

Integration of physics-derived memristor models with machine learning frameworks

Zhenming Yu^{*†}, Stephan Menzel^{*}, John Paul Strachan^{*†}, Emre Neftci^{*†}

^{*}Fakultät für Elektrotechnik und Informationstechnik, RWTH Aachen, Aachen, 52074, Germany

[†]Peter Grünberg Institut, Forschungszentrum Jülich GmbH, Jülich, 52425, Germany

{z.yu, e.neftci}@fz-juelich.de

Abstract—Simulation frameworks such MemTorch [1] [2], DNN+NeuroSim [3] [4], and aihwkit [5] are commonly used to facilitate the end-to-end co-design of memristive machine learning (ML) accelerators. These simulators can take device nonidealities into account and are integrated with modern ML frameworks. However, memristors in these simulators are modeled with either lookup tables or simple analytic models with basic nonlinearities. These simple models are unable to capture certain performance-critical aspects of device nonidealities. For example, they ignore the physical cause of switching, which induces errors in switching timings and thus incorrect estimations of conductance states. This work aims at bringing physical dynamics into consideration to model nonidealities while being compatible with GPU accelerators. We focus on Valence Change Memory (VCM) cells, where the switching nonlinearity and SET/RESET asymmetry relate tightly with the thermal resistance, ion mobility, Schottky barrier height, parasitic resistance, and other effects [6]. The resulting dynamics require solving an ODE that captures changes in oxygen vacancies. We modified a physics-derived SPICE-level VCM model [7] [8], integrated it with the aihwkit [5] simulator and tested the performance with the MNIST dataset. Results show that noise that disrupts the SET/RESET matching affects network performance the most. This work serves as a tool for evaluating how physical dynamics in memristive devices affect neural network accuracy and can be used to guide the development of future integrated devices.

I. INTRODUCTION

Because of their compact size, non-volatility, and low latency, memristive devices show great potential in ML and neuromorphic engineering. Digital, analog, and stochastic in-memory computing schemes have been developed that utilize the advantages of memristors [9]. However, memristors are subject to nonidealities like switching nonlinearity, SET/RESET asymmetry, device-to-device, and cycle-to-cycle variations. Naive training algorithms that don't take these into account suffer from performance loss. [10] To assist in co-designing memristive ML accelerators, simulation frameworks have been developed [11] with various memristor models.

Memristor models can be generally sorted into two categories: behavioral models and physics-derived models. With behavioral models, memristors are treated as black boxes. Experimental observations are fitted with simple equations, and the models are validated and improved in this process. In contrast, physics-derived models formulate physical equations stemming from an analysis of physical phenomena. The derived equations are often simplified and optimized to get the final solution. While physics-derived models that can accu-

rately produce voltage-dependent behaviors have been adopted in circuit design and validations [12] [13], only behavioral models were used in ML simulators like MemTorch [1] [2], DNN+NeuroSim [3] [4], and aihwkit [5].

Behavioral models can produce faithful results at a rather low compute cost, but they are not able to capture some aspects of device physics, which limits the application of ML simulators. For example, they do not model voltage-dependent switching behaviors, so the effect of different SET and RESET voltages cannot be investigated and optimized. They do not model noise based on variations of device parameters, so the simulation results cannot be used to guide material scientists for optimizing memristive devices. To improve on this situation, we integrated a physics-derived memristor model in an open-source ML simulator aihwkit [5] and investigated the effect of device nonidealities. With this approach, we provide a easy access to the voltage configurations as well as physic driven noises, and make it easy to investigate their impact on the network.

II. DEVICE MODELING

A. JART VCM Model

We choose an accurate physics driven model, the Jülich Aachen Resistive Switching Tools (JART) VCM model [6] as our starting point. The model includes device-to-device and cycle-to-cycle noise, and has also been validated with experimental results in [7]. In the JART model, the filament region of VCM devices are abstracted as a stack of different materials, and the layers of the stack as then modeled as circuit elements in series. As shown in Fig. 1, the model equivalent circuit consists 4 different parts: R_s , R_p , R_d and D_{Sch} .

R_s represents the series resistance, which consists of a fixed resistance of the titanium oxide layer R_{TiO_x} and a current-dependent line resistance R_l . The detailed expression is shown as:

$$R_s = R_{TiO_x} + R_l = R_{TiO_x} + R_0 (1 + \alpha_l R_0 I_M^2 R_{th,l}) \quad (1)$$

where R_0 is the line resistance under zero current, α_l is the temperature coefficient of the lines, I_M is the current through the memristor and $R_{th,l}$ is the thermal resistance of the lines.

The HfO_2 layer is divided into two regions, a plug region with fixed oxygen vacancy concentration N_p , and a disc region where the oxygen vacancy concentration N_d can change. The conductive plug region serves as a reservoir of oxygen

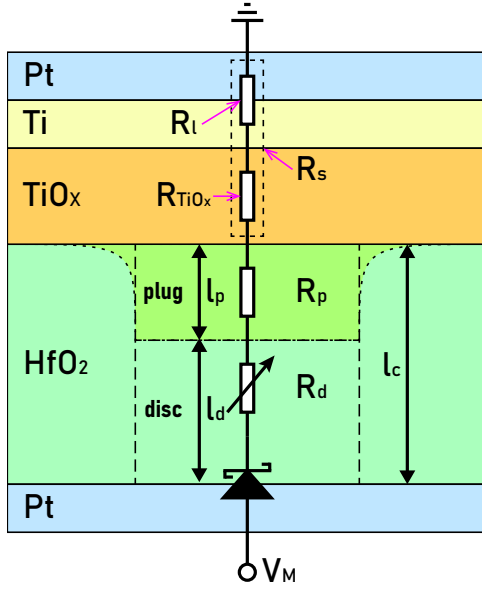


Fig. 1. Equivalent circuit diagram for the JART memristor model. Details of this device can be found in [14]

vacancies, which can then flow into or out of the disc region under the applied voltages, and change the conductance. The resistances of these layers are expressed as:

$$R_{p/d} = l_{p/d} (Z_{VO} e A N_{p/d} \mu_n) \quad (2)$$

where $l_{p/d}$ is the length of the regions, $Z_{VO} e = 2e$ is the charge of a oxygen vacancy, $A = \pi r_d^2$ is the cross section area of the filament area and μ_n is the electron mobility.

Finally, D_{Sch} represents the Schottky barrier formed between the disc region and the bottom electrode. The current-voltage relation of the resulting Schottky diode is shown as:

$$I_M = \begin{cases} -\sqrt{\pi W_{00} e \left(\frac{\phi_{Bn}}{\cosh^2 \left(\frac{W_{00}}{k_B T} \right)} - V_{Sch} \right)} \cdot \exp \left(\frac{-e \phi_{Bn}}{W_0} \right) \\ \cdot \left(\exp \left(\frac{-e V_{Sch}}{\epsilon'} \right) - 1 \right) \cdot A A^* \cdot \frac{T}{k_B} & \text{if } V_M < 0 \\ A A^* T^2 \exp \left(\frac{-e \phi_{Bn}}{k_B T} \right) \exp \left(\frac{e V_{Sch}}{k_B T} - 1 \right) & \text{if } V_M > 0 \end{cases} \quad (3)$$

where k_B is the Boltzman's constant, A^* is the Richardson's constant. The local temperature T is given by:

$$T = \begin{cases} I_M (V_d + V_p + V_{Sch}) R_{th,SET} + T_0 & \text{if } V_M < 0 \\ I_M (V_d + V_p + V_{Sch}) R_{th,RESET} + T_0 & \text{if } V_M > 0 \end{cases} \quad (4)$$

where T_0 is the ambient temperature, $R_{th,SET}$ and $R_{th,RESET}$ are the thermal resistances of the hafnium oxide layer, and W_{00} and W_0 are expressed as:

$$W_{00} = \frac{eh}{4\pi} \sqrt{\frac{Z_{VO} N_d}{m^* \epsilon}}, \quad W_0 = \frac{W_{00}}{\tanh \left(\frac{W_{00}}{k_B T} \right)} \quad (5)$$

and ϵ' as:

$$\epsilon' = \frac{W_{00}}{\frac{W_{00}}{k_B T} - \tanh \left(\frac{W_{00}}{k_B T} \right)} \quad (6)$$

where h is the Planck's constant, m^* is the electron effective mass and ϵ is the static oxide permittivity. The Schottky barrier height ϕ_{Bn} is given by:

$$\phi_{Bn} = \phi_{Bn0} - \sqrt[4]{\frac{Z_{VO} N_d (\phi_{Bn0} - \phi_n - \frac{V_{Sch}}{e})}{8\pi^2 \epsilon_{\phi_B}^3}} \quad (7)$$

where ϕ_{Bn0} is the nominal Schottky barrier height, ϕ_n is the energy level difference between the Fermi level and the conduction band edge and ϵ_{ϕ_B} is the hafnium oxide permittivity related to energy barrier lowering.

B. Simplified Model

The JART model described in Sec. II-A is accurate but expensive to compute. The current I_M described in Eq. 3 is only related to the voltage drop across the Schottky diode V_{Sch} . However, other parts of the device are modeled as resistors, where the voltage drop can only be calculated with the current I_M . Together with other effects, Eq. 1-7 form a complex set of nonlinear equations that can only be solved iteratively, which is very hard to parallelize. To solve this issue, we simplified the current calculation with a model [8] that is mathematically fitted to the JART model. The simplified model provides an estimation of I_M with the oxygen vacancy concentration in the disc region N_d and the applied voltage across the memristor V_M . If $V_M < 0$:

$$I_M = -a - \frac{b}{(1 + c^d)^f} \quad \text{where} \quad a = \frac{a_1 + a_0}{1 + e^{-\frac{V_M + a_2}{a_3}}} - a_0$$

$$b = b_1 (1 - e^{-V_M}) - b_0 V_M, \quad c = \frac{c_2 e^{-\frac{V_M}{c_3}} + c_1 V_M - c_0}{N_d \cdot 10^{-26}}$$

$$d = d_2 e^{-\frac{V_M}{d_3}} + d_1 V_M - d_0, \quad f = f_0 + \frac{f_1 - f_0}{1 + \left(\frac{-V_M}{f_2} \right)^{f_3}} \quad (8)$$

where a_x , b_x , c_x , d_x and f_x are fitting coefficients. If $V_M > 0$:

$$I_M = -\frac{g_0 (e^{-g_1 V_M} - 1)}{\left(1 + (h_0 + h_1 V_M + h_2 e^{-h_3 V_M}) \left(\frac{N_d}{N_{d,min}} \right)^{-j_0} \right)^{\frac{1}{k_0}}} \quad (9)$$

where g_x , h_x , j_x and k_x are fitting coefficients.

Using this simplified model, we can directly calculate I_M , and assign the voltage drop across the resistance layers with Eq. 1-2. The V_{Sch} can then be calculated as:

$$V_{Sch} = V_M - V_s - V_p - V_d \quad (10)$$

The updates can then be calculated with these voltages.

C. Conductance Update

The conductance update is calculated with the original JART model [6] [7]. The conductance change is caused by changing oxygen vacancy concentration in the disc region N_d , which is described by an ordinary differential equation:

$$\frac{dN_d}{dt} = -\frac{I_{ion}}{Z_{VO} e A l_d} \quad (11)$$

where the ionic current I_{ion} is given by:

$$I_{ion} = Z_{VO} e A c_{VO} \alpha \nu_0 F_{limit}(N_d) \cdot \left(\exp\left(-\frac{\Delta W_{Af}}{k_B T}\right) - \exp\left(-\frac{\Delta W_{Ar}}{k_B T}\right) \right) \quad (12)$$

α is the ion hopping distance, ν_0 is the attempt frequency and c_{VO} is the average vacancy concentration:

$$c_{VO} = \frac{N_p + N_d}{2} \quad (13)$$

Function $F_{limit}(N_d)$ scales the update, and limits N_d within the specified range between $N_{d,max}$ and $N_{d,min}$:

$$F_{limit}(N_d) = \begin{cases} 1 - \left(\frac{N_d}{N_{d,max}}\right)^{10} & \text{if } V_M < 0 \\ 1 - \left(\frac{N_{d,min}}{N_d}\right)^{10} & \text{if } V_M > 0 \end{cases} \quad (14)$$

And the energy barriers for ion hopping are described as:

$$\Delta W_{Af/r} = \Delta W_A \left(\sqrt{1 - \gamma^2} \mp \gamma \frac{\pi}{2} + \gamma \arcsin(\gamma) \right) \quad (15)$$

$$\gamma = \frac{e Z_{VO} \alpha E_{ion}}{\Delta W_A \pi} \quad (16)$$

$$E_{ion} = \begin{cases} V_d / l_d & \text{if } V_M < 0 \\ (V_p + V_d + V_{Sch}) / l_c & \text{if } V_M > 0 \end{cases} \quad (17)$$

where ΔW_A is the activation energy, l_d is the thickness of the disc region and l_c is the thickness of the hafnium oxide layer.

switch faster, but the intermediate states becomes less accessible. Lower pulse amplitudes can give us better accessibility to the intermediate states, but at the cost of switching delays. We matched the SET and RESET curves by hand tuning the voltages, and the results shown in Fig. 2(d) are almost identical.

To ease the use of the device in a on-chip learning scenario where the weights are first trained for task A and later adjusted for task B, we need to control the conductance range. If the conductance range is not controlled, a device might be pushed too far off towards the extreme. In this situation, it which will be hard to move back into the fast-switching region, with the low voltages that grant us access to the intermediate states. This could be implemented by control circuits that checks the device conductance, and skip some of the applied pulses that further push deice conductance beyond the specified range.

D. Noise Implementation

To account for stochasticity, we implement variations selectively on parameters that are physically noisy. However, as described in Sec. II-B, in the simplified model, physical values are hidden behind the fitting parameters $a_x - k_x$ in Eq. 8-9. So we only induced noise with conductance update described in Sec II-C. In the original JART model [7], noise are introduced in the oxygen vacancy boundaries: $N_{d,max}$, $N_{d,min}$ and the filament geometry parameters: r_d , l_d . Because we introduced peripheral circuits that control the conductance ranges, we also added device-to-device noise on the conductance boundaries set by the control circuits: G_{max} , G_{min} .

In [7], device-to-device noise are introduced upon initialization, by drawing from a Gaussian distribution with variance scaled by the mean:

$$X_{d2d} \sim \mathcal{N}(X_{mean}, X_{mean} \cdot \sigma) \quad (18)$$

and cycle-to-cycle noise is implemented as a random walk starting from the device-to-device initialization values, i.e. $X_0 = X_{d2d}$. For $N_{d,max}$ and $N_{d,min}$, cycle-to-cycle noise were directly inserted:

$$X_{t+1} = X_t + \Omega \cdot X_t \cdot \sigma, \quad \Omega \sim U(-1, 1) \quad (19)$$

and for the filament geometry parameters r_d , l_d , the cycle-to-cycle noise is scaled by the update magnitude. This produces more noise when a larger update is applied, and less noise otherwise. However, with our ML use case, most weight updates occur in the first few epochs. As learning progresses, the updates become small and the cycle-to-cycle noise becomes negligible. So we added another additive cycle-to-cycle noise to the JART implementation:

$$X_{t+1} = \begin{cases} X_t \cdot \left(1 + \Omega_1 \sigma_{add} + \Omega_2 \sigma_{mult} \left(\frac{N_d - N_{d,old}}{N_{d,max} - N_{d,old}} \right) \right) & \text{if } V_M < 0 \\ X_t \cdot \left(1 + \Omega_1 \sigma_{add} + \Omega_2 \sigma_{mult} \left(\frac{N_{d,old} - N_d}{N_{d,old} - N_{d,min}} \right) \right) & \text{if } V_M > 0 \end{cases}$$

$$\Omega_1, \Omega_2 \sim U(-1, 1). \quad (20)$$

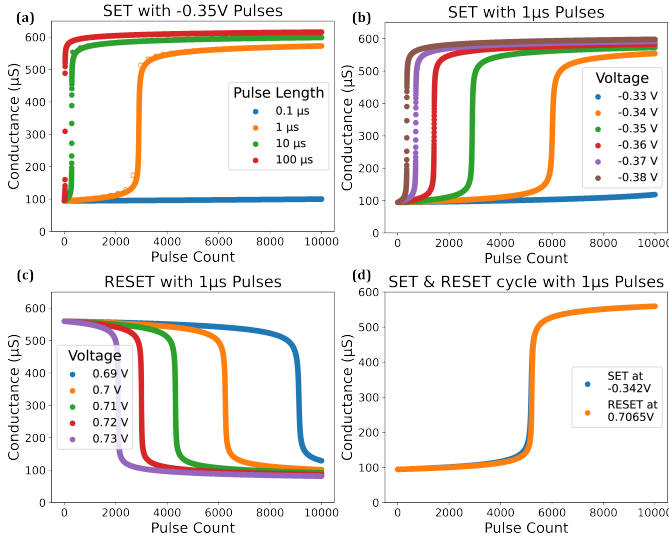


Fig. 2. **Simulation results of the memristor model in Python.** (a): Changing pulse length with fixed pulse amplitude. (b): Changing pulse amplitude with fixed pulse length for the SET direction(i.e. $V_M < 0$). (c): Changing pulse amplitude with fixed pulse length for the RESET direction(i.e. $V_M > 0$). (d): Hand-tuned SET and RESET matching.

With this physics-derived model, we can freely adjust the pulse length and the pulse amplitude to get various switching behaviour in both directions. As shown in Fig. 2(a-c), higher pulse amplitudes and longer pulse length can make the device

This additive noise accounts for the diffusion in the filament area, which happens with or without external stimulation.

Optional boundaries can be specified to truncate the device-to-device noise to reasonable ranges or to limit the cycle-to-cycle random walk process during training.

III. RESULTS

We integrated the model described in Sec. II with the IBM aihwkit simulator [5] and tested the results on MNIST dataset [15]¹. The model is integrated into a 3-layer fully connected network with sigmoid activation between the layers. We trained the network with stochastic gradient descent and a scheduler that decreases the learning rate by 50% for every 10 epochs. This is necessary because our device model has a very sharp transition in the middle of the conductance range (Fig. 2). If a constant learning rate is used, the network parameters oscillate towards the end of the training, which damages the performance.

As shown in Fig. 3(a), with the same network architecture, learning rate, and scheduler configuration, floating point weights achieved an accuracy of 95%. Our device model achieved an accuracy of 93.8% without any noise being applied, and with noise fitted experimentally in [7], we can get a performance of 88.3%.

From equations 11-17, we can derive that noise on different parameters have different effects. Noise on $N_{d,max}$, $N_{d,min}$, and l_d only affect the update magnitude, they don't change the sign of the update, so in the end they only affect the learning rate. Noise on r_d however, can affect the matching between the increment and decrement update steps. This is mainly due to the asymmetric change in E_{ion} and T with variable r_d . As shown in Eq. 17, with SET pulses, E_{ion} is related with r_d :

$$\begin{aligned} E_{ion} &= \frac{V_d}{l_d} = \frac{I_M \cdot R_d}{l_d} \\ &= \frac{I_M \cdot l_d (Z_{VO} e A N_d \mu_n)}{l_d} \\ &= I_M Z_{VO} e N_d \mu_n \pi r_d^2. \end{aligned} \quad (21)$$

However, with RESET pulses, E_{ion} is only related to noise-free parameters:

$$\begin{aligned} E_{ion} &= \frac{V_p + V_d + V_{Sch}}{l_c} = \frac{V_M - V_s}{l_c} \\ &= \frac{V_M - I_M \cdot (R_{TiO_x} + R_0 (1 + \alpha_l R_0 I_M^2 R_{th,l}))}{l_c}. \end{aligned} \quad (22)$$

Noise in r_d also affects the local temperature T through thermal resistances, and Eq. 4 becomes:

$$T = \begin{cases} I_M (V_M - V_s) R_{th,SET} \cdot \frac{r_d^2}{r_{d,Noisy}^2} + T_0 & \text{if } V_M < 0 \\ I_M (V_M - V_s) R_{th,RESET} \cdot \frac{r_d^2}{r_{d,Noisy}^2} + T_0 & \text{if } V_M > 0 \end{cases} \quad (23)$$

¹The code for the modified aihwkit simulator with JART model integration and the test scripts used for this work are publicly available on GitHub at <https://github.com/ZhenmingYu/aihwkit>.

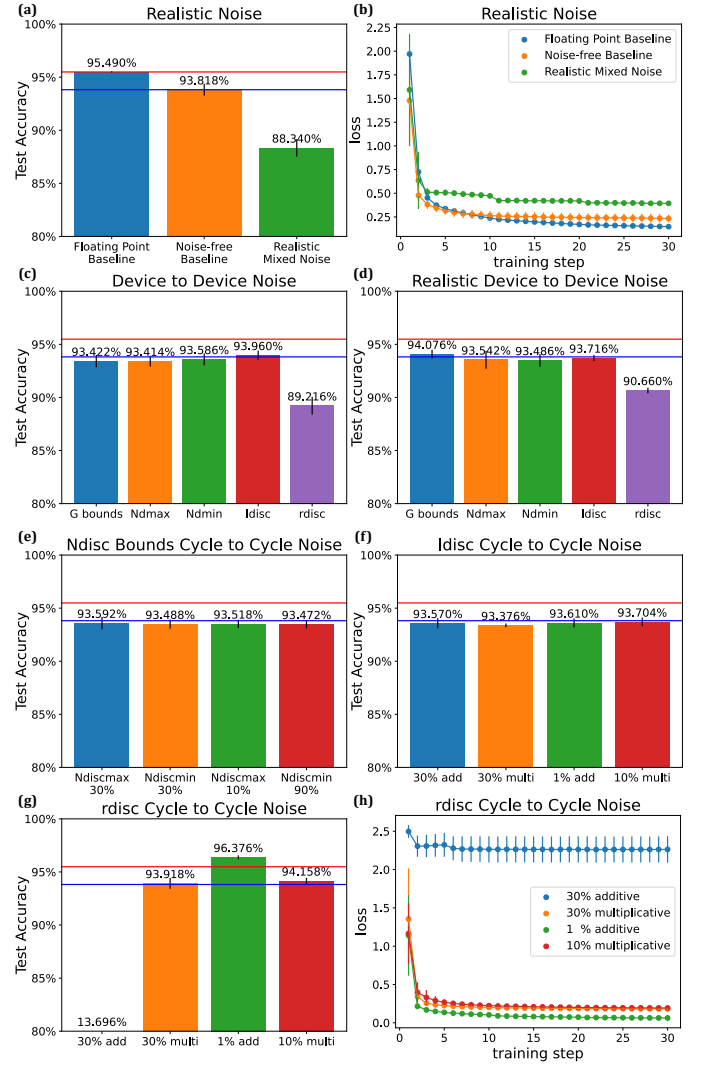


Fig. 3. **MNIST simulation results of accuracy (a) and loss (b)** for simulations with realistic mixed noise, without any noise, and with floating point weights. Accuracy of device-to-device noise at the same scale (c) and based on realistic estimation (d). Accuracy of cycle-to-cycle noise on $N_{d,max}$, $N_{d,min}$ (e) and on l_d . Accuracy (f) and loss (g) for cycle-to-cycle noise on r_d .

where $R_{th,SET}$ and $R_{th,RESET}$ are different values. As a result, the local temperature will scale differently during SET and RESET for the same $r_{d,Noisy}$, and will then affect the switching behaviour in different ways.

The effect of different noise can also be seen in the results. Fig. 3(c) and (d) show the accuracy of the network with device-to-device noise. When the device-to-device noise are scaled to the same extent of 30%, noise on most parameters don't affect the performance as much, and the network achieved accuracy close to the noise-free baseline. With device-to-device noise on r_d however, the network performance decreases significantly to about 89.2%. Using r_d noise based on a realistic estimation, where r_d is allowed a smaller spread, the performance is slightly increased to about 90.7%. Similar results can be seen with cycle-to-cycle noise. As shown in Fig. 3(e) and (f), noise on $N_{d,max}$, $N_{d,min}$ and l_d don't affect performance much.

The results are quite different for the cycle-to-cycle noise on r_d : The network is crippled with a large 30% noise directly applied in the random walk. The loss will not go down during training (Fig. 3(h)), and an accuracy of only 13.7% is achieved (Fig. 3(g)). However, with a small 1% noise directly applied, it becomes easier for the network to get out of local minima, and an accuracy beyond the baseline is achieved. Multiplicative noise scaled by the update have very little impact on the network performance, as they become negligible after a few training epochs.

Because we used the simplified fit model (Sec. II-B) for current calculation, the voltage estimation across different layers will not change with noise on the disc region length l_d . As a result, noise on l_d appear to have little impact on the network performance in our tests. However, in reality, l_d should also be able to affect the network by changing the voltage drop across different layers, as shown in Eq. 1-3. Nonetheless, we are still able to conclude that the matching between increment and decrement update steps is the most important property for training neural networks on memristors. This finding is also in line with previous publication [10] [16].

IV. CONCLUSION

We built a memristive machine learning simulator with physics-derived model and investigated the effect of noise from various sources based on it. This simulator unlocks many possibilities for the community. It opens up access to the pulse configurations, which enables us to explore better ways to match the performance-critical SET and RESET behaviours. It also exposes nonidealities rooted in different physical sources, which grants the ML community chances to develop new algorithms that target these nonidealities directly. On the other hand, it also gives material scientists a tool to evaluate the effect of different noise sources, which can guild them in developing novel device structures that alleviates noise on performance-critical parameters. With the random walk based cycle-to-cycle noise implementation, we can also disable the boundaries and train the model on different tasks. The parameters would then stray away from their original values, causing the device behavior to change after the training. This allow us to study the effect of device aging, and try to prevent performance loss while operating the devices.

ACKNOWLEDGMENT

This work was sponsored by the Federal Ministry of Education, Germany (project NEUROTEC-II grant no. 16ME0398K and 16ME0399). We thank Vasileios Ntinis for his help with the simplified model [8], and Malte J. Rasch for his generous support on aihwkit [5].

COPYRIGHT

© 2022 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in

any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

REFERENCES

- [1] C. Lammie and M. R. Azghadi, "Memtorch: A simulation framework for deep memristive cross-bar architectures," in *2020 IEEE international symposium on circuits and systems (ISCAS)*. IEEE, 2020, pp. 1–5.
- [2] C. Lammie, W. Xiang, B. Linares-Barranco, and M. R. Azghadi, "Memtorch: An open-source simulation framework for memristive deep learning systems," *Neurocomputing*, vol. 485, pp. 124–133, 2022.
- [3] X. Peng, S. Huang, H. Jiang, A. Lu, and S. Yu, "Dnn+ neurosim v2.0: An end-to-end benchmarking framework for compute-in-memory accelerators for on-chip training," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 40, no. 11, pp. 2306–2319, 2020.
- [4] S. Yu, H. Jiang, S. Huang, X. Peng, and A. Lu, "Compute-in-memory chips for deep learning: Recent trends and prospects," *IEEE Circuits and Systems Magazine*, vol. 21, no. 3, pp. 31–56, 2021.
- [5] M. J. Rasch, D. Moreda, T. Gokmen, M. Le Gallo, F. Carta, C. Goldberg, K. El Maghraoui, A. Sebastian, and V. Narayanan, "A flexible and fast pytorch toolkit for simulating training and inference on analog crossbar arrays," in *2021 IEEE 3rd international conference on artificial intelligence circuits and systems (AICAS)*. IEEE, 2021, pp. 1–4.
- [6] F. Cüppers, S. Menzel, C. Bengel, A. Hardtdegen, M. Von Witzleben, U. Böttger, R. Waser, and S. Hoffmann-Eifert, "Exploiting the switching dynamics of hfo2-based reram devices for reliable analog memristive behavior," *APL materials*, vol. 7, no. 9, p. 091105, 2019.
- [7] C. Bengel, A. Siemon, F. Cüppers, S. Hoffmann-Eifert, A. Hardtdegen, M. von Witzleben, L. Hellmich, R. Waser, and S. Menzel, "Variability-aware modeling of filamentary oxide-based bipolar resistive switching cells using spice level compact models," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 67, no. 12, pp. 4618–4630, 2020.
- [8] V. Ntinis, A. Ascoli, I. Messaris, Y. Wang, V. Rana, S. Menzel, and R. Tetzlaff, "Towards simplified physics-based memristor modeling of valence change mechanism devices," *IEEE Transactions on Circuits and Systems II: Express Briefs*, 2022.
- [9] D. Ielmini and H.-S. P. Wong, "In-memory computing with resistive switching devices," *Nature electronics*, vol. 1, no. 6, pp. 333–343, 2018.
- [10] C. Lee, K. Noh, W. Ji, T. Gokmen, and S. Kim, "Impact of asymmetric weight update on neural network training with tiki-taka algorithm," *Frontiers in neuroscience*, p. 1554, 2022.
- [11] C. Lammie, W. Xiang, and M. R. Azghadi, "Modeling and simulating in-memory memristive deep learning systems: An overview of current efforts," *Array*, p. 100116, 2021.
- [12] X. Fu, Q. Li, W. Wang, H. Xu, Y. Wang, W. Wang, H. Yu, and Z. Li, "High speed memristor-based ripple carry adders in 1t1r array structure," *IEEE Transactions on Circuits and Systems II: Express Briefs*, 2022.
- [13] M. Mayahinia, A. Singh, C. Bengel, S. Wiefels, M. A. Lebdeh, S. Menzel, D. J. Wouters, A. Gebregiorgis, R. Bishnoi, R. Joshi *et al.*, "A voltage-controlled, oscillation-based adc design for computation-in-memory architectures using emerging rerams," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 18, no. 2, pp. 1–25, 2022.
- [14] A. Hardtdegen, C. La Torre, F. Cüppers, S. Menzel, R. Waser, and S. Hoffmann-Eifert, "Improved switching stability and the effect of an internal series resistor in hfo 2/tio x bilayer reram cells," *IEEE transactions on electron devices*, vol. 65, no. 8, pp. 3229–3236, 2018.
- [15] Y. LeCun and C. Cortes, "The mnist database of handwritten digits," 2005.
- [16] T. Gokmen and W. Haensch, "Algorithm for training neural networks on resistive device arrays," *Frontiers in Neuroscience*, vol. 14, p. 103, 2020.