APL Electronic Devices

RESEARCH ARTICLE | SEPTEMBER 19 2025

High-accuracy inference using HfO_xS_y/HfS_2 memristors

Aferdita Xhameni ; Antonio Lombardo

Check for updates

APL Electronic Devices 1, 036124 (2025)
https://doi.org/10.1063/5.0286727

View Online
Export Citation

Articles You May Be Interested In

Band edge states, intrinsic defects, and dopants in monolayer HfS_2 and SnS_2

Appl. Phys. Lett. (February 2018)

A synaptic transistor based on van der Waals heterojunction $HfS_2/HfO_x/SnS_2$ with optical modulation properties

Appl. Phys. Lett. (April 2024)

Resistive switching behaviors and mechanisms of HfS_2 film memory devices studied by experiments and density functional theory calculations

Appl. Phys. Lett. (February 2020)

18 December 2025 10:27:04

AIP Publishing

APL Electronic Devices
Fostering connections across multiple disciplines in the broad electronics community
Now Open for Submissions
AIP Publishing

High-accuracy inference using HfO_xS_y/HfS_2 memristors

Cite as: APL Electron. Devices 1, 036124 (2025); doi: 10.1063/5.0286727
Submitted: 20 June 2025 • Accepted: 6 September 2025 •
Published Online: 19 September 2025

Aferdita Xhameni[1,2] and Antonio Lombardo[1,2,a)]

AFFILIATIONS

[1] London Centre for Nanotechnology, 19 Gordon St, London WC1H 0AH, United Kingdom
[2] Department of Electronic and Electrical Engineering, Malet Place, University College London, London WC1E 7JE, United Kingdom

[a)] Author to whom correspondence should be addressed: a.lombardo@ucl.ac.uk

ABSTRACT

We demonstrate high-accuracy classification for handwritten digits from the MNIST dataset (~98.00%) and RGB images from the CIFAR-10 dataset (~86.80%) by using resistive memories based on a 2D van der Waals semiconductor: hafnium disulfide (HfS_2). These memories are fabricated via dry thermal oxidation, forming vertical crossbar HfO_xS_y/HfS_2 devices with a highly ordered oxide-semiconductor structure. Our devices operate without electroforming or current compliance and exhibit multi-state, non-volatile resistive switching, allowing resistance to be precisely tuned using voltage pulse trains. Using low-energy potentiation and depression pulses (0.7–0.995 V, 160–350 ns), we achieve 31 (~5 bits) stable conductance states with high linearity, symmetry, and low variation over 100 cycles. Key performance metrics—such as weight update, quantization, and retention—are extracted from these experimental devices. These characteristics are then used to simulate neural networks with our resistive memories as weights. Neural networks are trained on state-of-the-art (SOTA) digital hardware (CUDA cores), and a baseline inference accuracy is extracted. IBM's Analog Hardware Acceleration Kit is used to modify and remap digital weights in the pretrained network based on the characteristics of our devices. Simulations account for factors like conductance linearity, device variation, and converter resolution. In both image recognition tasks, we demonstrate excellent performance, similar to SOTA, with only <0.07% and <1.00% difference in inference accuracy for the MNIST and CIFAR-10 datasets, respectively. The forming-free, compliance-free operation, fast switching, low energy consumption, and high-accuracy classification demonstrate the strong potential of HfO_xS_y/HfS_2-based resistive memories for energy-efficient neural network acceleration and neuromorphic computing.

© 2025 Author(s). All article content, except where otherwise noted, is licensed under a Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/). https://doi.org/10.1063/5.0286727

18 December 2025 10:27:04

I. INTRODUCTION

The economic and environmental costs of training and deploying neural networks for machine learning and artificial intelligence must be addressed.[1,2] From their conception, neural networks have taken inspiration from the brain to enable and improve performance in machine learning tasks. Arrangements of artificial neurons and synapses comprise the network, where the strengths of connections between different nodes in the network (artificial neurons) are represented by the weight values of branches connecting the layers (artificial synapses). Once a network has been trained to solve a particular task, its weights encode the network's ability to evaluate new, untested data and thus should be trainable, precise, and resilient to repeated programming and device aging. In the early days of machine learning, computation for neural networks was performed on digital hardware such as central processing units (CPUs), resulting in performance increases over time that generally followed Moore's Law. However, since 2012, computation for machine learning on digital hardware has been performed on graphical processing units (GPUs), from which a doubling of performance has been achieved every 3.4 months or fewer. Aside from improved algorithms and the increased parallelism offered by GPU cores, this rapid increase in performance can also be explained by the rate at which GPU hardware has improved, with NVIDIA GPUs improving in computational performance by a factor of 317 since 2012.[1] However, despite recent advances in more efficient algorithms and hardware architectures, a rethinking of machine learning systems at the most fundamental level is urgently required to address the ever-growing demand for computing power.[1] Taking inspiration from the brain, developing hardware that can co-locate processing and memory functions is key to breaking the von Neumann bottleneck, which limits computational efficiency by necessitating the shuttling of data back and forth between processing and memory units.[3]

Many different types of devices are potential candidates for accelerating performance in machine learning tasks and surpassing the von Neumann bottleneck. Memristors are one class of simple, two-terminal analog devices that have shown promise in hardware acceleration for neuromorphic computing and machine learning tasks when integrated into densely packed crossbar arrays.[4] Analogous to biological systems, where the transmission strength of incoming signals can be controlled at a synapse, most memristors can variably impede the flow of current due to modulation of their conductance between multiple states. Conductance states in a memristor can be modified by the application of electrical stress, such as a voltage or current pulse, where increasing the memristor's conductance state is referred to as potentiation, and decreasing it is referred to as depression. Similar to biological synapses, some memristors can retain programmed conductance states when electrical stress is removed, making them non-volatile memory devices. Hence, in most implementations where memristors are used for machine learning, the devices are integrated into crossbar arrays utilized for weight storage and update. The analog crossbar array can then be interfaced with digital integrated circuits via analog-to-digital converters (ADC) and digital-to-analog converters (DAC) for other processes in a machine learning task, such as applying activation functions. In existing digital hardware, weight values are calculated and stored in separate logic and memory units, respectively. However, analog crossbar arrays of memristors offer vastly increased parallelism and avoid shuffling data back and forth, as weight values can be both programmed and stored as non-volatile conductance states in the same memristive hardware. Therefore, by inputting voltages across the rows of a crossbar array where memristors have been programmed to precise conductance states and measuring the output currents along the columns, a memristor-based crossbar array can perform multiplication and accumulation operations (using Ohm's and Kirchhoff's laws) to enable fast matrix-vector multiplications (MVM). In some applications, such as compressed sensing, this presents a key advantage of memristor-based crossbar arrays, which is to allow for different matrix-vector multiplication (MVM) operations to execute in the same amount of time, regardless of the input data size [O(1) time complexity].[4] The same is not true for GPUs, in which execution time grows with n^2 for input data of size n in MVM operations. A good example of the performance enhancements offered by memristive hardware in machine learning tasks can be found in the work of Yao *et al.*[5] In this paper, network weights were first trained on digital hardware, then transferred to physically implemented crossbar arrays of $TiN/TaO_x/HfO_x/TiN$ memristors, with modifications made to network weights to account for the characteristics of their memristor hardware.[5] Their memristor-based neural network was able to correctly classify a large proportion of previously unseen handwritten numbers from the MNIST dataset,[6] resulting in a classification accuracy of 96.19%, close to a digital hardware baseline score of 97.99%.[5] The small difference in accuracy, together with the significant decrease in energy consumption and improved performance density compared to conventional digital hardware for storing weights, clearly demonstrates a strong use case for integrating memristor-based hardware accelerators with digital components. Therefore, owing to their potential for efficiency, scalability, and strong non-volatile memory performance, memristors are strong candidates for use as analog weight storage in crossbar arrays.[5,7]

Despite their excellent performance, there are a number of challenges associated with using memristive hardware for machine learning applications. Noise during conductance update or read steps can originate from a variety of sources and reduces the effectiveness of using memristors for weight storage or update in a neural network. Precisely programming and distinguishing states in noisy devices that show highly nonlinear conductance update within a limited conductance range can become impossible, leading to reduced machine learning accuracy. On the other hand, linear conductance update allows different conductance states to be more accurately distinguished during potentiation and depression, provided that there is low cycle-to-cycle variation and small drift in the devices. When trying to program memristors to represent specific weights in a neural network, this behavior facilitates high machine learning accuracy. Furthermore, not only should conductance update be linear in potentiation and depression, but the device should also show a high degree of symmetry in both schemes.

To enable the use of energy-efficient and scalable memristive hardware in machine learning tasks, tailored potentiation and depression pulsing schemes in which pulse widths and amplitudes are modulated by pulse number are often introduced.[8–12] Incrementally varying pulse widths, amplitudes, or otherwise tuning biasing pulse trains while being conscious of device characteristics can also improve cycle-to-cycle variation.[12] By operating the device in a biasing regime that does not result in sudden changes in conductance due to stochastic and unpredictable effects like filament formation or rupture, predictable and repeatable changes in conductance can be utilized to program the device.[12] Therefore, despite increasing computational latency and circuit area, schemes that incrementally update conductance with respect to programming pulse number can be used to more accurately and repeatably program resistive switching devices.[8,12]

Another set of challenges associated with implementing memristive hardware in neural networks includes the requirements of electroforming and current compliance for each device. Electroforming is a one-time initialization step typically required by a class of memristors called resistive random access memory (RRAM) devices, typically based on insulating metal oxides. RRAM devices have otherwise shown excellent performance, reliability, energy efficiency, and scalability. However, requiring electroforming presents a barrier to their adoption since it necessitates increased peripheral circuitry and hinders scaling.[13] Current compliance circuitry is required by most memristive devices that do not have a self-limiting mechanism to prevent high currents from damaging the device. In most implementations, memristive chips require 1-transistor-1-memristor architectures (1T1M), which limit integration density and increase computational complexity.[14,15] Defect engineering can be used to address electroforming by introducing a large number of defects or modifying the microstructure of the pristine device to provide pre-existing conductive pathways. Self-limiting currents in the memristor stack have been achieved by intentionally engineering heterostructures that slow the motion of charge carriers responsible for changing the device resistance, such as oxygen vacancies.[14,15]

18 December 2025 10:27:04

One class of materials that allows for precise defect engineering and facile control of heterostructures with pristine interfaces is two-dimensional layered materials (2DLMs). Memristors based on partially oxidized 2DLM semiconductors have shown promise in energy efficiency, fast, compliance-free, and electroforming-free operation,[16,17] along with strong non-volatile memory characteristics and independence from surrounding conditions such as water vapor, temperature, and oxygen.[17]

While there are a variety of candidates and methods for integrating memristors into neural networks to address many of these issues, it is necessary to have a means of rapidly and accurately evaluating the potential performance of new classes of memristive devices for hardware accelerators. Such devices may be able to provide a breakthrough in performance that could carve a niche for their use within machine learning and neuromorphic applications. However, due to the nature of device fabrication with novel materials and engineering methods, the path from individual device fabrication and performance evaluation to wide-scale chip fabrication and programming is often long, expensive, and does not allow for exploring the materials and parameter space for device optimization, limiting adoption by industry. The Analog Hardware Acceleration Kit (AIHWKIT) developed by IBM[18] is a Python-based, open-source library that enables estimation of the performance of analog memristive hardware in a variety of machine learning tasks by implementing a wide range of measured device performance parameters, non-idealities, and necessary peripheral circuitry in machine learning simulations.[19] While performance estimations based on AIHWKIT may not account for all possible challenges that may arise when exploring new hardware for analog in-memory computation, it allows for devices based on less mature technologies to demonstrate their potential performance in memristive chips. Consequently, it provides an excellent route toward rapid device optimization and materials screening without the need for complex fabrication of large arrays of devices.

In this work, we investigate the machine learning performance of a simulated network/crossbar chip whose elements are experimental HfO_xS_y/HfS_2 memristors. Such devices have shown high potential for ML applications as they combine sub-nJ switching, excellent thermal and environmental stability, current self-limiting (compliance-free), and forming-free operation.[17] These devices were investigated with tailored potentiation and depression pulses, producing highly linear and symmetric conductance update with low cycle-to-cycle variation. Such linearity, together with the non-volatility of the states, enables the use of our memristors in neural networks to store synaptic weights. The network weights are mapped into a number of programming pulses and stored in the memristors as resistance or conductance values. As a result, despite accounting for a range of measured device characteristics—such as ON/OFF ratio, cycle-to-cycle, and device-to-device variations—our simulations show high-accuracy classification scores with the MNIST dataset[6] and the more challenging CIFAR-10 dataset,[20] nearing state-of-the-art (SOTA) performance. The combination of the desirable figures of merit of the individual memristors we investigated, and their simulated performance in memristive chips for machine learning tasks provides strong motivation for further research on memristive chips based on 2DLM semiconductor-insulator structures.

18 December 2025 10:27:04

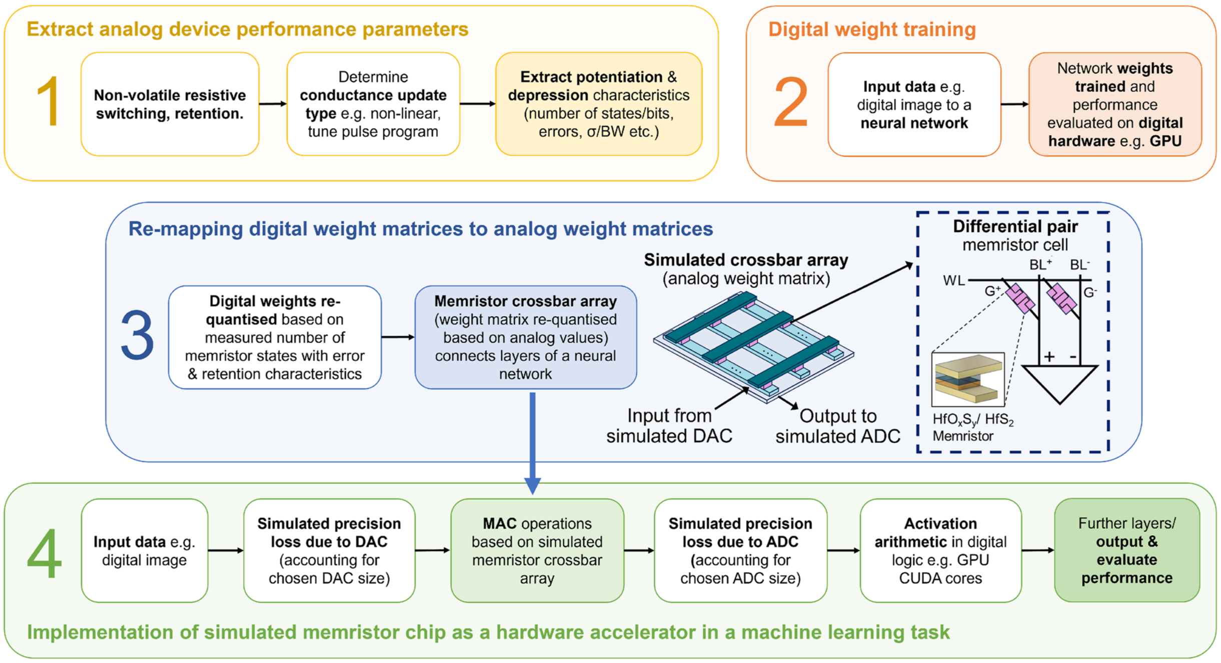

Extract analog device performance parameters
1
Non-volatile resistive switching, retention.
Determine conductance update type e.g. non-linear, tune pulse program
Extract potentiation & depression characteristics (number of states/bits, errors, σ/BW etc.)
Digital weight training
2
Input data e.g. digital image to a neural network
Network weights trained and performance evaluated on digital hardware e.g. GPU
Re-mapping digital weight matrices to analog weight matrices
3
Digital weights re-quantised based on measured number of memristor states with error & retention characteristics
Memristor crossbar array (weight matrix re-quantised based on analog values) connects layers of a neural network
Simulated crossbar array (analog weight matrix)
Input from simulated DAC
Output to simulated ADC
Differential pair memristor cell
WL
BL+
BL-
G+
G-
+
-
HfOxSy/ HfS2 Memristor
4
Input data e.g. digital image
Simulated precision loss due to DAC (accounting for chosen DAC size)
MAC operations based on simulated memristor crossbar array
Simulated precision loss due to ADC (accounting for chosen ADC size)
Activation arithmetic in digital logic e.g. GPU CUDA cores
Further layers/ output & evaluate performance
Implementation of simulated memristor chip as a hardware accelerator in a machine learning task

FIG. 1. Process flow for using the AIHWKIT[18] to evaluate the potential machine learning performance of an analog memory device intended for use in a crossbar array for weight storage or update.

II. EVALUATION OF ML PERFORMANCE OF NOVEL ANALOG MEMORIES

To evaluate the potential machine learning performance of our devices in a suitable task, we have used an open-source PyTorch toolkit developed by IBM called the Analog Hardware Acceleration Kit (AIHWKIT).[18] In Fig. 1, we outline the typical workflow for evaluating the use of memristors in a simulated crossbar array used for a machine learning task.[19]

Prior to using the toolkit, the performance of the device as a non-volatile memory should be evaluated, which involves testing the resistive switching of the device with voltage pulses and the non-volatility or retention characteristics of the programmed states. When designing a crossbar array of memristors for weight storage and update in a machine learning task, the conductance or resistance state of the device should then be programmed to increase and decrease in a linear and gradual manner (potentiation and depression, respectively). Since the weights held in digital logic can take values between -1 and 1, and conductance states represent weights in memristive hardware accelerators, a differential configuration of memristors (Fig. 1, panel 3) is often employed, as only positive conductances can be encoded in each individual device. Therefore, each memristor cell is composed of two memristors in the array, with one corresponding to positive weights and the other corresponding to negative weights. An example of the measurement and data extraction process is shown in the next section.

Once device characteristics have been determined, a machine learning task should be chosen, and an associated neural network can be programmed and evaluated in its ability to solve the task. By default, both training and inference will run on digital hardware such as CPU or GPU cores, locally or by utilizing cloud computing services. In our case, we ran our code locally on an NVIDIA RTX 3080 GPU, utilizing its compute unified device architecture (CUDA) cores.

Using the device conductance update and retention characteristics, a suite of device features and non-idealities can be configured for the digital hardware to simulate while solving the machine learning task. These include, but are not limited to, the device type (e.g., PCM, RRAM, etc.), the number of conductance states/bit resolution, voltage drops across rows and columns of the crossbar array due to interconnect resistance (IR drop), retention characteristics, update linearity and asymmetry, and peripheral circuitry features such as ADC/DAC size. In the context of simulating a crossbar array of memristors to store and update weights, all of these limitations and features in device performance are applied by modifying the way that weights (programmed in digital hardware) would change if the network were deployed onto analog hardware. Then, when the network deployed on simulated analog hardware is used for inference, the impact of programmable bit resolution, inaccurate weight programming, and limited retention on machine learning performance due to analog hardware characteristics can be evaluated by observing a difference in performance compared to the unmodified, SOTA digital hardware. It is worth mentioning that the network can also be trained on the simulated analog hardware.[21] In this case, Step 3 in Fig. 1 is performed before Step 2, such that the training is performed utilizing the performance parameters and non-idealities of the analog hardware. In the following sections, we explore several examples of evaluating the machine learning potential for memristive hardware based on our experimental HfO_xS_y/HfS_2 memristors.

III. POTENTIATION AND DEPRESSION

We extract relevant parameters for simulating image recognition performance of compliance-free and forming-free memristors based on a crystalline 2DLM semiconductor (HfS_2), which was partially dry oxidized to form the HfO_xS_y/HfS_2 structure shown in Fig. 2(a). Devices were measured on a Form Factor MPS150 probe station, connected to a Keysight B1500A Parameter Analyzer with remote sensing units and B1530A WGFMU (waveform generator/fast measurement unit) with a temporal resolution of 10 ns. The devices show stable non-volatile resistive switching when measured with fast voltage pulses [Fig. 2(b)].

Similar to other RRAM technologies,[5,8–10,18,22] our devices show non-linear conductance update characteristics when biased with repeated identical voltage pulses. However, although not ideal, this has been circumvented by using pulses with increasing voltage and pulse width [Fig. 2(c)] at the cost of increasing the required peripheral circuitry in a physical implementation of such a circuit.[8,9] Twenty pulses were used for both potentiation and depression to leave headroom to extract an optimal performance range, with one complete potentiating pulse train and one complete depressing pulse train constituting one programming cycle, therefore containing 40 programming pulses. To ensure robust characterization of our devices, read pulses were employed as -0.1 V, 30 μs pulses, 20 μs apart from programming pulses, avoiding any contribution to read currents from spurious charging or discharging capacitances due to the high-frequency operation. The voltages and pulse widths employed were low (<1 V and <350 ns, respectively) and are indicated in Fig. 2(c).

Potentiation and depression pulse trains were conducted on a single device for 100 cycles to determine the resilience of the device to repeated programming [Fig. 2(d)]. The raw conductance read data for each of the 100 cycles are shown superimposed on one complete programming cycle, with the average values plotted using a dashed line. From the data, we extract the conductance states obtained from potentiation and depression within the most linear range of both regimes [Fig. 2(d)]. We require a differential configuration of our devices to represent positive and negative weights; therefore, weight values (w) encoded by our devices must be represented by the difference in conductance of the memristors on a positive (G^+) and negative (G^-) branch ($w \propto G^+ - G^-$). In this configuration, each unique combination ($G^+ - G^-$) of quantized conductance states is assigned a "bin" number, and the separation between neighboring bins is defined as the bin width. This results in 31 total bins [or $\log_2(31) \sim 5$ bits] being accessible for reliable programming in both potentiation and depression combined [Fig. 2(e)]. We choose to program our devices to ~5 bits, as this is the point at which the precision of analog implementations can be superior to digital ones while not having much higher multiply-and-accumulate (MAC) energy, and is therefore a realistic application for our devices.[23]

In addition, the cycle-to-cycle variation, and consequently the standard deviation of each state, is also crucial for determining how reliably a memristor within a memristive chip can achieve a

18 December 2025 10:27:04

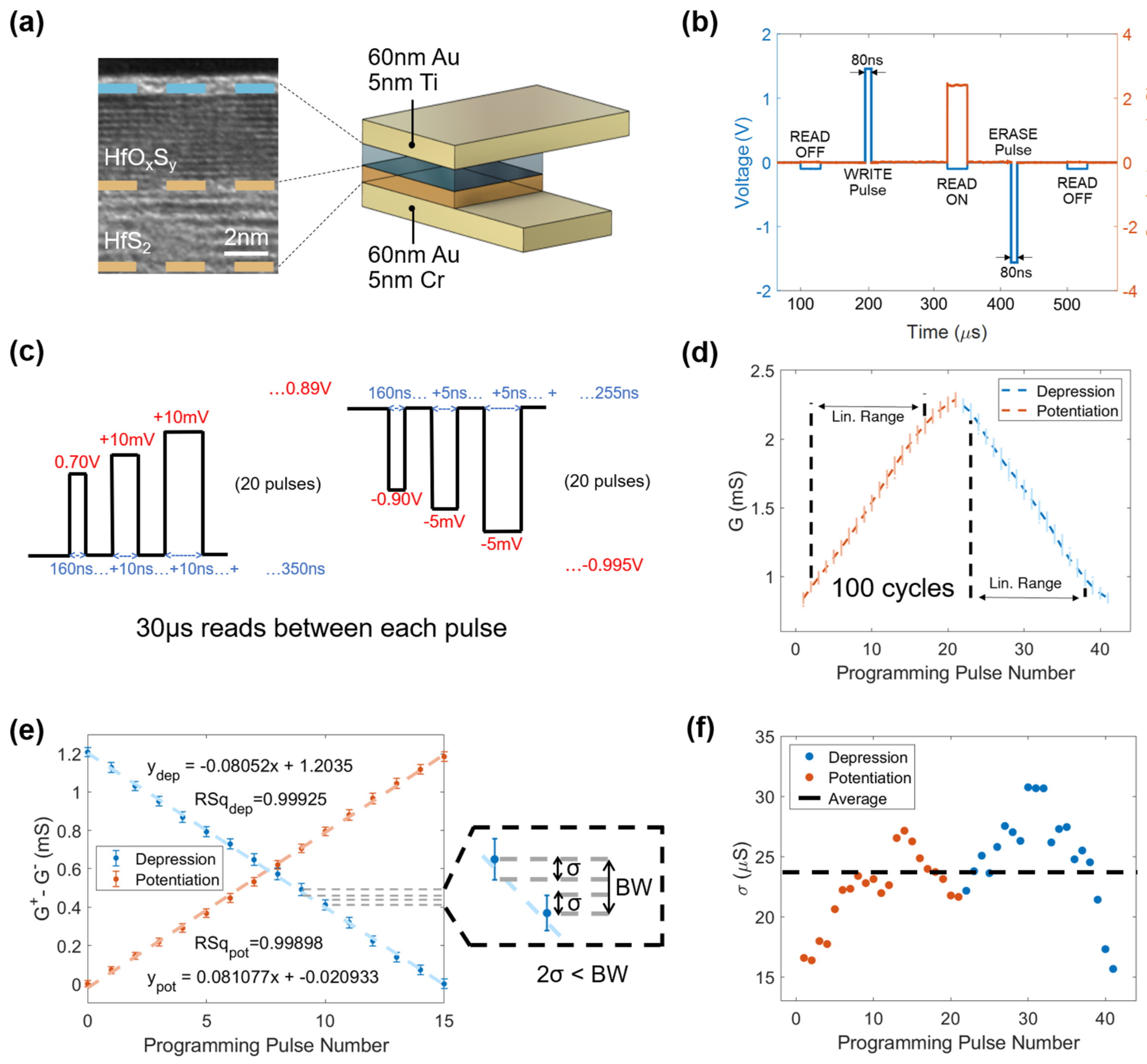


FIG. 2. (a) The forming-free, compliance-free device used in potentiation and depression experiments shows (b) robust non-volatile resistive switching with fast (80 ns) WRITE and ERASE pulses. (c) We employ a tailored pulsing scheme to achieve gradual and linear conductance update, which are required to attain the desired bit resolution for high-accuracy machine learning with our devices. (d) Distribution of conductances achieved using the pulsing scheme in (c), over 100 cycles. (e) Linear region extracted from (d), with average conductances and standard deviation fitted with linear functions. (f) A high degree of symmetry between potentiation and depression and low standard deviation at all conductance values allows for 31 bins (unique values of $G^+ - G^-$) to be defined with bin width (BW) > 2σ.

predicted or specified conductance when programmed with an associated pulse train. Within our linear range, each step between bins or average conductance states is encoded by the average bin width, 81.077 or 80.520 μS for programming in potentiation or depression, respectively. Crucially for machine learning accuracy, the standard deviations of neighboring states do not overlap. This is indicated between a pair of neighboring states in Fig. 2(e). Furthermore, comparing the gradients of the fitted lines for both potentiation and depression, we observe only a very small difference. A high degree of symmetry between potentiation and depression conductance update also positively influences machine learning accuracy and is present in our data. The high R-squared values for both lines (0.998 98 for potentiation and 0.999 25 for depression) also indicate how closely we can fit a linear conductance update model to our data, from which we will base our simulated crossbar array devices for machine learning. Figure 2(f) shows the standard deviation of each state, which can be taken at an average value of 23.700 μS. Overall, the device shows strong linear and symmetrical conductance update characteristics at low energy (23.74 ± 1.26 nJ total programming energy per complete potentiation/depression cycle, averaged over

18 December 2025 10:27:04

ten cycles), without requiring electroforming or current compliance, from which we can build a device model for simulating machine learning performance of a memristor chip.

IV. MNIST IMAGE CLASSIFICATION

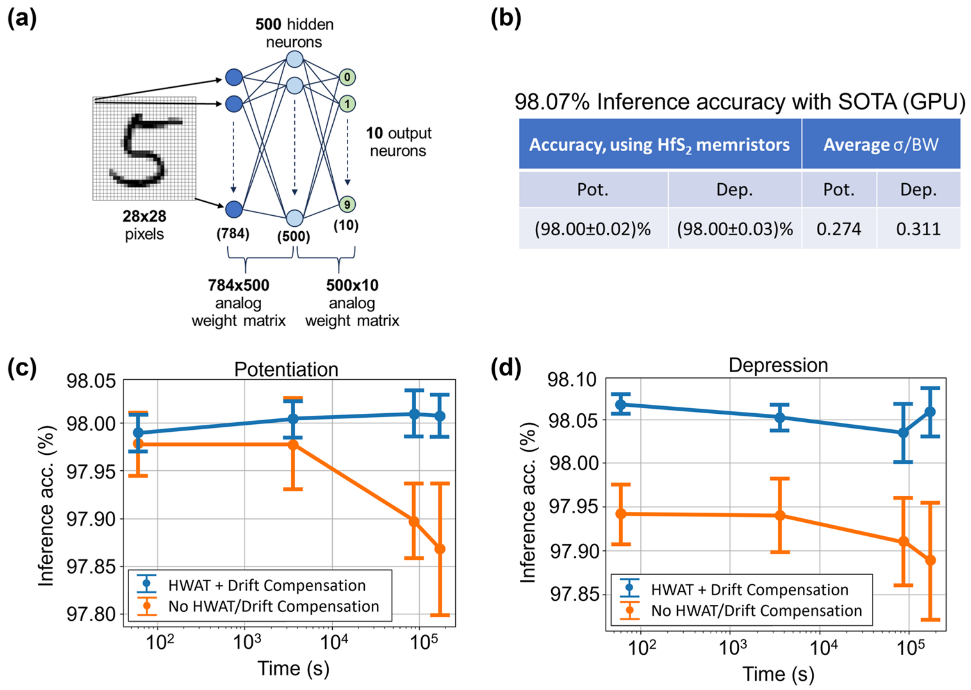
In Fig. 3(a), we show the network utilized to classify handwritten number images included in the MNIST dataset.[6] The 28 × 28 input images are flattened to a 1 × 784 vector in the first layer of the network. We use one hidden layer consisting of 500 neurons, and the network has a fully connected architecture where each neuron from one layer is connected to all neurons in the subsequent layer. We use this architecture as it has been shown to result in high-accuracy classification of handwritten numbers in the MNIST dataset.[24] Finally, in the last layer of the network in Fig. 3(a), a predicted handwritten number is determined from ten possible values (0–9). Deploying this network architecture on SOTA digital hardware (an NVIDIA RTX 3080 GPU) and training the network weights for 30 epochs results in an inference accuracy of 98.07%.

In the simulated analog implementation of this network, we employ a differential configuration of two memristors (one to represent positive weights and the other for negative weights) between each neuron, as described in the previous section and in Fig. 1, panel 3. When simulating deployment of the network on our memristive hardware, we map network weights to the number of programming pulses required to reach each corresponding analog weight within the linear operating range of the device, using the linear fit in Fig. 2(e). The "0" weight value is mapped to the start of the device's linear range.

Based on the electrical data in Fig. 2, we extracted relevant performance parameters pertaining to (i) cycle-to-cycle variation, (ii) linearity, (iii) symmetry, (iv) IR drop, (v) bit resolution, and (vi) the characteristics of conductance update in our HfO_xS_y/HfS_2 devices, which inform our device model. However, prior to deploying weights to our simulated arrays of HfO_xS_y/HfS_2 devices, we perform a further five epochs of hardware-aware training (HWAT) on the network. During HWAT, the network learns to ensure robust weight deployment to our HfO_xS_y/HfS_2 devices by retraining network weights on digital hardware for a small number of cycles while accounting for the characteristics of the analog hardware on which we wish to deploy the network. Although we have a good picture of device characteristics over a range of experimentally measured parameters (i–vi above), *in lieu* of data with similar statistical significance, we have simulated the impact of device-to-device variation informed by literature on arrays of HfO_2 memristors fabricated by a scalable method (ALD)[25] and arrays of hBN-based 2D layered memristors.[26] This has informed a baseline value of 30% for conductance update and how reliably we can achieve the minimum/maximum conductance states of our devices. Therefore, during HWAT, we account for our experimentally measured analog hardware characteristics (i–vi listed above) and for simulated device-to-device (DTOD) variation. Inference accuracy is subsequently extracted by simulating deployment of the HWAT-modified weights on our HfO_xS_y/HfS_2 devices and evaluating the proportion of correctly predicted handwritten numbers from an unseen test set from the MNIST dataset.

Simulations were conducted for both potentiation and depression, with separate noise characteristics corresponding to each programming mode. In both potentiation and depression (positive and negative weight update, respectively), we achieve 98.00% accuracy with low variation across five runs, only 0.07% lower than SOTA accuracy, showing the potential of this hardware to solve

18 December 2025 10:27:04



98.07% Inference accuracy with SOTA (GPU)

Accuracy, using HfS_2 memristors		Average σ/BW	
Pot.	Dep.	Pot.	Dep.
(98.00±0.02)%	(98.00±0.03)%	0.274	0.311

FIG. 3. (a) Network architecture employed for the MNIST image classification task. (b) High-accuracy inference results with five repetitions indicating consistent MNIST classification performance close to SOTA devices. Resilience of memristive hardware to drift in potentiation (c) and depression (d) given global drift compensation and hardware-aware retraining of the network weights.

© Author(s) 2025

machine learning tasks with high accuracy [Fig. 3(b)]. This is largely attributed to the linear operation and low σ/bin-width (BW) ratio in both programming regimes, extracted from Fig. 2(e).

In our simulations, we have also utilized the capability of the AIHWKIT to simulate the peripheral circuitry connecting analog memristor hardware and SOTA hardware, as memristive chips cannot be operated in isolation. We map analog weights to digital values and utilize 8-bit ADC and DACs, which, despite being higher resolution than our devices (which have shown ~5-bit operation), make our circuit more resilient to programming noise, albeit at a cost to total computation energy and chip area. Along with adaptive scaling of input data for the first few batches of training data (ensuring weights are represented as accurately as possible within the limitations of our analog hardware), this ensures that our simulation is not agnostic to the other peripheral hardware required to operate a memristive chip for use in a neural network for machine learning tasks.

To create an even more complete device model, we also utilized conductance drift data from the ReRAMWan2022 analog device model.[27] This model is based on electrical measurements of thousands of HfO_2 memristors, allowing us to reasonably predict inference accuracy drift for a future array of our own HfO_xS_y/HfS_2 devices fabricated by a scalable method. Over the same time span for which we observe <3% conductance drift across the whole range of conductances programmed in our devices,[17] the ReRAMWan2022 data show considerably more conductance drift, up to 30%,[27] which we take as a worst-case scenario baseline due to its higher statistical significance. We compare the retention of inference accuracy by simulating deployment on our HfO_xS_y/HfS_2 devices with HWAT and drift compensation against a baseline model, which we take to mean without HWAT and without any compensation for conductance drift described by the ReRAMWan2022 model[27] [Figs. 3(c) and 3(d)]. This shows that the impact of HWAT on MNIST inference accuracy is relatively small for our devices and therefore may be unnecessary. However, by applying global drift compensation, we observe superior retention of inference accuracy over the tested period compared to the baseline ReRAMWan2022 drift model[27] without compensation.

V. CIFAR-10 IMAGE CLASSIFICATION

To further evaluate the potential of our devices for machine learning applications, we chose an image classification task based on the CIFAR-10 dataset.[20] This dataset contains 60 000 32 × 32 pixel RGB images of ten different categorical items, including dogs, cats, frogs, and others. To classify the CIFAR-10 dataset, we implemented a convolutional neural network (CNN), shown in Fig. 4. This network is composed of three repeated blocks, where the input image is split into two branches. In the upper branch, it undergoes two convolutions—being scanned by a 3 × 3 filter to extract a number of reduced-dimension feature maps each time. After the first convolution in the upper branch, the feature maps are normalized with respect to a dynamically calculated mean and standard deviation (batch normalization, improving stability during training) and then passed through a rectified linear unit (ReLU) activation function, which introduces non-linearity into the data, improving the ability of the network to learn complex patterns. After the second convolution in the top branch, only a further ReLU operation is performed.

In the lower branch, only one convolution is performed, with a 1 × 1 filter to extract an equivalent number of feature maps to the other branch. This lower branch corresponds to the residuals, which are then combined with the result of the two convolutions in the upper branch and pooled to the maximum value in a 2 × 2 filter, which is passed over the different channels to reduce the dimensionality of the output of each block. The output of each preceding block becomes the input of the next. These blocks are one possible implementation of ResNet blocks, which have been shown to be successful in image recognition machine learning tasks.[28] We employ only three ResNet blocks, although many modern network architectures implement tens of these blocks to achieve very high accuracy in even more challenging tasks. However, this comes at the cost of increasing the number of trainable weights. Thus, we employ only three blocks to maintain the number of memristors required to implement this network relatively low. The final part of the network is a fully connected network with three layers, outputting a prediction from the ten possible categories. Given a differential configuration of memristors, in total, ~420 000 HfO_xS_y/HfS_2 devices would be required to store ~210 000 trained weights for the whole network, due to convolutions and other operations in each of the three blocks (see analog weight matrix dimensions in each panel, Fig. 4).

The network was first trained for 200 epochs to an inference accuracy of 87.51% [Fig. 5(a)] using SOTA hardware (an NVIDIA RTX 3080 GPU, as before). When simulating deployment of the network weights on our memristive hardware, as conducted for MNIST handwritten number classification, we observe a <0.9% decrease in accuracy despite all the memristor non-idealities we have implemented in our simulation, with only 20 hardware-aware retraining (HWAT) epochs of the deployed network weights.

We implemented device-to-device (DTOD) variation as in our MNIST simulations (Fig. 3). However, to inform future fabrication and estimate the impact of DTOD as well as other device non-idealities on inference accuracy, we varied the DTOD variation values and re-ran simulations for inference in CIFAR-10 [Fig. 5(b)]. The loss in accuracy of ~0.9%, which we report from our CIFAR-10 simulations, is achieved at 30% DTOD. Due to the low cycle-to-cycle variation, high linearity, and good symmetry in our potentiation/depression data, DTOD variation is the dominating factor causing loss in inference accuracy when simulating deployment on our HfO_xS_y/HfS_2 devices compared to SOTA digital hardware [Fig. 5(b)]. It is important to note that acceptable limits of accuracy loss compared to SOTA hardware are application-dependent. Despite not using a scalable fabrication method in our work, existing literature shows that our image classification accuracy scores are achievable within a realistic DTOD range of 30% for HfO_2-based memristors fabricated using a scalable method (ALD),[25] and for 2D materials-based memristors as well.[26] Similar to our MNIST simulations [Figs. 3(c) and 3(d)], we compare the network's resiliency to drift given a baseline network (uncompensated for drift and without HWAT) and a drift-compensated, HWAT-retrained network [Figs. 5(c) and 5(d)]. We observe a much larger variation in accuracy between the HWAT and baseline scores, of ~9%, highlighting the advantage of using HWAT in more challenging machine learning tasks to provide robust weight deployment on analog hardware.

18 December 2025 10:27:04

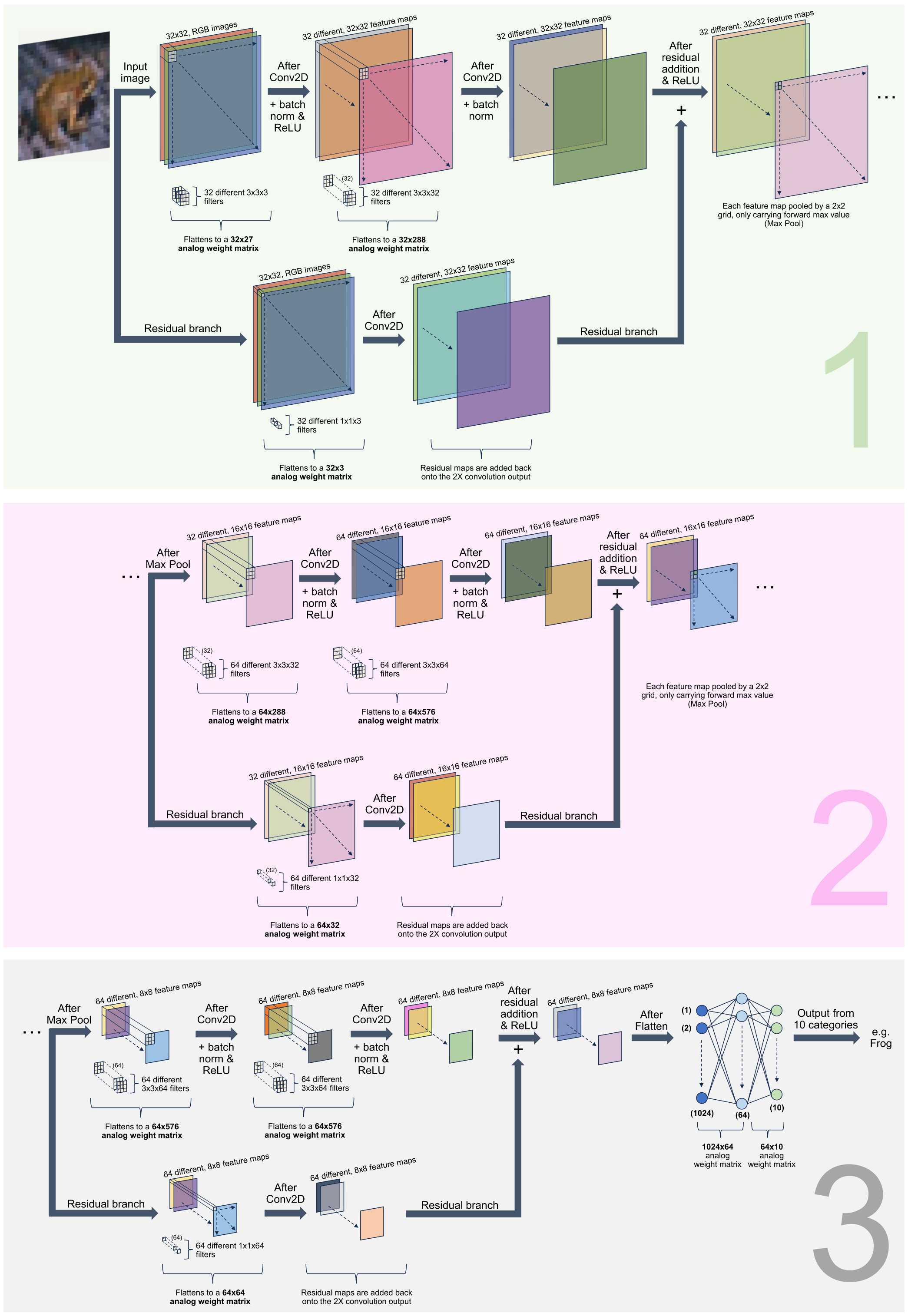
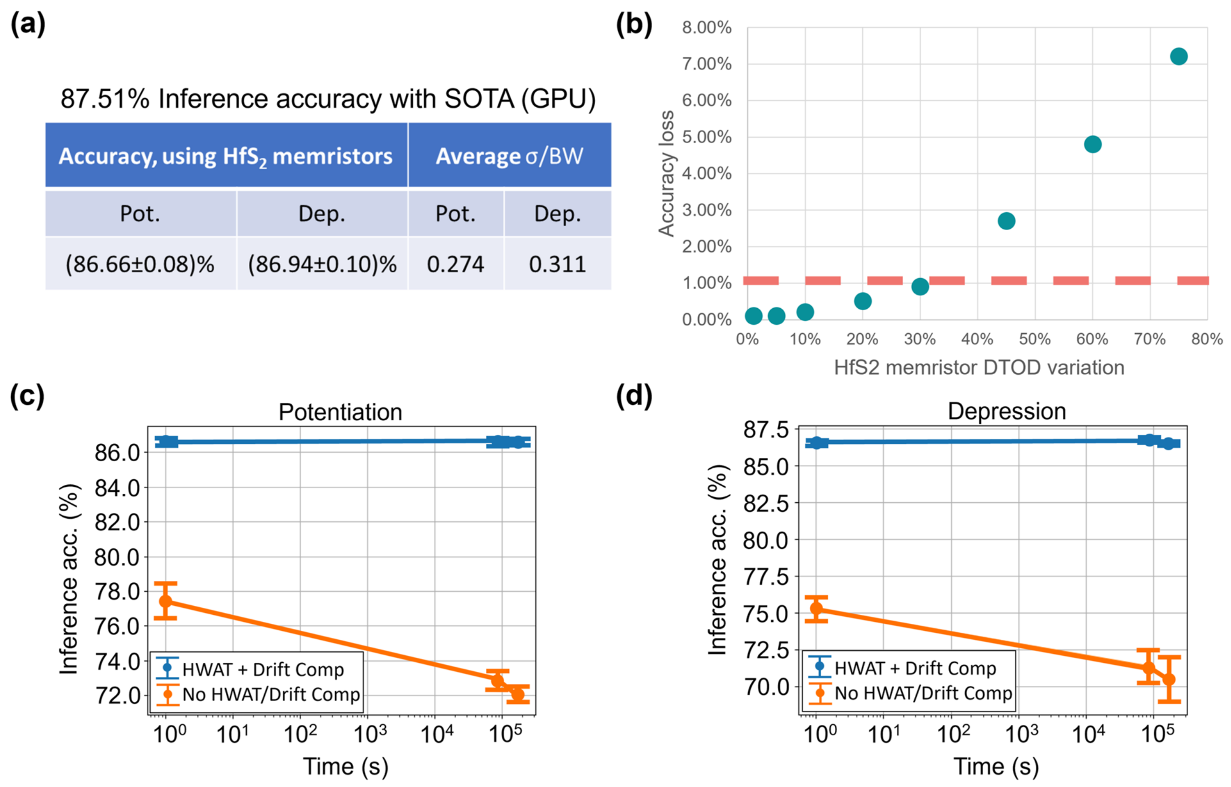

Input image
32x32, RGB images
After Conv2D + batch norm & ReLU
32 different, 32x32 feature maps
32 different 3x3x3 filters
Flattens to a 32x27 analog weight matrix
32 different 3x3x32 filters
Flattens to a 32x288 analog weight matrix
After Conv2D + batch norm
After residual addition & ReLU
Each feature map pooled by a 2x2 grid, only carrying forward max value (Max Pool)
Residual branch
After Conv2D
32 different 1x1x3 filters
Flattens to a 32x3 analog weight matrix
Residual maps are added back onto the 2X convolution output
1
After Max Pool
32 different, 16x16 feature maps
64 different, 16x16 feature maps
64 different 3x3x32 filters
Flattens to a 64x288 analog weight matrix
64 different 3x3x64 filters
Flattens to a 64x576 analog weight matrix
64 different 1x1x32 filters
Flattens to a 64x32 analog weight matrix
2
64 different, 8x8 feature maps
64 different 3x3x64 filters
Flattens to a 64x576 analog weight matrix
64 different 1x1x64 filters
Flattens to a 64x64 analog weight matrix
After Flatten
(1024) (64) (10)
1024x64 analog weight matrix
64x10 analog weight matrix
Output from 10 categories
e.g. Frog
3

FIG. 4. Convolutional neural network composed of three ResNet blocks and a fully connected layer. This network was trained with data from the CIFAR-10 dataset augmented with random horizontal flips, rotations, normalization, resizes, and crops. Simulated arrays of HfO_xS_y/HfS_2 device arrays act as analog weight matrices in the network.

18 December 2025 10:27:04


87.51% Inference accuracy with SOTA (GPU)

Accuracy, using HfS_2 memristors		Average σ/BW	
Pot.	Dep.	Pot.	Dep.
(86.66±0.08)%	(86.94±0.10)%	0.274	0.311

FIG. 5. (a) High-accuracy, low-variation inference results with five repetitions indicating consistent CIFAR-10 classification performance close to SOTA devices. (b) Device-to-device (DTOD) variation below 30% results in <1% drop in inference accuracy compared to SOTA devices. Resilience of memristive hardware to drift in potentiation (c) and depression (d) given global drift compensation and hardware-aware retraining of the network weights.

18 December 2025 10:27:04

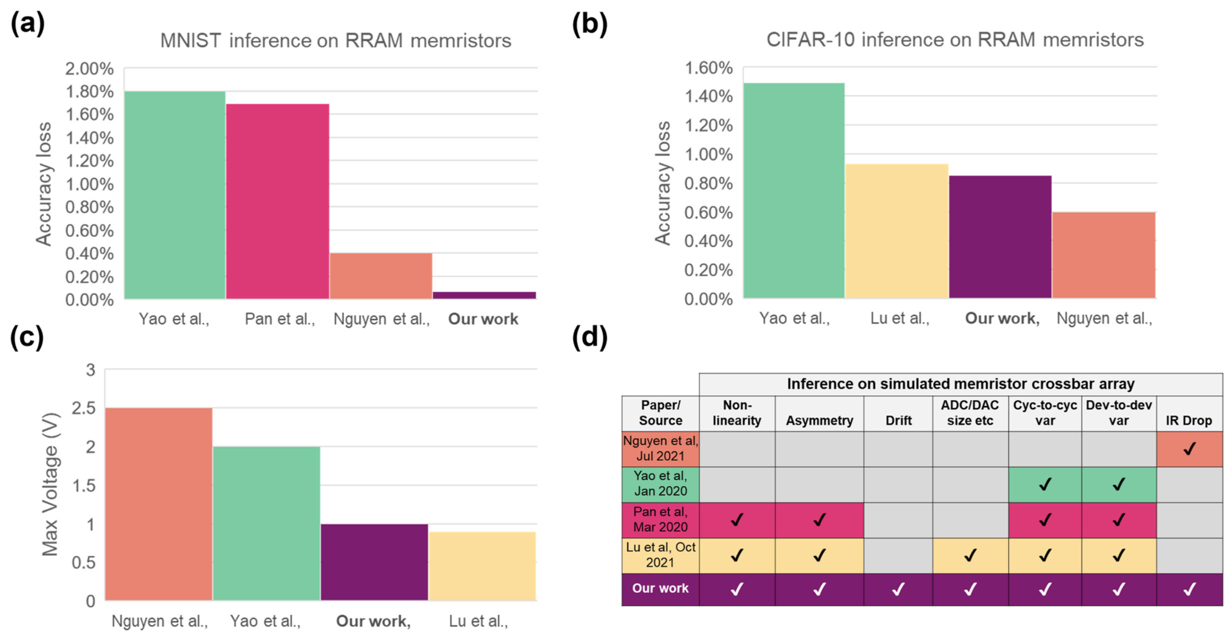
With global drift compensation applied, our network modeled on HfO_xS_y/HfS_2 memristor hardware retains its accuracy of 86.80%, compared to the baseline model without drift compensation and HWAT, which degrades in performance significantly over time.

VI. COMPARISON TO OTHER MEMRISTORS

Evaluating the performance of memristive devices in machine learning applications by simulating the impact of measured device characteristics is a common and important practice in the field. However, different authors use a variety of different toolkits, such as AIHWKIT,[18,21] XPESIM,[5,29] NeuroSIM,[10,30,31] and others.[22,32] In their simulations, authors deploy various neural network architectures to solve a variety of different machine learning tasks on different datasets and evaluate different types of memristive devices. Therefore, to meaningfully contextualize the performance of our devices, we compare our results to existing literature where potentiation and depression measurements of a few RRAM devices have been used to extract device parameters relevant for weight storage in a neural network. We believe the papers chosen are the most relevant, as Nguyen *et al.*[11] and Lu *et al.*[10] both employ pulsing schemes with increasing pulse heights in their potentiation and depression experiments. Furthermore, as in our work, Lu *et al.*[10] also use a chalcogenide switching layer in their Ag/SnS/Pt devices and do not require electroforming to operate their devices. Pan *et al.*[22] measure devices with a similar structure to ours ($TiN/HfO_2/Ti$), and Yao *et al.* also use a hafnia-based device stack ($TiN/TaO_x/HfO_x/TiN$). It is important to note that the MNIST inference result achieved by Yao *et al.* was performed fully in hardware, consisting of large memristor arrays connected to integrated programming and readout circuitry, highlighting a significant achievement in the field.[5] However, for inference on the CIFAR-10 dataset, the authors used a neural network with a much larger number of weights (which would require more memristors than they had fabricated). Therefore, for inference on the CIFAR-10 dataset, Yao *et al.* used a device model that considered the device-to-device and cycle-to-cycle variation they measured in their experimental hardware.[5]

In the examples chosen, inference accuracy of a network trained on the MNIST [Fig. 6(a)] and CIFAR-10 datasets [Fig. 6(b)] has been evaluated on both SOTA digital hardware and simulated RRAM hardware, allowing for comparison between the two.[5,10,11,22] Absolute accuracy was not used, as this depends strongly on the network architecture and size, which is not being evaluated here. We also compare the programming voltages used to update memristor weights in CIFAR-10 image classification [Fig. 6(c)]. While there are many other metrics by which the effectiveness of memristive

© Author(s) 2025



	Inference on simulated memristor crossbar array						
Paper/ Source	Non-linearity	Asymmetry	Drift	ADC/DAC size etc	Cyc-to-cyc var	Dev-to-dev var	IR Drop
Nguyen et al, Jul 2021							✓
Yao et al, Jan 2020					✓	✓	
Pan et al, Mar 2020	✓	✓			✓	✓	
Lu et al, Oct 2021	✓	✓		✓	✓	✓	
Our work	✓	✓	✓	✓	✓	✓	✓

FIG. 6. Loss in accuracy resulting from the deployment of neural networks on simulated analog memristive hardware compared to SOTA, in inference tasks based on test data from the (a) MNIST and (b) CIFAR-10 datasets. Using a simulated crossbar array of our devices, we observe low loss in accuracy for both datasets. (c) Our devices are able to retain high accuracy compared to similar existing literature, despite utilizing low programming voltages. (d) To increase the realism of our simulations, we account for a range of device non-idealities while being conscious of peripheral circuitry and drift.

18 December 2025 10:27:04

hardware for neural networks can be evaluated, accuracy degradation compared to SOTA in MNIST and CIFAR-10 classification and maximum voltage used during programming were three criteria that were available across a number of different works. Across these metrics, our devices show consistently low loss in accuracy compared to SOTA in both image classification tasks [Figs. 6(a) and 6(b)], while only requiring low programming voltages [Fig. 6(c)].

The table in Fig. 6(d) indicates the different device non-idealities that have been implemented in each image recognition simulation, as disclosed in the main texts or their associated supplementary material. Given all the non-idealities we have implemented in our simulation, we believe our simulations offer a realistic prediction of the performance of arrays of HfO_xS_y/HfS_2 devices in machine learning tasks. Our devices show promising machine learning performance and strong potential for maintaining high accuracy compared to SOTA hardware while being programmed with short pulses (<350 ns) at low voltages (<1.0 V). Despite the increased computational latency and chip area caused by requiring tailored pulsing schemes, which are used for many memristive devices,[8] including the work by Lu *et al.*,[10] Nguyen *et al.*,[11] and our own, our memristors are forming-free and compliance-free, which contributes toward enabling simplified operation and reduced area consumption for memristor-based chips.[33]

VII. CONCLUSIONS

We have shown that low-energy, fast-switching HfO_xS_y/HfS_2 memristor hardware can achieve highly linear and symmetric conductance update with high granularity, without requiring electroforming or current compliance. By using the IBM toolkit, we performed highly realistic simulations where not only the real device characteristics are considered but also the impact of a number of other factors, such as device-to-device variation, ADCs/DACs size, IR drop, and inference accuracy drift over time. The results show that high accuracy is achieved for inference on both the MNIST and CIFAR-10 datasets, showing the potential of resistive memories based on HfO_xS_y/HfS_2 semiconductor-insulator structures for future hardware accelerators. With further fine-tuning of device characteristics, our forming-free, compliance-free memristors based on HfO_xS_y/HfS_2 have the potential to enable energy-efficient, area-efficient, and highly accurate memristor chips for machine learning and neuromorphic computing.

ACKNOWLEDGMENTS

We acknowledge helpful discussions with Abin Varghese, Prabodh Katti, and Bipin Rajendran. We acknowledge funding from Grant No. EP/T517793/1.

AUTHOR DECLARATIONS

Conflict of Interest

The authors have no conflicts to disclose.

Author Contributions

Aferdita Xhameni: Conceptualization (equal); Data curation (lead); Formal analysis (equal); Investigation (lead); Methodology (lead); Software (lead); Writing – original draft (lead); Writing - review & editing (equal). **Antonio Lombardo**: Conceptualization (equal); Formal analysis (equal); Funding acquisition (lead); Investigation (supporting); Project administration (lead); Resources (lead); Supervision (lead); Writing - original draft (supporting); Writing – review & editing (equal).

DATA AVAILABILITY

The data that support the findings of this study are available from the corresponding author upon reasonable request.

REFERENCES

[1] A. Mehonic and A. J. Kenyon, "Brain-inspired computing needs a master plan," Nature **604**(7905), 255–260 (2022).

[2] N. Jones *et al.*, "How to stop data centres from gobbling up the world's electricity," Nature **561**(7722), 163–166 (2018).

[3] D. Marković, A. Mizrahi, D. Querlioz, and J. Grollier, "Physics for neuromorphic computing," Nat. Rev. Phys. **2**(9), 499–510 (2020).

[4] A. Mehonic, A. Sebastian, B. Rajendran, O. Simeone, E. Vasilaki, and A. J. Kenyon, "Memristors—From in-memory computing, deep learning acceleration, and spiking neural networks to the future of neuromorphic and bio-inspired computing," Adv. Intell. Syst. **2**(11), 2000085 (2020).

[5] P. Yao, H. Wu, B. Gao, J. Tang, Q. Zhang, W. Zhang, J. J. Yang, and H. Qian, "Fully hardware-implemented memristor convolutional neural network," Nature **577**(7792), 641–646 (2020).

[6] Y. LeCun, C. Cortes, C. Burges *et al.* (2010), "MNIST handwritten digit database," MNIST, see http://yann.lecun.com/exdb/mnist/.

[7] K. Zhu, S. Pazos, F. Aguirre, Y. Shen, Y. Yuan, W. Zheng, O. Alharbi, M. A. Villena, B. Fang, X. Li *et al.*, "Hybrid 2D–CMOS microchips for memristive applications," Nature **618**(7963), 57–62 (2023).

[8] P.-Y. Chen, B. Lin, I.-T. Wang, T.-H. Hou, J. Ye, S. Vrudhula, J.-s. Seo, Y. Cao, and S. Yu, "Mitigating effects of non-ideal synaptic device characteristics for on-chip learning," in *2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)* (IEEE, 2015), pp. 194–199.

[9] A. Gebregiorgis, A. Singh, S. Diware, R. Bishnoi, and S. Hamdioui, "Dealing with non-idealities in memristor based computation-in-memory designs," in *2022 IFIP/IEEE 30th International Conference on Very Large Scale Integration (VLSI-SoC)* (IEEE, 2022), pp. 1–6.

[10] X. F. Lu, Y. Zhang, N. Wang, S. Luo, K. Peng, L. Wang, H. Chen, W. Gao, X. H. Chen, Y. Bao *et al.*, "Exploring low power and ultrafast memristor on p-type van der Waals SnS," Nano Lett. **21**(20), 8800–8807 (2021).

[11] T. V. Nguyen, J. An, and K.-S. Min, "Memristor-CMOS hybrid neuron circuit with nonideal-effect correction related to parasitic resistance for binary-memristor-crossbar neural networks," Micromachines **12**(7), 791 (2021).

[12] A. Glukhov, V. Milo, A. Baroni, N. Lepri, C. Zambelli, P. Olivo, E. Pérez, C. Wenger, and D. Ielmini, "Statistical model of program/verify algorithms in resistive-switching memories for in-memory neural network accelerators," in *2022 IEEE International Reliability Physics Symposium (IRPS)* (IEEE, 2022), pp. 3C–3.

[13] Y. Sun, X. Yan, X. Zheng, Y. Liu, Y. Zhao, Y. Shen, Q. Liao, and Y. Zhang, "High on–off ratio improvement of ZnO-based forming-free memristor by surface hydrogen annealing," ACS Appl. Mater. Interfaces **7**(13), 7382–7388 (2015).

[14] S. Ng, R. A. John, J.-t. Yang, and N. Mathews, "Forming-less compliance-free multistate memristors as synaptic connections for brain-inspired computing," ACS Appl. Electron. Mater. **2**(3), 817–826 (2020).

[15] B. Yang, N. Xu, C. Li, C. Huang, D. Ma, J. Liu, D. Arumí, and L. Fang, "A forming-free ReRAM cell with low operating voltage," IEICE Electron. Express **17**(22), 20200343 (2020).

[16] A. A. AlMutairi, A. Xhameni, X. Guo, I. Chircă, V. Nicolosi, S. Hofmann, and A. Lombardo, "Controlled fabrication of native ultra-thin amorphous gallium oxide from 2D gallium sulfide for emerging electronic applications," Adv. Mater. Interfaces **12**, 2400481 (2024).

[17] A. Xhameni, A. A. AlMutairi, X. Guo, I. Chircă, T. Wen, S. Hofmann, V. Nicolosi, and A. Lombardo, "Forming and compliance-free operation of low-energy, fast-switching HfO_xS_y/HfS_2 memristors," Nanoscale Horiz. **10**, 616 (2025).

[18] 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.

[19] M. Le Gallo, C. Lammie, J. Büchel, F. Carta, O. Fagbohungbe, C. Mackin, H. Tsai, V. Narayanan, A. Sebastian, K. El Maghraoui, and M. J. Rasch, "Using the IBM analog in-memory hardware acceleration kit for neural network training and inference," APL Mach. Learn. **1**(4), 041102 (2023).

[20] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Technical Report, University of Toronto (2009).

[21] Y. Kim, H. Kim, S. Jeon, H. W. Kim, E. Hong, N. Kim, H. Choi, H. Park, J. Jeong, D. Lee, and J. Woo, "Linear synaptic weight update in selector-less HfO_2 RRAM using Al_2O_3 built-in resistor for neuromorphic computing systems," IEEE Trans. Electron Devices **71**(8), 4637–4643 (2024).

[22] W.-Q. Pan, J. Chen, R. Kuang, Y. Li, Y.-H. He, G.-R. Feng, N. Duan, T.-C. Chang, and X.-S. Miao, "Strategies to improve the accuracy of memristor-based convolutional neural networks," IEEE Trans. Electron Devices **67**(3), 895–901 (2020).

[23] B. Murmann, "Mixed-signal computing for deep neural network inference," IEEE Trans. VLSI Syst. **29**(1), 3–13 (2020).

[24] N. Manral, "MLP-training-for-MNIST-classification" (2019); https://github.com/nipunmanral/MLP-Training-For-MNIST-Classification?tab=readme-ov-file#readme/.

[25] E. Pérez, D. Maldonado, C. Acal, J. E. Ruiz-Castro, F. J. Alonso, A. M. Aguilera, F. Jiménez-Molinos, Ch. Wenger, and J. B. Roldán, "Analysis of the statistics of device-to-device and cycle-to-cycle variability in TiN/Ti/Al: HfO_2/TiN RRAMs," Microelectron. Eng. **214**, 104–109 (2019).

[26] S. S. Teja Nibhanupudi, A. Roy, D. Veksler, M. Coupin, K. C. Matthews, M. Disiena, Ansh, J. V. Singh, I. R. Gearba-Dolocan *et al.*, "Ultra-fast switching memristors based on two-dimensional materials," Nat. Commun. **15**(1), 2334 (2024).

[27] W. Wan, R. Kubendran, C. Schaefer, S. B. Eryilmaz, W. Zhang, D. Wu, S. Deiss, P. Raina, H. Qian, B. Gao *et al.*, "A compute-in-memory chip based on resistive random-access memory," Nature **608**(7923), 504–512 (2022).

[28] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (IEEE, 2016), pp. 770–778.

[29] W. Zhang, X. Peng, H. Wu, B. Gao, H. He, Y. Zhang, S. Yu, and H. Qian, "*Design guidelines of RRAM based neural-processing-unit: A joint device-circuit-algorithm analysis*," in *Proceedings of the 56th Annual Design Automation Conference 2019* (ACM, 2019), pp. 1–6.

[30] X. Peng, S. Huang, Y. Luo, X. Sun, and S. Yu, "DNN+ NeuroSim: An end-to-end benchmarking framework for compute-in-memory accelerators with versatile

18 December 2025 10:27:04

device technologies," in *2019 IEEE International Electron Devices Meeting (IEDM)* (IEEE, 2019), pp. 32–35.
[31]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 Trans. Comput.-Aided Des. Integr. Circuits Syst. **40**(11), 2306–2319 (2020).
[32]C. Lammie, W. Xiang, B. Linares-Barranco, and M. Rahimi Azghadi, "MemTorch: An open-source simulation framework for memristive deep learning systems," Neurocomputing **485**, 124–133 (2022).
[33]Y. Huang, T. Ando, A. Sebastian, M.-F. Chang, J. J. Yang, and Q. Xia, "Memristor-based hardware accelerators for artificial intelligence," Nat. Rev. Electr. Eng. **1**(5), 286–299 (2024).

18 December 2025 10:27:04