All-Passive Hardware Implementation of Multilayer Perceptron Classifiers

Akshay Ananthakrishnan and Mark G. Allen, *Fellow, IEEE*

***Abstract*— Bottom–up-fabricated crossbars promise superior circuit density and 3-D integrability compared with the traditional CMOS-based implementations. However, their inherent stochasticity presents difficulties in building complex circuits from components that demand precise patterning and high registration accuracies. With fewer terminals than active devices, passive components offer higher device densities and registration tolerances, making them amenable to bottom–up synthesized nanocrossbars. Motivated by this preference for passivity, we explore, in this article, neuromorphic classifiers based on passive neurons and passive synapses. We demonstrate via SPICE simulations how a shallow network of the diode–resistor-based passive rectifier neurons and resistive voltage summers, despite its inherent inability to buffer, amplify, and negate signals, can recognize MNIST digits with 95.4% accuracy. We introduce weight-to-conductance mappings that enable negative weights to be implemented in hardware without excessive memory overheads. The influences of soft and hard defects on the classification performance are evaluated, and the methods to boost fault-tolerance are proposed. The first-order evaluation of the area, speed, and power consumption of the passive multilayer perceptron classifiers is undertaken, and the results are compared with a benchmark study in neuromorphic hardware.**

***Index Terms*— Analog circuits, complex neuromorphic systems, crossbar architectures, diode–resistor networks, multilayer perceptron hardware, neural network hardware, passive neurons, shallow neural networks, supervised learning.**

I. Introduction

THREE-DIMENSIONALLY interconnected analog neuromorphic computers offer a promising route to implementing complex machine learning tasks in a hardware-efficient manner. In this context, a widely adopted technology, namely, the CMOS/Nanohybrid (CMOL), utilizes nanoscale crossbars to store synaptic weights and a relatively sparse, CMOS-based neuron layer to perform logical operations [1], [2]. However, the inherent challenge of interfacing CMOS with nanocrossbars, i.e., neuron–synapse connections, restricts the scalability of CMOL [3]. An alternative approach to build powerful neuromorphic computers is to reduce CMOS dependence and, instead, develop neuron circuits that can be integrated into nanocrossbars [4], [5]. Compared with conventional top–down fabrication, bottom–up synthesis, unstructured [6]–[8] and semistructured [9], [10], can produce much larger 3-D networks of crisscrossing wires at lower costs [5]. However, their inherent stochasticity establishes a preference for transistor-free, passive circuits built using two-terminal devices, such as diodes and resistors, which are more tolerant of registration inaccuracies [4], [5]. Previous studies [9], [11] have already successfully demonstrated the Boolean logic using resistive switches and diodes in bottom–up-fabricated crossbars. The preference for simplicity at the nanoscale raises an important question: is it possible to build functional neuromorphic computers using passive neurons and passive synapses?

To answer this question, we must first identify the shortcomings of passive neurons and the architectures that best support a functional network of passive neurons. Given that passive neurons cannot negate inputs or isolate different layers of an all-passive neural network, it is reasonable to expect that loading effects could greatly impact classification performances [5]. Among the main factors determining loading, network depth (i.e., shallow versus deep) and neuron fan-out are key. Using a single hidden layer with many units, shallow multilayer perceptrons (SMLPs) generate varied representations of inputs and combine them meaningfully to ascertain the decision boundaries that demarcate various data classes. With enough training data and hidden layer units, SMLPs are universal approximators, and they can generate arbitrarily accurate approximations to any function [12]. On the other hand, deep multilayer perceptrons (DMLPs) utilize several hidden layers to represent a target function as a composition of simpler functions. From a hardware standpoint, all-passive DMLPs are suboptimal since: 1) stacking multiple bufferless layers of passive neurons will likely amplify loading effects and distort the voltage outputs of inner layers and 2) the inability to access inputs to buried hidden layers in DMLPs will prevent resourceful implementation of negative weights since each passive neuron cannot, by itself, produce complementary (positive and negative) outputs. In contrast to DMLPs, all but one layer of SMLPs can be accessed externally, and this provides advantages in implementing negative weights (discussed later). From a performance viewpoint, model compression methods have shown that SMLPs can achieve similar accuracies as DMLPs and, in some cases,

Manuscript received February 18, 2019; revised July 7, 2019, April 14, 2020, and July 14, 2020; accepted August 8, 2020. *(Corresponding author: Akshay Ananthakrishnan.)*

Akshay Ananthakrishnan is with the Department of Mechanical Engineering and Applied Mechanics, University of Pennsylvania, Philadelphia, PA 19104 USA (e-mail: akshayan@seas.upenn.edu).

Mark G. Allen is with the Department of Electrical and Systems Engineering, University of Pennsylvania, Philadelphia, PA 19104 USA (e-mail: mallen@seas.upenn.edu).

This article has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the authors.

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

Digital Object Identifier 10.1109/TNNLS.2020.3016901

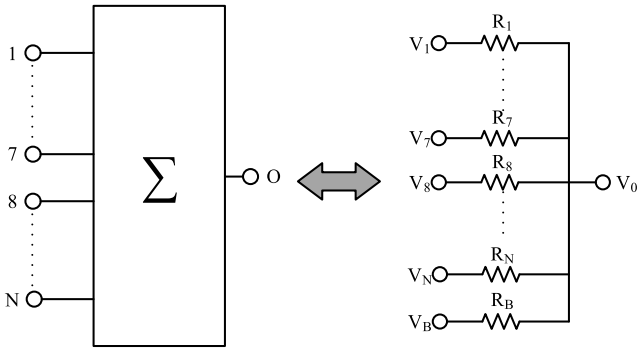

Fig. 1. PVS circuit (right) and its equivalent block diagram representation (left). Note that the block diagram implicitly includes the bias input V_B, without allocating a separate input terminal.

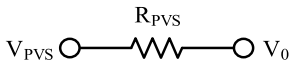

Fig. 2. Thevenin equivalent circuit of the PVS. Here, V_{PVS} and R_{PVS} are the Thevenin voltage and the Thevenin resistance, respectively.

without excessive parameters, i.e., the number of synaptic weights [13].

Motivated by these ideas, we present, in this article, a hardware implementation of all-passive SMLPs. We extend early theoretical works in analog diode logic [14], [15] to evolve passive circuit designs for neurons and synapses. Our passive rectifier neurons (PRNs) harness the piecewise linear approximation capabilities of diode networks to emulate "ReLU" activations. We show how the voltage summing properties of passive voltage summers (PVSs) can be utilized to implement resistive dot-product engines. We introduce methods that leverage the limited depth of SMLPs to implement signed synaptic weights in hardware using a nondifferential arrangement of resistors. These simple hardware representations present advantages for the fabrication of complex neuromorphic systems.

The remainder of this article is organized as follows. Section II introduces the building blocks of passive SMLPs, namely, the PVS and the PRN. Section III discusses the analytical core of passive SMLPs, namely, weight-to-conductance transformations, feature-to-input voltage mappings, and bias-voltage relationships. Section IV presents the passive SMLP performance on the MNIST digit classification task and assesses its susceptibility to defects. Strategies to improve fault-tolerance are discussed. Conclusions are presented in Section V.

II. Building Blocks of Passive Perceptron Hardware

A. Passive Voltage Summers

Fig. 1 describes a PVS comprised of parallel "synaptic" resistors R_1 to R_N and a "bias" resistor R_B. It receives voltage inputs V_1 to V_N and a constant bias input V_B and produces an output voltage V_0 across a high impedance load connected between output terminal O and ground (not shown in Fig. 1). Since the PVS is a resistive network, we use Thevenin's theorem to simplify the circuit in Fig. 1 to its equivalent representation in Fig. 2. Here, the Thevenin voltage V_{PVS} and the Thevenin resistance R_{PVS} are derived from the synaptic conductance vector G and the input voltage vector V as follows:

$$V_{\mathrm{PVS}} = \left(\sum_{i=1}^{N} \frac{G_i V_i}{G_{\mathrm{sum}}}\right) + \left(\frac{G_B V_B}{G_{\mathrm{sum}}}\right) \tag{1}$$

$$R_{\mathrm{PVS}} = \frac{1}{G_{\mathrm{PVS}}} \tag{2}$$

$$G_{\mathrm{PVS}} = G_{\mathrm{sum}} = \left(\sum_{i=1}^{N} G_i\right) + G_B. \tag{3}$$

The term G_{sum} (hereafter referred to as "conductance sum") in (3) is a constant that is set *a priori*. Unlike *in situ* training, where all Gs are optimized on-chip, the relatively 'low overhead' *ex situ* training approach adopted in this article determines all Gs from the corresponding weights of software-trained SMLPs. Note that the discussion so far, including the calculations in (1)–(3), considers the parasitic wire resistances to be negligible compared with any of the resistors in Fig. 1. While this assumption is true for the wire geometry adopted in this article (further discussed in section IV), it may not be reasonable for implementations with much smaller wire widths. In these cases, the vector-matrix multiplication operation by the PVS will be distorted, and V_{PVS} will deviate from (1). Optimizing the PVS performance in the presence of these distortions will require adopting hardware-aware training [16] or *in situ* training approaches [17].

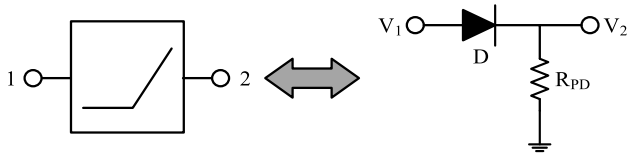

Fig. 3. Diode–resistor circuit design of a PRN (right) and its equivalent block diagram representation (left).

B. Passive Rectifier Neuron

A PRN receives an input from a PVS and produces an output that is a "rectified-linear" form of its input. This transformation guarantees that the magnitude of a PRN's output never exceeds that of its input, a property that enables its construction using the simple diode–resistor circuit shown in Fig. 3. Assuming the PRN to be ideal (i.e., the diode D is ideal) and in a "standalone" condition (i.e., when its output port is not connected to any load), we explain its operation by highlighting that: 1) for negative input voltages, i.e., $V_1 < 0$, the reverse-biased diode blocks the passage of current, and the PRN output is pulled down to ground potential, i.e., $V_2 = 0$ V and 2) when the input voltage is nonnegative, i.e., $V_1 \geq 0$, the ideal diode conducts, and this allows the PRN output to follow its input, i.e., $V_2 = V_1$. Mathematically, these two observations can be summarized as

$$V_2 = \max(0, V_1). \tag{4}$$

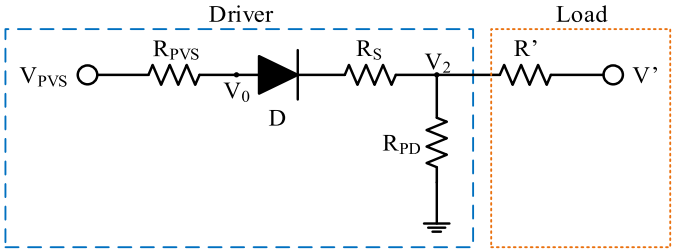

Fig. 4. Circuit diagram of a PRN receiving inputs from the PVS (represented by the Thevenin voltage V_{PVS} and the Thevenin resistance R_{PVS}) and driving a load (dotted orange block containing the Thevenin voltage V' and the Thevenin resistance R').

For a "real" PRN that utilizes a real diode, we adopt the piecewise linear diode model that accounts for the forward voltage drop V_F and the series resistance R_S of the diode. Although this model does not accurately capture: 1) the nonlinearity of diode I–V close to the turn-on voltage and 2) the finite reverse saturation current I_S, it is effective in identifying the key features of PRN networks (as seen in Section IV).

Consider a real PRN that receives its input from a PVS and drives a loading circuit comprised of voltage sources and resistors. Using the Thevenin-equivalent PVS circuit in Fig. 2 and representing the loading circuit by its Thevenin-equivalent form (i.e., a Thevenin voltage source V' in series with a Thevenin resistance R'), we obtain the circuit in Fig. 4. Here, we assume that the bias voltage input V_B to the PVS is $V_B + \Delta V_B$, where $\Delta V_B = V_F(G_{\mathrm{sum}}/G_B)$ [per (1)] compensates the finite forward voltage V_F. Therefore, we have not explicitly included a voltage source (supplying a voltage V_F with respect to ground) in Fig. 4. When the diode D in Fig. 4 is forward-biased, we can replace it with a "short," and under this condition, we can simplify the driving circuit (dotted blue box in Fig. 4) to its Thevenin-equivalent form, in Fig. 5. Here, V_{PRN} and R_{PRN} denote the Thevenin voltage and the Thevenin resistance, respectively, and they can be calculated as

$$V_{\mathrm{PRN}} = V_{\mathrm{PVS}}\left(\frac{\gamma}{\gamma+1}\right) \tag{5}$$

$$R_{\mathrm{PRN}} = (R_{\mathrm{PVS}} + R_S)\left(\frac{\gamma}{\gamma+1}\right) \tag{6}$$

where $\gamma = R_{\mathrm{PD}}/(R_{\mathrm{PVS}} + R_S)$. Using the voltage-divider formula and the expressions for V_{PRN} and R_{PRN} from (5) and (6), respectively, we get the PRN output voltage V_2 as

$$V_2 = \left(\frac{\gamma}{\gamma(\alpha+1)+1}\right)\left(V_{\mathrm{PVS}} + (\alpha)V'\right) \tag{7}$$

where $\alpha = (R_{\mathrm{PVS}} + R_S)/R'$. In contrast to the case discussed earlier, when the diode D is reverse-biased, we can replace it, effectively, with an "open," and this simplifies the circuit in Fig. 4 to its equivalent representation in Fig. 6.

Using the voltage-divider rule and the definitions of γ and α provided earlier, we can calculate the PRN output voltage

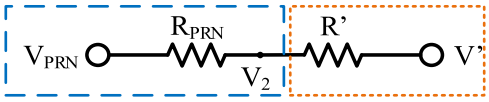

Fig. 5. Thevenin equivalent circuit (dashed blue block containing the Thevenin voltage V_{PRN} and the Thevenin resistance R_{PRN}) representing the combination of the PVS and the PRN (in a conducting state). The dotted orange block is the Thevenin equivalent load comprising of the Thevenin voltage V' in series with the Thevenin resistance R'.

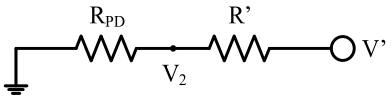

Fig. 6. PRN circuit when the neuron is in a nonconducting state, i.e., when the diode comprising the passive neuron is reverse biased.

V_2 as

$$V_2 = \left(\frac{\gamma\,\alpha}{1+\gamma\,\alpha}\right)V'. \tag{8}$$

From (7) and (8), we conclude that for both "conducting" and "nonconducting" states of the PRN, the resistors in Fig. 4 will cause the PRN output to deviate from the rectified-linear activation in (4), i.e., $V_2 = \max(0, V_{\mathrm{PVS}})$. A precise analytical description of this deviation is difficult to obtain, especially in the case of SMLPs, where the Thevenin-equivalent load resistor R', and hence α, for a specific PRN will depend on: 1) the PRN of interest; 2) the states of the remaining hidden PRNs; 3) R_{PVS} corresponding to other PVSs in the hidden layer; 4) R'_{PVS} corresponding to PVSs in the output layer; and 5) the constant value of R_{PD}. Therefore, it is only feasible to express α_k for the kth hidden PRN as $\alpha_k = f_k(\gamma, \lambda)$, where $\lambda = R'_{\mathrm{PVS}}/(R_{\mathrm{PVS}} + R_S)$ and γ are constants across all hidden PVSs (assuming a constant R_{PVS} for all hidden PVSs). Consequently, the outputs of hidden PRNs and the overall performance of the passive SMLP classifier will be a nontrivial function of the parameters G_{sum}, γ, and λ. Since these parameters determine the dc operating point of SMLPs, they also control the magnitudes of diode currents. For large SMLPs, the maximum permissible diode current will determine the lower bounds on γ and λ.

III. Weight-Conductance Transformations

We consider an *ex situ* training approach where we train a SMLP in software (hereafter referred to as "soft SMLP") and then implement it in hardware as a passive SMLP, by transforming: 1) the software-determined weights (hereafter referred to as "soft weights") in W into synaptic conductances in G and 2) the software-determined biases B (hereafter referred to as "soft bias") into voltage biases in V_B and conductances in G_B. This demands careful attention to the properties of PVSs described by (1)–(3). First, the strict nonnegativity of G_{sum}, G, and G_B implies that each "hardware synaptic weight," G_i/G_{sum}, and "hardware bias weight" G_B/G_{sum} are always nonnegative and less than unity. It is also imperative that the inequality $\sum_{i=1}^{N} G_i < G_{\mathrm{sum}}$ be satisfied so that the nonzero soft biases in B can be implemented in hardware by

V_B and G_B. These two critical aspects of PVS circuit design can be achieved by leveraging the limited depth of SMLPs and the scale-invariance property of PRNs (refer to Section S.I in the Supplementary Material for details).

Consider a soft SMLP classifier with "m" input nodes, "n" hidden layer nodes, and "p" output nodes that can be trained to learn the association between a feature vector $X(m \times 1)$ and its corresponding class labels $L(p \times 1)$. This knowledge is stored in the form of hidden weight $W(n \times m)$, the hidden soft bias $B(n \times 1)$, the output weight $W'(p \times n)$, and the output soft bias $B'(p \times 1)$. A passive SMLP hardware implementation of this soft SMLP utilizes "$2m$" voltage inputs, "n" pairs of PVS-PRN blocks in parallel in the hidden layer, and a total of "p" PVS blocks in the output layer. It is important to appreciate that classification tasks demand appropriate relative ordering of SMLP outputs without regard for their magnitudes, and this eliminates the need for neuron blocks in the output layer. Note that in comparison to the soft SMLP, the passive SMLP hardware utilizes "m" additional voltage inputs whereby out of the "$2m$" voltage inputs in $V_{\text{IN}}(2m \times 1)$, "$m$" are obtained from X, and the remaining "m" inputs are derived from $-X$. This scheme enables negative weights in W to be implemented with all positive conductances in G without requiring a differential configuration of resistors. By scaling the inputs in X appropriately, we obtain V_{IN} as follows:

$$V_{\text{IN}} = \left(KK'/K_V\right)\begin{bmatrix} X \\ -X \end{bmatrix} \tag{9}$$

$$K = \text{ceil}(T) + \epsilon \tag{10}$$

$$K' = \text{ceil}(T') + \epsilon' \tag{11}$$

$$T = \max_{1 \le i \le n}\left(\sum_{j=1}^{m}\left|W_{ij}\right|\right) \tag{12}$$

$$T' = \max_{1 \le i \le p}\left(\sum_{j=1}^{n}\left|W' + W'_{\text{SH}}\right|\right) \tag{13}$$

$$W'_{\text{SH}} = 1_p C^T \tag{14}$$

$$C_j = -\max_{1 \le i \le p}\left(W'_{ij}\right). \tag{15}$$

Here, "ceil' represents the ceiling function $\text{ceil}(x) = \min\{n \in Z | n \ge x\}$. The terms ϵ and ϵ' are nonnegative constants that must be set to arbitrary nonzero values when the respective values of T or T' are integers. This guarantees $K > ceil(T)$ and $K' > ceil(T')$ for arbitrary weight matrices W and W', thereby ensuring that all conductances in the hidden bias conductance vector G_B [obtained from (3)] have finite, nonzero values. Equations (14) and (15) define the weight-shift matrix W'_{SH} in terms of the minimum synaptic weight in each hidden layer PRN. The significance of W'_{SH} and its effect on SMLP outputs is discussed in the latter part of this section. Note that the voltage scale factor K_V in (9) is an arbitrary constant that restricts the input voltages in V_{IN} to a desired voltage range. While a large K_V can lower the power consumption of the SMLP, a concomitant drop in the output voltages will increase its susceptibility to noise. Thus, the optimal K_V emerges from a tradeoff between power consumption and classification accuracy.

From (9), we see that any negative soft weight W_{ij} between an input node i and a hidden node j in the soft SMLP classifier can be implemented in hardware by connecting the voltage input $(V_{\text{IN}})_{i+4}$ to the jth PRN through a resistor whose conductance depends only on the magnitude of the soft weight $|W_{ij}|$. Thus, by transferring the sign associated with each W_{ij} to the voltage inputs in V_{IN}, we can represent $|W_{ij}|$ using conductances in the input-hidden synaptic conductance matrix $G(2m \times n)$. We construct a block matrix representation of G as follows:

$$G = \begin{bmatrix} G_N^+ & G_N^- \end{bmatrix} \tag{16}$$

$$\left(G_N^+\right)_{ij} = \left(\frac{G_{\text{sum}}}{2K}\right)\left[1 + \text{sgn}\left(W_{ij}\right)\right]\left|W_{ij}\right| \tag{17}$$

$$\left(G_N^-\right)_{ij} = \left(\frac{G_{\text{sum}}}{2K}\right)\left[1 - \text{sgn}\left(W_{ij}\right)\right]\left|W_{ij}\right| \tag{18}$$

where G_{sum} is a constant that denotes the conductivity sum for each hidden PVS. From (18) and (19), we find that when $(G_N^+)_{ij} \neq 0$, then $(G_N^-)_{ij} = 0$, and when $(G_N^+)_{ij} = 0$, then $(G_N^-)_{ij} \neq 0$, and vice versa. This property ensures that G has exactly "$m \cdot n$" nonzero elements, each corresponding to a synaptic weight in W. Using the relationship between G, G_{sum}, and G_B provided in (3), we calculate the hidden bias conductance $G_B(n \times 1)$ as follows:

$$G_B = G_{\text{sum}} 1_n - G 1_{2m} \tag{19}$$

where 1_{2m} and 1_n are "all-ones" vectors with dimensions $2m \times 1$ and $n \times 1$, respectively. Using the hidden soft bias B and the hidden bias G_B from (19), we obtain the hidden bias voltage V_B as

$$V_B = G_{\text{sum}} D_B\left[\left(\frac{K'}{K_V}\right)B + V_F 1_n\right] \tag{20}$$

$$(D_B)_{ij} = \frac{\delta_{ij}}{(G_B)_i} \tag{21}$$

where $j = 1, \ldots, n$, δ is the Kronecker delta, and there is no implied summation over the indices. Note that the last term in (20) biases the diode to an operating point that is empirically chosen to maximize network accuracy.

The outputs of real PRNs are always nonnegative due to the activations they perform. Hence, the above approach cannot be employed for hardware implementation of negative weights in W'. To address this bottleneck, we propose a method that: 1) shifts all weights in W' by the magnitude of the most negative weight in W', thereby ensuring that all weights in W' are nonnegative and then 2) transforms these modified weights into conductances in $G'(n \times p)$. These two sequential steps can be expressed mathematically as

$$W'_P = W' + W'_{\text{SH}} \tag{22}$$

$$G' = \left(\frac{G'_{\text{sum}}}{K'}\right)W'_p \tag{23}$$

where G'_{sum} is a constant signifying the conductance-sum associated with each output PVS, and W'_{SH} (referred to as the weight-shift) is defined by (14) and (15). It is useful to

note that W'_{SH} introduces some sparsity in G', such that out of the np conductances in G', a minimum of "n" conductances are identically zero. The overall sparsity in G' could be further enhanced by pruning the network appropriately. These strategies optimize the required number of interconnects and, thereby, offer advantages for the fabrication of passive SMLP hardware. Using (3) and the value of G' calculated from (23), we compute the bias conductance G'_B for output PVSs as

$$G'_B = G'_{\mathrm{sum}} 1_p - G' 1_n \tag{24}$$

where 1_p and 1_n are all-ones vectors of dimensions $p \times 1$ and $n \times 1$, respectively. Since the output layer does not utilize PRNs, the output bias voltage V'_B can be calculated as

$$V'_B = \left(\frac{G'_{\mathrm{sum}}}{K_V}\right) D'_B B' \tag{25}$$

$$\left(D'_B\right)_{ij} = \frac{\delta_{ij}}{\left(G'_B\right)_i} \tag{26}$$

where $i, j = 1, \ldots, p$, δ is the Kronecker delta, and there is no implied summation over the indices.

The above approach assumes that shifting weights in W' according to (24) does not impact the accuracy of SMLP classifiers. To validate this, we combine (22) and (23) as follows:

$$G' = G'_0 + \Delta G'_0 \tag{27}$$

where $G'_0 = (G'_{\mathrm{sum}}/K')W'$ and $\Delta G'_0 = (G'_{\mathrm{sum}}/K')W'_{\mathrm{SH}}$.

Here, G'_0 represents the original synaptic conductance obtained directly from W', i.e., without shifting W' by W'_{SH}, and $\Delta G'_0$ is the offset to G'_0 caused by W'_{SH}. Assuming that the passive SMLP has a large output impedance (with respect to ground), we employ (1) and (27) to quantify the influences of G'_0 and $\Delta G'_0$ on the SMLP output V'(approximated to be equal to the open-circuit voltage due to the large output impedance) as

$$V' = \frac{G'V_H + G'_B V'_B}{G'_{\mathrm{sum}}} = V'_0 + \Delta V'_0 \tag{28}$$

where $V'_0 = (1/G'_{\mathrm{sum}})(G'_0 V_H + G'_B V'_B)$ and $\Delta V'_0 = (1/G'_{\mathrm{sum}})(\Delta G'_0 V_H)$.

Here, V_H is the voltage output of hidden PRNs, V'_0 is the original voltage output of output PVSs (using G'_0 instead of G'), and $\Delta V'_0$ is the voltage offset produced by the conductance offset $\Delta G'_0$. It is useful to note that since the soft-bias B' is a constant, it satisfies the equality $B'/K_V = G'_B V'_B = G'_{B0} V'_{B0}$. Here, G'_{B0} and V'_{B0} are the original output bias conductance and the output bias voltage, respectively, which are obtained by replacing G' in (24) and (26) with G'_0. Using (14), (27), and (28), we obtain $\Delta V'_0$ as

$$\Delta V'_0 = \left(\frac{C \cdot V_H}{K'}\right) 1_p = V_{\mathrm{OFS}} 1_p \tag{29}$$

where $V_{\mathrm{OFS}} = (C \cdot V_H/K')$. Here, $C \cdot V_H$ is the dot product between vectors C [defined by (15)] and V_H. From (29), we see that W'_{SH} offsets all "p" voltage outputs of the passive SMLP by the same amount V_{OFS}. Since a constant V_{OFS} does not disturb the relative ordering of passive SMLP outputs, we conclude that shifting weights in W' does not affect the classification accuracies of passive SMLPs.

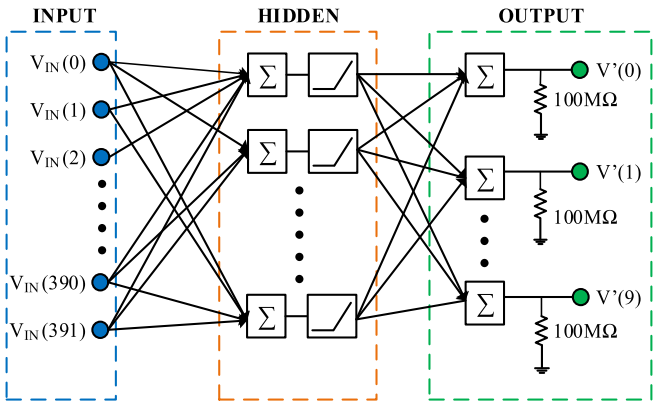

Fig. 7. Passive SMLP circuit for the MNIST digit classification problem. The inputs $V_{\mathrm{IN}}(196) - V_{\mathrm{IN}}(391)$ are corresponding negatives of $V_{\mathrm{IN}}(0) - V_{\mathrm{IN}}(195)$. Note that all output weights are nonnegative, and thus, negated values of hidden layer outputs are unnecessary. The 100 MΩ output resistors were chosen to be larger than all other resistors in the network.

IV. Passive Multilayer Perceptron Classifier

Using the PVS and PRN circuits discussed previously, we implemented a passive SMLP that classifies the MNIST digits. As a part of preprocessing, the original 28-pixel × 28-pixel MNIST images were cropped to 20-pixel × 20-pixel and then resized to 14-pixel × 14-pixel. To restrict the magnitudes of voltage inputs to the SMLP, all pixel intensities were rescaled from their original values in the [0 255] interval to corresponding values in the [−2 2] interval. These intensity-rescaled images were then unrolled into vectors of sizes 196 × 1 and provided as inputs to a 196 – 60 – 10 soft SMLP. Neurons in the hidden layer of the soft SMLP employed "ReLU" activations, while those in the output layer implemented "softmax" activations. The soft SMLP was trained on the first 60,000 gray-scale MNIST images while constraining the maximum L_2 norms of weights and biases to 0.8 and 0.2, respectively. Such a constrained training approach ensured that the scale factors in (10) and (11) and the bias voltage inputs V_B and V'_B weren't large. Overall, the trained soft SMLP exhibited a 95.87% classification accuracy on the MNIST test set.

Fig. 7 shows the hardware instantiation of the soft SMLP used for SPICE simulations. Each hidden node of the hardware SMLP utilized a serial arrangement of PVS and PRN blocks, and output nodes employed a PVS block each. The weight-to-conductance transformations introduced in Section III, along with the optimized values of λ and γ (discussed later in this section), were used to calculate the "ideal" conductances of all resistors. These were then quantized into 65 discrete conductance "bins" between 1 and 500 μS based on practically achievable analog states [17]–[20]. For SPICE simulations of our passive SMLPs, we modeled all memristors as resistors (with a temperature coefficient of resistance $TC_1 = 0.0015[1/\mathrm{K}]$ and nominal temperature $T_{\mathrm{NOM}} = 300$ K) in parallel with a parasitic capacitor $C_m = \epsilon_0 \epsilon_r w^2/d = 10$ fF, where $\epsilon_r = 60$, $w = 0.5$ μm is the electrode width, and $d = 15$ nm is the thickness of TiO_{2-x} active layer [21] [22]. The diodes comprising the PRNs in Fig. 7 were modeled as

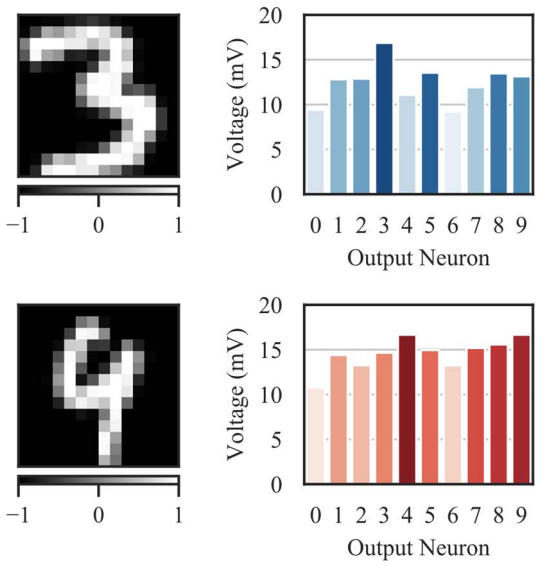

Fig. 8. Operation of the passive SMLP showing voltage-encoded inputs (left column) and the corresponding output voltages (right column). The first row describes the accurate classification of input "3" (left), and the second row shows how input "9" (left) was misclassified as "4" (red bar). Note that the color shades in the bar plots represent bar heights, where darker shades represent taller bars.

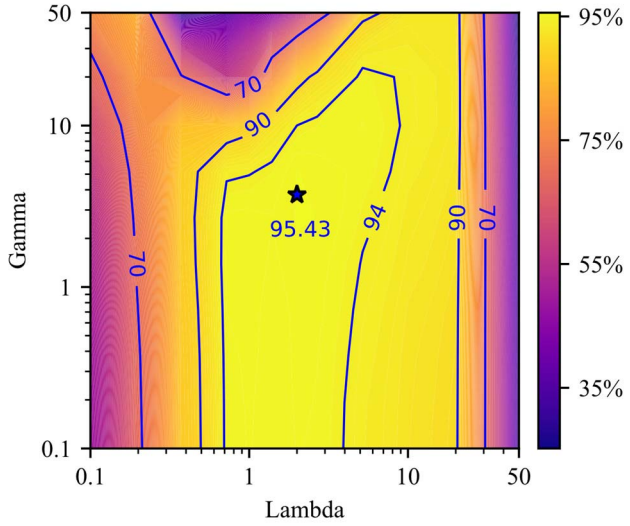

Fig. 9. Classification accuracy of the passive SMLP as a function of lambda λ and gamma γ, where λ is the ratio of the effective input admittance and the conductance sum of output PVSs, and γ is the ratio of the effective input admittance and the conductance of the pull-down resistor in the PRN circuit.

real nanoscale diodes [23], [24], and their SPICE parameters, namely, the reverse saturation current I_0, the ideality factor n, and the series resistance R_s, were obtained by fitting the diode equation $I = I_0 \exp((V - IR_s)/nV_T)$ to its I–V characteristic. The following empirically-determined diode parameters were utilized in all steady-state SPICE simulations: $I_0 = 0.69$ μA, $n = 4.76$, and $R_s = 286$ Ω. For transient analysis, we estimated the diode junction capacitance C_J from the reported switching time delay $T_S = 20$ ns and R_S as $C_J = (T_S/5R_S) = 14$ pF, where we assumed that it takes a total time $T_S = 5R_SC_J$ for the diode to reach steady-state following a step-voltage excitation. For a crossbar implementation of resistors based on nanowires having width $w = 0.5$ μm, thickness $h = 25$ nm, pitch $p = 0.5$ μm [18], and resistivity $\rho_{\text{wire}} = 4.77$ $\mu\Omega \cdot$ cm [22], we estimated the wire resistance per crossbar cell $R_{\text{wire}} = \rho l/wh = 4$ Ω. Since R_{wire} was much smaller compared with the remaining resistors in Fig. 7, it was not included in any of the SPICE simulations.

As shown in Fig. 8, the passive SMLP was presented with a 14-pixel $\times$ 14-pixel image input as a vector of 392 voltage values between -1 and 1 V (with a 10 mV resolution). These voltage inputs (along with the bias voltages V_B and V'_B) propagated through the network, and generated voltage outputs $V'(0)$ to $V'(9)$, each corresponding to a unique digit in [0 9]. Then, the largest voltage was identified, and the corresponding digit was predicted as the label for the specific input image. An example of accurate classification is shown in Fig. 8 (row 1), where a voltage-encoded "3" (left) was shown to the passive SMLP, and the corresponding voltage bar emerged as the tallest (dark blue bar on the right). Fig. 8 also illustrates an example of inaccurate classification (row 2), where a voltage encoded "9" was provided as the input, but the SMLP wrongly classified it as "4" (dark red bar on the right). Note that from a hardware implementation standpoint, the millivolt-level SMLP outputs will need to be amplified before comparison.

As discussed in Section III, the overall accuracy of passive SMLPs depends on λ and γ. Fig. 9 reveals an optimal range for λ and a certain upper bound for γ. For very small values of λ, i.e., $\lambda \ll 1$, the small resistors comprising the output PVSs draw large currents, and this promotes substantial deviations in the PRN outputs. On the other hand, for very large λ, the magnitudes of some of the output conductances diminish beyond the achievable lower bound, i.e., 1 μS in this case, thereby leading to inaccurate implementation of SMLP weights and biases. In the optimal λ regime, we speculate that a large value of γ/λ promotes the unwanted loading of PRN outputs by output layer PVSs. Overall, our simulations showed that the optimal combination of $\lambda = 2$ and $\gamma = 3.73$ allows passive SMLPs to achieve an "ideal" classification accuracy of 95.43%, a performance that is on-par with soft SMLPs, i.e., 95.87%. Note that the accuracies reported in Fig. 9 may vary depending upon the exact values of soft weights and biases obtained from *ex situ* training and the problem under consideration.

Using the parameters $\lambda = 2$ and $\gamma = 3.73$ in the weight-to-conductance transformations of Section III, we obtained the target conductances in G, G_B, G', and G'_B as well as the input and bias voltages of the network. In practice, however, these target values and, consequently, the hardware weights and biases can only be implemented up to a finite relative accuracy. From Fig. 10(a), we see that as the "conductance-programming accuracy" decreases, i.e., when the coefficient of variation of conductances increases, the classification accuracy of the passive SMLP decreases from its ideal value (dotted blue line). Noting that the state of the art in memristor technologies can implement conductances up to a relative

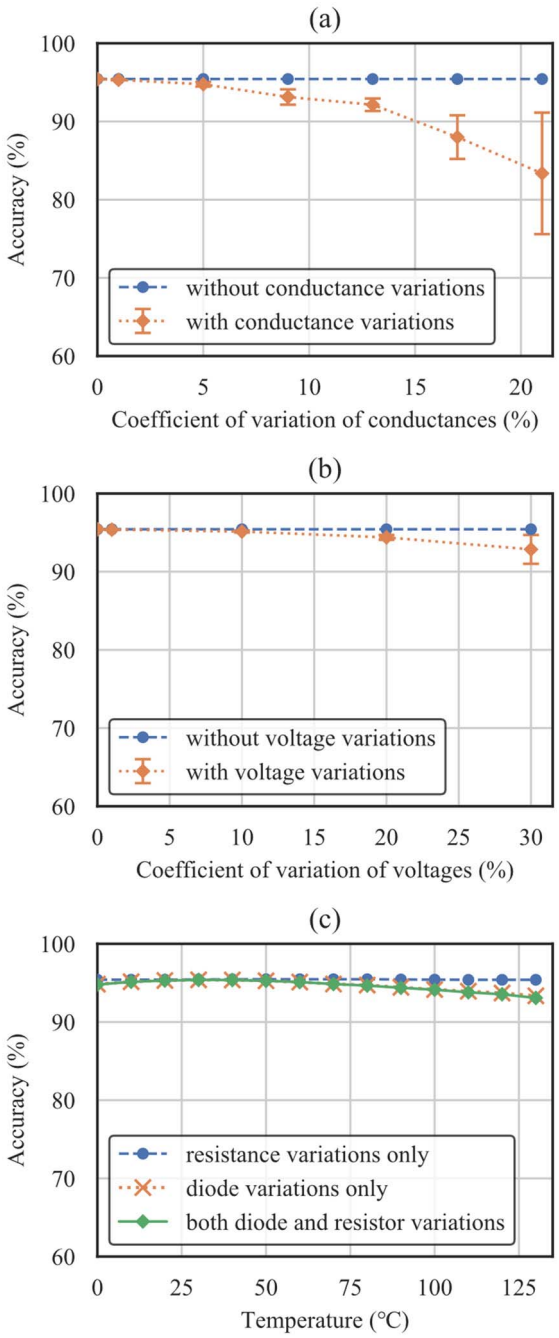

Fig. 10. Performance of passive SMLPs as a function of (a) conductance-, (b) voltage-, and (c) temperature-induced variations. Error bars indicate standard deviations across ten simulation runs.

accuracy of 1% [18], [25], we expect passive SMLPs to achieve accuracies of around 95.3% $\pm$ 0.1% even in the presence of 1% conductance variations [see Fig. 10(a)]. Apart from inaccurate conductances, variations in the input and bias voltages also influence the passive SMLP performance. Fig. 10(b) confirms that the impact of voltage disturbances on classification accuracy becomes noticeable only when the magnitudes of observed variations exceed 20%. However, modern DACs can provide analog input and bias voltages to relative accuracies better than 1%–2%; thus, we expect the effect of voltage variations on SMLP accuracy to be minimal.

Given the temperature dependence of resistor and diode properties, we investigated the SMLP performance as a function of operating temperature. From Fig. 10(c), we see that the SMLP accuracy (green solid line) decreases with increasing temperature (reference temperature: 27 °C). Note that this simulation assumed the temperature coefficient of resistance (TCR) of all resistors to be equal to 0.0015 [1/K] [26]. From Fig. 10(c), we observe that the accuracy obtained by modeling both diodes and resistors as temperature-dependent elements (green solid line) is nearly identical to that obtained by modeling the temperature dependence of diodes only (orange dotted line). This implies that the SMLP accuracy is relatively independent of temperature-induced resistance variations [blue dashed line in Fig. 10(c)], a result that can be explained by the following key points: 1) passive SMLPs encode soft weights as conductance ratios, not absolute conductances and 2) normalized changes in resistances have been assumed to vary linearly with temperature (with a constant TCR) [27]. Note that some resistive RAM technologies [26], [28] exhibit a more complicated $R - T$ relationship, where the TCR is not constant but depends on the conductance state, negative values at low conductances, and positive values at high conductances. However, even in these cases, we found the SMLP accuracy trend followed the solid green line in Fig. 10(c).

The sources of performance deviations discussed until now can be broadly classified as "soft defects" as they do not catastrophically impact system performances. We now attend to "hard defects," which, depending upon the nature of the defect, can inflict drastic performance degradations. Our simulations considered stuck at fault (SAF) defects of the following kinds: 1) stuck-at-open: the device is permanently in high impedance state and 2) stuck-at-short: the device is permanently in a low impedance state. We modeled defective diodes and resistors of types 1) and 2) by replacing them with large (100 MΩ) and small (100 Ω) resistors, respectively. A comparison of Fig. 11(a) and (b) reveals that passive SMLPs are more tolerant of faulty diodes than faulty resistors. In fact, from Fig. 11(a), we see that it is possible to achieve 80% accuracy even when 50% of the diodes are stuck-at-open. In both Fig. 11(a) and (b), we notice that the decline of accuracy with defect rate is sharper in the case of stuck-at-short faults (dotted green line) than in the case of stuck-at-open faults (dashed orange line) although the precise manner of performance degradation is contingent on important factors such as the selection of forward-biasing voltage V_F [see (20)] for defective and nondefective neurons. It is important to note that the results in Fig. 11 were obtained using $V_F = 0.4$ V, a value that was found optimal for nondefective networks, i.e., networks with a 0% defect rate. In the case of stuck-at-short resistors, we find that just 1% defective resistors plummeted the classification accuracy from 95.43% to less than 20%. Although the susceptibility of neuromorphic hardware to stuck-at-short faults is well known [29], [30], we expect them to pose a greater challenge for passive SMLPs that lack interlayer isolation. To improve their robustness to such hard defects, we adopted a combination

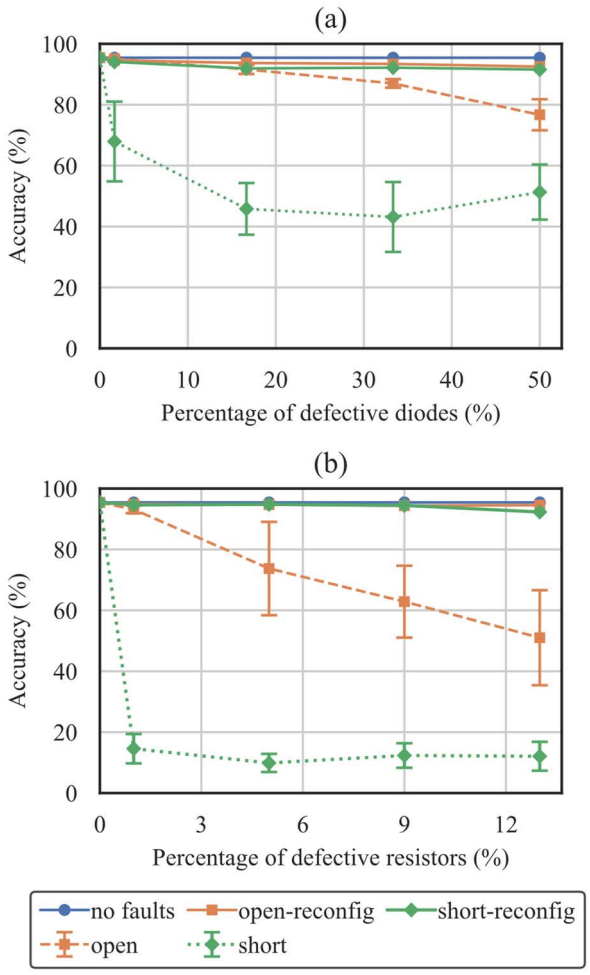

Fig. 11. Performance of passive SMLPs in the presence of (a) defective diodes and (b) defective resistors where the green dotted lines and orange dashed lines represent "stuck-at-short" and "stuck-at-open" defects, respectively. "Short-reconfig" (green solid lines) and "Open-reconfig" (orange solid lines) show the network performances achieved using the proposed three-pronged fault tolerance strategy. Note that the error bars indicate standard deviations across ten simulation runs.

of the following software and hardware level fault-tolerance strategies.

1) *Retraining:* Having identified defect locations, soft SMLPs were trained again to evolve a new set of defect-aware optimal weights for the network [30], [31].
2) *Redundancy:* To ensure a high probability of obtaining 60 defect-free pull-down resistors, we employed a 4X redundancy scheme, whereby each ground resistor had four potential substitutes [31]. Adopting this method incurs modest hardware overheads as it requires adding only four additional grounded columns/rows to the crossbar implementation of the circuit in Fig. 7.
3) *Cross Point Fuses:* Adding a fuse (or two) in series with resistors at each cross point is a widely adopted strategy for mitigating short defects in RRAM cells [32]–[34]. In this case, when a "short" resistor draws large currents while being programmed, the fuse connected to it will blow up, thereby disconnecting only that defective resistor from the PVS circuit. This prevents the outputs of PVSs containing "short" resistors from being pinned to erroneous voltage levels during inference. Note that for, nondefective cross-points, serial fuses will increase the total cross-point resistance, e.g., by about 800 Ω in the case of MnO_2 fuses [33]. However, as all the main resistances in Fig. 7 were much larger than the fuse resistance, we found the latter did not impact the passive SMLP performance.

The solid lines in Fig. 11 confirm how the three-pronged strategy discussed earlier makes passive SMLPs significantly more resilient to hard defects. In addition to defect-tolerance, we also evaluated the impact of poor retention characteristics of memristors, i.e., decay in conductances value over time on SMLP accuracy. In the absence of detailed state-dependent retention models, we adopted the first-order approximation where conductance drift was modeled as a fixed percentage decrease of all conductances relative to their respective programed values [35]. Simulations showed (results not shown here) that passive SMLPs were robust, and their accuracies decreased only by about 0.1% and 1.5% for 4X and 9X decreases in conductances, respectively.

Besides classification accuracy, we note that the areal footprint, speed, and power consumption are important yardsticks for comparing different hardware SMLPs. Based on demonstrated crossbar implementations of memristors [18] and diodes [24] that utilize conductive lines with width $l_w = 0.5\ \mu$m, line spacing $l_s = 0.5\ \mu$m, inputs $N_{\text{inp}} = 392$, hidden units $N_{\text{hid}} = 60$, and outputs $N_{\text{out}} = 10$, we estimate: 1) area of synaptic resistor crossbars $A_{\text{syn}} = (l_w + l_s)^2 \cdot [(N_{\text{inp}} \times N_{\text{hid}}) + (N_{\text{hid}} \times N_{\text{out}})] = 0.024\ \text{mm}^2$; 2) area of bias and pull-down resistor crossbars $A_{\text{bias}} = (l_w + l_s)^2 \cdot [N_{\text{hid}}^2 + (N_{\text{hid}} + N_{\text{out}}) + N_{\text{out}}^2] = 0.004\ \text{mm}^2$; and 3) area of the diode $A_{\text{diode}} = (l_w + l_s)^2 \cdot (N_{\text{hid}} \times 1) = 6 \times 10^{-5}\ \text{mm}^2$, giving us a total area of the computational core $A_{\text{core}} = A_{\text{syn}} + A_{\text{bias}} + A_{\text{diode}} \cong 0.028\ \text{mm}^2$. Based on this design, we estimate that a larger 1568-60-10 network (N_{inp} similar to benchmark's) will occupy $A_{\text{syn}} = 0.095\ \text{mm}^2$, and therefore, $A_{\text{core}} \cong 0.1\ \text{mm}^2$ (A_{bias} and A_{diode} values remain the same). Note that these calculations assume a conservative PLA-like design based on coplanar synapse and diode crossbars instead of a fully stacked 3-D configuration amenable to the all-passive hardware proposed in this article [36]. To evaluate how fast passive SMLPs can accomplish a single MNIST digit classification task, we determined the average time delay T_D of the network from the temporal responses of outputs $V'(0)$ to $V'(9)$ when all the input and bias voltages were suddenly switched on at $t = 10$ ns (assuming a 1 ns rise time). From Fig. 12, we see that all outputs reached steady states approximately 180 ns after stimulation (dashed black line) giving us $T_D = 180$ ns. With regards to power consumption, we found, based on the nodal voltages and branch currents, that the passive SMLP consumes an average static power $P_{\text{av}} = 134$ mW for a single classification task. Note that our limited computational resources precluded the detailed distributed modeling of nanowire-related parasitic elements, and hence, we did not include the nanowire resistances ($R_{\text{wire}} \cong 4\ \Omega$/cell) and capacitances (wire-substrate and wire-wire estimated at few

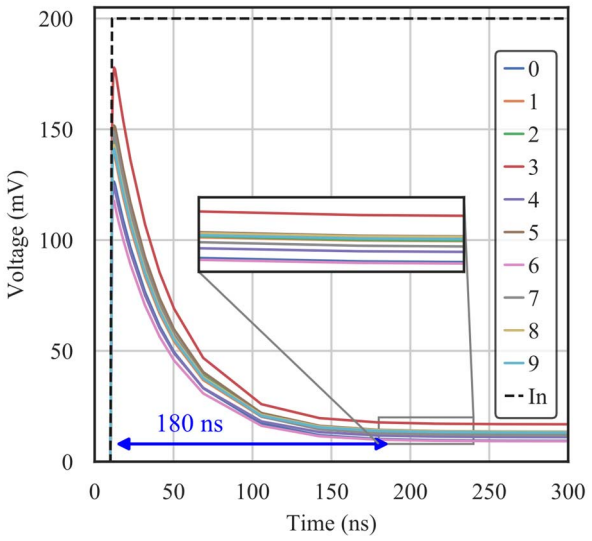

Fig. 12. Transient response of the passive SMLP when all voltage inputs are switched on at $t = 10$ ns (dashed black line depicts one such input with 1 ns rise time). The blue arrow depicts the average time delay of the passive MLP (∼180 ns).

tens of attofarads [22]) in our speed and power simulations. Based on literature-reported wire delays (for 5 nm resistive crossbars) of a few nanoseconds [22], [37], we expect the total delay in all-passive SMLPs to marginally exceed 180 ns. This suggests that all-passive SMLPs can retain their speed advantage at smaller scales too. Also, since $R_{\text{wire}}(4\ \Omega)$ is very small compared with the most conductive resistor (2000 Ω), we expect its power dissipation to be negligible. However, crossbars with much smaller wire-widths than that used in our simulations (wire-width $= 0.5\ \mu$m), will have a larger R_{wire}; as a result, maintaining sufficiently high output voltages will require boosting the input and bias voltages.

Compared with the state of the art in "active" neuromorphic implementations [38], we find that passive SMLPs are 6.7 times more power-hungry, 2.4 times faster (ideally), and up to three times more compact. Thus, the average power delay product per MNIST digit for the passive SMLPs $E_{\text{av}} = P_{\text{av}} T_{\text{MLP}} = 24$ nJ is almost 2.7 times that of the selected benchmark. Note that these comparisons are optimistic since they ignore overheads from the output amplification stage. Based on their speed advantage, all-passive neuromorphic systems can be useful in low latency information processing systems for autonomous lane detection [39], obstacle avoidance [40], [41], and event-driven motion prediction [42]. Looking ahead, we expect the next generation of all-passive SMLPs to be adaptive and capable of unsupervised "online" learning albeit at the cost of additional circuitry and related overheads [22], [29], [30], [43].

V. Conclusion

This article demonstrated how the combination of shallow network architectures and rectified-linear activations can be harnessed to build hardware MLP classifiers from all-passive building blocks, namely, diode–resistor neurons and resistive synapses. We identified two nondimensionless parameters that determine classification performance and showed that for an optimal choice of these parameters, all-passive MLPs can classify MNIST digits with 95.43% accuracy. Although passive MLP performances are susceptible to defects, we identified fault-tolerance strategies that address this drawback effectively. While this article discusses passive MLP classifiers only, the work presented here can be extended to all-passive regressors too, albeit with suitable modifications such as eliminating offsets in the output voltages. By demonstrating the possibility of building neuromorphic systems from simple circuit primitives, this work lays the foundation for more scalable deep neuromorphic computers.

References

[1] B. Chakrabarti *et al.*, "A multiply-add engine with monolithically integrated 3D memristor crossbar/CMOS hybrid circuit," *Sci. Rep.*, vol. 7, no. 1, pp. 1–10, Feb. 2017.

[2] D. B. Strukov and R. S. Williams, "Four-dimensional address topology for circuits with stacked multilayer crossbar arrays," *Proc. Nat. Acad. Sci. USA*, vol. 106, no. 48, pp. 20155–20158, 2009.

[3] S. Bhat, S. Kulkami, J. Shi, M. Li, and C. A. Moritz, "SkyNet: Memristor-based 3D IC for artificial neural networks," in *Proc. IEEE/ACM Int. Symp. Nanosc. Archit. (NANOARCH)*, Jul. 2017, pp. 109–114.

[4] M. M. Ziegler and M. R. Stan, "CMOS/nano co-design for crossbar-based molecular electronic systems," *IEEE Trans. Nanotechnol.*, vol. 2, no. 4, pp. 217–230, Dec. 2003.

[5] M. R. Stan, P. D. Franzon, S. C. Goldstein, J. C. Lach, and M. M. Ziegler, "Molecular electronics: From devices and interconnect to circuits and architecture," *Proc. IEEE*, vol. 9, no. 11, pp. 1940–1957, Nov. 2003.

[6] H. Tanaka *et al.*, "A molecular neuromorphic network device consisting of single-walled carbon nanotubes complexed with polyoxometalate," *Nature Commun.*, vol. 9, no. 1, p. 2693, Jul. 2018.

[7] A. Z. Stieg, A. V. Avizienis, H. O. Sillin, C. Martin-Olmos, M. Aono, and J. K. Gimzewski, "Emergent criticality in complex Turing B-type atomic switch networks," *Adv. Mater.*, vol. 24, no. 2, pp. 286–293, Jan. 2012.

[8] J. M. Tour *et al.*, "Nanocell logic gates for molecular computing," *IEEE Trans. Nanotechnol.*, vol. 1, no. 2, pp. 100–108, Jun. 2002.

[9] Y. Huang, X. Duan, Q. Wei, and C. M. Lieber, "Directed assembly of one-dimensional nanostructures into functional networks," *Science*, vol. 291, no. 5504, pp. 630–633, Jan. 2001.

[10] H. Finkelstein, P. M. Asbeck, and S. Esener, "Architecture and analysis of a self-assembled 3D array of carbon nanotubes and molecular memories," in *Proc. 3rd IEEE Conf. Nanotechnol. IEEE-NANO*, Aug. 2003, pp. 441–444.

[11] W. Lu and C. M. Lieber, "Nanoelectronics from the bottom up," *Nature Mater.*, vol. 6, no. 11, pp. 841–850, Nov. 2007.

[12] M. Leshno, V. Y. Lin, A. Pinkus, and S. Schocken, "Multilayer feedforward networks with a nonpolynomial activation function can approximate any function," *Neural Netw.*, vol. 6, no. 6, pp. 861–867, Jan. 1993.

[13] J. Ba and R. Caruana, "Do deep nets really need to be deep," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 2654–2662.

[14] R. H. Wilkinson, "A method of generating functions of several variables using analog diode logic," *IEEE Trans. Electron. Comput.*, vol. EC-12, no. 2, pp. 112–129, Apr. 1963.

[15] T. E. Stern, *Theory of Nonlinear Networks and Systems*. New York, NY, USA: Addison-Wesley, 1965.

[16] M. E. Fouda, S. Lee, J. Lee, A. Eltawil, and F. Kurdahi, "Mask technique for fast and efficient training of binary resistive crossbar arrays," *IEEE Trans. Nanotechnol.*, vol. 18, pp. 704–716, 2019.

[17] F. Alibart, E. Zamanidoost, and D. B. Strukov, "Pattern classification by memristive crossbar circuits using ex situ and *in situ* training," *Nature Commun.*, vol. 4, no. 1, pp. 1–7, Oct. 2013.

[18] G. C. Adam, B. D. Hoskins, M. Prezioso, F. Merrikh-Bayat, B. Chakrabarti, and D. B. Strukov, "3-D memristor crossbars for analog and neuromorphic computing applications," *IEEE Trans. Electron Devices*, vol. 64, no. 1, pp. 312–318, Jan. 2017.

[19] 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, May 2015.
[20] I. Boybat *et al.*, "Neuromorphic computing with multi-memristive synapses," *Nature Commun.*, vol. 9, no. 1, pp. 1–12, Dec. 2018.
[21] M. D. Stamate, "Dielectric properties of TiO2 thin films deposited by a DC magnetron sputtering system," *Thin Solid Films*, vol. 372, nos. 1–2, pp. 246–249, Sep. 2000.
[22] M. E. Fouda, A. M. Eltawil, and F. Kurdahi, "Modeling and analysis of passive switching crossbar arrays," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 65, no. 1, pp. 270–282, Jan. 2018.
[23] M.-J. Lee *et al.*, "2-stack 1D-1R cross-point structure with oxide diodes as switch elements for high density resistance RAM applications," in *IEDM Tech. Dig.*, Jan. 2007, pp. 771–774.
[24] B. S. Kang *et al.*, "High-current-density CuO $_x$/InZnO$_x$ thin-film diodes for cross-point memory applications," *Adv. Mater.*, vol. 20, no. 16, pp. 3066–3069, 2008.
[25] F. Alibart, L. Gao, B. D. Hoskins, and D. B. Strukov, "High precision tuning of state for memristive devices by adaptable variation-tolerant algorithm," *Nanotechnology*, vol. 23, no. 7, Feb. 2012, Art. no. 075201.
[26] J. Borghetti, D. B. Strukov, M. D. Pickett, J. J. Yang, D. R. Stewart, and R. S. Williams, "Electrical transport and thermometry of electroformed titanium dioxide memristive switches," *J. Appl. Phys.*, vol. 106, no. 12, pp. 1–5, 2009.
[27] F. M. Bayat, M. Prezioso, B. Chakrabarti, H. Nili, I. Kataeva, and D. Strukov, "Implementation of multilayer perceptron network with highly uniform passive memristive crossbar circuits," *Nature Commun.*, vol. 9, no. 1, pp. 1–7, Dec. 2018.
[28] F. Miao *et al.*, "Anatomy of a nanoscale conduction channel reveals the mechanism of a high-performance memristor," *Adv. Mater.*, vol. 23, no. 47, pp. 5633–5640, Dec. 2011.
[29] F. M. Bayat, M. Prezioso, B. Chakrabarti, I. Kataeva, and D. Strukov, "Memristor-based perceptron classifier: Increasing complexity and coping with imperfect hardware," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2017, pp. 549–554.
[30] C. Li *et al.*, "Efficient and self-adaptive *in-situ* learning in multilayer memristor neural networks," *Nature Commun.*, vol. 9, no. 1, pp. 7–14, Dec. 2018.
[31] L. Xia *et al.*, "Stuck-at fault tolerance in RRAM computing systems," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 8, no. 1, pp. 102–115, Mar. 2018.
[32] H. W. Cheng *et al.*, "A novel rewritable one-time-programming OTP (RW-OTP) realized by dielectric-fuse RRAM devices featuring ultra-high reliable retention and good endurance for embedded applications," in *Proc. Int. Symp. VLSI Technol., Syst. Appl. (VLSI-TSA)*, Apr. 2018, pp. 1–2.
[33] M. M. Zhang, J. Yang, and R. S. Williams, "Protective elements for non-volatile memory cells in crossbar arrays," U.S. Patent 10 147 762, Dec. 4, 2018. [Online]. Available: https://patents.google.com/patent/US10147762B2
[34] L. T. Tran, T. C. Anthony, and F. A. Perner, "One-time programmable memory using fuse/anti-fuse and vertically oriented fuse unit memory cells," U.S. Patent 6 584 029, Jun. 24, 2003. [Online]. Available: https://patents.google.com/patent/US6584029B2
[35] S. Gi, I. Yeo, M. Chu, S. Kim, and B. Lee, "Fundamental issues of implementing hardware neural networks using memristor," in *Proc. Int. SoC Design Conf. (ISOCC)*, Nov. 2015, pp. 215–216.
[36] A. Dehon, "Nanowire-based programmable architectures," *ACM J. Emerg. Technol. Comput. Syst.*, vol. 1, no. 2, pp. 109–162, Jul. 2005.
[37] C. Yakopcic, R. Hasan, T. M. Taha, and D. Palmer, "SPICE analysis of dense memristor crossbars for low power neuromorphic processor designs," in *Proc. Nat. Aerosp. Electron. Conf. (NAECON)*, Jun. 2015, pp. 305–311.
[38] F. Merrikh-Bayat, X. Guo, M. Klachko, M. Prezioso, K. K. Likharev, and D. B. Strukov, "High-performance mixed-signal neurocomputing with nanoscale floating-gate memory cell arrays," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 10, pp. 4782–4790, Oct. 2018.
[39] S. Kim and T.-G. Chang, "Neuromorphic hardware accelerated lane detection system," *IEICE Trans. Inf. Syst.*, vol. E100.D, no. 12, pp. 2871–2875, 2017.
[40] L. Salt, G. Indiveri, and Y. Sandamirskaya, "Obstacle avoidance with LGMD neuron: Towards a neuromorphic UAV implementation," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2017, pp. 1–4.
[41] C. Wang *et al.*, "A braitenberg vehicle based on memristive neuromorphic circuits," *Adv. Intell. Syst.*, vol. 2, no. 1, Jan. 2020, Art. no. 2070001.
[42] H. Akolkar, S. Ieng, and R. Benosman, "Real-time high speed motion prediction using fast aperture-robust event-driven visual flow," 2018, *arXiv:1811.11135*. [Online]. Available: http://arxiv.org/abs/1811.11135
[43] C. D. Schuman *et al.*, "A survey of neuromorphic computing and neural networks in hardware," 2017, *arXiv:1705.06963*. [Online]. Available: http://arxiv.org/abs/1705.06963

Akshay Ananthakrishnan received the B.E. degree (Hons.) in mechanical engineering from the Birla Institute of Technology and Science, Pilani, India, in 2013, and the M.S. degree in mechanical engineering and applied mechanics from the University of Pennsylvania, Philadelphia, PA, USA, in 2015, where he is currently pursuing the Ph.D. degree in mechanical engineering and applied mechanics.

His current research interests include the design and fabrication of neuromorphic systems and neural electrodes.

Mark G. Allen (Fellow, IEEE) received the B.A. degree in chemistry, the B.S.E. degree in chemical engineering, and the B.S.E. degree in electrical engineering from the University of Pennsylvania, Philadelphia, PA, USA, both B.S.E. in 1984, and the M.S. and Ph.D. degrees from the Massachusetts Institute of Technology, Cambridge, MA, USA, in 1986 and 1989, respectively.

In 1989, he joined the Faculty of the School of Electrical and Computer Engineering, Georgia Institute of Technology (Georgia Tech), Atlanta, GA, USA, ultimately holding the rank of Regents' Professor and the J.M. Pettit Professorship in microelectronics, as well as a joint appointment with the School of Chemical and Biomolecular Engineering. In 2013, he left Georgia Tech to become the Alfred Fitler Moore Professor of Electrical and Systems Engineering and Scientific Director of the Singh Nanotechnology Center, University of Pennsylvania, Philadelphia, PA, USA. He is a Co-Founder of multiple micro-electro-mechanical systems (MEMS) companies, including Cardiomems, Atlanta, Axion Biosystems, Atlanta, and Enachip, Jamesburg, NJ, USA. His research interests are in the development and application of new microfabrication and nanofabrication technologies, as well as MEMS.

Dr. Allen received the IEEE 2016 Daniel P. Noble Award for contributions to research and development, clinical translation, and commercialization of biomedical microsystems. He was the Co-Chair of the IEEE/ASME MEMS Conference. He was the Editor-in-Chief of the *Journal of Micromechanics and Microengineering*.