Perspective on training fully connected networks with resistive memories: Device requirements for multiple conductances of varying significance

Giorgio Cristiano,[1,2] Massimo Giordano,[1,2] Stefano Ambrogio,[1] Louis P. Romero,[1] Christina Cheng,[1] Pritish Narayanan,[1] Hsinyu Tsai,[1] Robert M. Shelby,[1] and Geoffrey W. Burr[1,a)]
[1]*IBM Research AI, IBM Research—Almaden, 650 Harry Road, San Jose, California 95120, USA*
[2]*École Polytechnique Fédérale de Lausanne, Route Cantonale, 1015 Lausanne, Switzerland*

(Received 1 June 2018; accepted 23 July 2018; published online 25 September 2018)

Novel Deep Neural Network (DNN) accelerators based on crossbar arrays of non-volatile memories (NVMs)—such as Phase-Change Memory or Resistive Memory—can implement multiply-accumulate operations in a highly parallelized fashion. In such systems, computation occurs in the analog domain at the location of weight data encoded into the conductances of the NVM devices. This allows DNN training of fully-connected layers to be performed faster and with less energy. Using a mixed-hardware-software experiment, we recently showed that by encoding each weight into four distinct physical devices—a "Most Significant Conductance" pair (MSP) and a "Least Significant Conductance" pair (LSP)—we can train DNNs to software-equivalent accuracy despite the imperfections of real analog memory devices. We surmised that, by dividing the task of updating and maintaining weight values between the two conductance pairs, this approach should significantly relax the otherwise quite stringent device requirements. In this paper, we quantify these relaxed requirements for analog memory devices exhibiting a saturating conductance response, assuming either an immediate or a delayed steep initial slope in conductance change. We discuss requirements on the LSP imposed by the "Open Loop Tuning" performed after each training example and on the MSP due to the "Closed Loop Tuning" performed periodically for weight transfer between the conductance pairs. Using simulations to evaluate the final generalization accuracy of a trained four-neuron-layer fully-connected network, we quantify the required dynamic range (as controlled by the size of the steep initial jump), the tolerable device-to-device variability in both maximum conductance and maximum conductance change, the tolerable pulse-to-pulse variability in conductance change, and the tolerable device yield, for both the LSP and MSP devices. We also investigate various Closed Loop Tuning strategies and describe the impact of the MSP/LSP approach on device endurance. *Published by AIP Publishing.* https://doi.org/10.1063/1.5042462

I. INTRODUCTION

Deep Neural Networks (DNNs) have recently allowed computers to perform many challenging recognition and classification tasks at near- or even beyond-human performance levels.[1] However, the training of such networks—the process by which "weights" in the network are iteratively adjusted in order to slowly improve DNN accuracy—can take several days, even on large graphics processing unit (GPU) clusters.

During this supervised training, known data examples are "forward propagated" through the network, generating a set of outputs which represent what the neural network "thinks" about a particular example. These generated outputs are then compared against the labels (the expected output), and any resulting errors are "backpropagated" through the network.[2] As a result, the "weights" throughout the network can each be adjusted to help the network get better at recognizing that example. As many different data examples are presented over and over and the weights adjusted adiabatically, DNNs can go beyond just memorizing the trained examples and can achieve high "generalization" accuracy on test examples the network did not get to see during training. However, since achieving such high "test" accuracies requires both large networks and large amounts of training data, training times can range from days to weeks on real-world problems, even with cutting-edge GPU-based hardware.

One way to accelerate this training process, particularly for fully-connected layers, is by using dense crossbar arrays of Non-Volatile Memory (NVM).[3–17] The conductance of NVMs can be changed by applying voltage or current pulses. Various physical mechanisms can exhibit suitable resistive switching characteristics, including the temperature-driven transition between amorphous and crystalline states in Phase-Change Memory (PCM),[18,19] reversible breakdown in Resistive Switching Memory (RRAM),[20] changes in the polarization state in Ferroelectric Field Effect Transistors,[21] or switching of magnetization in Magnetic tunnel junction or MRAM device.[22] In many cases, the conductance change is both gradual and non-volatile, making such devices suitable candidates for encoding DNN weights. (Signed weights can be encoded by either a pair of conductances or a programmable conductance and an intermediate reference current.) Since the multiply-accumulate operation at the heart of DNN computations can be implemented directly on crossbar arrays of

a)Electronic mail: gwburr@us.ibm.com. Tel.: (408) 927-1512.

NVMs in a massively parallel fashion at the location of the data,[3–5] orders-of-magnitude improvements in both speed and energy efficiency over digital architectures constrained by the "von Neumann" bottleneck between memory and computation can be achieved.[4,23–25]

However, NVMs are subject to a range of non-idealities. Some of these issues are well-known from the use of NVMs for digital data storage,[26] including limited endurance and retention, limited dynamic range, resistance drift,[27] and imperfect yield. Other issues are unique for the DNN training application, including non-linearity and variability in the conductance response (from cycle to cycle), variability in the maximum achievable conductance for a given pulse condition (from device to device), and asymmetry between weight changes in the up and down directions. In addition, non-yielding devices can impose different problems based on whether they have failed into a high conductance state ("stuck-on") or a low conductance state ("dead"). These latter issues have meant that, until recently, training DNNs using large arrays of analog NVM devices were not achieving the same high DNN accuracies that could be achieved with software-based training performed using conventional digital computers.

Recently, we proposed a novel scheme with multiple conductances of varying significance that was able to match, at least on small- and medium-size datasets, the test accuracy provided by a software simulator (TensorFlow) using conventional Von Neumann computational hardware.[25,28] (An earlier implementation described a similar approach, but with both conductance pairs implemented with the same RRAM devices.[28]) In our scheme, the weight is encoded in two pairs of devices, each of them performing different roles and thus having differing specifications. In this paper, we quantify these desired properties, and how these specifications can be traded off against each other while still achieving software-equivalent generalization or "test" accuracies.

The first part of the paper defines a general resistive device model based on the concept of a "Jump-Table" (JT), followed by a description of the Open-Loop Tuning (OLT) and Closed-Loop Tuning (CLT) used to adjust the conductances of the Least Significant Pair (LSP) and Most Significant Pair (MSP) of analog resistive memory devices. We then study the Closed Loop Tuning of the MSP and provide strategies for optimizing this process based on the device properties and the target weight. Then, we explore the impact of various device non-idealities on a Fully Connected (FC) DNN trained on the MNIST dataset of handwritten digits, showing that the multiple-conductances-of-varying-significance concept[25] relaxes the device specifications. Finally, we provide a quantitative table of updated specifications for both the LSP and MSP devices.

II. CONDUCTANCES OF VARYING SIGNIFICANCE

Figure 1 shows a FC DNN with four neuron layers [Fig. 1(a)], and its mapping onto three different crossbar arrays [Fig. 1(b)]. The arrows show the flow of information during forward propagation (from left to right) and backpropagation (from right to left). Two neighboring weights, each encoded into the difference in conductances between a pair of devices (or device elements) called G^+ and G^-, are schematically represented in Fig. 1(c). The "excitation" of each neuron is represented by a voltage $V_i(t)$ with constant amplitude and varying pulse-width. Aggregate read-current is integrated on capacitors located at the edge of the array. Note that FC layers, as used in Multi-Layer Perceptron (MLP) and recurrent network DNN topologies, are the focus of this work—these are the network layers that map most efficiently to crossbar arrays while also posing the biggest challenge for efficient implementation with digital accelerators.[5,25]

In our improved implementation[25] (see also Ref. 28), each weight is encoded with two pairs of NVMs, as shown in Fig. 2(a). The overall weight is the sum of the difference of g^+ and g^- conductances (called the Least Significant Pair, LSP) together with the difference between G^+ and G^- (called the Most Significant Pair, MSP) up-scaled by a factor $F > 1$. (Further extensions could call for even more pairs of Intermediate Significance, if desired.) During training, all weight updates are applied to the LSP, and the MSP is unchanged. A DNN may request a small increase (decrease) in a particular weight over many hundreds or even thousands of examples by applying a large number of positive (negative) update pulses and an almost equal number of negative (positive) update pulses. This necessitates LSPs offering conductance changes that are (i) linear and independent of the absolute conductance values and (ii) matched in terms of upward and downward conductance change, in order to exactly cancel out these numerous weight increase and

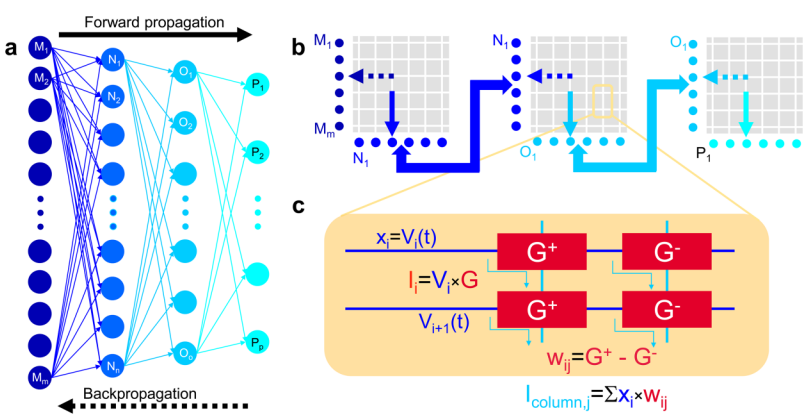

FIG. 1. (a) Schematic representation of a Fully Connected (FC) Neural Network with 4 neuron layers and 3 weight layers. (b) This network can be efficiently mapped on three NVM array-cores. Solid and dashed arrows show the direction of forward and back-propagation operations, respectively. (c) shows how a weight W is encoded in the network as the difference, $W = G^+ - G^-$, between the conductances of a pair of analog resistive memory devices, G^+ and G^-.

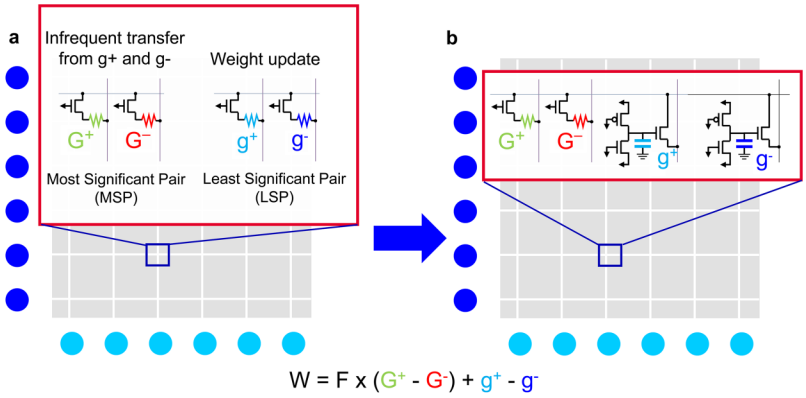

FIG. 2. (a) Synaptic weight is encoded using four resistive devices: a Most Significant Pair (MSP) of devices (G^+, G^-) and a Least Significant Pair (LSP) of devices (g^+, g^-). Read currents from the MSP are multiplied by a gain factor F, further increasing the available weight dynamic range. The LSP devices are frequently updated (using Open Loop Tuning, OLT) during training. Occasionally, the aggregate weight changes added to the LSP are completely transferred to the MSP (using Closed Loop Tuning, CLT). We recently showed a mixed-hardware-software demonstration in which the LSP devices were implemented (using SPICE simulations) with CMOS-based 3T1C structure, enabling extremely high weight update linearity and granulatity and thus higher neural network accuracies after training.[25]

decrease requests. These characteristics are difficult to achieve with today's NVM devices. For this reason, our first implementation used a 3-Transistor 1-Capacitor (3T1C) LSP structure as shown in Fig. 2(b). Our hope is that this CMOS structure can eventually be replaced by a pair of suitable NVM devices.

After every few thousand training examples, we perform a weight-transfer operation, which consists of programming the entire net weight $W = F(G^+ - G^-) + g^+ - g^-$ into the MSP. (While this was performed on all devices in our mixed hardware-software experiment,[25] it would likely be more efficient to perform this weight-transfer operation on one column at a time in a rolling fashion across each array.) The transfer process is described in Fig. 3 using the "G-diamond" formalism.[3] In the case illustrated here, training operations since the last transfer event have led to a net positive change in the LSP [$g^+ > g^-$, as represented by the projections to the respective axes, and by the pale blue and blue block arrows in Fig. 3(a)]. This net positive weight change needs to be transferred to the MSP.

Ideally, this could be performed by just increasing the G^+ conductance, leaving the G^- untouched as shown in Fig. 3(b). However, as we will see shortly, any combination of G^+ and G^- that leads to the right conductance difference encodes the target weight. The LSP may then be zeroed out, allowing it to regain its full dynamic range for further training. Note that this step favors LSP devices that can be easily programmed into the same conductance state, either by an abrupt RESET operation (present work) or by other means. For instance, for 3T1C devices,[25] the two programming transistors can be briefly configured as a transmission gate in order to rapidly charge all the capacitors to a desired target voltage.

By first copying the target weight $W = F(G^+ - G^-) + g^+ - g^-$ into peripheral circuitry and then zeroing out the LSP, the updated weight on the array $W = F(G^+ - G^-)$ can be repeatedly read and compared to the target value temporarily preserved in the periphery. Multiple write-verify or "retry" cycles allow more accurate tuning of the weight value. This is only possible because weight-transfer is performed fairly infrequently (in the present work, after every 4000 training examples). In our approach, we do not use any analog-to-digital conversion.[29] Our existing neuromorphic circuitry can also be applied for these weight transfer operations without incurring much if any additional circuit overhead. In contrast, weight updates onto the LSP during training are performed on every example and therefore require a "blind" write-without-verify procedure we refer to as Open-Loop Tuning (OLT), in order to preserve high-speed processing.

Upon completion of the Closed Loop Tuning (CLT) operation yet before the resumption of OLT, any residual error in the net weight can be reduced by similarly tuning the LSP, referred to as Post-Transfer Tuning (PTT). While this PTT operation obtains lower weight transfer error and thus preserves the trained weights better, it also tends to sacrifice some of the dynamic range of the LSP available for subsequent training.

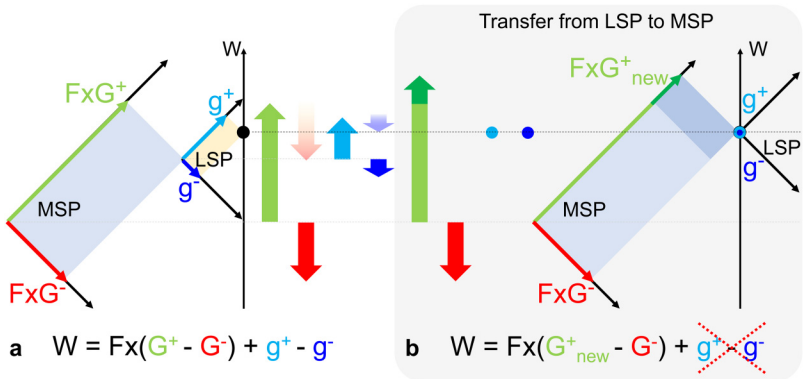

FIG. 3. (a) G-diamond representation of a positive weight with non-zero MSP and LSP components. The weight is represented on a vertical axis, while each contributing conductance is plotted on diagonal axes. The projection of each conductance onto the vertical axis indicates its contribution to the total weight (thick arrows). For "negative" conductances, the exact projection is shown, as well as a shifted copy (lighter colored arrows) to illustrate the vector sum implementing the equation shown at the bottom of the figure. Note that the same weight can be obtained with numerous different combinations of individual conductances. (b) The Closed Loop Tuning (CLT) transfer operation stores the entire weight in the MSP, leaving no weight contribution left in the LSP. Small residual errors after CLT can be compensated by additional LSP programming during Post Transfer Tuning (PTT).

III. DEVICE MODELING

From the preceding discussion, it is clear that the MSP and LSP devices have fundamentally different roles[25] and thus distinctly different device requirements. The purpose of this paper is to quantify these requirements. We present detailed simulation studies at multiple levels: for individual devices, for "basic" operations such as the transfer operation described earlier, and for full DNN network simulation. We employ the Jump-Table (JT) formalism[3] to capture the full extent of variability and stochasticity in NVM devices. An example JT is shown in Fig. 4(a). The horizontal axis shows the device conductance G^* before the programming operation, and the vertical axis shows the conductance change ΔG^* achieved upon firing one programming pulse. Both axes are represented in terms of the percentage with respect to either the maximum G value or the maximum ΔG value, since both of these values can vary from device to device. The colormap of the JT shows the cumulative probability that, given a particular initial G^*, a jump of size ΔG^* or smaller is obtained. Any variable represented with a * shows a relative or normalized value. The same variable can be transformed into absolute conductance units by multiplying the relative variable by $\langle G_{max} \rangle = 50\,\mu S$.

We describe two generic JT behaviors reminiscent of real NVM devices characterized by gradual SET transition (from low to high conductance) and abrupt RESET transition (from high to low conductance). Figure 4(a) shows the JT for a Large Initial Step (LIS) model with larger steps at small initial conductances and then decreasing-size steps at higher conductances, thus describing the saturation effects observed in many real analog memory devices (including RRAM and PCM devices). The initial step-size $\Delta G_0{}^*$ models the conductance step from the lowest (or RESET) conductance state. $\Delta G_0{}^*$ also determines the "slope" S of the JT, thus providing the degree of device non-linearity (a perfectly linear device corresponds to a flat JT). Intra-device or "pulse-to-pulse" variability is parameterized by σ^*_{intra}, which describes the write noise during device programming, e.g., the randomness of the conductance change obtained. In Fig. 4(b), we also introduce an S-shape model which is highly evocative of PCM devices previously observed in our laboratory.[3] Here, the presence of a deep PCM RESET state can lead to initially small conductance changes before the onset of steep

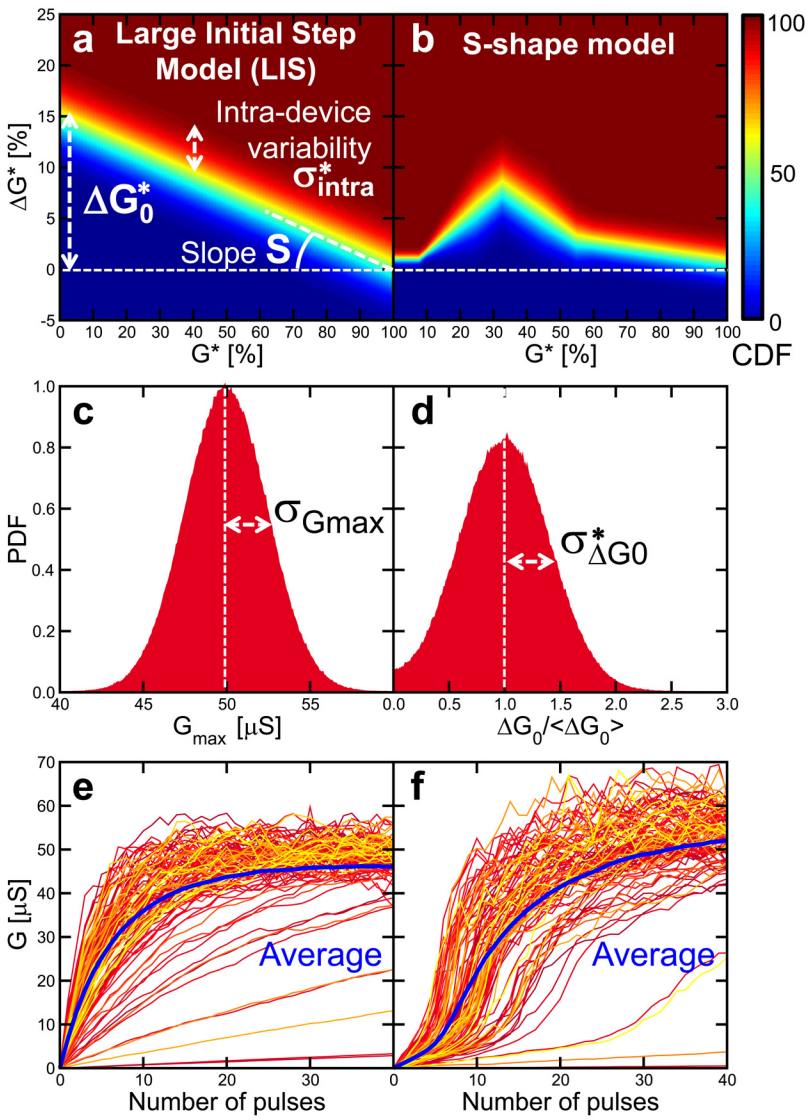

FIG. 4. Statistical model of SET transition for resistive analog-memory devices. Each "Jump-table" (JT) shows the cumulative probability that, given an initial conductance G^*, a step of size ΔG^* or smaller will occur, for Large Initial Step (a) and S-shape (b) models. The Large Initial Step (LIS) model is parameterized by the large initial step $\Delta G_0{}^*$ (here $\Delta G_0{}^* = 15\%$) from the lowest or "RESET" conductance state. Since each conductance response saturates at 100%, the value of $\Delta G_0{}^*$ can also be described as the "slope" S of the JT. Intra-device or "pulse-to-pulse" variability is captured by the transition region defined by $\sigma^*_{intra} = 2.5\%$. Inter-device or "device-to-device" variability is captured by distributions of (c) the maximum device conductance G_{max} and (d) the mean $\Delta G_0{}^*$. Shown here are $\sigma_{Gmax} = 2.5\,\mu S$ and $\sigma_{\Delta G0}{}^* = 40\%$. Example curves of device conductance G as a function of the number of programming pulses since RESET are shown for (e) the LIS and (f) S-shape models, together with the nominal behavior.

Feature	Model	Section	Related Work
MSP Programming	Closed Loop Tuning (CLT)	IV Figs. 6-10	[15,17]
LSP Programming	Open Loop Tuning (OLT)	V Figs. 11-12	[4,7,8,9,10,12,13]
Initial Step-size (Linearity)	ΔG_0	VI.A Fig. 14	[4,7,12,13]
Intra-device Variability	σ_{intra}	VI.B Fig. 15	[4,11,12,13,14,16]
Inter-device Variability	σ_{Gmax} $\sigma^*_{\Delta G0}$	VI.B Figs. 16-17	[4,14]
Endurance	Set/Reset distributions	VI.C Fig. 18	[6,16]
Faulty devices	Dead Devices	VI.D Fig. 19	[6]
	Stuck On Devices	VI.D Fig. 20	

FIG. 5. Table describing the list of device non-idealities addressed in this paper, the corresponding modeling and section, and related work by other groups.

conductance changes and finally conductance saturation. In addition to the intra-device variability modeled by the JT, additional Gaussian distributions of the maximum conductance G_{max} [Fig. 4(c)], characterized by the standard deviation σ_{Gmax}, and the maximum relative conductance jump $\Delta G_0{}^*$ [Fig. 4(d)], defined by $\sigma_{\Delta G0}{}^*$, are used to capture the inter-device or "device-to-device" variability observed in real devices.

In all of the studies performed in this paper, the RESET operation is assumed to be abrupt. For filamentary RRAM, for which the SET operation is abrupt and only RESET is gradual, these results can be directly adapted by swapping the names "SET" and "RESET" and by considering motion across the G-diamonds in the opposite direction (from right-to-left rather than the left-to-right motion representative of the PCM-like devices studied here). A truly bidirectional

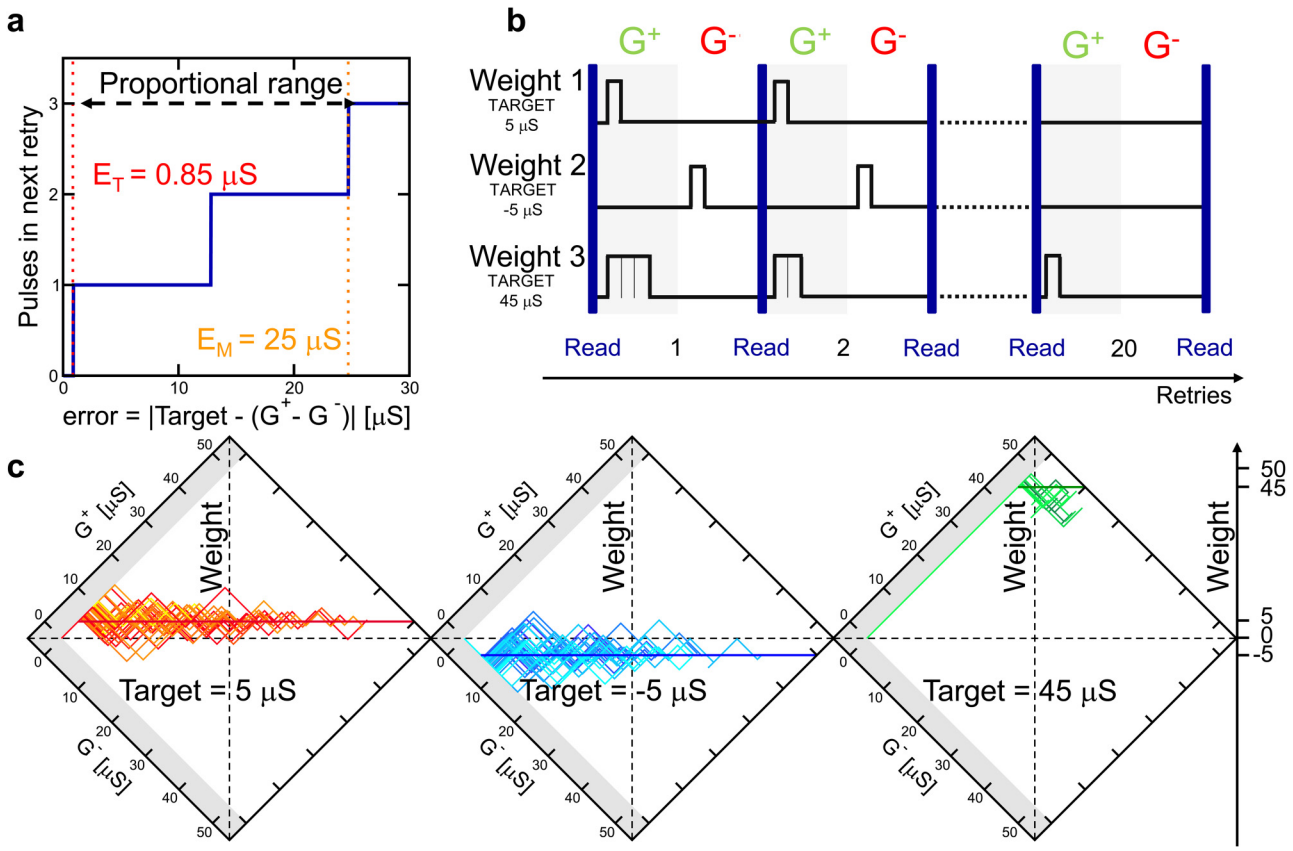

FIG. 6. Description of Closed-Loop Tuning. (a) shows how the number of pulses to be fired during any one retry is chosen depending on the magnitude of the error between the measured weight ($G^+ - G^-$) and the target weight value. If the CLT error is smaller than the tolerance threshold E_T, no pulses are fired. For errors larger than E_M, three pulses are always fired. (The threshold for firing two pulses is always at the average between E_T and E_M.) (b) Schematic description of CLT: during each retry, the entire column of weights is read, compared to the target value stored in the periphery, and for each row, either G^+ or G^- is programmed accordingly. (c) *G*-diamond representations of CLT (assuming LIS devices as described in Fig. 4) for weight targets of 5, −5, and 45 μS, each showing 50 simulated CLT trajectories.

analog memory might require some modifications to our analysis and the tuning procedures, but the overall conclusions for which specifications are critical should still be mostly valid.

Monte Carlo simulations of conductance vs. number of pulses for these two models are shown in Figs. 4(e) and 4(f). The nominal behavior is also shown for clarity. To summarize the device aspects that will be studied in the remaining of the paper, Fig. 5 provides a list of non-idealities, describing how each is modeled or represented, and listing the related section in the paper where these results are discussed. Some references to related work by other groups are also provided.

IV. CLOSED-LOOP TUNING

The programming of MSP devices during transfer is performed with the Closed Loop Tuning (CLT) procedure described in Fig. 6. For simplicity, we will refer here to the weight as $W = G^+ - G^-$, omitting the value of F (since it appears in both the target and the currently measured read current) and of g^+ and g^- (since the LSP values are assumed to have been RESET to equal values at the beginning of the CLT process). The CLT process begins with two initial reads of the current weight value before and after the LSP contributions have been removed. The former is the desired weight target that we wish to program the MSP to, and the latter is the current weight stored in the MSP pair. The difference between these is referred to below as the "error." One or the other of G^+ or G^- is then programmed with a small number of pulses proportional to the magnitude of the error, as shown in Fig. 6(a). After each programming attempt, the error is recalculated by issuing another read. In the case that the magnitude of the error falls below a confidence threshold, E_T, no pulses are fired. For all errors with magnitude larger than a maximum error, E_M, three pulses are fired. The sign of the weight error determines which device (G^+ or G^-) is programmed.

Every programming retry is divided into time intervals dedicated for reads and for programming operations on G^+ and G^-, with the constraint that every weight can only be programmed on G^+ or G^- in a given retry but not both [Fig. 6(b)]. This allows the CLT operation to take place in parallel on an entire column of devices within a large device array. CLT operations continue until all errors are below E_T or until a maximum number of retries are reached. We typically allow 20 retries unless otherwise indicated.

Figure 6(c) shows three examples of CLT performed on weights with both G^+ and G^- initially in a RESET state, each showing 50 simulated trajectories plotted on a G-diamond. Since the objective is to tune the weight—not the individual conductance values—to first order, any point along the horizontal line across the G-diamond representing the target weight is equally suitable. Since we assume PCM-like behavior where only conductance increases are gradual, we start with a full abrupt RESET of both MSP devices (lefthand corner of the G-diamond) in order to maximize the capabilities of the left-to-right motion across the G-diamond induced by partial SET operations. Targets of +5 and $-5\,\mu S$ show similar behavior since the weight is symmetric for positive and negative targets. In contrast, a larger target of $45\,\mu S$ calls for a strong initial increase of G^+, followed by small adjustments to both G^+ and G^-.

To gain deeper insight, Figs. 7(a) and 7(b) report the statistical distribution of retries that synapses, starting from

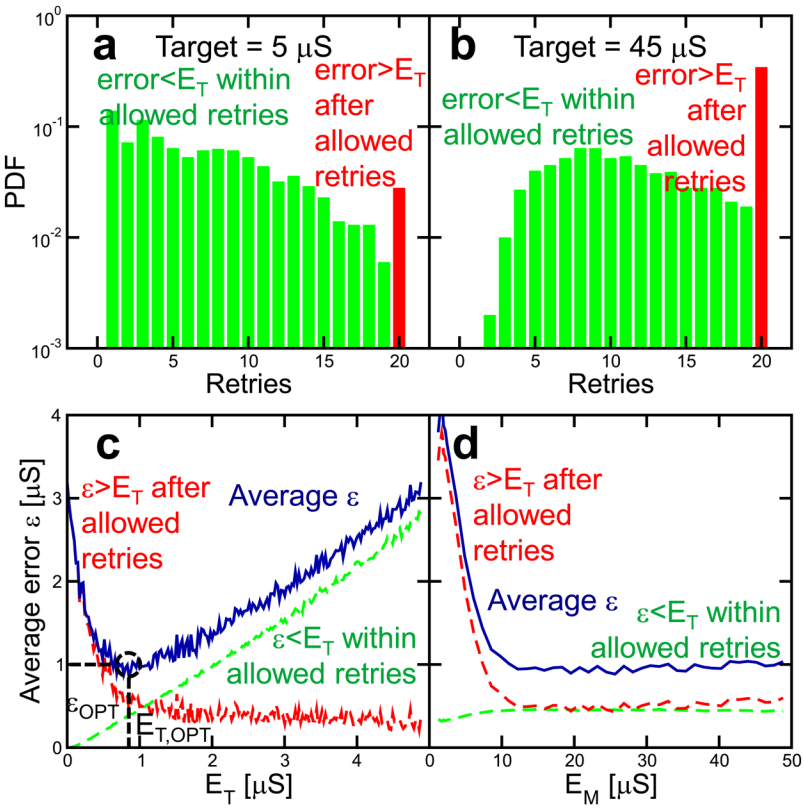

FIG. 7. Statistics of the number of CLT retries needed (assuming LIS devices as in Fig. 4, starting with both G^+ and G^- in the low-conductance RESET state), for targets of (a) $5\,\mu S$ and (b) $45\,\mu S$. Here, $E_T = 0.85\,\mu S$ and $E_M = 25\,\mu S$. Green bars show the number of devices that reach the confidence level within the maximum number of retries allowed, while red bars show how many devices have an error still larger in magnitude than E_T after 20 retries. The average error ϵ obtained after CLT—the sum of the error within allowed retries and the error after allowed retries, for 100 000 simulated trajectories uniformly distributed across the range of target weights—is shown (c) as a function of E_T and (d) as a function of E_M. The minimum error, ϵ_{OPT}, is achieved at the best choice of threshold, $E_{T,OPT}$.

both G^+ and G^- in RESET state, require to reach targets of 5 and 45 μS, respectively (using 10 000 simulated CLT trajectories for each target weight value). The green bars represent weights which manage to achieve an error smaller in magnitude than the confidence threshold E_T within the maximum number of retries (20), while the red bar represents those weights which have still not yet succeeded in reaching an error smaller than E_T, even after the last retry. Higher-conductance targets tend to lead to a larger number of retries, since more steps are needed to program the associated device (G^+ for positive weights, G^- for negative weights) from the initially low-conductance RESET state to higher values. After CLT, the residual difference between the target weight and the actual weight represents the CLT error.

We characterize this error in terms of the average CLT error ϵ, across 100 000 simulated trajectories uniformly distributed across all possible target weights between $-50\,\mu S$ and $+50\,\mu S$. Figure 7(c) plots this error ϵ as a function of the confidence threshold E_T, showing that ϵ is the sum of two components. The first component is the error due to weights which manage to reach the target within a threshold E_T [green curve in Fig. 7(c), green bars in Figs. 7(a) and 7(b)]. This error increases linearly with E_T, since a larger E_T tolerates larger errors by terminating retries earlier. The second contribution comes from weights which have still not reached the threshold E_T after 20 retries [red curve in Fig. 7(c), red bars in Figs. 7(a) and 7(b)]. Here, small E_T can lead to large errors, because getting sufficiently close to the target weight within 20 retries becomes increasingly unlikely. The combination of both these components define an optimal choice of confidence threshold, $E_{T,OPT}$, at which ϵ reaches its minimum value, ϵ_{OPT}.

Figure 7(d) shows the dependence of ϵ on the outer threshold E_M, the error threshold above which three programming pulses are applied. (Note that the threshold at which two programming pulses are fired is always halfway between E_T and E_M.) Very small E_M values cause a large error since, even for cases in which the error is small and the weight is close to the target, the CLT procedure calls for three pulses [as shown in Fig. 6(a)] instead of one, forcing a large change in the weight. Larger E_M values do not substantially affect ϵ, however. Excessively large E_M can cause an increase of ϵ measured at the maximum allowed number of retries, since the number of applied pulses may not be enough to reach the highest conductances. This should pose an upper limit on the choice of E_M, but at very high values: even for the largest value shown in Fig. 7(d), this effect is still extremely subtle.

A. Design choices for Closed-Loop Tuning

The previous discussion shows that the CLT procedure involves many design choices. In this section, we explore the impact of the available parameters on ϵ. While the value of E_M does not provide particular challenges as long as E_M is not too small [Fig. 7(d)], the choice of E_T must be made carefully. Figure 8(a) shows the correlation between the optimal threshold $E_{T,OPT}$ and the pulse-to-pulse variability σ^*_{intra} introduced in Fig. 4(a). Since that parameter is in units of percentage, here we use $\sigma_{intra} = \sigma^*_{intra} \cdot \langle G_{max} \rangle$ to discuss the the pulse-to-pulse variability in units of conductance applicable to a device with a particular maximum conductance, $\langle G_{max} \rangle = 50\,\mu S$.

Figure 8(a) clearly shows a linear dependence:

$$E_{T,OPT} = \frac{\sigma_{intra}}{2}, \tag{1}$$

relating the optimal choice of confidence threshold, $E_{T,OPT}$, and the pulse-to-pulse variability, σ_{intra}. Since the device has a maximum precision inversely proportional to σ_{intra}, the confidence interval, spanning $2E_T$ to cover errors of both positive and negative sign, is optimum when equal to σ_{intra}, leading to Eq. (1).

This relationship holds for most reasonable values of ΔG_0. The inset in Fig. 8(a) shows the fitting-error between the actual $E_{T,OPT}$ observed after simulated CLT trajectories [similar to the blue line in Fig. 8(a)] and the expectations of Eq. (1), for different initial steps ΔG_0. The curve is mainly flat, confirming Eq. (1) as a simple yet useful relationship, except for very small values of ΔG_0, which provide too little conductance change to reach larger weight targets within 20 retries.

Another trade-off comes from the choice of the number of iterations during the CLT procedure. Figure 8(b) shows that ϵ_{OPT} strongly decreases for increasing number of iterations, but it cannot go below the limit imposed by $\sigma_{intra}/2$ due to intrinsic pulse-to-pulse variability. As a result, a larger number of iterations cannot help reduce ϵ_{OPT}. Instead, the use of a large number of retries extends the required duration of the CLT procedure and also causes the average final position of G^+ and G^- within the G-diamond to be farther to the right. The resulting higher read current (shown here for a read voltage $V_{read} = 0.2$ V) increases read power consumption and could drive the aggregate read current to levels that could trigger electromigration.

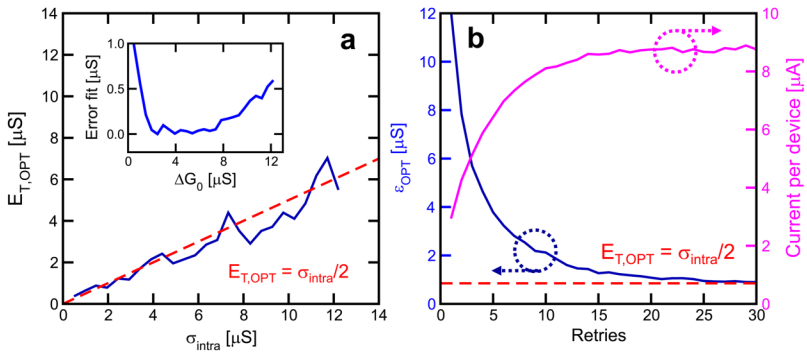

FIG. 8. (a) Optimal confidence threshold, $E_{T,OPT}$, as a function of σ_{intra} (using LIS devices as in Fig. 4). The dashed line suggests that the optimal confidence $E_{T,OPT}$ corresponds to $\sigma_{intra}/2$. Inset shows the fitting-error for $E_{T,OPT} = \sigma_{intra}/2$ as a function of the initial conductance jump, ΔG_0, showing that this linear approximation holds well except for very small values of ΔG_0. (b) Average error (blue line) resulting from CLT as a function of the number of allowed retries, when using the optimal choice of $E_{T,OPT}$. The pink line shows the average read current per device obtained by reading G^+ and G^- with a read voltage $V_{read} = 0.2$ V.

B. Impact of single device programming

In previous sections, both G^+ and G^- were used during CLT for weight tuning. However, another choice would be to RESET both G^+ and G^- and then use only one of these devices to reach the target. This "uncoupled" procedure has the advantage of reducing both circuit overhead and power consumption, since one of the two devices does not need to be programmed. Figure 9(a) shows ϵ_{OPT} as a function of the target weight for "uncoupled" programming operations (blue curve) performed on one device (G^+ for positive targets, G^- for negative targets), as well as "coupled" programming operation (red curve) using both G^+ and G^-, assuming an LIS device as in Fig. 4. Since coupled CLT causes both G^+ and G^- to reach large conductances where each step is small due to conductance saturation, the resulting increased precision in CLT causes ϵ_{OPT} to be generally smaller for coupled programming. In contrast, uncoupled programming cannot leverage this effect, leading to higher ϵ_{OPT}.

However, for weight targets above $40\,\mu S$, the roles are inverted. Now, conductance steps are smaller for uncoupled programming, since the primary conductance (G^+ for positive weights, G^- for negative weights) is almost saturated. In contrast, as soon as the CLT procedure calls for a small correction on G^- in a low conductance state, coupled programming invariably increases ϵ_{OPT} due to the large G^- initial step, causing a large error. Successive CLT retries cannot compensate for this error, since G^+ cannot be further increased due to device saturation. In this case, based on the desired target, two programming conditions can be implemented as shown in Fig. 9(c): coupled programming for small and medium targets and uncoupled programming for large targets. Figure 9(b) shows the behavior for an S-shaped device. In this case, there is no intersection of the programming curves so that coupled programming represents the best solution for all target weight values.

C. Impact of initial RESET during transfer

The previous sections assumed that the first step during transfer was to always perform a RESET operation on both G^+ and G^-. This makes the entire dynamic range of both devices available, leading to a more accurate CLT. However, this also implies that device endurance is degraded, since endurance strongly depends on RESET transitions. For this reason, reducing the RESET events would be desired to ensure a longer life-time of the devices. Figure 10(a) shows ϵ_{OPT} as a function of the number of transfers during DNN training. In the first case, both G^+ and G^- undergo a RESET operation at every transfer, giving a constant ϵ_{OPT} over successive transfers. In the second case, G^+ and G^- are never programmed in a RESET state, as shown in Fig. 10(b). Considering an initial RESET condition, during the first two transfers ϵ_{OPT} is equivalent for both conditions, since conductances are not yet saturated. However, successive transfers cause conductance saturation, increasing the weight-transfer errors for the case without RESET at every transfer. Therefore, to provide an acceptable trade-off between CLT accuracy and device endurance, device RESET should probably be performed at least every 2-3 transfers. A third option is to perform a RESET only for weights where $|W - W_{target}| > E_{T,OPT}$, where W is the weight before transfer and W_{target} is the weight which must be programmed during transfer, Fig. 10(c). Since the programming precision cannot go below ϵ_{OPT}, reprogramming such weights shows no advantages, as described by the green and blue curves in

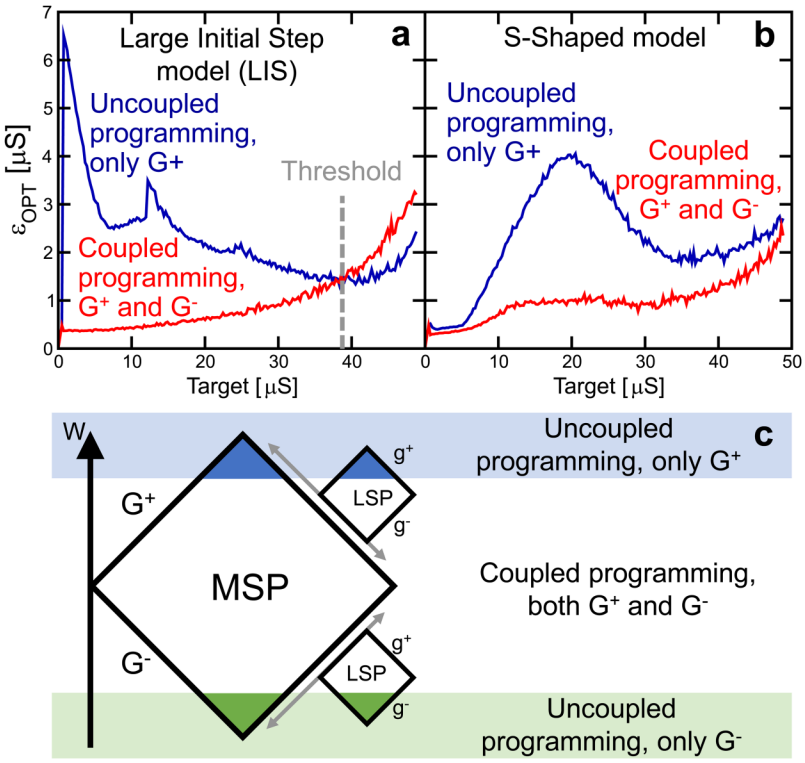

FIG. 9. (a) Best possible MSP weight error after CLT, ϵ_{OPT}, as a function of the target to be encoded in the MSP (assuming LIS devices as in Fig. 4). The red line corresponds to "coupled" programming operation using both G^+ and G^-, while the blue line corresponds to "uncoupled" programming where only G^+ (for positive weights) or G^- (for negative weights) is changed. (b) Similar plot but for devices following an S-shaped JT. (c) shows the different regimes encountered during weight programming of a LIS device, showing the region where "coupled" programming is preferred (small- and medium-weights) as opposed to "uncoupled" programming (large magnitude weights). The exact thresholds between these regions can be expected to scale with the size of the large initial jump.

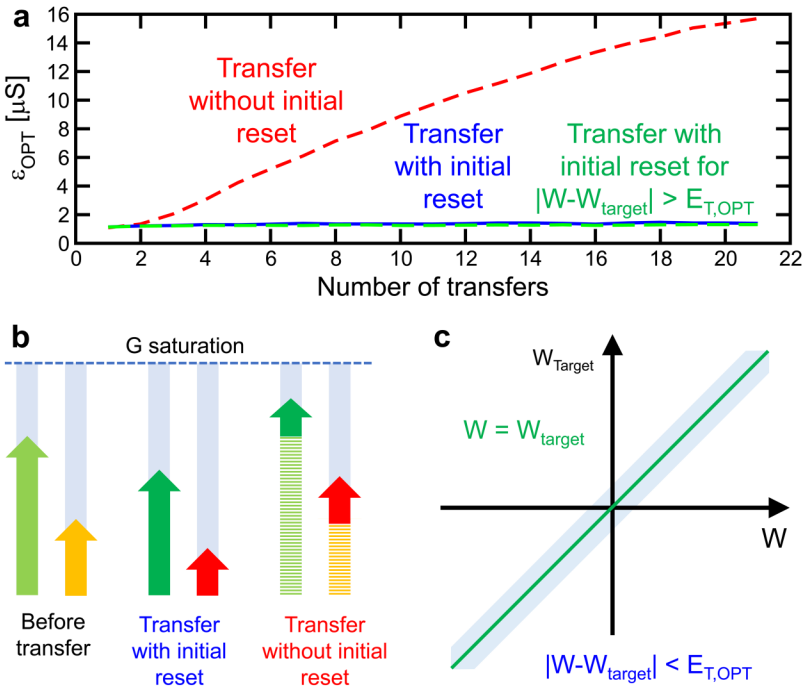

FIG. 10. (a) ϵ_{OPT} as a function of the number of transfers between RESET operations, using LIS devices as in Fig. 4. Three cases are described, all starting in an initial low-conductance RESET state. As described in (b), either G^+ and G^- are RESET before every transfer (blue solid line), or G^+ and G^- are never RESET (red dashed line). In the latter case, the error increases rapidly due to saturation of both G^+ and G^-. Thus, RESET operations need to be applied on every transfer or after every few transfers. Weights which are already close to the target weight W_{target} within $E_{T,OPT}$ can be left unchanged (c), providing the same ϵ_{OPT} as with full RESET and programming.

Fig. 10(a) being overlapped. In this third case, which is also the one used in the rest of the paper, devices are less degraded, since RESET operations are reduced, while still maintaining the minimum ϵ_{OPT}. This design choice is also applied during Post Transfer Tuning so that any transfer error smaller than $E_{T,OPT}$ is discarded, speeding up the tuning procedure.

V. THE CHALLENGE WITH OPEN LOOP TUNING

The ability to train DNNs to high accuracy depends strongly on the linearity of the devices receiving the weight update.[4,30] Typically, the weights exhibit an oscillatory behavior during training, with slow changes emerging over hundreds or thousands of training examples. Each weight tends to receive many update requests which almost cancel each other. For this reason, it is crucial that weight updates be symmetric so that these opposing requests can cancel each other. If this does not happen, the weight is then biased towards higher or lower conductances, leading to saturation and preventing the network from further update. If weights are updated by bidirectional programming of a single conductance (with the other one a shared reference current), then a nonlinear but symmetric conductance update—where the derivative of conductance with respect to applied pulses was symmetric in magnitude for all conductance values—would be acceptable. When 3T1C structures are used as the LSP device, the charge added to the capacitor by the CMOS devices is highly linear, with a nearly flat change in voltage as a function of the capacitor voltage.[25]

In the present case, where each weight is tuned by increasing the conductance of one of the two devices in the LSP, the best scenario is when the size of the conductance change is independent of the conductance value, e.g., linear. However, directly relating device properties and neural network accuracy is not a simple task, since the backpropagation algorithm tends to counteract device non-linearity by adjusting the weight updates it requests. This introduces a complicated feedback loop between device properties, network size, topology, and the selected dataset.

As shown in Fig. 11(a), the Training Loop [Fig. 11(a)] involves forward inference of neuron excitations, reverse propagation of backpropagated deltas, and then weight-update based on those excitations and delta values. Fine-tuning the parameters involved in training for maximum accuracy can be represented as a second, "Optimization" Loop [Fig. 11(b)]. The single device conductance step ΔG directly influences the learning rate adopted during weight update. This defines an average weight step which broadens the distribution of the neuron excitations, and thus the aggregate excitation applied to the neuron activation function (such as the *logistic*, *tanh* or *ReLU* functions). This input distribution also depends on the number of weights contributing into each hidden neuron, which scales with the size of the neuron layers. Finally, all these factors affect the overall backpropagated errors, which also depend on the number of neuron layers.

We can perform a simple analysis of the impact of device properties by observing weight updates through the combination of Open Loop Tuning (OLT) and weight transfer with CLT and PTT. Here, we use a sequence of pulses with a known number of weight-increase and -decrease requests, generated by random permutation rather than by the backpropagation algorithm. As such, device response that markedly deviates from the intended weight changes are not counteracted by feedback from the backpropagation algorithm (which would be present during normal neural network training).

Figure 12 describes the impact of such random pulse-sequences containing 20 positive and 30 negative pulse

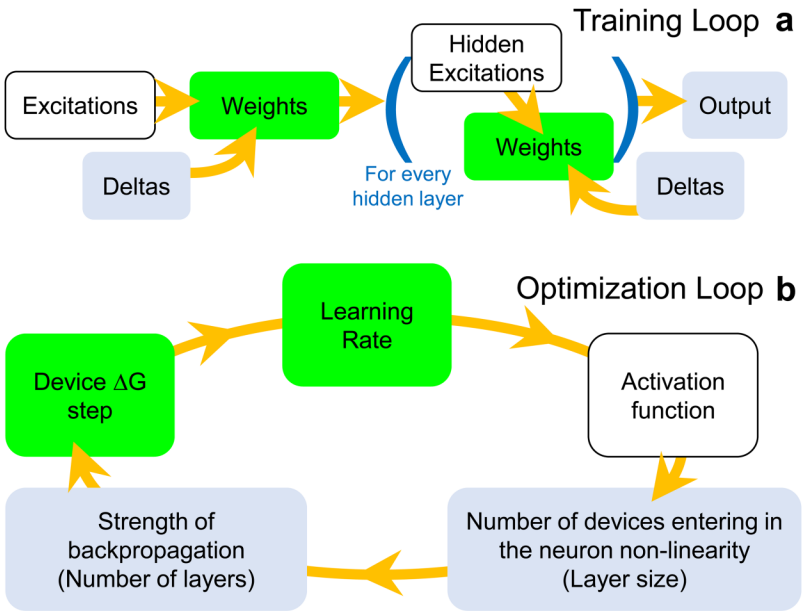

FIG. 11. (a) Training Loop during back-propagation and (b) corresponding Optimization Loop to improve the training performance. The changes on each weight, directly dependent upon the device ΔG step and learning rate, then affect the behavior of the network at different levels, through the neuron non-linearity, the layer size, and the number of network layers.

requests, applied to a synapse comprised of an identical MSP and LSP using LIS devices described as in Fig. 4. A single weight-transfer event is triggered after 4000 training examples, programming MSP devices with a CLT procedure. This allows us to incorporate the polarity inversion technique,[25] which strongly suppresses the negative effects of inter-device variability in the LSP by swapping the roles of g^+ and g^- during each weight-transfer event. As a result, a weight which was originally expressed as $W = F(G^+ - G^-) + g^+ - g^-$ is now treated as $W = F(G^+ - G^-) - (g^- - g^+)$, for both read and OLT write events. Since each individual LSP device alternates between serving in the role of g^+ and g^-, any fixed asymmetry between the two devices in a given LSP tends to cancel out. If weight-increases get favored in

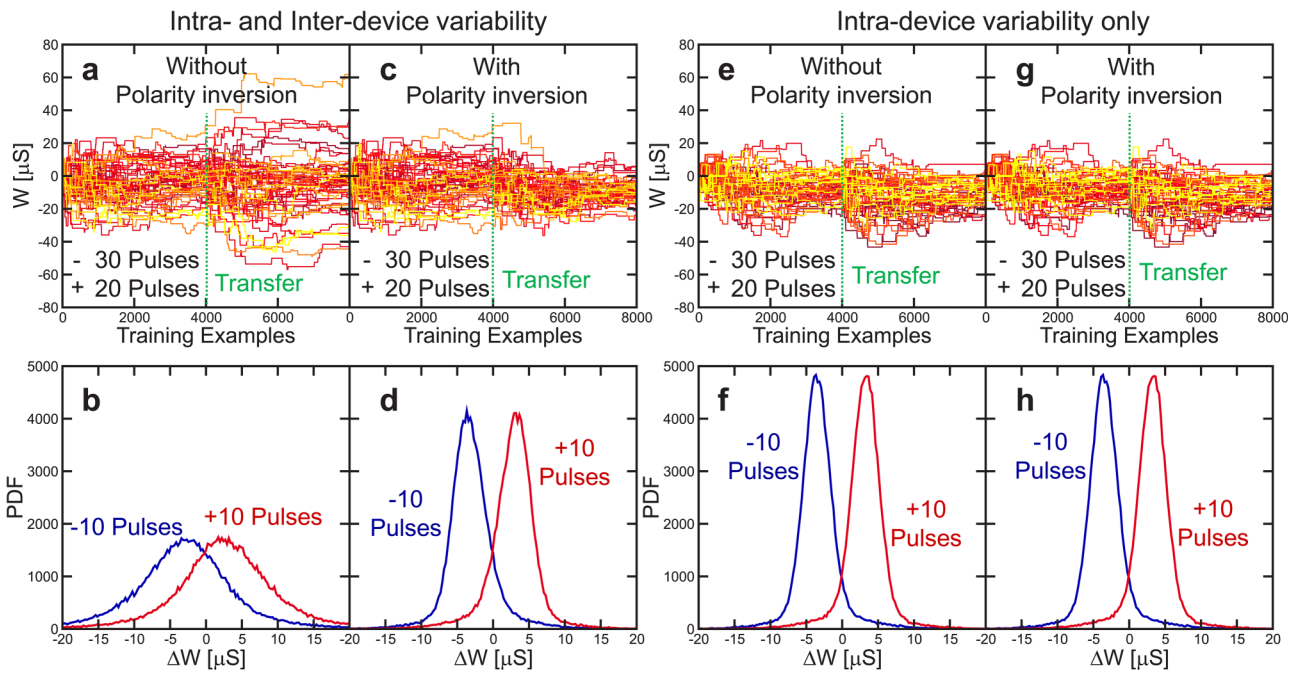

FIG. 12. Simulated weight response for random, pre-synthesized weight-update pulse sequences, similar to those seen during OLT except without the feedback that the backpropagation algorithm would introduce. Both MSP and LSP assume LIS devices as in Fig. 4. (a) shows 50 conductance trajectories updated with 50 LSP programming pulses, 20 up and 30 down, randomly permutated and distributed over 8000 training examples, where transfer onto the MSP and post-transfer tuning occur after 4000 examples. (b) shows the distributions (over 10^5 weight trajectories) of the overall weight update $\Delta W = W(8000) - W(0)$. Initial conditions are randomly distributed across the entire dynamic range of the MSP, with 30 pulses on G^+ and 20 on G^- for the red curve and the opposite for the blue curve. Polarity inversion [(c) and (d)], performed as part of the CLT after example #4000, clearly mitigates the distribution broadening by reducing the impact of inter-device (device-to-device) variability. In contrast, when only intra-device (pulse-to-pulse) variability is present [(e), (f), (g), and (h)], polarity inversion cannot improve the weight-update trajectories.

one transfer interval, that same fixed asymmetry will then tend to create weight-decreases over the next transfer interval, thus cancelling out on average. One of the key features that enables polarity inversion is the fact that the CLT process transfers all of the weight information from the LSP into the MSP. It is at exactly this instant that the polarity of the LSP can easily be inverted, not just for one device but for an entire column of devices.

Figure 12(a) shows the weight-update trajectories for the case without polarity inversion, revealing a strong broadening of curves. This broadening is also noticeable in the histograms in Fig. 12(b), showing the distribution of $\Delta W = W(8000) - W(0)$. These histograms not only consider weight-trajectores that start from W=0 as shown in Fig. 12(a) but also take into account all possible initial weight conditions from $-50\,\mu S$ to $50\,\mu S$. The application of polarity inversion strongly reduces the broadening of both the weight-trajectories and the resulting weight-change distributions [Figs. 12(c) and 12(d)], by cancelling the inter-device fixed asymmetries between LSP devices. However, polarity inversion provides no correction for problems introduced by intra-device or pulse-to-pulse variability, as shown by the very similar weight-update trajectories shown in Figs. 12(e) and 12(f) and Figs. 12(g) and 12(h), without and with polarity inversion, respectively. In this case, the random variability is introduced as each programming pulse is applied, leading to variable conductance updates as described by the Jump-Table transition region of width σ_{intra}. Swapping the roles of the two LSP devices cannot cancel out this kind of variability. Polarity inversion is thus a technique which helps make neural network training much more tolerant to significant device-to-device variability for the LSP devices, but not more tolerant to pulse-to-pulse variability.

In general, tight ΔW distributions such as those shown in Fig. 12(d) are correlated with high DNN training accuracy, since this shows that the weight-changes actually implemented tend to closely follow the requested weight-changes. In contrast, broad or overlapping distributions of ΔW suggest that a device may not offer precise enough weight-changes to support high training accuracy. However, only DNN training studies can quantitatively establish if a device is suitable for high-accuracy hardware training, so we turn to such studies in Sec. VI.

VI. IMPACT ON TRAINING ACCURACY

Our description of the programming methods has already made clear that for the OLT weight-update requests applied to the devices within the LSP, the most important device criteria are the linearity and granularity (large number of steps) of the conductance update. In contrast, while the devices within the MSP must provide an accurate CLT, aspects such as conductance update linearity are not immediately relevant. The two pairs of devices also support each other: the presence of the MSP allows polarity inversion to reduce the impact of device-to-device variability on the LSP devices, while the presence of Post Transfer Tuning allows the LSP to compensate for sloppiness in the CLT weight-transfer into the MSP.

Section VI A quantifies the impact of the various device non-idealities on the accuracy of training of a FC DNN, which is composed of four neuron layers with size 528-250-125-10 as shown in Fig. 1(a). The weights are described with MSP and LSP modeled with LIS devices. We used the MNIST dataset of handwritten digits[31] to train the network. The purpose of this section is to define the set of properties that enable a synapse with feasible MSP and LSP devices to match a software implementation of the training of a neural network (of the same size, trained on the same MNIST dataset). From that baseline, we explore the impact of each non-ideality, one at a time, to estimate the maximum tolerable non-ideality.

To obtain software equivalent accuracy, we assume two significantly different pairs of LIS devices, with corresponding Jump-Tables shown in Fig. 13(a) for MSP, and in Fig. 13(b) for LSP. The $\Delta G_0{}^*$ and $\sigma_{intra}{}^*$ parameters were chosen (as shown in the figure) in order to remain achievable with real memory devices yet still match the generalization accuracy after training with a software implementation. Figure 13(c) shows the resulting training and test accuracies during 20 epochs of MNIST training, compared with different runs of Tensorflow simulations on a network of the same size. All successive simulations will plot the final generalization accuracy after 20 epochs, compared to the median software test accuracy of 98%. The acceptable tolerance below this median accuracy, based on the spread of 20 Tensorflow runs initiated with different random seeds, is roughly a 0.5% accuracy loss. This target accuracy region is indicated by the light blue region in Fig. 13 and on all subsequent plots.

The significant differences between the two LIS devices shown in Fig. 13 already foreshadow some of the important conclusions to follow: the MSP needs to offer large steps, while the LSP should offer much smaller conductance steps. These observations are roughly independent of the choice of F, which in all studies is set to F=3.0.

A. Dynamic range

We first analyze the impact of the dynamic range of both kinds of devices, as impacted by the granularity and linearity of conductance change of each device. An ideal, perfect device would show a linear behavior, which corresponds to a horizontal JT. This would imply that, irrespective of the initial conductance G, the device would show a conductance step ΔG with constant magnitude. However, real resistive devices always show a maximum conductance saturation and a general decrease of ΔG for higher G, which translates into a JT with median ΔG value that reaches zero at the maximum conductance, as shown in Fig. 14(a). We then varied the slope of this JT median, by changing the initial step ΔG_0, first on MSP, Fig. 14(a), with the baseline LSP described by the JT of Fig. 13(b). Final test accuracy results after 20 epochs of MNIST training are shown in Fig. 14(b) as a function of ΔG_0 (or equivalently, as a function of the slope of the JT). This increase in the JT slope reduces the number of available steps that the LSP can take, starting from its lowest conductance state, before exceeding some higher conductance state. Marked on the figure are the number of

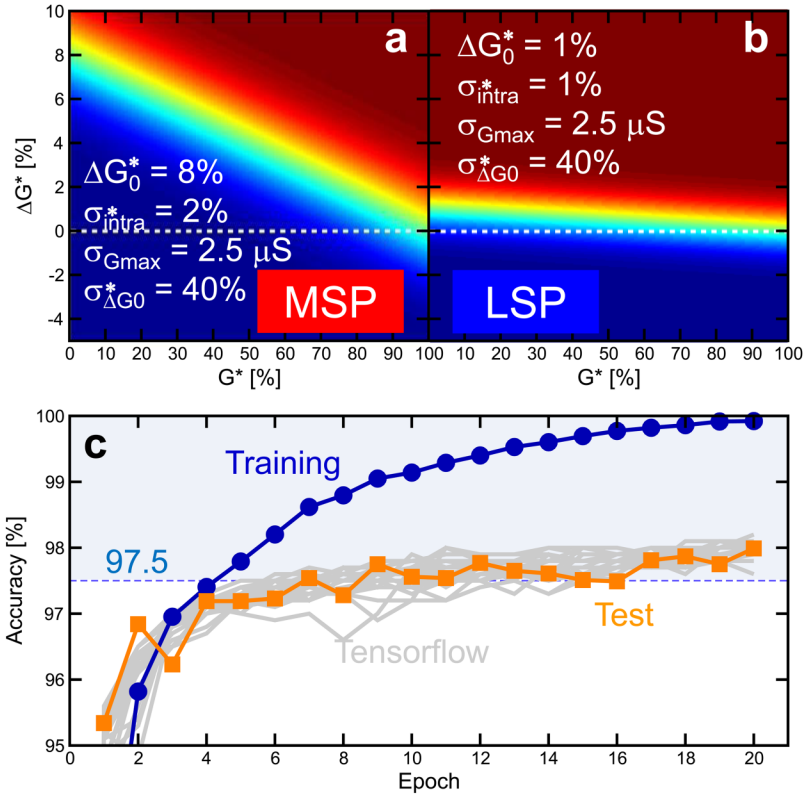

FIG. 13. To match software accuracy, we assume two different types of LIS devices with different Jump-Table parameters for MSP (a) and LSP (b). The corresponding training and test accuracies are shown in (c), revealing a close agreement with 20 Tensorflow runs.[25]

steps, using the median line of the JT, between the 0% and 95% conductance states.

The MSP exhibits a relatively large resilience to the exact ΔG_0 value, since the MSP is only programmed during transfer with a CLT procedure that is then followed by Post Transfer Tuning. As discussed earlier, the CLT procedure provides accurate results if, for increasing number of retries, the MSP devices are able to provide smaller and smaller ΔG steps. This is naturally obtained with a JT with decreasing median for increasing conductance, thus providing relaxed dynamic range constraints on the device. Note that for very small slopes, accuracy decreases sharply. At this point, ΔG steps have become so small that the CLT procedure described in Fig. 6 cannot reliably achieve larger weight values within the maximum number of allowed retries. Note that for such MSP devices offering very small conductance steps, it might be appropriate to redesign the CLT procedure so that the MSP devices were only RESET when absolutely necessary.

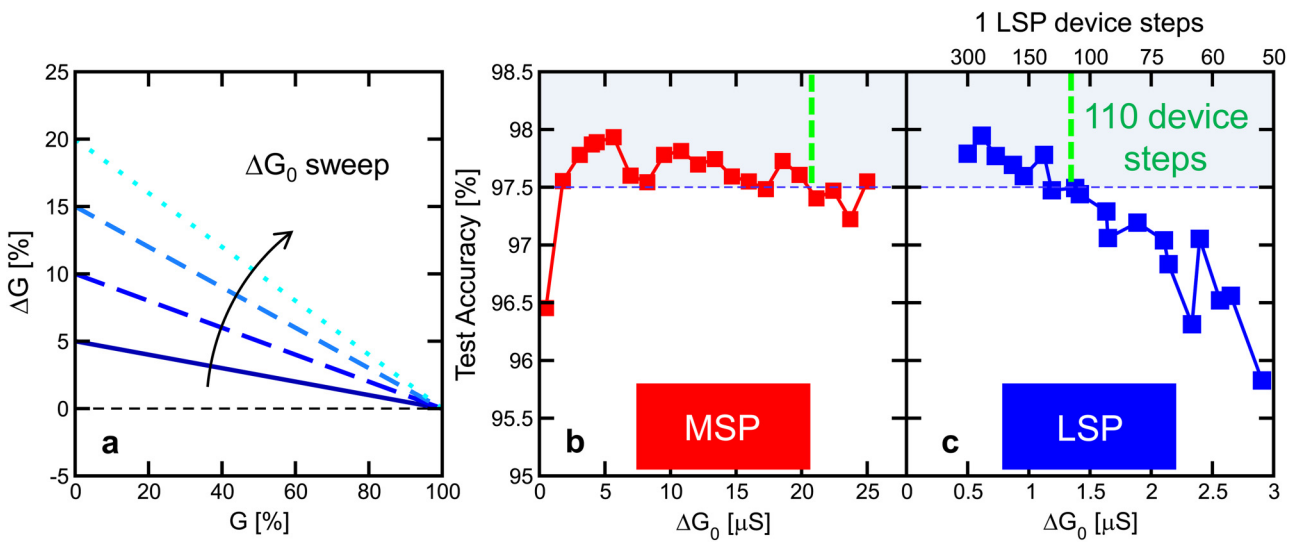

FIG. 14. Impact of the size of the initial step ΔG_0 (a) of MSP or LSP on training accuracy of a 528-250-125-10 neurons FC DNN. (b) Accuracy as a function of MSP ΔG_0, considering LSP as in Fig. 13(b). Results show a weak dependence on MSP dynamic range, since MSP only changes with CLT and errors left after CLT are then corrected with PTT. (c) Accuracy as a function of LSP ΔG_0, considering MSP as in Fig. 13(a). Results show a larger sensitivity to LSP linearity. The minimum number of acceptable LSP steps is ~110.

This would allow a highly granular MSP to copy the small weight changes from the LSP within a reasonable number of steps.

Figure 14(c) shows the change in accuracy as a function of the dynamic range of the LSP devices, plotted as a function of the initial jump ΔG_0 of the LSP (e.g., its JT slope), assuming the baseline MSP devices [Fig. 13(a)]. In this case, the network is much more sensitive to loss of dynamic range, and ~110 steps are required in order to achieve acceptable generalization acuracy. However, this represents a considerable improvement over the original single-pair RPU concept, which required 1000 conductance steps from minimum to maximum.[4] Thus, the MSP + LSP structure significantly relaxes the requirements on each individual device. This comes about because the n_{lsp}=110 steps across the dynamic range of the LSP only represent a part of the entire dynamic range for the synapse. The maximum positive weight is $F(G_{max} - G_{min}) + (g_{max} - g_{min})$, and the effective stepsize of the LSP is $(g_{max} - g_{min})/n_{lsp}$. For simplicity, we can assume that $G_{min} \sim g_{min} \sim 0$, and thus the total dynamic range of the synapse is

$$2\left(F\frac{G_{max}}{g_{max}} + 1\right)n_{lsp} \sim 2(F+1)n_{lsp}, \tag{2}$$

where the factor of 2 comes from spanning both positive and negative weights, and where we have further assumed that the maximum conductances of the individual MSP and LSP devices (e.g., before applying the gain factor F to the read currents) are similar. Thus, the transition shown in Fig. 14(c) for $n_{lsp} \sim 110$ corresponds to an effective total synaptic dynamic range of 880—almost exactly the original device specification.[4] Note that there is a second-order impact of the dynamic range of the MSP on the choice of F. A large F can be chosen only if the MSP device can be trusted to get close enough to the target weight that the LSP can correct this error and still have enough dynamic range available for subsequent training operations.

B. Variability

Variability represents an important issue which can degrade the tuning precision in both CLT and OLT, thus affecting both MSP and LSP conductance update. As shown in Fig. 4, intra-device or pulse-to-pulse variability, schematically represented by σ_{intra}, and inter-device or device-to-device variability, described through G_{max} and ΔG_{max} distributions in Fig. 4(c) and 4(d), can be distinguished and studied separately.

Figure 15 targets pulse-to-pulse variability, providing MNIST accuracies for variable σ_{intra}, as described in Fig. 15(a). As σ_{intra} increases for both the MSP [Fig. 15(b)] and the LSP [Fig. 15(c)], accuracy drops rapidly, since both CLT and OLT suffer from increasing programming errors affecting either the quality of the weight transfer into the MSP, or the accuracy of LSP weight update during OLT. Since optimum confidence threshold $E_{T,OPT}$ is proportional to σ_{intra}, we adapted $E_{T,OPT}$ using Eq. (1), to provide a fair comparison as σ_{intra} increases.

Figures 16 and 17 show the much smaller effects of device-to-device variability. In particular, Fig. 16 shows the dependence on G_{max} broadening σ_{Gmax}, Fig. 16(a). In this case, results reveal that, for both MSP [Fig. 16(b)] and LSP [Fig. 16(c)], the network can tolerate a large σ_{Gmax} up to 20% of the entire single device dynamic range of $50\,\mu S$ for the MSP, and up to 25% for the LSP. The latter is due to polarity inversion which provides large robustness to fixed asymmetries. For the MSP, the nature of the weight distributions reduces the number of large weights that are difficult to encode into asymmetric MSP pairs. For instance, there is only a small probability that an MSP with a particularly weak G^+ (G^-) would be asked to encode one of the very few large positive (negative) weights.

Similarly, the network is relatively good at tolerating variations in ΔG_{max}, where one of the member conductances of an LSP or MSP might exhibit a much more (less) steep JT than its companion conductance. This robustness is shown as a function of $\sigma_{\Delta G0}$ in Fig. 17(a). Results in Fig. 17(b) for

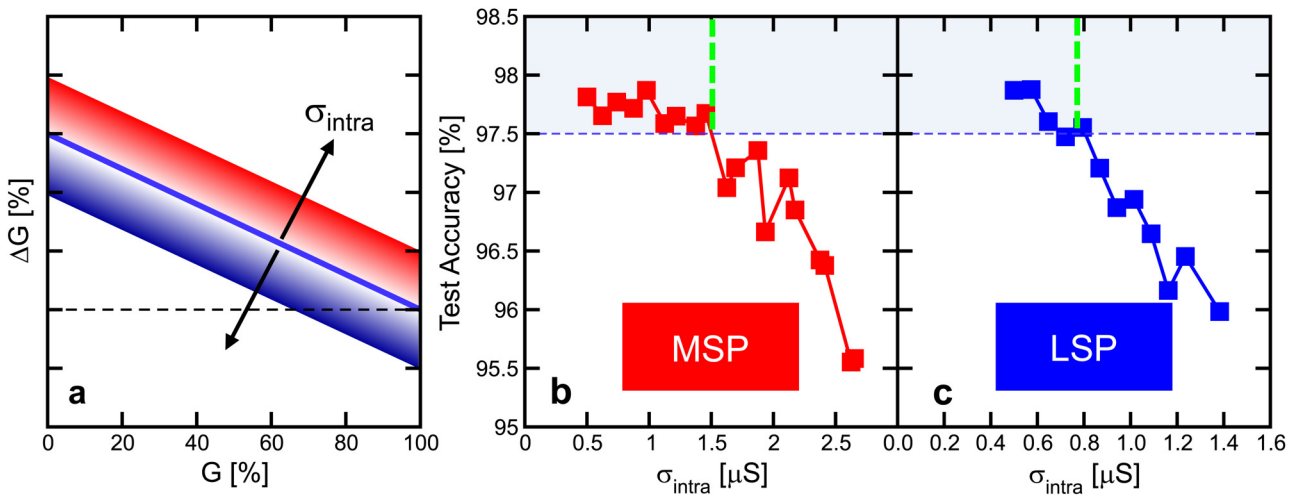

FIG. 15. Impact of pulse-to-pulse variability σ_{intra} (a) of MSP or LSP on training accuracy of an FC DNN with 528-250-125-10 neurons. (b) Test accuracy as a function of MSP σ_{intra}, considering LSP as in Fig. 13(b), showing an accuracy degradation due to noisy CLT procedure. (c) Accuracy as a function of LSP σ_{intra}, considering MSP as in Fig. 13(a), showing a strong sensitivity. This provides specific low-variability requirements on both MSP and LSP devices.

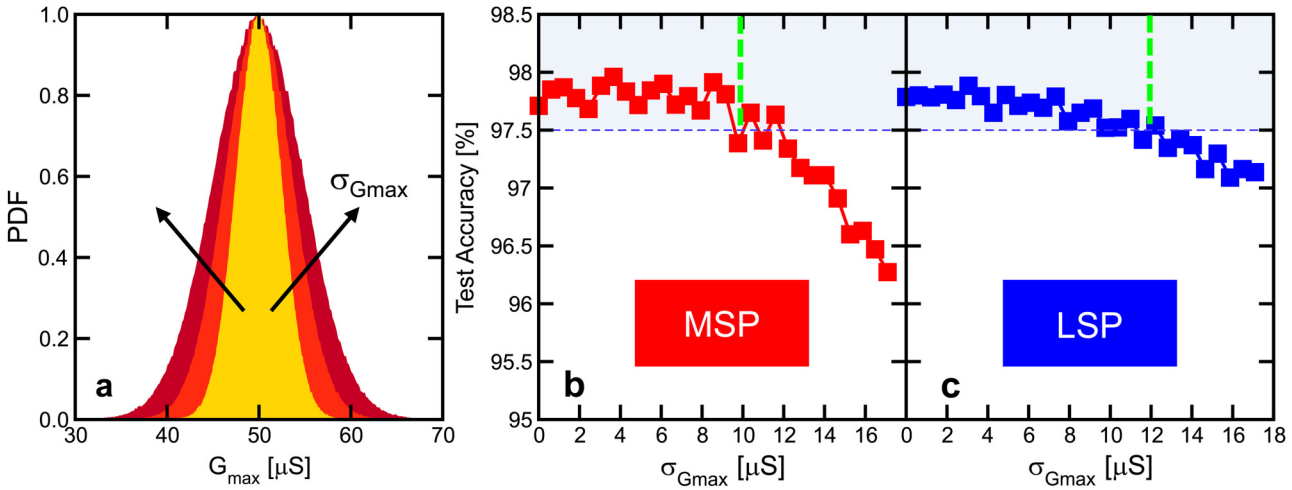

FIG. 16. Impact of G_{max} variation (a) of MSP or LSP on training accuracy of a 528-250-125-10 neurons FC DNN. (b) Accuracy as a function of MSP G_{max}, considering LSP as in Fig. 13(b). (c) Accuracy as a function of LSP G_{max}, considering MSP as in Fig. 13(a). Both cases show a weak dependence of accuracy on $\sigma_{G_{max}}$ due to polarity inversion, with MSP being strongly robust due to CLT.

MSP and Fig. 17(c) for LSP reveal strong robustness in MSP variations and an acceptable accuracy for increased $\sigma_{\Delta G0}$ in LSP. The high robustness of MSP is dictated by the CLT procedure which does not depend at all on the match in initial first conductance steps between the two member devices of the MSP. The robustness of the LSP can again be ascribed to polarity inversion, allowing device-to-device variability between the two member devices of the LSP to cancel out over successive transfer intervals.

C. Endurance

Resistive devices generally show a limited endurance, that is, a limited number of SET and RESET cycles before permanent degradation. While different technologies exhibit substantially different endurance characteristics, it is generally best to try to limit the overall number of programming requests. In addition, the abrupt RESET transition tends to degrade PCM devices more rapidly that SET operations, because phase separation of the constituent atoms mostly occurs while the PCM material is in the molten state. Figure 18(a) shows distributions of the number of SET transitions that individual MSP and LSP devices undergo during 20-epochs of training on the MNIST dataset. MSP devices show a reduced number of SET transitions since programming only occurs during occasional CLT operations, while LSP devices need to deal with a larger number of SET transitions due to continuous OLT during training.

Figure 18(b) shows the number of RESET operations incurred on the individual MSP and LSP devices. Since the training procedure used here calls for a RESET of all LSP devices after every transfer, the corresponding LSP distribution is a delta function with no statistical variations. In contrast, MSP devices exhibit a broad distribution extending all the way to zero, since weights which do not need to be reprogrammed during transfer are left unchanged. Since the

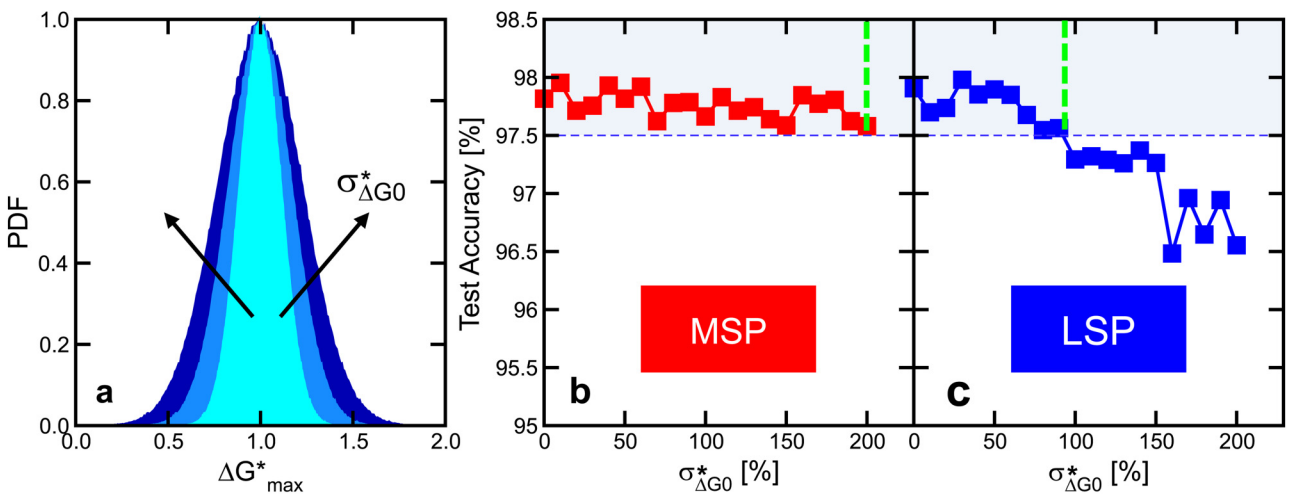

FIG. 17. Impact of $\sigma_{\Delta G0}{}^{*}$ (a) of MSP or LSP on training accuracy of a 528-250-125-10 neurons FC DNN. (b) Accuracy as a function of MSP $\sigma_{\Delta G0}{}^{*}$, considering LSP as in Fig. 13(b). (c) Accuracy as a function of LSP $\sigma_{\Delta G0}{}^{*}$, considering MSP as in Fig. 13(a). Both cases show a weak dependence of accuracy on $\sigma_{\Delta G0}{}^{*}$ due to polarity inversion, with MSP being strongly robust due to CLT.

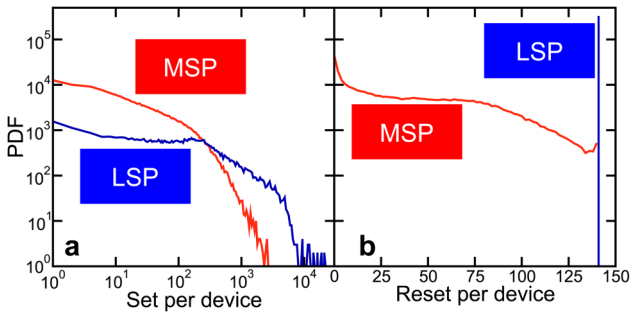

FIG. 18. Distribution of SET (a) and RESET (b) operations per single LSP and MSP LIS device during training, for 20 epochs, of a 528-250-125-10 FC DNN on MNIST dataset.

typical endurance of PCM devices is 10^8 RESET cycles, this shows that a single training run only incurs a tiny fraction of the endurance lifetime of the device.

D. Faulty devices

While conventional CMOS logic technology generally provides close to 100% device yield, emerging memory arrays typically suffer from some number of non-operational devices that simply do not change in conductance. There can be some number of non-operational devices due to issues during fabrication, as well as some devices that are capable of operation but only at higher programming voltages, and are thus non-responsive at the chosen switching voltages. The number of such devices typically increases during operation due to occasional early endurance failures. Non-operational devices can include both "dead" devices, with a read conductance near zero, or "stuck-on," with a read conductance that typically corresponds to a fully SET device.

Both of these types of non-operational devices reduce the network accuracy, both by reducing the number of functional weights the network has available to work with and by introducing an unwanted bias in the neuron activations Σxw, steering this quantity towards zero (large) values for dead (stuck-on) devices, respectively.

Figure 19 shows the accuracy dependence of the previously mentioned FC DNN (528-250-125-10 neurons, trained on the MNIST dataset) on the dead G rate for the MSP (a)

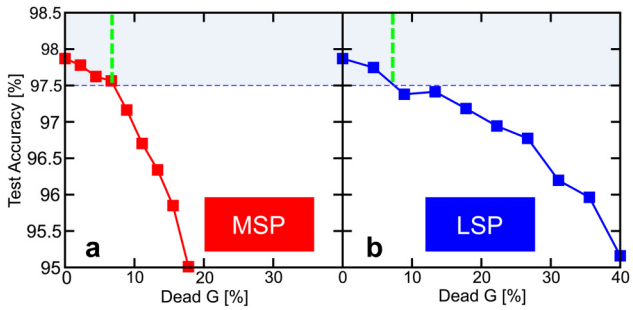

FIG. 19. Impact of dead G rate of MSP or LSP on training accuracy of a 528-250-125-10 neurons FC DNN. (a) Accuracy as a function of MSP dead G rate, considering LSP as in Fig. 13(b). (b) Accuracy as a function of LSP dead G rate, considering MSP as in Fig. 13(a). Both cases show a strong dependence of accuracy on dead G rate.

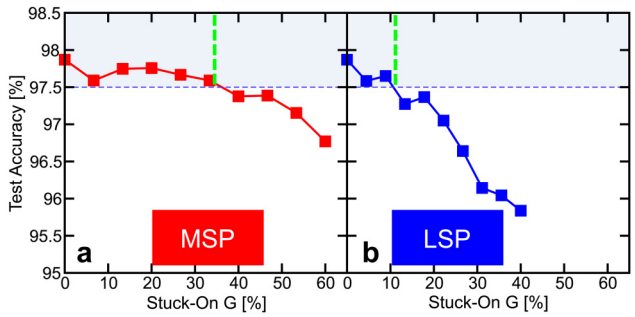

FIG. 20. Impact of stuck-on G rate of MSP or LSP on training accuracy of a 528-250-125-10 neurons FC DNN. (a) Accuracy as a function of MSP stuck-on G rate, considering LSP as in Fig. 13(b). (b) Accuracy as a function of LSP stuck-on G rate, considering MSP as in Fig. 13(a). MSP accuracy shows a larger robustness than LSP accuracy.

and the LSP (b). Each curve assumes that the other device pair has perfect yield. Both show a strong dependence on dead devices, which is consistent with a dependence of accuracy on the number of functional weights. As we study elsewhere,[32] some portion of this degradation can be recovered by increasing the number of hidden units to restore a larger number of functional weights.

Similarly, Fig. 20 plots the dependence of accuracy on stuck-on G rate for MSP (a) and LSP (b). In this case, the impact is stronger on LSP, while MSP accuracy shows a weak dependence due to the CLT which compensates the error introduced by the stuck-on G^+ or G^- with proper tuning of the functional conductances, g^+ and g^-.

VII. CONCLUSION

In this paper, we have provided a detailed description of the device requirements needed to obtain software-equivalent training accuracies on Fully Connected DNNs. The novel weight structure we recently proposed[25] is composed of two pairs of devices, a Most Significant Pair (MSP) and a Least Significant Pair (LSP). This concept substantially reduces device requirements by splitting the training task between the device pairs. The MSP is responsible for long-term weight storage and must be capable of rapid Closed-Loop Tuning (CLT) during weight transfer; the LSP is responsible for example-by-example weight programming and thus must be capable of rapid Open-Loop Tuning (OLT) conductance update. Figure 21 shows a table providing a summary of the quantitative device tolerance requirements obtained here, where the green boxes correspond to the most relaxed condition between MSP and LSP.

For the MSP, retention and stability of the conductance states, and fast, reliable conductance programming are of primary importance. Low pulse-to-pulse variability is much more important than device-to-device variability, endurance, or device yield. While the number of levels available to the MSP does not directly impact the dynamic range of the overall synapse, lower dynamic range forces a smaller F factor and thus limits the efficacy of the multiple-conductances-of-varying-significance approach. The MSP does not need to support small conductance updates, although this could be helpful if the CLT procedure were

Specifications	Parameter	MSP	LSP
Initial Step-size	ΔG_0 (ΔG_0^*)	< 21 μS (42%)	< 1.4 μS (2.8%)
Intra-device Variability	σ_{intra}	< 1.5 μS	< 0.8 μS
Inter-device Variability	σ_{Gmax}	< 10 μS	< 12 μS
	$\sigma^*_{\Delta G0}$	< 200%	< 95%
Faulty devices	Dead C.R.	< 7%	< 7%
	Stuck On C.R.	< 35%	< 10%
Dynamic range	Number of levels	> 13	> 110
Retention	Time before data loss	Higher	Lower
Endurance	Number of Set/Reset	Lower	Higher

FIG. 21. Table summarizing device requirements to enable software equivalent training accuracy for FC DNNs when using multiple conductances of varying significance. Green boxes highlight the most relaxed constraints between LSP and MSP requirements, assuming devices capable of abrupt conductance decrease (RESET) and a conductance response which is initially steep but then saturates as conductance increases (SET).

modified accordingly. An ideal MSP might be one that was capable of very large but still granular conductance changes in one direction (either SET or RESET) and much smaller conductance changes in the other direction, and which exhibited this behavior at all possible conductance states.

In contrast, the LSP needs to offer small conductance changes with low pulse-to-pulse variability. Because the overall dynamic range of the synapse scales directly with the dynamic range of the LSP device, a large dynamic range is still desired. However, the presence of the MSP reduces the required size of this dynamic range by a factor of 8× [from 1000 (Ref. 4) to ∼110]. An LSP device needs fast programming speed for the very small conductance updates and reasonable endurance. However, the presence of MSP and the use of polarity inversion means that poor retention or high device-to-device variability is not a problem.

Future work will need to focus on the development of novel memory devices which target either MSP or LSP requirements. The results of this paper can be directly applied for devices capable of abrupt conductance decrease (RESET) and a conductance response which is initially steep but then saturates as conductance increases (partial SET), or for the exact opposite (e.g., abrupt SET and partial RESET). The methodology of this paper can be used for other kinds of devices offering bidirectional programming. For such devices, the CLT approach would need to be slightly redesigned to exploit the strengths (and to finesse the weaknesses) of such MSP and LSP devices.

The multiple-conductances-of-varying significance approach we recently introduced can eliminate the need to develop one single device that must satisfy a very demanding list of device requirements. In its place, we need two different devices, to populate a Least Significant Pair and Most Significant Pair, calling for very different yet substantially relaxed device requirements, as we have quantified in the present paper. This is an important step towards the industrial development of hardware accelerators based on analog resistive memory devices.

[1]Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," Nature **521**(7553), 436–444 (2015).
[2]D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by backpropagating errors," Nature **323**, 533–536 (1986).
[3]G. Burr, R. M. Shelby, C. di Nolfo, J. Jang, R. Shenoy, P. Narayanan, K. Virwani, E. U. Giacometti, B. Kurdi, and H. Hwang, "Experimental demonstration and tolerancing of a large-scale neural network (165 000 synapses), using phase-change memory as the synaptic weight element," in *IEDM Technical Digest* (2014), p. T29.5.
[4]T. Gokmen and Y. Vlasov, "Acceleration of deep neural network training with resistive cross-point devices: Design considerations," Front. Neurosci. **10**, 333 (2016).
[5]H. Tsai, S. Ambrogio, P. Narayanan, R. M. Shelby, and G. W. Burr, "Recent progress in analog memory-based accelerators for deep learning," J. Phys. D **51**, 283001 (2018).
[6]S. Yu, Z. Li, P.-Y. Chen, H. Wu, B. Gao, D. Wang, W. Wu, and H. Qian, "Binary neural network with 16 Mb RRAM macro chip for classification and online training," in *2016 IEEE International Electron Devices Meeting (IEDM)* (IEEE, 2016), pp. 16.2.1–16.2.4.
[7]S. Yu, P. Y. Chen, Y. Cao, L. Xia, Y. Wang, and H. Wu, "Scaling-up resistive synaptic arrays for neuro-inspired architecture: Challenges and prospect," in *IEDM Technical Digest* (2015), pp. 17.3.1–17.3.4.
[8]M. Hu, J. P. Strachan, Z. Li, E. M. Grafals, N. Davila, C. Graves, S. Lam, N. Ge, J. J. Yang, and R. S. Williams, "Dot-product engine for neuromorphic computing: Programming 1T1M crossbar to accelerate matrix-vector multiplication," in *2016 53rd ACM/EDAC/IEEE Design Automation Conference (DAC)* (IEEE, 2016), pp. 1–6.
[9]J. Woo, K. Moon, J. Song, S. Lee, M. Kwak, J. Park, and H. Hwang, "Improved synaptic behavior under identical pulses using AlO_x/HfO_2 bilayer RRAM array for neuromorphic systems," IEEE Electron Device Lett. **37**(8), 994–997 (2016).
[10]J. W. Jang, S. Park, G. W. Burr, H. Hwang, and Y. H. Jeong, "Optimization of conductance change in $Pr_{1-x}Ca_xMnO_3$-based synaptic devices for neuromorphic systems," IEEE Electron Device Lett. **36**(5), 457–459 (2015).
[11]F. M. Bayat, M. Prezioso, B. Chakrabarti, I. Kataeva, and D. Strukov, "Implementation of multilayer perceptron network with highly uniform passive memristive crossbar circuits," Nat. Commun. **13**, 2331 (2018).
[12]S. Agarwal, S. J. Plimpton, D. R. Hughart, A. H. Hsia, I. Richter, J. A. Cox, C. D. James, and M. J. Marinella, "Resistive memory device requirements for a neural algorithm accelerator," in *2016 International Joint Conference on Neural Networks (IJCNN)* (2016), pp. 929–938.
[13]S. R. Nandakumar, M. Le Gallo, I. Boybat, B. Rajendran, A. Sebastian, and E. Eleftheriou, "Mixed-precision architecture based on computational memory for training deep neural networks," IEEE ISCAS Proc., 1–5 (2018).
[14]I. Boybat, M. Le Gallo, S. R. Nandakumar, T. Moraitis, T. Parnell, T. Tuma, B. Rajendran, Y. Leblebici, A. Sebastian, and E. Eleftheriou, "Neuromorphic computing with multi-memristive synapses," Nat. Commun. **9**, 2514 (2018).
[15]N. Papandreou, H. Pozidis, A. Pantazi, A. Sebastian, M. Breitwisch, C. Lam, and E. Eleftheriou, "Programming algorithms for multilevel phase-change memory," in *2011 IEEE International Symposium of Circuits and Systems (ISCAS)* (2011), pp. 329–332.
[16]D. Garbin, O. Bichler, E. Vianello, Q. Rafhay, C. Gamrat, L. Perniola, G. Ghibaudo, and B. DeSalvo, "Variability-tolerant convolutional neural network for pattern recognition applications based on OxRAM synapses," in *IEDM Technical Digest* (2014), pp. 28.4.1–28.4.4.
[17]P. Yao, H. Wu, B. Gao, S. B. Eryilmaz, X. Huang, W. Zhang, Q. Zhang, N. Deng, L. Shi, H.-S. P. Wong *et al.*, "Face classification using electronic synapses," Nat. Commun. **8**, 15199 (2017).
[18]D. Wouters, R. Waser, and M. Wuttig, "Phase-change and redox-based resistive switching memories," Proc. IEEE **103**(8), 1274–1288 (2015).
[19]G. W. Burr, B. N. Kurdi, J. C. Scott, C. H. Lam, K. Gopalakrishnan, and R. S. Shenoy, "Overview of candidate device technologies for storage-class memory," IBM J. Res. Dev. **52**(4.5), 449–464 (2008).

[20]H.-S. P. Wong, H.-Y. Lee, S. Yu, Y.-S. Chen, Y. Wu, P.-S. Chen, B. Lee, F. T. Chen, and M.-J. Tsai, "Metal-oxide RRAM," Proc. IEEE **100**(6), 1951–1970 (2012).
[21]H. Mulaosmanovic, J. Ocker, S. Müller, M. Noack, J. Müller, P. Polakowski, T. Mikolajick, and S. Slesazeck, "Novel ferroelectric FET based synapse for neuromorphic systems," in *2017 Symposium on VLSI Technology* (2017), pp. T176–T177.
[22]J. Torrejon, M. Riou, F. A. Araujo, S. Tsunegi, G. Khalsa, D. Querlioz, P. Bortolotti, V. Cros, K. Yakushiji, A. Fukushima, H. Kubota, S. Yuasa, M. D. Stiles, and J. Grollier, "Neuromorphic computing with nanoscale spintronic oscillators," Nature **547**, 428–431 (2017).
[23]G. W. Burr, P. Narayanan, R. M. Shelby, S. Sidler, I. Boybat, C. di Nolfo, and Y. Leblebici, "Large-scale neural networks implemented with non-volatile memory as the synaptic weight element: Comparative performance analysis (accuracy, speed, and power)," in *IEDM Technical Digest* (2015), p. 4.4.
[24]S. Agarwal, T.-T. Quach, O. Parekh, A. H. Hsia, E. P. DeBenedictis, C. D. James, M. J. Marinella, and J. B. Aimone, "Energy scaling advantages of resistive memory crossbar based computation and its application to sparse coding," Front. Neurosci. **9**, 484 (2016).
[25]S. Ambrogio, P. Narayanan, H. Tsai, R. M. Shelby, I. Boybat, C. di Nolfo, S. Sidler, M. Giordano, M. Bodini, N. C. P. Farinha, B. Killeen, C. Cheng, Y. Jaoudi, and G. W. Burr, "Equivalent-accuracy neuromorphic hardware acceleration of neural network training using analog memory," Nature **558**(7708), 60 (2018).
[26]T. Y. Liu, T. H. Yan, R. Scheuerlein, Y. Chen, J. K. Lee, G. Balakrishnan, G. Yee, H. Zhang, A. Yap, J. Ouyang, T. Sasaki, A. Al-Shamma, C. Chen, M. Gupta, G. Hilton, A. Kathuria, V. Lai, M. Matsumoto, A. Nigam, A. Pai, J. Pakhale, C. H. Siau, X. Wu, Y. Yin, N. Nagel, Y. Tanaka, M. Higashitani, T. Minvielle, C. Gorla, T. Tsukamoto, T. Yamaguchi, M. Okajima, T. Okamura, S. Takase, H. Inoue, and L. Fasoli, "A 130.7-mm^2 2-layer 32-Gb ReRAM memory device in 24-nm technology," IEEE J. Solid-State Circuits **49**(1), 140–153 (2014).
[27]D. Ielmini, A. L. Lacaita, and D. Mantegazza, "Recovery and drift dynamics of resistance and threshold voltages in phase-change memories," IEEE Trans. Electron Devices **54**(2), 308–315 (2007).
[28]S. Agarwal, R. B. J. Gedrim, A. H. Hsia, D. R. Hughart, E. J. Fuller, A. A. Talin, C. D. James, S. J. Plimpton, and M. J. Marinella, "Achieving ideal accuracies in analog neuromorphic computing using periodic carry," in *2017 Symposium on VLSI Technology* (IEEE, 2017), pp. T174–T175.
[29]P. Narayanan, A. Fumarola, L. Sanches, S. Lewis, K. Hosokawa, R. M. Shelby, and G. W. Burr, "Towards on-chip acceleration of the backpropagation algorithm using non-volatile memory," IBM J. Res. Dev. **61**(4/5), 11:1–11 (2017).
[30]G. W. Burr, R. M. Shelby, S. Sidler, C. Di Nolfo, J. Jang, I. Boybat, R. S. Shenoy, P. Narayanan, K. Virwani, E. U. Giacometti *et al.*, "Experimental demonstration and tolerancing of a large-scale neural network (165 000 synapses) using phase-change memory as the synaptic weight element," IEEE Trans. Electron Devices **62**(11), 3498–3507 (2015).
[31]Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," Proc. IEEE **86**(11), 2278–2324 (1998).
[32]L. P. Romero, S. Ambrogio, M. Giordano, G. Cristiano, M. Bodini, P. Narayanan, H. Tsai, R. M. Shelby, and G. W. Burr, "Training fully connected networks with resistive memories: Impact of device failures," Faraday Discuss. (to be published).