔG_{i}. This approach is referred to as the *smart pulsing strategy*.

B. Demonstration of Stochastic Linear Regression

In our demonstration, a dataset of size 1000 is artificially generated to be used as training of the crossbar array network. The conductance values are randomly initialized within a range from 10 to 60 μS. The learning rate α is initially set to 1, and for improved convergence is reduced by 3% after each training step. Each iteration corresponds to presenting a single sample from the dataset followed by the adjustment of the conductance for each memristor based on the calculated ΔG_{i}. Figure 5(a) summarizes the results of the memristor crossbar implementation of the stochastic multivariable linear regression algorithm (*without variation*). In Figure 5(a), the blue dots are the dataset corresponding to price of pizza plotted as a function of two independent variables, x_1, number of ingredients, and x_2, size of the pizza. The algorithm is conducted five different times and for each case the initial and final conductance states are recorded. The red mesh surfaces represent the model prediction based on the initial (random) state of memristor conductances in the crossbar. The green mesh surfaces represent the prediction after 1000 training steps (i.e., after all data samples have been presented to the array). The different final predictions for each case results from the different random initial states along with random shuffling in the sampling process. The results show a significant improvement in the model prediction of the data set after training as indicated by the green mesh surfaces overlapping the data points. Figure 5(b) plots the evolution of conductance for each memristor in the array as a function of the algorithm iteration step during training. It indicates larger updates in conductance during the initial steps and a settling as convergence is achieved.

Figure 5(c) compares the convergence as indicated by the prediction mean squared error (MSE) as a function of iteration number for the three different versions of the pulsing strategy. It is clear from the slope of MSE vs. iteration number that the pulsing strategy with larger number of pulses (i.e., 1-4-16) has a faster initial convergence rate (can reach lower MSE with fewer iterations during the initial training steps). However, as training advances, the convergence rate slows down and eventually all three pulsing strategies achieve small MSE. We note that a fast initial convergence rate may be desirable for specific training applications. The proposed pulsing approach can achieve a fast initial convergence rate without compromising high prediction accuracy of the fully trained crossbar array. Finally, we examine the impact of variability on the stochastic multivariable linear regression algorithm memristor crossbar implementation. Figure 5(d) plots the prediction mean squared error (MSE) as a function of iteration number for a pulsing strategy of 1-4-16, with and without memristor variation. For the case of no variation (shown in green), we include the average MSE vs. iteration from 10 simulations (solid line) and the range between maximum and minimum MSE (shaded green region). For the case with variation, we only show the average MSE vs. iteration number (solid red line). While the convergence is still good even with memristor variability, we note the following effects: 1) The results indicate that the convergence is slower (error is reduced at a slower rate with training), 2) The accuracy is degraded (average error after training is slightly larger than what was obtained when neglecting variation), 3) The most significant issue appears to be an impact on precision. The results in Figure 5(d) show noticeable fluctuation in average error when variation is included. We interpret these fluctuations as an impact on the algorithm precision resulting from variability in the programming of memristor conductance states. It should be noted that even with these detrimental effects of memristor variability, the prediction error is still converging (i.e., error reduces with training) to about 3-5%. This is a promising result for memristor crossbars implementations

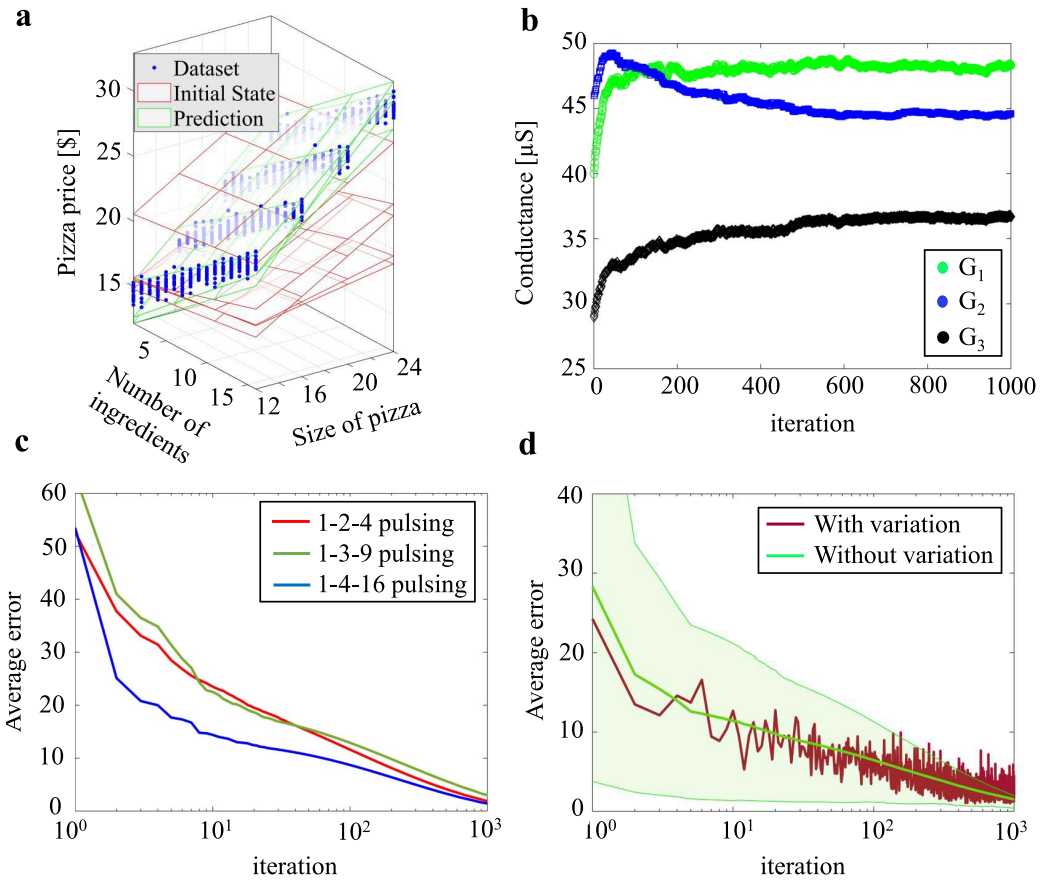

Fig. 5. (a) Results of stochastic multi-variable linear regression algorithm on a memristor crossbar (5 simulations). The blue dots are the dataset, the red mesh surface is the prediction based on the initial memristor conductance, and the green mesh surface is the final prediction after 1000 iterations. (b) Conductance evolution for each memristor from the 3 × 1 crossbar as a function of iteration (training step). (c) Comparison of convergence rate (error vs. iteration) for three different cases of the smart pulsing strategy. (d) Convergence rate (with and without variation) for the 1-4-16 pulsing strategy. Solid green line is average error from 10 simulations, shaded region indicates the range of maximum and minimum values from all 10 individual cases.

of regression algorithms that appears to indicate some level of immunity to device variability at the algorithm-level. The same simulation was repeated to compare the individual effects of dispersion in model parameters I_0, and γ_0 on the algorithm performance (not shown). We discover that the observed impact on precision is due mainly to dispersion in I_0, correlating with variation in conductance, and not to dispersion in γ_0 which mostly correlates to dispersion in set/reset voltages (see Figure 3(a-d)). This observation is reasonable as the algorithm implementation is based on pulsed programming where the amplitudes of the applied voltage pulses (+1.8 V/−1.5 V) have sufficient margins above/below the set/reset thresholds.

C. Stochastic Logistic Regression

This section describes the implementation of stochastic logistic regression in a memristor crossbar for classification of 5 × 5-pixel binary images that represent characters 'S', 'M', 'R', and 'T'. Figure 6(a) is a flowchart describing the logistic regression implementation. The data set is artificially generated and includes "noisy" samples or images where two of the binary pixels have been flipped (see Figure 6(b)). Separate data were generated for training and to test the classification accuracy at fixed training intervals (i.e., after a fixed amount of training images have been presented to the network). Figure 6(c) is a graphical representation of the neural network that is being implemented by the memristor crossbar for this classification task. The crossbar schematic is shown in Figure 6(d). Here, each synaptic connection is implemented by a memristor differential pair. The effective conductance for each differential pair is given by: $G_{ij} = G_{ij}^{+} - G_{ij}^{-}$. This enables negative weights to be implemented with the crossbar array (all conductances are positive). To perform the classification of the 5 × 5 images, a 25 × 8 memristor crossbar is simulated. During training, images that correspond to different characters (S, M, R, or T) are randomly presented to the array, so all 4 neurons are simultaneously trained to recognize their assigned character.

As discussed in the previous section, the linear range for the I-V characteristics of the memristors falls between the range of −0.25 to 0.25 V. Thus, during the "read" operation, each pixel from the binary image is mapped to a crossbar input voltage signal of 0.1 V for white pixels and −0.1 V for black pixels. The output current on each neuron is essentially a dot product of the input voltage vector and the effective conductance vector from the corresponding column pair. Mathematically,

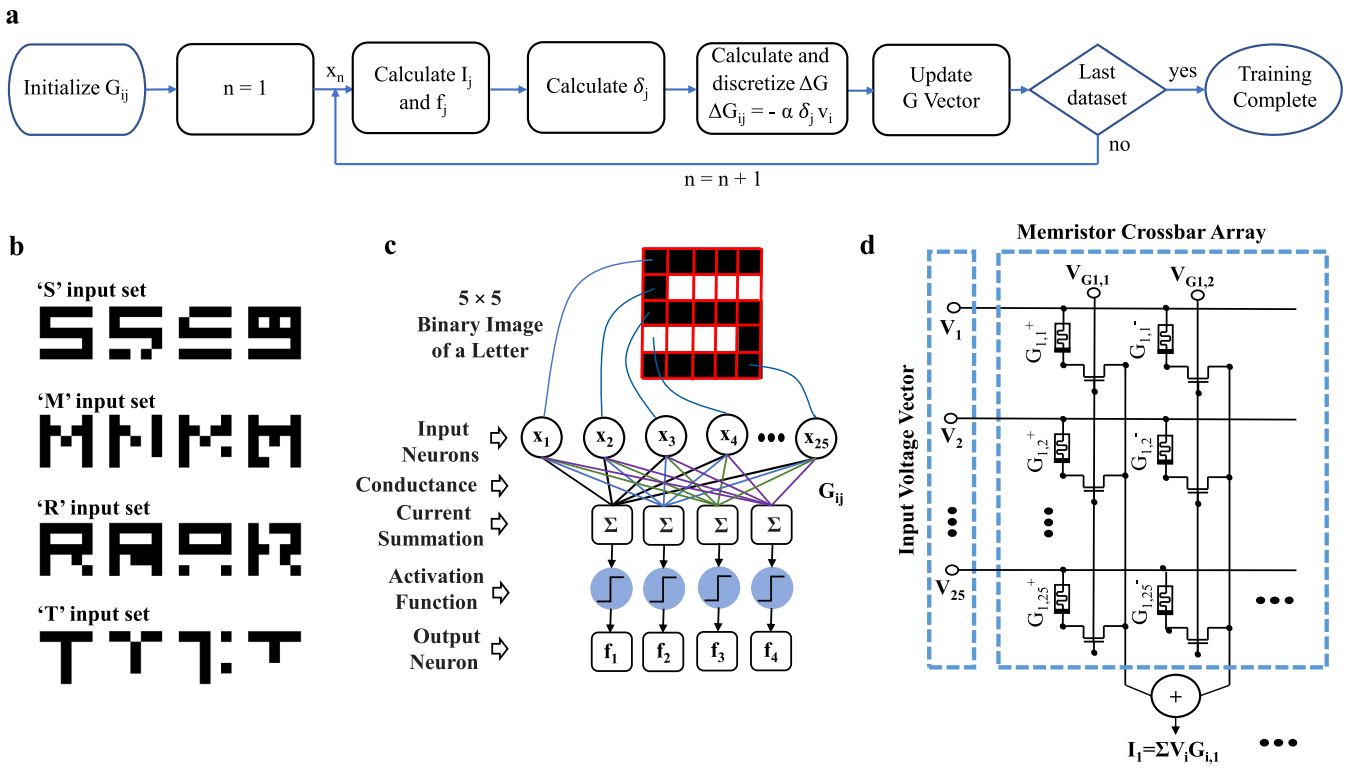

Fig. 6. Image classification experiment (physical-level description). (a) flow chart for the stochastic logistic regression algorithm memristor crossbar implementation. (b) The 5 × 5 input binary images (original image following the noisy ones). (c) Graph representation of the neural network implementation for classification image classification. (d) partial schematic of the 25 × 8 memristor crossbar simulated using HSpice + Verilog-A showing memristor differential pair.

the output currents are given by $I_j = \sum_{i=1}^{25} v_i G_{ij}$ where G_{ij} are the adjustable effective conductance and v_i are the input voltages. The output current is normalized and then goes through the sigmoid activation function which returns the value of $f_j = \frac{1}{1+e^{-I'_j}}$. Here, I'_j is the normalized output current of each column. In this normalization, the original current (I_j) is simply divided by a constant factor and presented to logistic function as I'_j. The sigmoid function gives an output ranging between 0 and 1. In this implementation, the classification error (δ_j) is calculated for each neuron as: $\delta_j = f_j - y_j$, where y_j is determined by the label in the training data set (equals 1 for the neuron that corresponds to the training image and zero for other neurons).

D. Demonstration of Logistic Regression

Similar to the linear regression demonstration, a *smart pulsing strategy* is used where different number of pulses are applied at each iteration based on the required conductance update given by $\Delta G_{ij} = -\alpha \times \delta_j v_i$. For ΔG_{ij} greater than ± 0.01 μS, five positive/negative pulses are applied, for ΔG_{ij} between ± 0.005 and ± 0.01, two positive/negative pulses are applied and for ΔG_{ij} Smaller than ± 0.005 μS, a single pulse is applied. This pulsing strategy is illustrated in Figure 7(a).

In this demonstration, the programming pulses have amplitudes of +1.4 V and –1.35 V, and widths of 20 ns and 10 ns respectively, and the learning rate, α, is constant with the value of 0.5. A single image from the dataset is presented to the network during each training step, followed by an adjustment of the effective conductance through the application of consecutive voltage pulses determined based on ΔG_{ij}. For example, if the effective conductance (ΔG_{ij}) needs to be increased, positive pulses are applied to the positive memristor (G^+_{ij}) and negative pulses are applied to the negative memristor (G^-_{ij}) in the differential pair. This increases the effective conductance. Similarly, if the effective conductance needs to be decreased, negative pulses are applied to the positive memristor and positive pulses are applied to the negative memristor in the differential pair. The accuracy of the prediction is evaluated at fixed training intervals using a separate dataset that consists of 400 5×5 binary images (100 noisy images for each character).

Figure 7 summarizes the results of the classification algorithm. We first compare the smart pulsing strategy against the constant pulse update approach described in [28] and implemented in a real crossbar in [29]. The constant pulsing approach is indicated by the dashed blue line in Figure 7(a), where a single pulse is applied independent of the value of ΔG_{ij}. Figure 7(b) shows the number of mismatched patterns for character "S" in the evaluation set as a function of training steps. With increasing training steps, the percentage of mismatched patters decreases. Red lines indicate the smart pulsing strategy proposed in this paper whereas blue shows the results for the constant pulsing method [28], [29]. The solid red line corresponds to the average mismatch from 5 different trials with different initial states and *without variation*. The

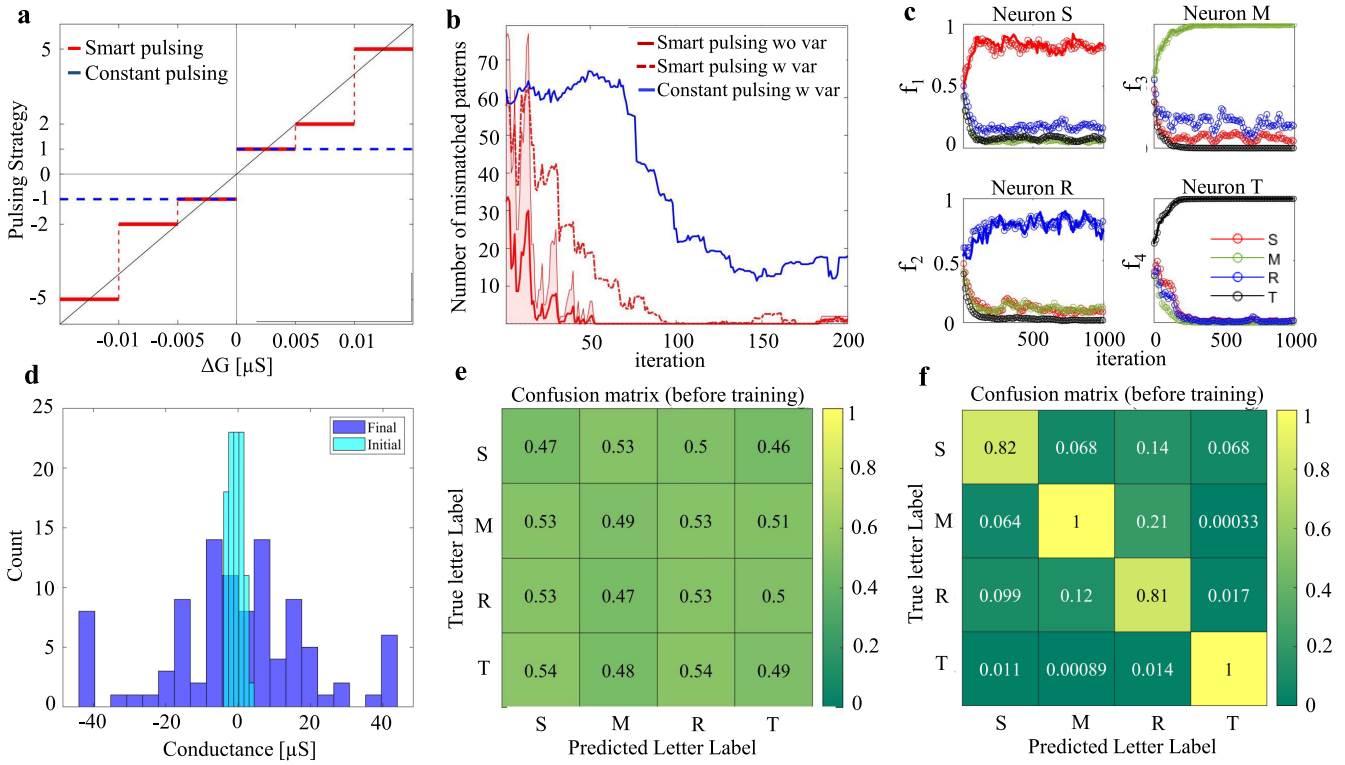

Fig. 7. (a) Pulsing strategy for the logistic regression algorithm implementation. Solid red line is the smart pulsing strategy, and blue dashed line is the constant pulsing. (b) Number of mismatched patterns (out of 100 test images) for neuron "S" during 200 initial training steps. Solid red line is the average number of mismatched patters from 5 different simulation runs for smart pulsing strategy without variation. Shaded region indicates the range of maximum and minimum values. The dotted red line is average with variation for smart pulsing. Constant blue line is the number of mismatched for the constant pulsing with variation. (c) Evolution of convergence for each output neuron for smart pulsing. Data represent output of sigmoid activation function (range between 0 and 1) for each output neuron. Scattered dots correspond to simulation without variation, and solid line corresponds to simulation with variation (average from 5 simulation runs with random initial states). (d) Histogram for the distribution of the initial and final effective conductance (weight) states for smart pulsing strategy with variation (iteration 1000) for neuron "S" (e) Confusion matrix for neuron "S" before training with effective conductance value initialized randomly for smart pulsing. (f) Confusion matrix for neuron "S" after training showing the final values of f_j for each neuron for smart pulsing.

shaded region indicates the range of maximum and minimum mismatch from all 5 individual runs. The red dotted line is the average mismatch *with variation*. As observed, it takes longer for the case with variation to reach almost perfect classification (zero mismatched patterns). The blue line is the average mismatch including memristor variability for the constant pulsing approach. *Compared to the smart pulsing strategy, convergence rate and accuracy are reduced*. For the smart pulsing strategy, Figure 7(c) shows the evolution of convergence based on the average output of the sigmoid function (f_j) at each of the neurons and for each of the different characters in the evaluation data set (100 noisy images for each character). For example, for the neuron assigned to character 'S' (labeled "Neuron S" in Figure 7(c)), f_j converges to a value close to 1 for images corresponding to the character 'S' and to values close to 0 for others. In Figure 7(c), the results shown with open circles are the average of 5 different trials (each up to 1000 training steps) *without memristor variation*. For comparison, solid lines plot the case *with variation* (only shown for results from images that match the assigned character to each neuron). These results indicate that memristor variation appears to have minimal impact on classification accuracy but affects precision by introducing more fluctuations as a function of training step (consistent with results from linear regression). Figures 7(e) and 7(f) show the confusion matrix corresponding to f_j values for each neuron before and after training. Before training, the f_j values for each neuron are randomly distributed around 0.5 based on the initial random effective conductances (see Figure 7(d) for distribution of initial and final effective conductance). The final values of f_j, after the training is complete (1000 steps), are shown in Figure 7(f). The results are in accordance with Figure 7(c), where corresponding neurons converge towards 1 and the non-corresponding neurons approach 0.

Our results indicates that with memristor variability, which is the realistic case for actual physical crossbars, more iterations are necessary to converge to a desirable classification accuracy. For more complex patterns, this gap may be large. From Figure 7(c), it can be concluded that because of the nature of logistic regression, where the output current (weighted sum of inputs) goes through the logistic function (in this case), the variation does not have outstanding impact in the learning process. It is important to point out that in some cases, small levels of device variation (noise) can help achieve improvements in accuracy as it may act as a form of regularization to prevent overfitting to the training set. This has been demonstrated in [50] for MNIST datasets where small levels of variability improved accuracy but was ultimately degraded

for larger levels of variation. Another technique to prevent overfitting and overreliance on individual devices is *dropout regularization* and is commonly used in multi-layer neural networks and was recently proposed to alleviate stuck-at-faults in memristor crossbar implementations [51]. Moreover, the work in [52]–[54] pointed out in the context of spiking neural networks that noise symmetrically distributed about a mean of zero will integrate out when trained across many samples. Another well-known source of noise that is neglected in the present analysis results from quantization of bit-line currents as typical implementations of the logistic function use digital circuits [47].

IV. Conclusion

A circuit-level analysis of 1T1R crossbar implementation of linear and logistic regression algorithms using a physics-based, variation-aware, and experimentally verified compact model for memristors is presented in this paper. The analysis includes the impact of device variability on convergence, as well as on prediction/classification accuracy and precision. The algorithm implementations are based on crossbar vector matrix multiplication, which is the core operation of typical neuromorphic computing platforms [29]. Moreover, this work presents an improved gradient-descent approach that is compatible with realistic hardware. This approach can achieve a fast initial convergence rate without compromising high prediction accuracy of the fully trained network. The results of this work indicate that our proposed *smart pulsing strategy* can be adjusted to accelerate training in real crossbar architectures. Our analysis regarding the impact of memristor variability on algorithm performance revealed the following: In linear regression, memristor variability does not appear to significantly affect prediction accuracy (can still achieve high accuracy), but convergence rate (how fast accuracy improves with training) and precision are noticeably degraded. The impact on precision is readily observed from fluctuations in the prediction error as a function of training steps (algorithm iterations). Similarly, in logistic regression, classification accuracy is not significantly affected by the memristor variability but a slower convergence rate and fluctuations in error as a function of algorithm iteration (impact on precision) were observed. We have also compared our proposed pulsing strategy to previous methods [28], [29] where a single positive or negative pulse is applied based on the sign of required update at each iteration. The proposed technique achieves faster convergence and better accuracy on classification of noisy binary images even in the presence of memristor variability. The findings of this paper are important to understand the impact of device variability algorithm performance and on the viability of memristor crossbars for prediction and classification tasks.

References

[1] A. Prakash, D. Jana, and S. Maikap, "TaO_x-based resistive switching memories: Prospective and challenges," *Nanosc. Res. Lett.*, vol. 8, no. 1, pp. 1–17, Dec. 2013.

[2] H.-S. P. Wong *et al.*, "Metal–oxide RRAM," *Proc. IEEE*, vol. 100, no. 6, pp. 1951–1970, Jun. 2012.

[3] V. Gupta, S. Kapur, S. Saurabh, and A. Grover, "Resistive random access memory: A review of device challenges," *IETE Tech. Rev.*, vol. 37, no. 4, pp. 377–390, Jul. 2020.

[4] L. Zhu, J. Zhou, Z. Guo, and Z. Sun, "An overview of materials issues in resistive random access memory," *J. Mater.*, vol. 1, no. 4, pp. 285–295, 2015.

[5] R. Patel, S. Kvatinsky, E. G. Friedman, and A. Kolodny, "Multistate register based on resistive RAM," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 23, no. 9, pp. 1750–1759, Sep. 2015.

[6] S. Yin *et al.*, "Monolithically integrated RRAM- and CMOS-based in-memory computing optimizations for efficient deep learning," *IEEE Micro*, vol. 39, no. 6, pp. 54–63, Nov. 2019.

[7] S. Yu, W. Shim, X. Peng, and Y. Luo, "RRAM for compute-in-memory: From inference to training," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 68, no. 7, pp. 2753–2765, Jul. 2021.

[8] D. Ielmini, "Resistive switching memories based on metal oxides: Mechanisms, reliability and scaling," *Semicond. Sci. Technol.*, vol. 31, no. 6, May 2016, Art. no. 063002.

[9] P. Lorenzi, R. Rao, and F. Irrera, "Role of the electrode metal, waveform geometry, temperature, and postdeposition treatment on SET and RESET of HfO_2-based resistive random access memory 1R-cells: Experimental aspects," *J. Vac. Sci. Technol. B, Nanotechnol. Microelectron., Mater., Process., Meas., Phenomena*, vol. 33, no. 1, Jan. 2015, Art. no. 01A107.

[10] C. Ye *et al.*, "Physical mechanism and performance factors of metal oxide based resistive switching memory: A review," *J. Mater. Sci. Technol.*, vol. 32, no. 1, pp. 1–11, Jan. 2016.

[11] D. Fey, "Memristors divide to conquer device variability," *Nature Electron.*, vol. 1, no. 8, pp. 438–439, Aug. 2018.

[12] M. K. Mahadevaiah *et al.*, "Reliability of CMOS integrated memristive HfO_2 arrays with respect to neuromorphic computing," in *Proc. IEEE Int. Rel. Phys. Symp. (IRPS)*, Mar. 2019, pp. 1–4.

[13] V. Milo *et al.*, "Multilevel HfO_2-based RRAM devices for low-power neuromorphic networks," *APL Mater.*, vol. 7, no. 8, Aug. 2019, Art. no. 081120.

[14] X. Hong *et al.*, "A novel geometry of ECM-based RRAM with improved variability," *J. Phys. D, Appl. Phys.*, May 2018, [Online]. Available: https://doi.org/10.1088/1361-6463/aac2b4

[15] A. Belmonte *et al.*, "Voltage-controlled reverse filament growth boosts resistive switching memory," *Nano Res.*, vol. 11, no. 8, pp. 4017–4025, 2018.

[16] J. Guy *et al.*, "Guidance to reliability improvement in CBRAM using advanced KMC modelling," in *Proc. IEEE Int. Rel. Phys. Symp. (IRPS)*, Apr. 2017, p. 2.

[17] J. Radhakrishnan *et al.*, "Impacts of Ta buffer layer and Cu–Ge–Te composition on the reliability of GeSe-based CBRAM," *IEEE Trans. Electron Devices*, vol. 66, no. 12, pp. 5133–5138, Dec. 2019.

[18] L. Goux *et al.*, "Key material parameters driving CBRAM device performances," *Faraday Discuss.*, vol. 213, pp. 67–85, Feb. 2019.

[19] M. Zhao, B. Gao, J. Tang, H. Qian, and H. Wu, "Reliability of analog resistive switching memory for neuromorphic computing," *Appl. Phys. Rev.*, vol. 7, no. 1, Mar. 2020, Art. no. 011301.

[20] P.-Y. Chen and S. Yu, "Reliability perspective of resistive synaptic devices on the neuromorphic system performance," in *Proc. IEEE Int. Rel. Phys. Symp. (IRPS)*, Mar. 2018, pp. 5C.4-1–5C.4-4.

[21] P.-Y. Chen *et al.*, "Mitigating effects of non-ideal synaptic device characteristics for on-chip learning," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2015, pp. 194–199.

[22] J. Tang *et al.*, "ECRAM as scalable synaptic cell for high-speed, low-power neuromorphic computing," in *IEDM Tech. Dig.*, Dec. 2018, pp. 1–13.

[23] G. Pedretti and D. Ielmini, "In-memory computing with resistive memory circuits: Status and outlook," *Electronics*, vol. 10, no. 9, p. 1063, Apr. 2021.

[24] Y. Liu, J. Gao, F. Wu, H. Tian, and T.-L. Ren, "The origin of CBRAM with high linearity, on/off ratio, and state number for neuromorphic computing," *IEEE Trans. Electron Devices*, vol. 68, no. 5, pp. 2568–2571, May 2021.

[25] A. Ali *et al.*, "Thickness-dependent monochalcogenide GeSe-based CBRAM for memory and artificial electronic synapses," *Nano Res.*, pp. 1–15, Sep. 2021. [Online]. Available: https://doi.org/10.1007/s12274-021-3793-1

[26] A. Chen and M.-R. Lin, "Variability of resistive switching memories and its impact on crossbar array performance," in *Proc. Int. Rel. Phys. Symp.*, Apr. 2011, pp. MY.7.1–MY.7.4.

[27] P.-Y. Chen and S. Yu, "Compact modeling of RRAM devices and its applications in 1T1R and 1S1R array design," *IEEE Trans. Electron Devices*, vol. 62, no. 12, pp. 4022–4028, Dec. 2015.

[28] M. V. Nair and P. Dudek, "Gradient-descent-based learning in memristive crossbar arrays," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2015, pp. 1–7.

[29] M. Prezioso, F. Merrikh-Bayat, B. D. Hoskins, G. C. Adam, K. K. Likharev, and D. B. Strukov, "Training and operation of an integrated neuromorphic network based on metal-oxide memristors," *Nature*, vol. 521, no. 7550, pp. 61–64, 2015.
[30] L. Chen *et al.*, "Accelerator-friendly neural-network training: Learning variations and defects in RRAM crossbar," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Mar. 2017, pp. 19–24.
[31] A. Malhotra, S. Lu, K. Yang, and A. Sengupta, "Exploiting oxide based resistive RAM variability for Bayesian neural network hardware design," *IEEE Trans. Nanotechnol.*, vol. 19, pp. 328–331, 2020.
[32] S. Kim, H.-D. Kim, and S.-J. Choi, "Impact of synaptic device variations on classification accuracy in a binarized neural network," *Sci. Rep.*, vol. 9, no. 1, pp. 1–7, Dec. 2019.
[33] K. Moon *et al.*, "RRAM-based synapse devices for neuromorphic systems," *Faraday Discuss.*, vol. 213, pp. 421–451, Feb. 2019.
[34] F. L. Aguirre, S. M. Pazos, F. Palumbo, J. Suñé, and E. Miranda, "SPICE simulation of RRAM-based cross-point arrays using the dynamic memdiode model," *Frontiers Phys.*, vol. 9, p. 548, Sep. 2021.
[35] E. Salvador, M. B. Gonzalez, F. Campabadal, J. Martin-Martinez, R. Rodriguez, and E. Miranda, "SPICE modeling of cycle-to-cycle variability in RRAM devices," *Solid-State Electron.*, vol. 185, Nov. 2021, Art. no. 108040.
[36] J. Reuben, M. Biglari, and D. Fey, "Incorporating variability of resistive RAM in circuit simulations using the Stanford–PKU model," *IEEE Trans. Nanotechnol.*, vol. 19, pp. 508–518, 2020.
[37] Y. Zhao, R. Chen, P. Huang, and J. Kang, "Modeling-based design of memristive devices for brain-inspired computing," *Frontiers Nanotechnol.*, vol. 3, p. 18, Apr. 2021.
[38] I. Messaris *et al.*, "NbO_2-Mott memristor: A circuit-theoretic investigation," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 68, no. 12, pp. 4979–4992, Dec. 2021.
[39] S. Vahdat, M. Kamal, A. Afzali-Kusha, and M. Pedram, "Reliability enhancement of inverter-based memristor crossbar neural networks using mathematical analysis of circuit non-idealities," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 68, no. 10, pp. 4310–4323, Oct. 2021.
[40] Y. Y. Chen *et al.*, "Balancing set/reset pulse for $>10^{10}$ endurance in HfO_2/Hf 1T1R bipolar RRAM," *IEEE Trans. Electron Devices*, vol. 59, no. 12, pp. 3243–3249, Dec. 2012.
[41] Y. Y. Chen *et al.*, "Improvement of data retention in HfO_2/Hf 1T1R RRAM cell under low operating current," in *IEDM Tech. Dig.*, Dec. 2013, pp. 10.1.1–10.1.4.
[42] A. Fantini *et al.*, "Intrinsic switching variability in HfO_2 RRAM," in *Proc. 5th IEEE Int. Memory Workshop*, May 2013, pp. 30–33.
[43] *Predictive Technology Model (PTM)*, Arizona State Univ., Tempe, Arizona, 2006.
[44] K. Mbarek, F. O. Rziga, S. Ghedira, and K. Besbes, "On the design and analysis of a compact array with 1T1R RRAM memory element," *Anal. Integr. Circuits Signal Process.*, vol. 102, no. 1, pp. 27–37, Jan. 2020.
[45] M. Marinella *et al.*, "Energy efficient neuromorphic algorithm training with analog memory arrays," Sandia Nat. Lab (SNL-CA), Livermore, CA, USA, Tech. Rep. SAND2019-8314C, 677565, Jul. 2019.
[46] Y. van de Burgt *et al.*, "A non-volatile organic electrochemical device as a low-voltage artificial synapse for neuromorphic computing," *Nature Mater.*, vol. 16, no. 4, pp. 414–418, 2017.
[47] J.-S. Seo *et al.*, "On-chip sparse learning acceleration with CMOS and resistive synaptic devices," *IEEE Trans. Nanotechnol.*, vol. 14, no. 6, pp. 969–979, Nov. 2015.
[48] M. Musisi-Nkambwe, S. Afshari, H. Barnaby, M. Kozicki, and I. S. Esqueda, "The viability of analog-based accelerators for neuromorphic computing: A survey," *Neuromorphic Comput. Eng.*, vol. 1, no. 1, Sep. 2021, Art. no. 012001.
[49] M.-F. Chang *et al.*, "Challenges and circuit techniques for energy-efficient on-chip nonvolatile memory using memristive devices," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 5, no. 2, pp. 183–193, Jun. 2015.
[50] Y.-C. Chen, J. K. Eshraghian, I. Shipley, and M. Weiss, "Analog synaptic behaviors in carbon-based self-selective RRAM for in-memory supervised learning," in *Proc. IEEE 71st Electron. Compon. Technol. Conf. (ECTC)*, Jun. 2021, pp. 1645–1651.
[51] Q. Xu, J. Wang, H. Geng, S. Chen, and X. Wen, "Reliability-driven neuromorphic computing systems design," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Feb. 2021, pp. 1586–1591.
[52] J. K. Eshraghian *et al.*, "Training spiking neural networks using lessons from deep learning," 2021, *arXiv:2109.12894*.
[53] T. P. Lillicrap, A. Santoro, L. Marris, C. J. Akerman, and G. Hinton, "Backpropagation and the brain," *Nature Rev. Neurosci.*, vol. 21, no. 6, pp. 335–346, 2020.
[54] A. Gholami, S. Kim, Z. Dong, Z. Yao, M. W. Mahoney, and K. Keutzer, "A survey of quantization methods for efficient neural network inference," 2021, *arXiv:2103.13630*.
[55] Z. Sun, G. Pedretti, A. Bricalli, and D. Ielmini, "One-step regression and classification with cross-point resistive memory arrays," *Sci. Adv.*, vol. 6, no. 5, Jan. 2020, Art. no. eaay2378.

Sahra Afshari received the B.S. degree in computer and electrical engineering from Azad University, Mashhad, Iran, and the M.S. degree in electrical engineering from Arizona State University (ASU), Tempe, AZ, USA, where she is currently pursuing the Ph.D. degree in electrical engineering with the School of Electrical, Computer, and Energy Engineering. She has a few years of experience as an Automation Design Engineer in the power plant industry. She specializes in the semiconductor domain.

Mirembe Musisi-Nkambwe (Senior Member, IEEE) received the B.S. and M.S. degrees in electrical engineering from the Rochester Institute of Technology, NY, USA. She is currently pursuing the Ph.D. degree in electrical engineering with the School of Electrical, Computer, and Energy Engineering, Arizona State University (ASU), Tempe, AZ, USA. She is a 17 year veteran in the semiconductor industry as a Circuit Design Engineer at Intel Corp and previously at Freescale Semiconductor (now NXP Semiconductors), Chandler, AZ, USA.

Ivan Sanchez Esqueda (Senior Member, IEEE) received the Ph.D. degree from Arizona State University in 2011. He worked as a Research Scientist with the University of Southern California for seven years. He is currently an Assistant Professor of electrical engineering with Arizona State University. His current research interests include low-dimensional materials, nanoelectronics, and the development of new solid-state technologies for computing and memory applications.