electronics

MDPI

Article

PySpice-Simulated In Situ Learning with Memristor Emulation for Single-Layer Spiking Neural Networks

Sorin Liviu Jurj

Computers and Information Technology, "Politehnica" University of Timisoara, 2 V. Parvan Blvd, 300223 Timisoara, Romania; jurjsorinliviu@yahoo.de

Abstract: This paper presents a novel approach to in situ memristive learning by training spiking neural networks (SNNs) entirely within the circuit using memristor emulators in SPICE. The circuit models neurons using Lapicque neurons and employs pulse-based spike encoding to simulate spike-timing-dependent plasticity (STDP), a key learning mechanism in SNNs. The Lapicque neuron model operates according to the Leaky Integrate-and-Fire (LIF) model, which is used in this study to model spiking behavior in memristor-based SNNs. More exactly, the first memristor emulator in PySpice, a Python library for circuit simulation, was developed and integrated into a memristive circuit capable of in situ learning, named the "In Situ Memristive Learning Method for Pattern Classification". This novel technique enables time-based computation, where neurons accumulate incoming spikes and fire once a threshold is reached, mimicking biological neuron behavior. The proposed method was rigorously tested on three diverse datasets: XPUE, a custom non-dominating 3×3 image dataset; a 3×5 digit dataset ranging from 0 to 5; and a resized 10×10 version of the Modified National Institute of Standards and Technology (MNIST) dataset. The neuromorphic circuit achieved successful pattern learning across all three datasets, outperforming comparable results from other in situ training simulations on SPICE. The learning process harnesses the cumulative effect of memristors, enabling the network to learn a representative pattern for each label efficiently. This advancement opens new avenues for neuromorphic computing and paves the way for developing autonomous, adaptable pattern classification neuromorphic circuits.

Keywords: neuromorphic circuits; spiking neural networks; in situ training; memristive learning; PySpice; Ngspice; memristor emulator; memristors

check for updates

Citation: Jurj, S.L. PySpice-Simulated In Situ Learning with Memristor Emulation for Single-Layer Spiking Neural Networks. *Electronics* **2024**, *13*, 4665. https://doi.org/10.3390/electronics13234665

Academic Editor: Hyungjin Kim

Received: 4 October 2024
Revised: 16 November 2024
Accepted: 22 November 2024
Published: 26 November 2024

Copyright: © 2024 by the author. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

The brain is regarded as the most essential organ for many living species experiencing life on this planet, admired for its extraordinary learning capabilities and coordination skills, which have inspired generations. Memristive devices have shown promise in replicating synaptic functions, enabling neuromorphic systems that mimic the adaptability and efficiency of biological neural networks [1]. These devices demonstrate the capacity to emulate various synaptic behaviors, resulting in energy-efficient neural networks that can perform complex tasks with learning efficiencies comparable to those observed in the brain [2]. Additionally, memristive hardware has the potential to transcend the limitations of conventional von Neumann architectures, particularly within artificial intelligence, by offering compact, power-efficient alternatives [3].

The transition from simple memristive devices to fully developed neuromorphic systems has provided AI systems with increased efficiency and scalability, paving the way for more advanced, brain-inspired computing models [4]. The ability of memristors to emulate synaptic plasticity—the capacity of biological synapses to strengthen or weaken over time in response to activity—is integral to creating neural networks that can learn and adapt similarly to biological systems [5]. By leveraging the processing capabilities of emerging memristive devices, neuromorphic computing is evolving to mimic the complex, parallel

processing characteristics of the brain, enabling a transformative impact on AI architectures [6]. Furthermore, memristive synapses serve as key components for constructing scalable, energy-efficient neural networks, facilitating the development of neuromorphic AI that approaches the brain's learning and coordination capacities [7]. This can be easily seen in the various scientific publications made in recent decades in different domains through the artificial intelligence (AI) algorithms, systems, and applications proposed by researchers in academia and industry [8]. The fascination for the human brain can be seen in the Human Brain Project (HBP), which ended in September 2023 and was funded by the European Union with around EUR 600 million for ten years, involving about 500 scientists publishing thousands of research articles [9,10]. Due to these tremendous collective efforts, our society is developing at a faster rate than ever before. On one hand, this is happening thanks to modern computers being more powerful than ever in processing the vast amount of digital data available using modern GPUs, and on the other hand, due to AI-based software being included in almost every existent domain to obtain the most benefits possible out of these data.

However, as Moore's law is predicted to come to an end soon, recently, more research efforts [11] have been made in the direction of solving the three main problems our modern computers based on the von Neumann architecture have which include: (a) data movement and memory bottleneck—the problem of data processing taking too much time due to the processing elements (CPU) and the memory being located separately in hardware; (b) the problem of energy efficiency, especially regarding AI training and inference; and (c) memory is volatile. All of these problems exist mainly due to physical (mechanical and electrical) limitations, more precisely, due to the two-dimensional (2D) silicon-based PCBs primarily being used, and due to the overuse of circuit elements such as transistors, which are volatile and lose data once the power source is absent [12–14].

1.1. Overview of Neuromorphic Computing

Recently, neuromorphic computing has emerged as a promising domain for addressing the limitations of traditional computing architectures. This approach encourages the development of brain-inspired algorithms, methods, and applications, focusing on both hardware and software advancements [15,16]. Neuromorphic computing technologies are particularly valuable for their potential to enhance computational efficiency and scalability, with characteristics that make them attractive as future computing paradigms. As recent studies highlight, the roadmap for neuromorphic computing involves not only addressing the current computing challenges but also exploring emerging technologies, innovative materials, and novel computing concepts, paving the way for future advancements in AI [16]. Furthermore, as the field progresses, neuromorphic systems are being optimized to support a range of AI algorithms and applications, demonstrating their transformative potential in enabling energy-efficient, adaptable computing models that mimic brain-like processing [15]. Despite many AI solutions being proposed in software, only a few were proposed in hardware. This is a problem because AI models still run training and inference mainly on von Neumann architectures, thus limiting their potential and scope. This is where the memristor, also commonly referred to as Resistive Random-Access Memory (RRAM) [17], comes into play, claimed by many researchers to be the fourth fundamental electrical component after the resistor, capacitor, and inductor. This claim, among other things, was challenged in 2023 by the authors in [18], who emphasize distinguishing between physical reality and mathematical fiction, advocating for a grounded approach that aligns with experimental evidence and physics principles. The memristor was first described and named in 1971 by Leon Chua [19], with its first physical conceptualization taking place in 2008 by a group of engineers from HP Labs [20,21]. The memristor has a non-volatility property: remembering the last electrical charge that passed through it and in what direction, even when the power is turned off [22,23]. In addition, memristors can store multi-bit data, have reduced power consumption, and increased performance, being able to accelerate matrix–vector multiplication, a fundamental operation in neural

networks, by leveraging the unique voltage–current relationship and memory retention characteristics depending on the history of the voltage and current that has been applied to them according to Ohm's Law and Kirchhoff's Current Law [24–26]. In recent years, different types of memristors have been realized that have more than ten years of data storage retention time, 10^5 orders of magnitude on/off resistance ratios, over 10^{12} long endurance, and fast switching speeds of less than 15 nanoseconds [1], with Leon Chua themself proposing a homemade memristor that costs under USD 10 in summer of 2023, called Poor Man's Memristor: Chua Corsage Memristor [27,28]. Memristors hold substantial promise for neuromorphic architectures due to their memory retention and synaptic emulation capabilities, addressing limitations in conventional computing architectures. Its amazing properties made researchers curious to look into nature, where they recently discovered that mushrooms are also memristors [29,30], having the potential to advance the field of hybrid electronic systems.

Many efforts are made in developing neuromorphic circuits and AI applications using memristors [31–39]. Neuromorphic computing is increasingly seen as a promising approach to solving the limitations of traditional von Neumann architectures, such as energy inefficiency and data bottlenecks. The ability of memristors to offer non-volatility, low power consumption, and efficient storage makes them central to the development of new neuromorphic architectures.

1.2. Sustainability and Energy Efficiency

Knowing that the human brain consumes around 20 Wh of power while weighing only 1.4 kg [3] whereas the world's fastest supercomputers (e.g., Hewlett Packard Enterprise Frontier, or OLCF-5) have a power consumption between 20 and 30 Megawatts [40] highlights the significant gap in energy efficiency between biological systems and von Neumann-based computers. This inefficiency raises concerns about the impact that traditional computing architectures have on sustainability, particularly with the rise in large-scale AI models. For example, training models like GPT-3, a large language model with 175 billion parameters, were estimated to consume around 1.287 MWh of electricity, generating 552 tons of CO_2 [41,42]. In contrast, the inference of AI systems increases power consumption and CO_2 emissions even further [43]. Consequently, there has been a growing movement in academia and industry toward green AI [44–48], which aims to mitigate the environmental impact of AI technologies. Neuromorphic computing, inspired by the energy efficiency of biological brains, offers a potential solution. It is predicted that the neuromorphic computing market will grow from USD 200 million in 2025 to USD 20 billion by 2035, illustrating its expected role in addressing these sustainability challenges [49].

Additionally, printed electronics [50–54] offer a scalable and flexible approach for fabricating memristor-based circuits, reducing production costs and enabling customization for neuromorphic applications. The combination of printed technology with memristors holds the potential to advance adaptable, low-cost neuromorphic systems, facilitating a shift towards sustainable and high-performance computational designs.

The main contributions of this paper are as follows:

- Introduction of the "In Situ Memristive Learning Method for Pattern Classification": This work introduces a novel method utilizing memristor emulation [55] in PySpice for real-time learning and pattern classification directly within the circuit. The memristor emulator is a key technical advancement, as it allows the network to perform in situ learning based on the Hebbian learning principle, bypassing the need for a separate training phase.
- Creation of the XPUE dataset: A custom dataset of 3 × 3 binary images, called the XPUE dataset, was developed. This dataset is unique because no image dominates another, providing a challenging benchmark for testing the network's pattern recognition capabilities.
- Successful testing on multiple datasets: The proposed method was evaluated on three datasets: XPUE, a 012345 digit dataset, and a resized 10 × 10 MNIST dataset. The

network achieved 100% accuracy on the XPUE and 012345 datasets, and a noteworthy 60% test accuracy on the MNIST dataset using the first 100 samples. This is a significant improvement over previous SPICE-based approaches, which achieved only 30% accuracy.

- PySpice-based simulation: One of the key contributions is the development and demonstration of the first memristor emulator in PySpice. This is a pivotal tool that allows for advanced circuit simulation in neuromorphic hardware, making it accessible for further research and development.
- Groundwork for 3D neuromorphic circuits with printed memristors: While the current work is focused on simulation, it lays the groundwork for the future development of 3D neuromorphic circuits using printed memristors [56]. The simulations indicate that transitioning to 3D designs could lead to significant reductions in energy consumption and space requirements, further enhancing the scalability and efficiency of neuromorphic computing systems.

The paper is organized as follows: In Section 2, the related work is presented. Section 3 details the proposed method for in situ memristive learning for pattern classification. Section 4 describes the experimental setup and results. Section 5 presents the limitations of this study and future work. Finally, in Section 6, the conclusions of this paper are presented.

2. State-of-the-Art

2.1. Memristors and Spiking Neural Networks in Neuromorphic Computing

The field of memristor-based neuromorphic computing is extensively studied, with key research highlighting the potential of these devices in bridging the gap between biological and artificial intelligence (AI). Seminal works have laid the groundwork for using memristors in spiking neural networks (SNNs) and demonstrated practical advancements in pattern recognition, energy efficiency, and real-time adaptability. Studies show that by leveraging mechanisms like spike-timing-dependent plasticity (STDP) and other Hebbian learning algorithms, memristors can emulate essential neural behaviors with significant energy savings.

Despite the availability of diverse neural network architectures, SNNs stand out due to their biological resemblance [57]. They have surged in popularity recently [58–62] because they interact via spikes (analogous to action potentials of approximately 100 mV) and demonstrate energy efficiency compared to traditional non-spiking neural networks [63,64]. Particularly, SNNs become more effective when integrated with Artificial Neural Networks (ANNs) [65,66]. A range of training algorithms is available, including direct training, temporal backpropagation, and ANN-to-SNN conversion [67]. Among these, STDP is the most commonly used unsupervised method for adjusting synaptic weights in response to neuron firing patterns [68,69]. In terms of neuron functionality, models such as Integrate and Fire (IF), Leaky Integrate and Fire (LIF), Izhikevich, and Hodgkin–Huxley have achieved a strong resemblance to biological neurons in neuromorphic systems [60,70,71].

2.2. Hardware Implementations and Architecture of Memristor-Based ANNs

A comprehensive review of the hardware implementations of memristor-based ANNs, particularly for image classification, covers the key architectural components, including image capture and input vector formation, vector–matrix multiplication (VMM) cores, sensing electronics, and synaptic weight updates [72]. This review details both ex situ and in situ training methodologies, ANN training algorithms, and weight programming techniques, alongside the fabrication and integration of ANN chips across multiple abstraction levels. This work provides valuable insights for researchers interested in designing robust memristive neural networks for practical applications.

Researchers have also developed CMOS-based circuits to emulate memristor behavior. For instance, a compact CMOS circuit was presented in [73] to replicate the pinched hysteresis loop characteristic of memristors holding analog memory states. This design enables efficient analog computation and provides a low-risk, scalable alternative to emerg-

ing memristive devices. Similarly, a full CMOS transistor-level circuit implementing an unsupervised SNN with STDP learning on-chip was developed for pattern recognition tasks, demonstrating the feasibility of combining CMOS technology with memristive functionalities in large-scale neuromorphic circuits [74].

2.3. Advances in Printed and Flexible Memristor Technology

Printed memristors represent a flexible, cost-effective solution for neuromorphic systems, enabling adaptable and lightweight designs. The progress in organic and hybrid memristors, highlighted in recent reviews [75], emphasizes solution-based printing methods suitable for flexible substrates. For example, an inkjet-printed tungsten oxide memristor demonstrates dual capabilities as a memory and synaptic device, offering robust performance for neuromorphic applications [76]. Another study introduced a completely printed, flexible neuron constructed on a polyimide substrate employing graphene ink and In2O3, designed for low-power, self-powered portable systems [77]. These advancements suggest promising avenues for energy-efficient, high-density neuromorphic systems.

2.4. Structural Innovations and Applications of Memristor-Based Neuromorphic Systems

Several studies emphasize the architectural innovations enabled by memristor-based systems, overcoming von Neumann bottlenecks to facilitate parallel processing and efficient data handling [78–84]. Also, several studies show that memristor-based systems can directly support ANNs [85–88], enabling the circulation of data throughout their circuits and aiding the implementation of the networks' forward and backward propagation strategies. For instance, the "Mosaic" architecture integrates distributed memristors for efficient routing and computation within memory, enabling high-precision performance in edge computing tasks through a unique network topology [89]. Other works explore energy-efficient designs, including the development of solar-powered memristor-based systems [90].

A noteworthy example is the STELLAR neuromorphic chip, which achieves 97% energy reduction in edge learning tasks. Utilizing a novel architecture, this chip showcases on-chip learning enhancements across various tasks like motion control and image classification [91]. This architecture highlights the potential of memristor-based circuits in advanced AI applications by achieving software-level accuracy with minimal hardware costs.

2.5. Emerging 3D Memristor Arrays for High-Density Neuromorphic Circuits

Conventional 2D crossbar arrays limit the potential of neuromorphic systems, prompting research into 3D architectures. Recent studies propose innovative methods for constructing 3D memristor arrays, allowing for more complex neural networks and efficient parallel operations [92,93]. For example, an eight-layer 3D circuit configuration has been designed to implement Convolutional Neural Networks (CNNs) directly, achieving competitive accuracy in image recognition tasks like the Modified National Institute of Standard and Technology (MNIST) dataset [92]. Another study introduced a 3D crossbar design that reduces area and increases synaptic density by folding the crossbar array, effectively lowering parasitic effects and enhancing neuromorphic performance [94].

Furthermore, the integration of advanced 3D neuromorphic systems that utilize monolithic integration techniques offers significant improvements in energy efficiency and latency, making them highly suitable for real-time applications. For instance, a 3D neuromorphic system that employs the Whetstone training algorithm combined with NeuroSIM for co-design simulations has demonstrated reductions in power consumption and latency, positioning it as a promising solution for advanced computing needs [95].

These studies emphasize the need for adaptive circuits capable of in situ training, a core focus of this work.

This work distinguishes itself from the related work mentioned above by proposing a novel in situ memristive learning method for pattern classification using a PySpice [96] memristor emulator. The proposed approach focuses on real memristors rather than modified emulators and demonstrates the potential for neuromorphic 3D circuits using printed

memristors. More precisely, the first memristor emulator in PySpice [96] was developed, a pioneering achievement that enables the simulation and analysis of memristive circuits in a Python-based environment. This emulator consists of an NMOSFET, a capacitor, a voltage-controlled current source, and a switch controlled by an additional input voltage, allowing for the accurate modeling of memristor behavior.

The proposed in situ memristive learning method leverages the cumulative effect of memristors to learn representative patterns for each label in a given dataset. This method was tested on three datasets: XPUE, a custom non-dominating 3×3 image dataset; a 3×5 digit dataset from 0 to 5 (based on [74]); and a resized 10×10 version of MNIST. This circuit successfully learned the patterns in all three datasets, achieving 100% accuracy on the XPUE and "012345" datasets and 60% test accuracy on the first 100 samples from the MNIST test set, surpassing the results obtained by other SPICE-based simulations (e.g., [97], which obtained a test accuracy of 30%).

Although the neuromorphic circuit was not physically printed in this study, this work lays the foundation for the future development of neuromorphic 3D circuits using printed memristors. The experimental results demonstrate the possibility of creating real neuromorphic 3D circuits that can learn various tasks using less energy and area footprint compared to their 2D counterparts. The 3D design can potentially improve learning by increasing the number of connections between neurons, similar to the biological brain, overcoming standard approaches that only increase the number of neurons and layers in AI training.

This research opens new frontiers for efficient, performant, and scalable neuromorphic solutions, laying the groundwork for exploring memristive learning circuits and their applications in SNNs and 3D neuromorphic circuits using printed memristors. By addressing the limitations of the proposed method, such as the requirement for non-dominating inputs and the current restriction to single-layer networks, and exploring future research directions, including backpropagation of errors, multi-layer generalization, and more technical STDP learning, the in situ memristive learning method for pattern classification can be further developed into a powerful and versatile neuromorphic computing approach with potential applications in various domains.

3. Proposed In Situ Memristive Learning Method for Pattern Classification

For designing, simulating, and verifying the functionality of SNNs and other neural network circuits based on memristors, simulators such as LTSpice and Cadence Spectre are mainly used, whereas for circuit layout design, Altium Designer (regarding printed circuit board) and Cadence Virtuoso (regarding integrated circuits) are preferred [85,98,99]. In this work, we utilize PySpice [96], a Python-based simulation framework integrated with NgSpice, to model and simulate memristive SNNs. PySpice is a versatile tool for simulating complex circuits, with demonstrated applications in electronic circuit simulation [100,101], highlighting its integration with Python for flexible, high-performance scripting. PySpice offers key advantages over traditional tools like LTSpice and Cadence Spectre. First, it allows flexible scripting via Python, enabling the automation of simulations, parameter sweeps, and advanced data analysis using libraries such as NumPy and Matplotlib. This makes complex simulations more manageable compared to manually adjusting SPICE netlists in other tools. Second, PySpice provides dynamic control over simulation conditions, essential for in situ learning. The real-time adjustments of voltages and currents across memristors allow us to accurately simulate STDP, enhancing the learning process within the network. Third, PySpice supports the development of custom models, such as our memristor emulator and Lapicque neuron, offering greater flexibility compared to pre-built models in other SPICE tools. This was crucial for accurately modeling memristive devices in neuromorphic circuits. Additionally, as an open-source tool, PySpice allows for seamless integration with machine learning frameworks like PyTorch, enabling hybrid neural models. Its cost-effective nature also removes the licensing barriers present in commercial software.

In the following, we will briefly present the memristors in general and later also detail how exactly they are implemented in this work with the help of PySpice [96]. We will then present the proposed XPUE dataset. Finally, we will present the proposed in situ memristive learning method for pattern classification.

A summarized view of the proposed solution can be seen in Figure 1 below.

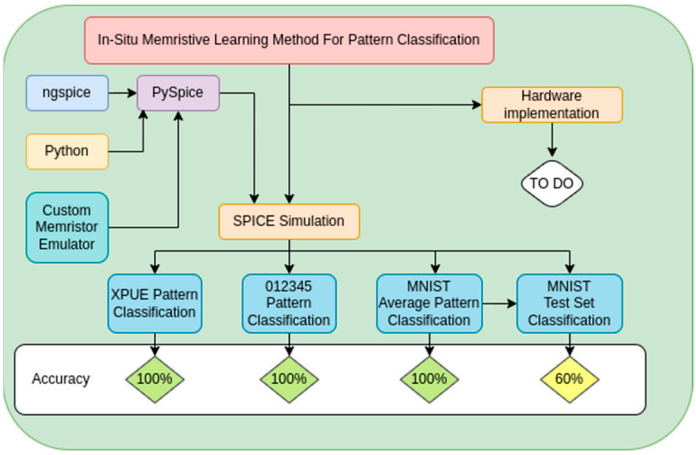


Figure 1. Overview of the In Situ Memristive Learning Framework for Neuromorphic Pattern Classification: This schematic illustrates the proposed method's workflow, incorporating software tools (Ngspice v43, PySpice v1.5, and Python v3.12), a custom-designed memristor emulator, and SPICE simulation for in situ learning on various datasets. The framework achieves 100% accuracy on simpler datasets (XPUE and 012345), with 60% accuracy on more complex datasets (resized MNIST and MNIST trial sets). Future directions include implementing this method on hardware for real-world neuromorphic computing applications.

3.1. General Overview of Memristors

Memristors are uniquely suited to neuromorphic applications, offering both memory retention and synaptic emulation capabilities. These properties provide an avenue to address the limitations of conventional computing architectures, including the von Neumann bottleneck and energy inefficiency in AI model training. By using memristors, neuromorphic circuits can emulate synaptic behavior, making them foundational to the development of high-performance neuromorphic computing systems. More exactly, memristors [102–104] are a class of passive two-terminal circuit elements that can be described as adaptive resistors with memory. The memristor resistance can be modified, which is useful for applications in neuromorphic computing, non-volatile memory, and signal processing. The memristor model can be described by the following Equation (1):

$$v(t) = R(x(t))i(t) \tag{1}$$

where $v(t)$ is the voltage across the memristor, $i(t)$ is the current through the memristor, $R(x(t))$ is the resistance of the memristor, and $x(t)$ is the state variable that determines the resistance of the memristor.

The memristor's resistance is a function of the state variable $x(t)$, which can be modified by the current across the memristor. The state variable $x(t)$ evolves according to the following Equation (2):

$$\frac{dx(t)}{dt} = f(x(t), i(t)) \tag{2}$$

where $f(x(t), i(t))$ is a function that describes the evolution of the state variable $x(t)$ in response to the voltage across the memristor.

Memristors are particularly well suited for neuromorphic computing applications due to their ability to emulate synapses in ANNs. The adaptive resistance of memristors can

be used to model the plasticity of synapses, which is essential for learning and memory in biological neural networks.

By rewriting the memristor model equation to compute $i(t)$ we have Equation (3):

$$i(t) = \frac{v(t)}{R(x(t))} \tag{3}$$

If we replace the memristor resistance $R(x(t))$ with a conductance $G(x(t)) = \frac{1}{R(x(t))}$, we can rewrite the equation as seen in Equation (4):

$$i(t) = G(x(t))v(t) \tag{4}$$

Therefore, if we consider $v(t)$ as the input voltage and $i(t)$ as the output current, we can see that the memristor acts as a synapse in an ANN, where the conductance $G(x(t))$ models the synaptic strength between neurons.

To understand the equations, we can consider a first memristor model, which is the linear memristor model, where the resistance of the memristor is a linear function of the state variable $x(t)$, as seen in Equation (5):

$$G(x(t)) = x(t) \tag{5}$$

and the evolution of the state variable $x(t)$ is also linear on the current $i(t)$, as seen in Equation (6):

$$\frac{dx(t)}{dt} = i(t) \tag{6}$$

Equation (6) represents the rate of change in the memristor's state variable $x(t)$ in response to the input current $i(t)$, modeling how the memristor's internal state evolves with current flow, critical for in situ learning.

Replacing $i(t)$ in Equation (6), we have Equation (7):

$$\frac{dx(t)}{dt} = G(x(t))v(t) = x(t)v(t) \tag{7}$$

Therefore, if we apply a sinusoidal voltage $v(t) = \sin(t)$ across the memristor, the state variable $x(t)$ will evolve according to the following Equation (8):

$$\frac{dx(t)}{dt} = x(t)\sin(t) \tag{8}$$

Equation (8) can be solved by Equation (9):

$$x(t) = C\exp(-\cos(t)) \tag{9}$$

Therefore, the equation for $i(t)$ can be written as in Equation (10):

$$i(t) = x(t)v(t) = C\exp(-\cos(t))\sin(t) \tag{10}$$

If we then plot the current–voltage relationship for the linear memristor model, we obtain a pinched hysteresis loop, which is a characteristic of memristors, as we can see in Figure 2.

Here, the pinched hysteresis loop is a signature of the memory effect in memristors, which is essential for neuromorphic computing applications. Applying a positive voltage differential across the memristor will increase its conductance while applying a negative voltage differential will decrease it. This behavior is similar to the long-term potentiation (LTP) and long-term depression (LTD) of synapses in biological neural networks, which are essential for learning and memory. In the case of non-volatile memristors, the memristor will retain its conductance even after the voltage differential is removed. This behavior

allows for long-term memory. Memristors can be used in unsupervised learning algorithms, such as Hebbian learning, where the synaptic strength between neurons is modified based on the correlation between their activities. The adaptive resistance of memristors allows for implementing Hebbian learning rules in hardware, which is essential for unsupervised learning in ANNs. STDP is another unsupervised learning rule that can be implemented using memristors, where the synaptic strength between neurons is modified based on the relative timing of the spikes in the pre- and post-synaptic neurons. Memristors can also be used in supervised learning algorithms, such as backpropagation, where the synaptic strength between neurons is modified based on the error between the predicted and actual outputs of the neural network. The adaptive resistance of memristors allows for implementing backpropagation learning rules in hardware, which is essential for supervised learning in ANNs [72]. Interestingly, Titanium dioxide (TiO_2) is a material that has been widely used in the fabrication of memristors due to its high resistance and low switching voltage. TiO_2 memristors are composed of a thin film of TiO_2 sandwiched between two metal electrodes, which allows for the modulation of the memristor's resistance by applying an external voltage. TiO_2 memristors have been shown to exhibit non-volatile memory behavior, which is essential for applications in neuromorphic computing and non-volatile memory. By doping the TiO_2 material with oxygen vacancies, it is possible to create a conductive filament that can be modulated by an external voltage. The conductive filament is formed by migrating oxygen vacancies in response to an external voltage, which changes the memristor's resistance. The conductive filament can be dissolved by applying an opposite voltage, allowing the memristor to be reset to its initial state.

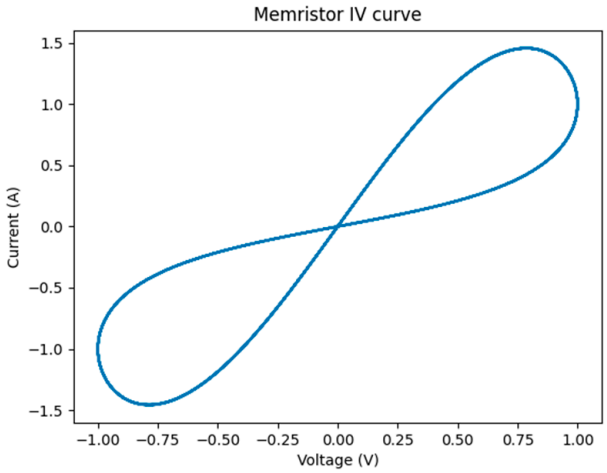


Figure 2. Current–voltage relationship for the linear memristor model using a sinusoidal voltage difference across the memristor of 1 V amplitude.

3.2. Proposed PySpice Implementation of the Memristor Emulator

PySpice [96] is a Python library that provides a Python interface to the popular SPICE circuit simulation tool, in this case, ngspice [105]. For the purposes of this work, the benefit of using PySpice is that it provides a simple and intuitive way to scale up the complexity of the SNNs in terms of the number of neurons, layers, and synapses, as proved earlier also by the work seen in [106]. In order to implement the memristor model in PySpice, we need to find a suitable circuit that emulates the behavior of a memristor. Mainly, we need to find a circuit that can be used to model the resistance of the memristor as a function of the state variable $x(t)$ and the evolution of the state variable $x(t)$ in response to the current $i(t)$. More exactly, we will use a circuit similar to the memristor emulator circuit proposed by the work in [73]. The proposed memristor emulator circuit used in this work can be seen in Figure 3.

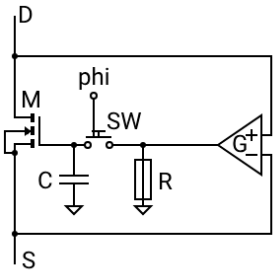


Figure 3. The memristor emulator circuit used in this work.

Here, the circuit consists of an NMOSFET (M), a capacitor (C), and a voltage-controlled current source (G). The memristor's resistance is modeled by the NMOSFET, which is controlled by the voltage across the capacitor. The evolution of the state variable $x(t)$ is modeled by the voltage across the capacitor, which is controlled by the current through the memristor. The circuit also includes a switch (SW) controlled by an additional input voltage (phi) that acts as a switch to block the current through the capacitor when the input voltage is low in order to stabilize the state variable $x(t)$ when desired. We also added a high resistance (R) in parallel with the capacitor to allow the current to flow through the circuit when the input voltage is low, which helps to stabilize the state variable $x(t)$.

While the circuit description provides a good overview of the memristor emulator's components, we can further clarify the learning process by presenting the mathematical model that underpins the circuit's behavior during learning.

Regarding memristor state dynamics, the state of the memristor is governed by the voltage across the capacitor $V_C(t)$, which evolves according to the current through the memristor. The capacitor's behavior follows Equation (11):

$$I_C(t) = C\frac{dV_C(t)}{dt} \tag{11}$$

where $I_C(t)$ is the current through the capacitor and C is the capacitance. The NMOSFET's resistance, which models the memristor's behavior, is controlled by $V_C(t)$, allowing the state variable to evolve based on the input signals. We model the memristor's resistance (or memductance) $G(t)$ as a function of the capacitor voltage, as seen in Equation (12):

$$G(t) = G_0 + \beta q(t), \quad q(t) = \int_0^t I(t)dt \tag{12}$$

where G_0 is the initial memductance and $q(t)$ is the charge that has passed through the memristor. Here, β serves as the learning rate, controlling the rate at which the memductance changes in response to inputs, helping to stabilize learning and ensuring convergence. The memductance adjusts based on the history of the charge, allowing the circuit to simulate synaptic weight changes in response to input signals.

The learning process in this memristor emulator can be described by a Hebbian learning rule, which updates the memristance (or memductance) based on the product of the input signal $x(t)$ and the output signal $y(t)$ seen in Equation (13):

$$\Delta G(t) = \eta x(t) y(t) \tag{13}$$

where η is a learning rate that controls the speed of the memristor's adaptation. This rule captures how synaptic strengths (represented by memductance) are adjusted based on the activity of the connected neurons (input and output).

The voltage-controlled switch (SW) regulates the learning process by allowing the system to stabilize the state variable when necessary. The switch operates as seen in Equation (14):

$$I_{SW} = \begin{matrix} 0 & if\ V_\phi < V_{th} \\ I_{mem} & if\ V_\phi \geq V_{th} \end{matrix} \quad (14)$$

When the input voltage V_ϕ is below a certain threshold V_{th}, the current through the capacitor is blocked, which stabilizes the memristor's state by preventing further updates. This mechanism allows for controlled learning and state stabilization. The addition of a high resistance R in parallel with the capacitor ensures that current can continue to flow through the circuit even when the switch is off. This helps prevent the total isolation of the capacitor and further aids in stabilizing the state variable by allowing for a slow discharge through the high resistance. The memristor emulator, as modeled here, allows for in situ learning by dynamically adjusting the memristance in response to input signals. The capacitor and switch together regulate the learning process, enabling both continuous adjustment during learning phases and stabilization when desired. This makes the emulator suitable for learning tasks where both short-term adaptation and long-term retention are required.

It is important to mention that this is the first PySpice [96] implementation of a memristor model found in the literature. While this emulator accurately mimics the expected voltage–current characteristics of memristors, it is important to note that the use of a capacitor introduces volatility into the system. The code snippet in Figure 4 implements the proposed memristor model in PySpice.

```
class Memristor(SubCircuit):
    def __init__(self, name, K, Kp, C, Vto, Roff, Ron):
        __nodes__ = ('S', 'D', 'phi')
        SubCircuit.__init__(self, name, *__nodes__)
        self.VCCS('m', self.gnd, 'o', 'D', 'S', K)
        self.R('R2', 'o', self.gnd, 10**12)
        self.VoltageControlledSwitch('sw', 'o', 'Cmem', 'phi', self.gnd, model='switch')
        self.model('switch', "SW", Ron=Ron, Roff=Roff, Vt=0.5)
        self.C('C', self.gnd, 'Cmem', C)
        self.model('memristor_nmos', 'NMOS', level=3, Vto=Vto, Kp=Kp)
        self.MOSFET(1, 'D', 'Cmem', 'S', self.gnd, model='memristor_nmos')
```

Figure 4. Python code implementation snippet of the proposed memristor model in PySpice. The complete code can be found in [107].

Table 1 shows the parameters and their default values for the memristor emulator circuit implementation.

Table 1. Memristor emulator circuit parameters.

Parameter	Value	Explanation
K	0.0002 $\frac{A}{V}$	G value
V_{t0}	0 V	Threshold voltage of NMOSFET
K_p	0.00013	NMOSFET parameter
C	0.1 pF	The capacitance of the capacitor
R	10^{12} Ω	High resistance of resistor
$V_{phi_{on}}$	0.5 V	Switch ON voltage
$V_{phi_{off}}$	0 V	Switch OFF voltage
R_high	10^{-3} Ω	ON resistance of the switch
R_low	10^{20} Ω	OFF resistance of the switch

With the values of the parameters from Table 1, we can simulate the memristor emulator circuit in PySpice and obtain the current–voltage relationship for the memristor model when applying a voltage difference across the memristor of sinusoidal form. The result can be seen in Figure 5.

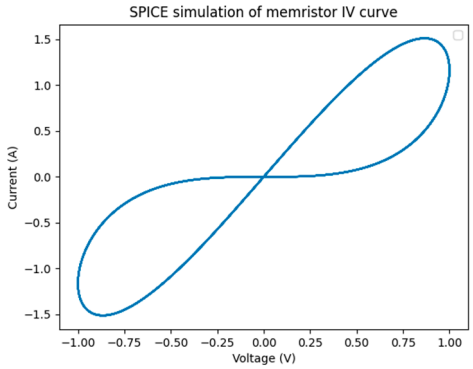


Figure 5. Current–voltage relationship for the memristor emulator circuit using a sinusoidal voltage difference across the memristor of 1 V amplitude.

In Figure 6, we can see the voltage across the capacitor in the memristor emulator circuit when applying a sinusoidal voltage difference across the memristor of 1 V amplitude.

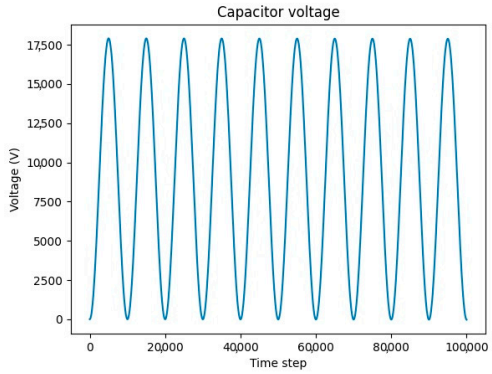


Figure 6. The voltage across the capacitor in the memristor emulator circuit using a sinusoidal voltage difference across the memristor of 1 V amplitude. Note: The 17.5 kV voltage in this simulation is an idealized parameter and not representative of real-world memristor operating conditions, which are much lower. Here, the time step units are in milliseconds (ms).

Here, we can see that the voltage across the capacitor is also sinusoidal, spanning from 0 V to approximately 17.5 kV. In Figure 7, we can see the input voltage applied to the memristor emulator circuit, which is a sinusoidal voltage difference of 1 V amplitude.

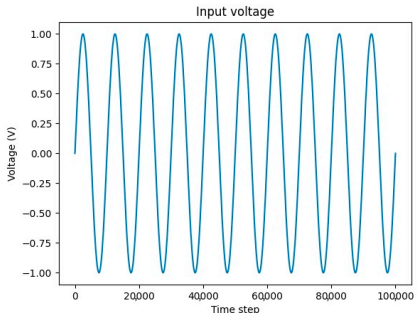


Figure 7. The sinusoidal voltage difference across the memristor of 1 V amplitude. Here, the time step units are in milliseconds (ms).

3.3. Proposed XPUE Dataset

The XPUE dataset consists of a set of four 3×3 images designed by us to be a simple test for the performance of SNNs in image classification tasks. In Figure 8, we can see the four images that compose this dataset.

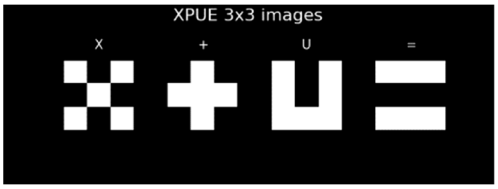


Figure 8. The proposed XPUE dataset.

One important feature of this set of images is that no image, thought of as input, dominates any other, i.e., no positive input is entirely inside the other. The dominance criterion ensures balanced learning across all the input patterns, preventing any single pattern from disproportionately influencing the network's weight adjustments.

The non-dominance feature can be defined as follows:

Let X_i be the i-th image or input, whose values are binary (either 0 or 1), and $X_{i_{k*n+l}}$ be the pixel at position (k, l) of the image X_i. Here, n acts as an index multiplier, organizing pixel data in the dataset for proper alignment and processing within the network. In general, X_{i_j} is the j-th element of (flattened if necessary) input X_i.

We say that X_1 dominates X_2 if it reflects Equation (15):

$$\sum_{s=0}^{N} X_{1_s} X_{2_s} \geq \sum_{s=0}^{N} X_{1_s} \tag{15}$$

where $N + 1$ is the number of pixels in the image (or the dimension of the input).

Considering that, we have Equation (16):

$$\sum_{s=0}^{N} X_{1_s} = \sum_{s=0}^{N} X_{1_s} X_{1_s} \tag{16}$$

because $X_{1_s} = 0$ or $X_{1_s} = 1$, so we can rewrite the dominance condition as seen in Equation (17):

$$\sum_{s=0}^{N} X_{1_s} X_{2_s} \geq \sum_{s=0}^{N} X_{1_s} X_{1_s} \tag{17}$$

Therefore, if matrix M is defined as $M_{ij} = \sum_{s=0}^{N} X_{i_s} X_{j_s}$, the dominance condition equals to finding a row where the diagonal element is not the maximum.

In other words, we want the matrix M to have the following property: $\forall i, \forall j \neq i, M_{ij} < M_{ii}$. As long as this property is satisfied, we say that no input dominates any other.

We observe this property in the XPUE dataset, which can be seen if we first write the flattened arrays of each image as follows (18):

$$\begin{aligned} X_1 &= [1,\ 0,\ 1,\ 0,\ 1,\ 0,\ 1,\ 0,\ 1] \\ X_2 &= [0,\ 1,\ 0,\ 1,\ 1,\ 1,\ 0,\ 1,\ 0] \\ X_3 &= [1,\ 0,\ 1,\ 1,\ 0,\ 1,\ 1,\ 1,\ 1] \\ X_4 &= [1,\ 1,\ 1,\ 0,\ 0,\ 0,\ 1,\ 1,\ 1] \end{aligned} \tag{18}$$

So, we obtain the following matrix M seen in (19):

$$M = \begin{bmatrix} \mathbf{5} & 1 & 4 & 4 \\ 1 & \mathbf{5} & 3 & 2 \\ 4 & 3 & \mathbf{7} & 5 \\ 1 & 2 & 5 & \mathbf{6} \end{bmatrix} \tag{19}$$

where the bold numbers are the diagonal elements of the matrix.

We can see that no row has a diagonal element that is not the maximum in that row; therefore, no input dominates any other. The property can be generalized to real values (as

opposed to just binary numbers). The dominance condition stays the same, but the matrix will have real values instead of integers. The authors of the work seen in [74] use a set of six 3×5-pixel images depicting the numbers 0 to 5 for pattern classification using a CMOS circuit implementation of SNNs. The flattened arrays of the images are as follows (20):

$$\begin{aligned} X_0 &= [1,1,1,1,0,1,1,0,1,1,0,1,1,1,1] \\ X_1 &= [0,1,0,0,1,0,0,1,0,0,1,0,0,1,0] \\ X_2 &= [1,1,1,0,0,1,1,1,1,1,0,0,1,1,1] \\ X_3 &= [1,1,1,0,0,1,1,1,1,0,0,1,1,1,1] \\ X_4 &= [1,0,1,1,0,1,1,1,1,0,0,1,0,0,1] \\ X_5 &= [1,1,1,1,0,0,1,1,1,0,0,1,1,1,1] \end{aligned} \tag{20}$$

The matrix M for this set of images is seen in (21):

$$M = \begin{bmatrix} \mathbf{12} & 2 & 10 & 10 & 8 & 10 \\ 2 & \mathbf{5} & 3 & 3 & 1 & 3 \\ 10 & 3 & \mathbf{11} & 10 & 7 & 9 \\ 10 & 3 & 10 & \mathbf{11} & 8 & 10 \\ 8 & 1 & 7 & 8 & \mathbf{9} & 7 \\ 10 & 3 & 9 & 10 & 7 & \mathbf{11} \end{bmatrix} \tag{21}$$

Here, we can see that no row has a diagonal element that is not the maximum in that row, so no input dominates any other. However, if we decided to add a 3×5 image of the number 6, we would have the following flattened array (22):

$$X_6 = [1,1,1,1,0,0,1,1,1,1,0,1,1,1,1] \tag{22}$$

The matrix M for the set of images 0 to 6 would be as seen in (23):

$$M = \begin{bmatrix} \mathbf{12} & 2 & 10 & 10 & 8 & 10 & 11 \\ 2 & \mathbf{5} & 3 & 3 & 1 & 3 & 3 \\ 10 & 3 & \mathbf{11} & 10 & 7 & 9 & 10 \\ 10 & 3 & 10 & \mathbf{11} & 8 & 10 & 9 \\ 8 & 1 & 7 & 8 & \mathbf{9} & 7 & 8 \\ 10 & 3 & 9 & 10 & 7 & \mathbf{11} & \mathbf{11} \\ 11 & 3 & 10 & 9 & 8 & 11 & \mathbf{12} \end{bmatrix} \tag{23}$$

In this case, we can see that the row corresponding to the image 5 has a value that is equal to the diagonal element (row 6, column 7), so the image 6 dominates the image 5.

3.4. In Situ Memristive Learning Method for Pattern Classification

A fully connected memristive SNN consists of an arbitrary number of layers, each layer containing an arbitrary number of neurons and connecting to the next layer through a memristive crossbar array. The memristive crossbar array is a matrix of memristors that connect the outputs of the neurons in one layer to the inputs of the neurons in the next layer. The memristive crossbar array allows for the parallel computation of the weighted sum of the inputs to each neuron in the next layer, essential for efficiently implementing neural networks in hardware. In between the layers, there is some additional circuitry to manipulate the output signals of the neurons, perform the activation function, etc., and finally send the output to the next layer. A general version of the memristive crossbar array is shown in Figure 9, where $M_{i,j,N}$ are the memristors representing negative weights and $M_{i,j,P}$ the positive ones. V_0 to V_n are the inputs of the layer, and the outputs are attached to the bottom ends of the vertical wires.

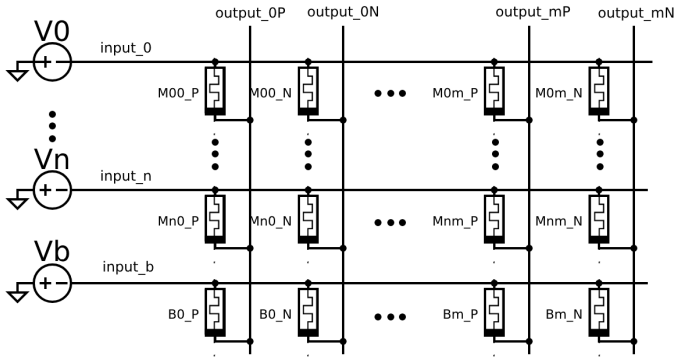


Figure 9. Proposed memristive crossbar array. Adapted from [106].

In the general case, we allow the possibility of negative weights by duplicating the number of outputs: one for representing positive values and one for representing negative values. We also allow for the possibility of a bias voltage on each layer, which is also optional. When these negative weights are used, each negative output will be subtracted from the positive output, and the result will be the respective input to the next layer. The in situ memristive learning method for pattern classification is a learning algorithm that uses the memristive crossbar array to perform the learning process directly on the hardware, without needing a separate training phase, based on a given set of data inputs and their respective labels. The in situ memristive learning method is based on the principle of Hebbian learning, where the synaptic strength between neurons is modified based on the correlation between the activities of the neurons. More exactly, this work applies a Hebbian learning rule variant, which adjusts synaptic weights based on the temporal correlation between pre- and post-synaptic spikes, known as STDP. In this model, the synaptic strength is modified proportionally to the relative timing of neuron spikes. When a pre-synaptic neuron fires just before a post-synaptic neuron, the synaptic weight increases; if the reverse occurs, it decreases. This rule effectively captures associative learning patterns in a biologically plausible manner directly within the circuit.

Let us consider a one-layer SNN with n inputs and m outputs, where n is the size of the input data and m is the number of unique labels. The network weights will be determined by the memristors in the crossbar array. The inputs are V_0 to V_n and the outputs are attached to the bottom ends of the vertical wires. The memristors are connected to the inputs and outputs, as shown earlier in Figure 9. Additional components are needed to implement the fully connected memristive SNN, such as operational amplifiers, Lapicque neurons, and Schmitt triggers. The neuron model used in this study is based on the Lapicque neuron, which operates under the LIF model. This model is well suited for SNNs as it captures the essential characteristics of biological neurons by accumulating input voltage until a threshold is reached, at which point it "fires" a spike. Schmitt triggers, which we implemented using op-amps, are employed to stabilize the spiking behavior. They transform the analog output from the neuron model into a digital signal, thus providing clear spike events in our circuit. The Schmitt trigger's hysteresis helps prevent oscillations, ensuring reliable operation within the neuromorphic system. More exactly, the same configuration as in the [106] is used to implement the additional components. Operational amplifiers are used in the memristive SNN at the end of the crossbar array to combine the weighted sum of the inputs to each output. The outputs from the memristive crossbar, after being processed by the operational amplifiers, are fed into the Lapicque neurons. The Lapicque neuron is a simple model of a spiking neuron that can be used to model the behavior of neurons in SNNs. The Lapicque neuron consists of a capacitor, a resistor, a voltage-controlled current source, and a Schmitt trigger. The capacitor models the membrane potential of the neuron, the resistor models the leakage of the membrane potential, the voltage-controlled current source (VCCS) models the input current to the neuron, and the Schmitt trigger models the spiking behavior of the neuron. The Schmitt trigger is a circuit that converts an analog input signal into a digital output signal. In

the circuit, it receives the output of the Lapicque neuron and outputs a digital signal that represents the neuron's spiking behavior. The Schmitt trigger consists of two operational amplifiers in a series to obtain a binary output signal. In Figure 10, we can see the fully connected memristive SNN for the XPUE dataset.

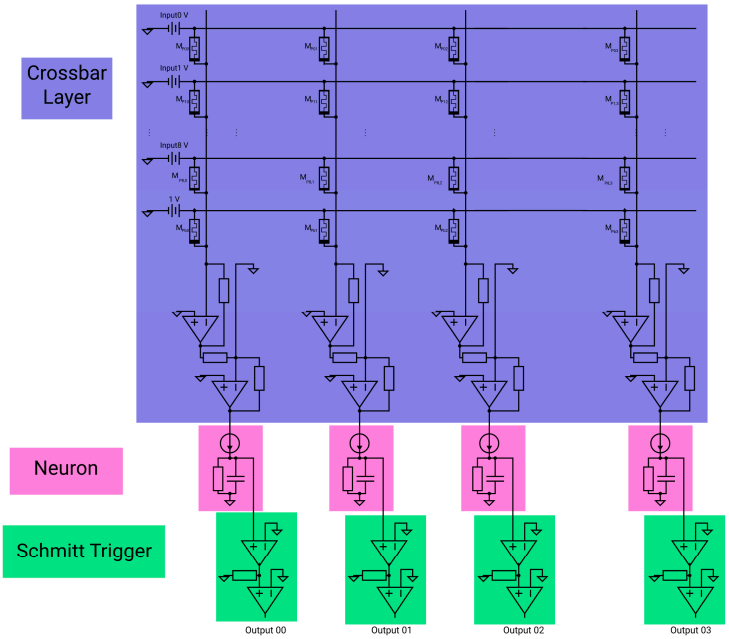


Figure 10. Fully connected memristive SNN for the XPUE dataset. Adapted from [106]. The figure depicts two inputs per neuron, representing the structured layout of the network for dataset compatibility.

Here, the network consists of a single layer with 9 input neurons and 4 output neurons, corresponding to the 4 images in the dataset. The components list for the XPUE circuit can be seen in the Appendix A section of this paper.

In Figure 11, we can see the fully connected memristive SNN for the "012345" dataset.

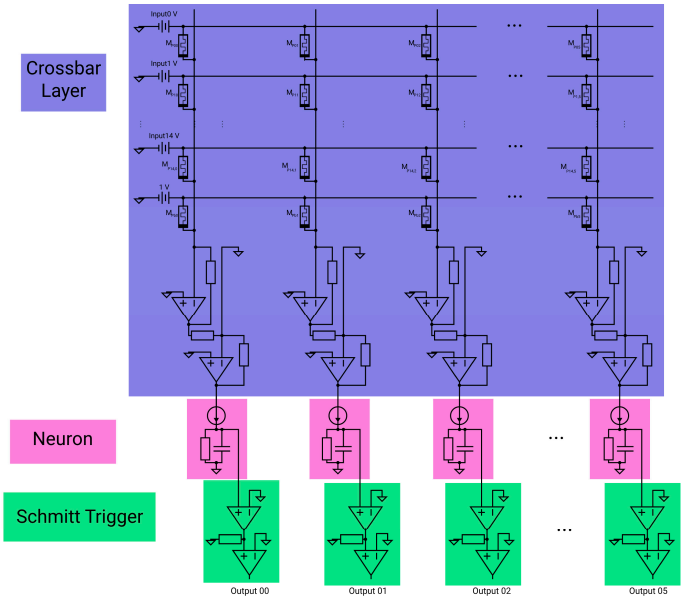


Figure 11. Fully connected memristive SNN for the "012345" dataset. Adapted from [106]. The figure depicts two inputs per neuron, representing the structured layout of the network for dataset compatibility.

Here, the network consists of a single layer with 15 input neurons and 6 output neurons, corresponding to the 6 images in the dataset. The components list for the 012345 circuit can be seen also in the Appendix A section of this paper.

In Figure 12, we can see the fully connected memristive SNN for the MNIST dataset.

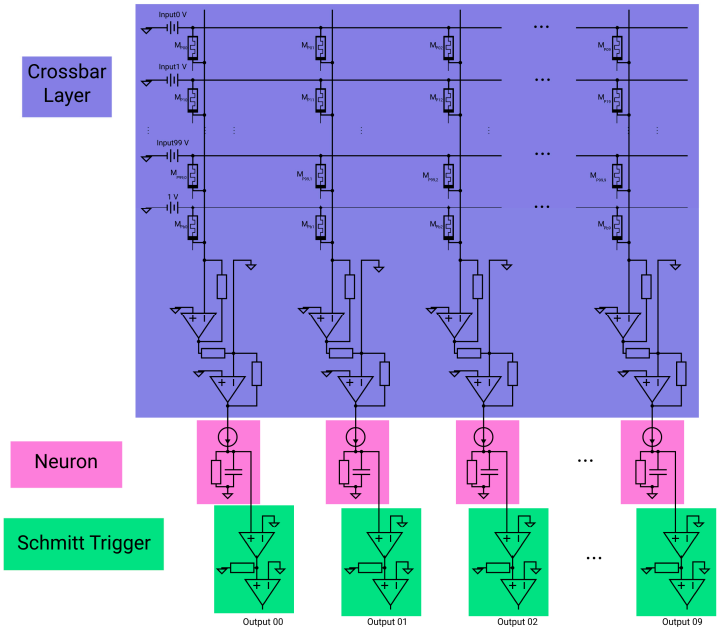


Figure 12. Fully connected memristive SNN for the MNIST dataset. Adapted from [106]. The figure depicts two inputs per neuron, representing the structured layout of the network for dataset compatibility.

Here, the network consists of a single layer with 100 input neurons and 10 output neurons, corresponding to the 10 digits in the dataset. The components list for the MNIST circuit can be seen also in the Appendix A section of this paper. Considering this is an in situ learning method, the training will be controlled by the input voltages fed into the network. The input voltages consist not only of the input data voltages V_0 to V_n but also of the switches attached to each of the memristors $\phi_{i,j}$ for $i = 1, \ldots, n$ and $j = 1, \ldots, m$. The switches are controlled by the input voltages $\phi_{i,j}$, which act as a switch to block the current through the memristor when the input voltage is low in order to stabilize the state variable $x(t)$ when desired. Therefore, to define the learning method, we just need to define a sequence of input voltages that will be fed into the network. We begin with a simple case in which there is a single example for each label. In this case, the learning phase consists of feeding the input data voltages for each example and setting $\phi_{i,j}$ to 1 if the j-th output is the correct one for the given input data and 0 otherwise. This means that the memristors associated with the correct output will be modified to increase their conductance, while the memristors associated with the incorrect outputs will not. This process is repeated for each example in the training set.

4. Experimental Setup and Results

The experimental results presented here are derived from simulations within the PySpice environment. While these simulations provide valuable insights into the potential of memristor-based circuits for neuromorphic computing, they do not yet reflect the actual hardware performance. The memristor models used here assume ideal conditions to demonstrate the feasibility of in situ learning within SNNs, but further empirical validation with physical devices is required to substantiate these findings. Real-world memristors may exhibit varying resistance and current characteristics under operational conditions,

influencing their learning dynamics. This work serves as a conceptual model, illustrating a path forward for practical neuromorphic systems.

In the following, the proposed in situ memristive learning method for pattern classification is tested on the XPUE dataset, the "012345" patterns, and the MNIST dataset. Finally, information about how to go from SPICE to hardware implementation using memristors and the testing modules is provided.

4.1. Testing Method with XPUE Dataset

Regarding the proposed XPUE dataset, the input data voltages for each example are as follows (24):

$$\begin{aligned} X_1 &= [1,\ 0,\ 1,\ 0,\ 1,\ 0,\ 1,\ 0,\ 1] \\ X_2 &= [0,\ 1,\ 0,\ 1,\ 1,\ 1,\ 0,\ 1,\ 0] \\ X_3 &= [1,\ 0,\ 1,\ 1,\ 0,\ 1,\ 1,\ 1,\ 1] \\ X_4 &= [1,\ 1,\ 1,\ 0,\ 0,\ 0,\ 1,\ 1,\ 1] \end{aligned} \tag{24}$$

In Figure 13, we can see the resulting input data voltages.

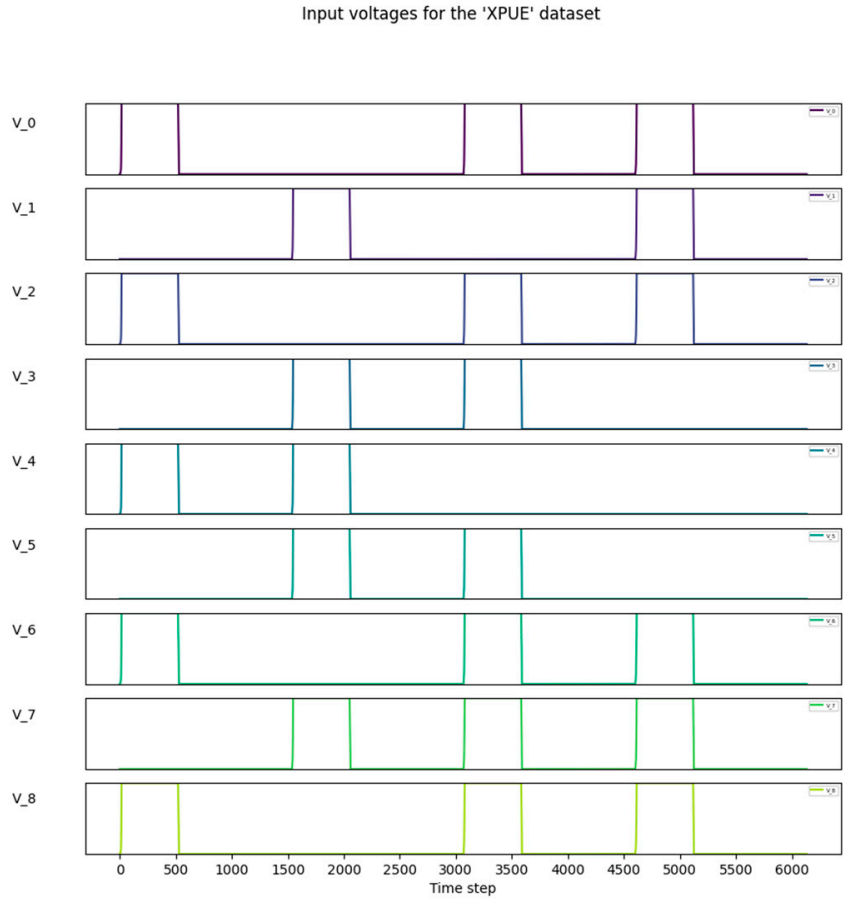


Figure 13. Input voltages for the XPUE dataset examples. Here, the time step units are in milliseconds (ms).

Also, in Figure 14, we can see the control voltages $\phi_{i,j}$ for each example.

The control voltages are set to 1 for the correct output and 0 for the incorrect output. We can see that the voltages are applied in intervals of 1500 steps, with a duration of 500 steps for each voltage, to avoid interference between the examples. With this training configuration, the results shown in Table 2 are obtained.

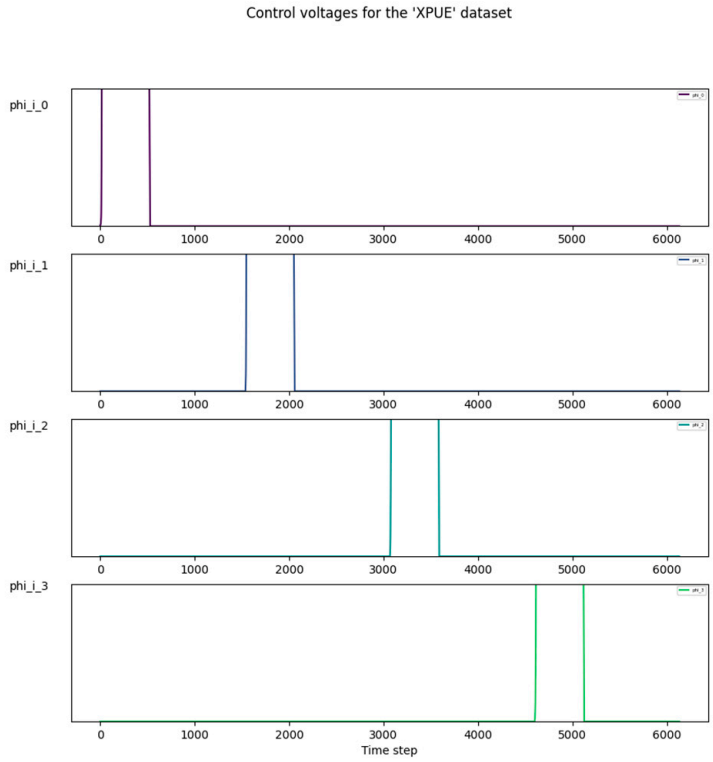


Figure 14. Control voltages for learning the XPUE dataset. Here, the time step units are in milliseconds (ms).

Table 2. Normalized voltage across the capacitor in the memristor emulator circuit for the XPUE dataset examples.

$X_{j,i}$	j = 0	j = 1	j = 2	j = 3
i = 0	1.0000	−0.9416	0.9523	0.9906
i = 1	−0.9431	0.9998	−1.0000	0.9894
i = 2	1.0000	−0.9416	0.9523	0.9906
i = 3	−0.9413	0.9998	0.9503	−0.9548
i = 4	0.9999	0.9997	−1.0000	−0.9539
i = 5	−0.9413	0.9998	0.9503	−0.9548
i = 6	1.0000	−0.9416	0.9523	0.9906
i = 7	−0.9431	0.9998	0.9503	0.9894
i = 8	1.0000	−0.9416	0.9523	0.9906

Here, the voltage values are normalized to a range of [−1 V, 1 V] to align with standard practices in modeling capacitive and memristive systems. This adjustment ensures that the results reflect realistic operating conditions, avoiding overdrive.

In Figure 15, we can see the evolution of these values over time in the training phase.

In order to evaluate the performance of the SNN after the training phase, we can apply the input data voltages for each example and observe the output voltages of the neurons in the last layer. We will consider a first-to-win strategy, where the output neuron which first reaches a voltage of 1 V will be considered the winner. For the inference, the voltages $phi_{i,j}$ will be set to 0 for all i and j in order to not alter the training phase. With these training results, we obtain the output voltages shown in Figure 16.

As we can see, given each input image, the first output neuron to reach 1 V is the one corresponding to the correct label, which means that the SNN has learned to classify the images correctly with 100% accuracy.

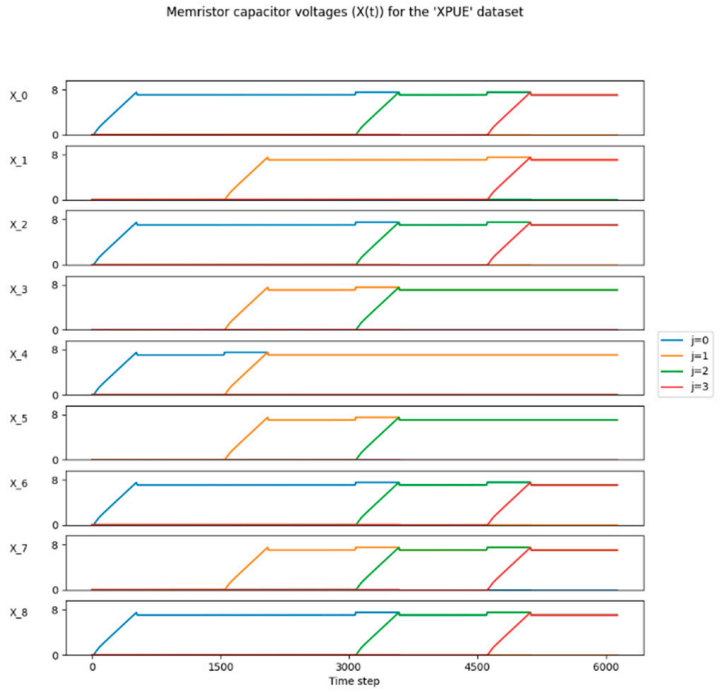


Figure 15. The voltage across the capacitor in the memristor emulator circuit for the XPUE dataset examples. Here, the time step units are in milliseconds (ms).

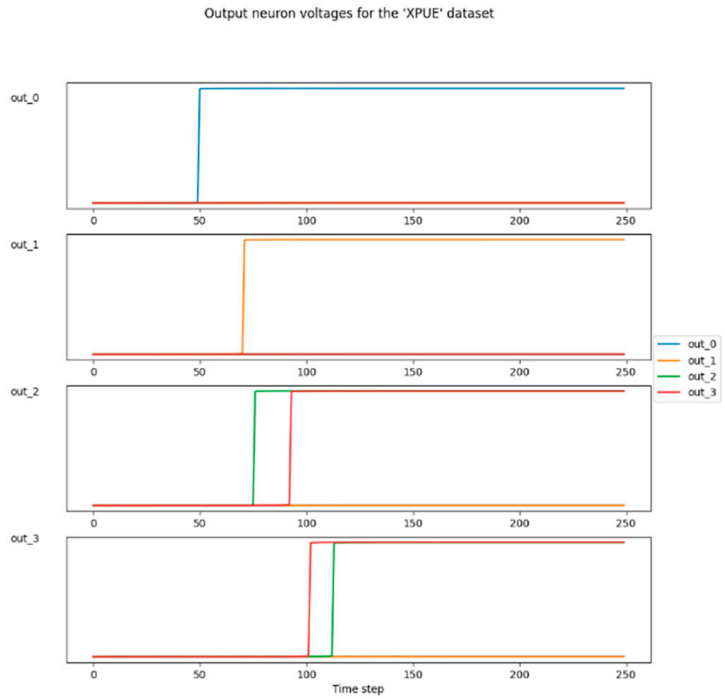


Figure 16. Output voltages for the XPUE dataset examples. Here, the time step units are in milliseconds (ms).

4.2. Testing Method with "012345" Patterns

Here, the in situ memristive learning method for pattern classification is also tested, this time with the "012345" patterns from the work seen in [74]. In this case, we have six examples, one for each label (the digits from 0 to 5). The input data voltages for each example are as follows (25):

$$\begin{aligned} X_0 &= [1,\ 1,\ 1,\ 1,\ 0,\ 1,\ 1,\ 0,\ 1,\ 1,\ 0,\ 1,\ 1,\ 1,\ 1] \\ X_1 &= [0,\ 1,\ 0,\ 0,\ 1,\ 0,\ 0,\ 1,\ 0,\ 0,\ 1,\ 0,\ 0,\ 1,\ 0] \\ X_2 &= [1,\ 1,\ 1,\ 0,\ 0,\ 1,\ 1,\ 1,\ 1,\ 1,\ 0,\ 0,\ 1,\ 1,\ 1] \\ X_3 &= [1,\ 1,\ 1,\ 0,\ 0,\ 1,\ 1,\ 1,\ 1,\ 0,\ 0,\ 1,\ 1,\ 1,\ 1] \\ X_4 &= [1,\ 0,\ 1,\ 1,\ 0,\ 1,\ 1,\ 1,\ 1,\ 0,\ 0,\ 1,\ 0,\ 0,\ 1] \\ X_5 &= [1,\ 1,\ 1,\ 1,\ 0,\ 0,\ 1,\ 1,\ 1,\ 0,\ 0,\ 1,\ 1,\ 1,\ 1] \end{aligned} \tag{25}$$

In Figure 17, we can see the images corresponding to each label.

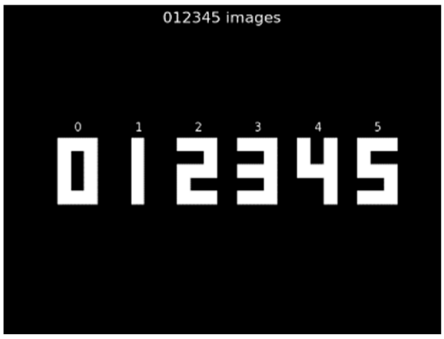


Figure 17. The "012345" pattern images.

In Figure 18, we can see the resulting input data voltages.

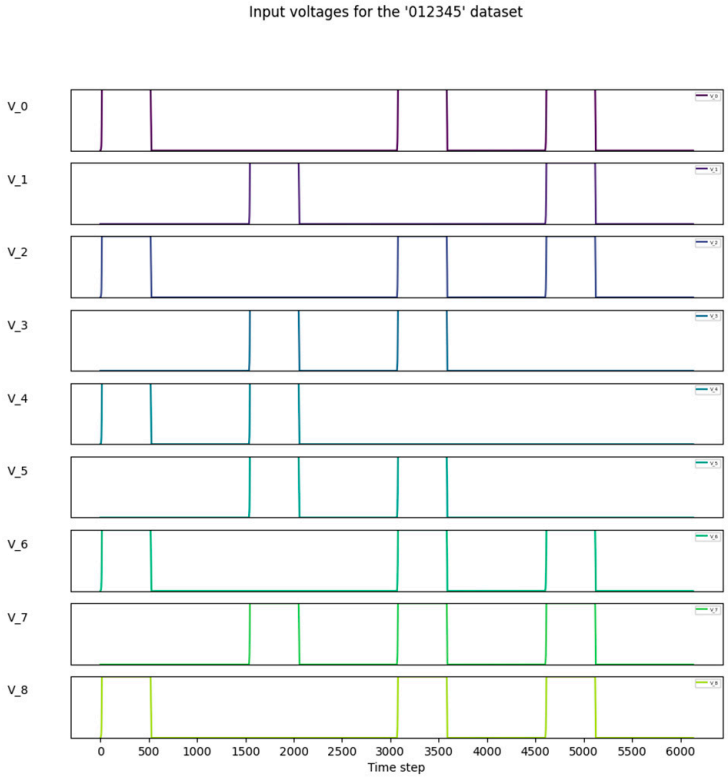


Figure 18. Input voltages for the "012345" dataset examples. Here, the time step units are in milliseconds (ms).

In Figure 19, we can see the control voltages $\phi_{i,j}$ for each example.

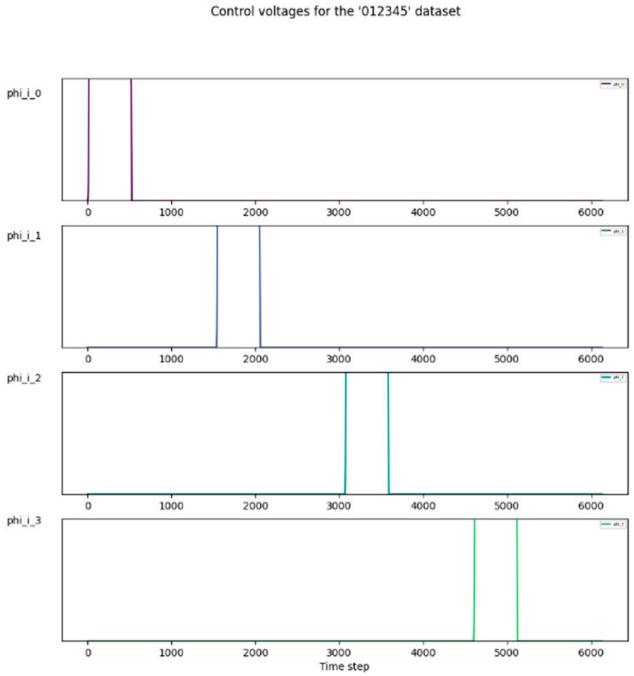


Figure 19. Control voltages for learning the "012345" dataset. Here, the time step units are in milliseconds (ms).

Here, the control voltages are set to 1 for the correct output and 0 for the incorrect output. In Figure 20, we can see the evolution of the internal memristor value $X(t)$ over time in the training phase.

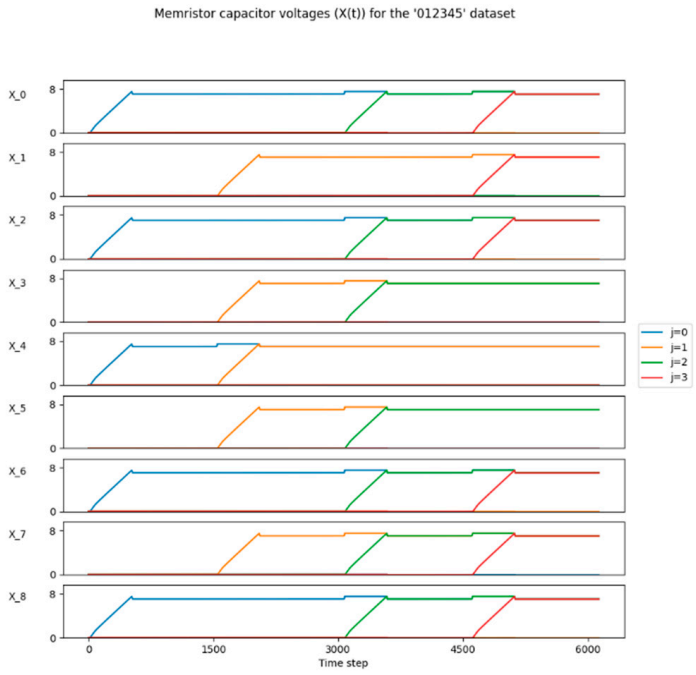


Figure 20. The voltage across the capacitor in the memristor emulator circuit for the "012345" dataset examples. Here, the time step units are in milliseconds (ms).

Finally, we obtain the output voltages shown in Figure 21.

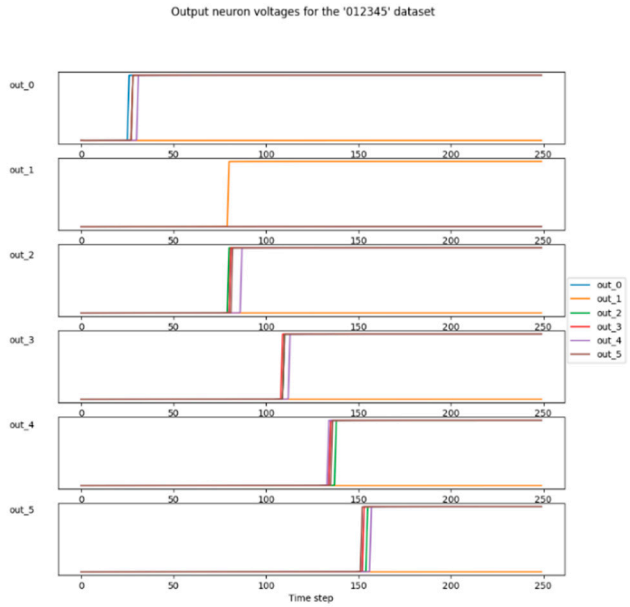


Figure 21. Output voltages for the "012345" dataset examples. Here, the time step units are in milliseconds (ms).

As we can see, given each input image, the first output neuron to reach 1 V is the one corresponding to the correct label, which means that the SNN has learned to identify the images correctly with 100% accuracy for this dataset as well. Note: The 100% accuracy observed may be due to using identical training and test sets, potentially leading to overfitting. Future work will incorporate separate test sets to evaluate generalization.

4.3. Testing Method with MNIST Dataset

The in situ memristive learning method for pattern classification has a cumulative effect on each trained example. This means that the weights of the network are modified by each example in the training set irrespective of the current weights, and the final weights are the result of the cumulative effect of all the examples. This means that if we trained for multiple examples per label, the final weights would be the result of the cumulative effect of all the examples. In other words, the weights will be the sum of the weights obtained for each example. We can use this feature to our advantage and create a representative example for each label by summing the input data voltages for each example with that label, which reduces the multiple examples per label to a single example per label, and then train the SNN with these representative examples using the in situ memristive learning method for pattern classification. We applied this method to the MNIST dataset, which is a dataset of handwritten digits from 0 to 9. More exactly, we created a representative example for each label by summing the input values of each class. We then resized the images to 10×10 pixels and kept the 20 highest pixel values for each image as 1 and the rest as 0. In Figure 22, we can see the resulting images corresponding to each label.

After applying the in situ memristive learning method for pattern classification to the resulting patterns, we obtained a training accuracy of 100%. We then tested the network on the first 100 samples from the MNIST test set and obtained a test accuracy of 60%. This result should be considered in context, as the SNN was trained fully in situ in a SPICE simulation (with no external calculation) without any pre-training or fine-tuning, and the test set was not used in the training phase. Furthermore, the training only takes a few seconds to complete, leveraging the capabilities of the memristors. These results are twice as good as those obtained by the work seen in [97], which obtained a test accuracy of 30% in a SPICE-based simulation. The lower MNIST accuracy (60%) observed in this study contrasts with the 89% accuracy in prior work [106] due to differences in training methodology. This study uses in situ training without pre-trained weights, which limits

initial performance, while ex situ training in prior work [106] involved pre-trained weights, yielding higher initial accuracy. This distinction highlights the trade-off between in situ learning flexibility and performance.

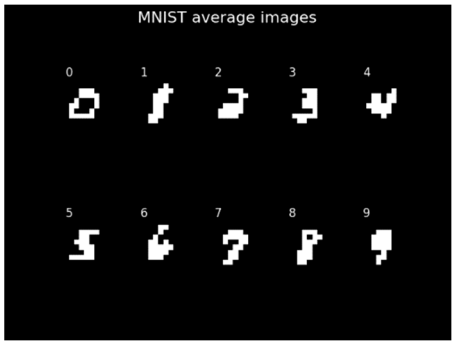


Figure 22. Average images for the MNIST dataset to use as representative examples.

In Figure 23, we can see the confusion matrix for the MNIST test set.

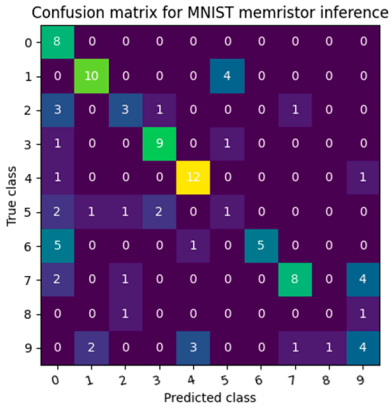


Figure 23. Confusion matrix for the MNIST test set. Here, bright yellow color represents the highest number of correct predictions, bright green indicates moderately high correct predictions, darker shades represent fewer occurrences, and deep purple/black indicates no predictions.

The confusion matrix shows that the SNN can classify the digits with good accuracy, with most of the digits being classified correctly, but it has some difficulty distinguishing the 8s, for instance.

5. Limitations of This Study and Future Work

While the in situ memristive learning method for pattern classification presented in this paper demonstrates promising results, there are several limitations to the current approach that should be addressed in future work.

One of the main limitations of the proposed method is the requirement that no input from the training set can dominate any other input. This constraint limits the types of patterns that the system can learn to distinguish, as it may not be able to effectively learn patterns with significant variations in pixel intensity or patterns where certain features are more prominent than others. Another limitation arises from the cumulative nature of the learning process. Since the learning has a blind cumulative effect on each trained example, the network can effectively only learn one representative pattern per label. This means that the network may not perform well on patterns that deviate significantly from the average of the examples for each label, as it lacks the ability to capture and learn from the variability within each class. Consequently, this approach may not support true generalization, which is essential for recognizing different representations of the same pattern, as observed when the network struggles with the MNIST dataset due to multiple variations in each digit.

Moreover, there is concern that the network's predictions might rely on the manual tuning of the synaptic weights for specific patterns rather than allowing the network to autonomously learn and generalize from a diverse set of examples. This limitation is particularly evident in simpler datasets like XPUE and "012345", where only one case per pattern is presented, enabling the network to memorize the pattern rather than generalize across different instances. As such, this method in its current form does not fully demonstrate robust in situ training where the network can adapt and generalize to new inputs.

To address these issues, future work should focus on designing learning algorithms that allow the network to learn from multiple examples of the same class, thereby improving its ability to generalize and handle variability within each pattern. Techniques such as data augmentation or introducing intra-class variability in the training phase could help the network capture the diversity within each class and perform better on more complex datasets like MNIST.

Furthermore, the current implementation of the in situ memristive learning method is limited to a single-layer SNN. This restriction hinders the network's ability to learn more complex and hierarchical features, which are often essential for achieving high accuracy on challenging datasets.

Another limitation arises from the choice of the memristor emulator. While the use of this emulator has facilitated the testing of in situ learning, it differs from well established, experimentally validated models of real memristors. The I–V characteristics observed in the emulator, such as its perfect symmetry for positive and negative voltages and the unrealistically high current requirements (up to -1.5 A), do not align with those observed in real memristive devices, which typically operate in the microampere to milliampere range. A memristor with such high current requirements would be unsuitable for most real-world, energy-efficient applications. Future work will need to focus on either refining the memristor emulator to better approximate the non-linear, asymmetric I–V behavior of real memristors or directly integrating real, experimentally validated memristors into the system.

Additionally, simulations have not yet been compared to experimental data from real memristors, which would be essential for validating the approach. While the memristor emulator served as a useful tool for initial testing, future work should focus on directly comparing the proposed model's behavior with real memristive devices to ensure the findings are transferable to physical implementations. This comparison would involve not only I–V characteristics but also the scalability of the proposed method in hardware environments.

To address these limitations, future work should focus on developing techniques to handle input patterns with varying levels of dominance, allowing the network to learn from a wider range of datasets and real-world scenarios. Investigating methods to incorporate intra-class variability into the learning process, enabling the network to capture and learn from the diversity within each label, could involve exploring techniques such as data augmentation, regularization, or more advanced learning rules. Extending the in situ memristive learning method to support multi-layer SNNs would require developing techniques for the backpropagation of errors through the memristive crossbar arrays and designing efficient architectures for inter-layer communication and information flow.

Exploring the integration of the proposed method with other learning paradigms, such as unsupervised learning or reinforcement learning, could enhance the network's ability to learn from unlabeled data or adapt to dynamic environments. Investigating the scalability and robustness of the in situ memristive learning method when applied to larger and more complex datasets, such as ImageNet or real-world sensor data, is another important direction for future research. Studying the energy efficiency and computational performance of the proposed method compared to other neuromorphic computing approaches and identifying potential optimizations to improve its efficiency and scalability further could help advance the field of neuromorphic computing.

Regarding the current limitations of the memristor emulator, future research should aim to optimize the current range to make the device more suitable for energy-efficient applications. Real memristors typically operate in the microampere to milliampere range, and reducing the current requirements for switching would make the method more applicable for environmentally sustainable systems. Additionally, refining the memristor model to reflect the asymmetric switching characteristics commonly observed in real devices will help improve the fidelity of the simulations, thereby making the approach more practical for hardware implementation.

While our current design focuses on 2D memristive circuits for SNNs, emerging technologies such as printed electronics and 3D circuit integration offer promising avenues for further optimization [54,108,109]. Printed electronics, with their ability to layer materials like polymers and nanoparticles using methods like gravure and inkjet printing [50,51], could enable the development of large-scale, flexible neuromorphic systems. Furthermore, the ability to construct 3D circuits using automated cutting and folding techniques [52] could significantly improve packing density and performance [110–115]. Future neuromorphic architectures, including systems like SpiNNaker, IBM TrueNorth, Intel Loihi, and Tianjic [11,116], could benefit from this shift toward 3D designs, providing higher efficiency for neuromorphic computing and AI applications [114,117–119].

By addressing these limitations and exploring these future research directions, the in situ memristive learning method for pattern classification can be further developed into a more powerful and versatile neuromorphic computing approach, with potential applications in various domains such as computer vision, robotics, and edge computing.

6. Conclusions

This paper presents a novel approach to in situ memristive learning for neuromorphic systems, leveraging memristor emulators to perform real-time learning and pattern classification. A key contribution is the development of the first memristor emulator in PySpice, applied to a memristive circuit capable of in situ learning, which we refer to as the "In Situ Memristive Learning Method for Pattern Classification".

The proposed method was evaluated on three datasets: XPUE, 012345, and a resized 10×10 MNIST dataset. The neuromorphic circuit demonstrated its efficacy by achieving 100% training and test accuracy on both the XPUE and 012345 datasets. For the MNIST dataset, the network reached 100% training accuracy and 60% test accuracy on the first 100 samples, which represents a substantial improvement over previous SPICE-based simulations, such as those in [97], which reported a 30% test accuracy.

The in situ memristive learning method leverages the cumulative effect of memristors, allowing the network to efficiently learn a representative pattern for each label. This approach simplifies the training process by reducing the need for multiple examples per label, lowering the computational complexity. While this feature has proven effective for simple datasets, there is potential to expand the learning method to handle more complex tasks, as outlined in our future research directions.

This work also highlights the potential of the proposed method for use in 3D neuromorphic circuits, particularly through the integration of printed memristors. The scalability and flexibility of such circuits could open up new possibilities for neuromorphic computing applications. Although this study focuses on simulations, future implementations with real memristors could bring further improvements in scalability, performance, and energy efficiency.

Lastly, while the memristor emulator has demonstrated strong performance in specific tasks, future advancements will involve transitioning from emulators to real memristors. This can be achieved by employing voltage-controlled switches in conjunction with physical memristors, allowing current to flow based on input signals, or by incorporating a physical switch to regulate the flow of current. Such adaptations will be essential for realizing the full potential of this approach in hardware-based neuromorphic computing systems.

All the implementation files regarding this work are made available in [107].

Funding: This research received no external funding.

Data Availability Statement: The original contributions presented in the study are included in the article, further inquiries can be directed to the corresponding author.

Conflicts of Interest: The author declares no conflicts of interest.

Abbreviations and Symbols

Abbreviation/Symbol	Definition
SNNs	spiking neural networks
MNIST	Modified National Institute of Standards and Technology
AI	artificial intelligence
HBP	Human Brain Project
RRAM	Resistive Random-Access Memory
ANNs	Artificial Neural Networks
STDP	spike-timing-dependent plasticity
IF	Integrate-and-Fire neuron
LIF	Leaky Integrate-and-Fire neuron
2D	two-dimensional
3D	three-dimensional
VMM	vector–matrix multiplication
NMOSFET	n-channel MOSFET
TiO_2	Titanium dioxide
ms	milliseconds unit
$v(t)$	the voltage across the memristor
$i(t)$	the current through the memristor
$R(x(t))$	the resistance of the memristor
$x(t)$	the state variable that determines the resistance of the memristor
$f(x(t), i(t))$	a function that describes the evolution of the state variable $x(t)$ in response to the voltage across the memristor
$v(t) = \sin(t)$	sinusoidal voltage across the memristor
X_i	the i-th image or input, whose values are binary (either 0 or 1)
$X_{i_{k*n+l}}$	the pixel at position (k, l) of the image X_i
X_{i_j}	the j-th element of (flattened if necessary) input X_i
N	the number of pixels in the image (or the dimension of the input)
M	matrix
$M_{i,j,N}$	the memristors representing negative weights
$M_{i,j,P}$	the memristors representing positive weights
V_0 to V_n	the inputs of the layer
n	the size of the input data
m	the number of unique labels
$\phi_{i,j}$	control voltages
$G(x(t))$	the conductance which models the synaptic strength between neurons

Appendix A

Neuromorphic Circuits Components List

	XPUE	012345	MNIST
Component Type	**Details**		
Memristors	40×	96×	1010×
Batteries	26× 1 V, and 16× 20 V	40× 1 V, and 24× 20 V	111× 1 V, and 40× 20 V
Resistors	16× 1000 Ω, and 4× 10 Ω	24× 1000 Ω, and 6× 10 Ω	40× 1000 Ω, and 10× 10 Ω
Capacitors	4× 0.15 pF	6× 0.15 pF	10× 0.15 pF
Operational Amplifiers	8× (working with $> 15V\|vccs\|$ voltage, gain irrelevant, $V_{offset} = 0$), and 8× (working with 1V $\|vccs\|$ voltage, $V_{offset} = -5$ V)	12× (working with $> 15V\|vccs\|$ voltage, gain irrelevant, $V_{offset} = 0$), and 12× (working with 1V $\|vccs\|$ voltage, $V_{offset} = -5$ V)	20× (working with $> 15V\|vccs\|$ voltage, gain irrelevant, $V_{offset} = 0$), and 20× (working with 1V $\|vccs\|$ voltage, $V_{offset} = -5$ V)

References

1. Huang, W.; Xia, X.; Zhu, C.; Steichen, P.; Quan, W.; Mao, W.; Yang, J.; Chu, L.; Li, X. Memristive Artificial Synapses for Neuromorphic Computing. *Nano-Micro Lett.* **2021**, *13*, 85. [CrossRef] [PubMed]
2. Weilenmann, C.; Ziogas, A.; Zellweger, T.; Portner, K.; Mladenović, M.; Kaniselvan, M.; Moraitis, T.; Luisier, M.; Emboras, A. Single Neuromorphic Memristor Closely Emulates Multiple Synaptic Mechanisms for Energy Efficient Neural Networks. *arXiv* **2024**, arXiv:2402.16628. Available online: https://arxiv.org/abs/2402.16628 (accessed on 10 July 2024). [CrossRef] [PubMed]
3. Sung, C.; Hwang, H.; Yoo, I.K. Perspective: A Review on Memristive Hardware for Neuromorphic Computation. *J. Appl. Phys.* **2018**, *124*, 151903. [CrossRef]
4. Huang, Y.; Kiani, F.; Ye, F.; Xia, Q. From Memristive Devices to Neuromorphic Systems. *Appl. Phys. Lett.* **2023**, *122*, 110501. [CrossRef]
5. Patel, P.; Patel, M.; Solanki, A.; Roy, M. Memristor-Based Neuromorphic Computing and Artificial Neural Networks for Computer Vision and AI—Applications. In *Biomedical Imaging*; Gogoi, A., Mazumder, N., Eds.; Springer: Singapore, 2024; pp. 1–20. [CrossRef]
6. Wang, J.; Zhu, Y.; Zhu, L.; Chen, C.; Wan, Q. Emerging Memristive Devices for Brain-Inspired Computing and Artificial Perception. *Front. Nanotechnol.* **2022**, *4*, 940825. [CrossRef]
7. Gül, F. Memristive Synapses as Building Blocks of Neuromorphic Artificial Intelligence (AI) Hardware. In Proceedings of the 2024 8th International Artificial Intelligence and Data Processing Symposium (IDAP), Malatya, Turkiye, 21–22 September 2024; pp. 1–5. [CrossRef]
8. Martín, A.; Camacho, D. Recent advances on effective and efficient deep learning-based solutions. *Neural Comput. Appl.* **2022**, *34*, 10205–10210. [CrossRef]
9. Amunts, K.; Ebell, C.; Muller, J.; Telefont, M.; Knoll, A.; Lippert, T. The Human Brain Project: Creating a European research infrastructure to decode the human brain. *Neuron* **2016**, *92*, 574–581. [CrossRef]
10. Amunts, K.; Axer, M.; Banerjee, S.; Bitsch, L.; Bjaalie, J.G.; Brauner, P.; Brovelli, A.; Calarco, N.; Carrere, M.; Caspers, S.; et al. The coming decade of digital brain research: A vision for neuroscience at the intersection of technology and computing. *Imaging Neurosci.* **2024**, *2*, 1–35. [CrossRef]
11. Mehonic, A.; Kenyon, A.J. Brain-inspired computing needs a master plan. *Nature* **2022**, *604*, 255–260. [CrossRef]
12. Rabaey, J.M. *Digital Integrated Circuits: A Design Perspective*; Prentice-Hall, Inc.: Hoboken, NJ, USA, 1996.
13. Taur, Y.; Ning, T.H. *Fundamentals of Modern VLSI Devices*, 3rd ed.; Cambridge University Press: Cambridge, UK, 2021.
14. Gargini, P. The International Technology Roadmap for Semiconductors (ITRS): "Past, present and future". In Proceedings of the 22nd Annual Technical Digest GaAs IC Symposium (Cat. No.00CH37084), Seattle, WA, USA, 5–8 November 2000; pp. 3–5. [CrossRef]
15. Schuman, C.D.; Kulkarni, S.R.; Parsa, M.; Mitchell, J.P.; Date, P.; Kay, B. Opportunities for neuromorphic computing algorithms and applications. *Nat. Comput. Sci.* **2022**, *2*, 10–19. [CrossRef]
16. Mehonic, A.; Ambrogio, S.; Valov, I.; Waser, R.; Ielmini, D. Roadmap to Neuromorphic Computing with Emerging Technologies. *APL Mater.* **2024**, *12*, 100801. [CrossRef]
17. Yang, X.; Taylor, B.; Wu, A.; Chen, Y.; Chua, L.O. Research Progress on Memristor: From Synapses to Computing Systems. *IEEE Trans. Circuits Syst. I Regul. Pap.* **2022**, *69*, 1845–1857. [CrossRef]
18. Di Ventra, M.; Pershin, Y. *Memristors and Memelements: Mathematics, Physics and Fiction*; Springer: Cham, Switzerland, 2023. [CrossRef]
19. Chua, L. Memristor—The missing circuit element. *IEEE Trans. Circ. Theory* **1971**, *18*, 507–519. [CrossRef]
20. Strukov, D.; Snider, G.; Stewart, D.; Williams, R.S. The missing memristor found. *Nature* **2008**, *453*, 80–83. [CrossRef] [PubMed]
21. Williams, R.S. Aftermath of finding the memristor. In *Handbook of Memristor Networks*; Springer: Cham, Switzerland, 2019.
22. Xiao, Y.; Jiang, B.; Zhang, Z.; Ke, S.; Jin, Y.; Wen, X.; Ye, C. A review of memristor: Material and structure design, device performance, applications and prospects. *Sci. Technol. Adv. Mater.* **2023**, *24*, 2162323. [CrossRef]
23. Wabnitz, M.; Gemmeke, T. Toolflow for the algorithm-hardware co-design of memristive ANN accelerators. *Mem.-Mater. Devices Circuits Syst.* **2023**, *5*, 100066. [CrossRef]
24. Li, J.; Ren, S.-G.; Li, Y.; Yang, L.; Yu, Y.; Ni, R.; Zhou, H.; Bao, H.; He, Y.; Chen, J.; et al. Sparse matrix multiplication in a record-low power self-rectifying memristor array for scientific computing. *Sci. Adv.* **2023**, *9*, eadf7474. [CrossRef]
25. Jo, S.; Chang, T.; Ebong, I.; Bhadviya, B.; Mazumder, P.; Lu, W. Nanoscale Memristor Device as Synapse in Neuromorphic Systems. *Nano Lett.* **2010**, *10*, 1297–1301. [CrossRef]
26. Pi, S.; Li, C.; Jiang, H.; Xia, W.; Xin, H.; Yang, J.J.; Xia, Q. Memristor crossbar arrays with 6-nm half-pitch and 2-nm critical dimension. *Nat. Nanotechnol.* **2019**, *14*, 35–39. [CrossRef]
27. Jin, P.; Wang, G.; Liang, Y.; Chen, L.; Iu, H.H.-C.; Chua, L.O. Poor Man's Memristor: Chua Corsage Memristor. *IEEE Trans. Circuits Syst. II Express Briefs* **2023**, *70*, 3139–3143. [CrossRef]
28. Chua, L.O. Homemade US$10 Chua Corsage Memristor: Use It to Make the Poor Man's Biomimetic Neurons. *IEEE Electron Devices Mag.* **2023**, *1*, 10–22. [CrossRef]
29. Beasley, A.E.; Powell, A.L.; Adamatzky, A.I. Memristive Properties of Mushrooms. *arXiv* **2020**, arXiv:2002.06413.
30. Beasley, A.E.; Abdelouahab, M.S.; Lozi, R.; Tsompanas, M.A.; Powell, A.L.; Adamatzky, A. Mem-fractive properties of mushrooms. *Bioinspir. Biomim.* **2022**, *16*, 066026. [CrossRef]

31. Miranda, E.; Suñé, J. Memristors for Neuromorphic Circuits and Artificial Intelligence Applications. *Materials* **2020**, *13*, 938. [CrossRef]
32. Ye, L.; Gao, Z.; Fu, J.; Ren, W.; Yang, C.; Wen, J.; Wan, X.; Ren, Q.; Gu, S.; Liu, X.; et al. Overview of Memristor-Based Neural Network Design and Applications. *Front. Phys.* **2022**, *10*, 839243. [CrossRef]
33. Yao, P.; Wu, H.; Gao, B.; Tang, J.; Zhang, Q.; Zhang, W.; Yang, J.J.; Qian, H. Fully hardware-implemented memristor convolutional neural network. *Nature* **2020**, *577*, 641–646. [CrossRef]
34. Jia, X. *Design and Training of Memristor-Based Neural Networks*; UC Santa Cruz: Santa Cruz, CA, USA, 2020. Available online: https://escholarship.org/uc/item/52n874z5 (accessed on 3 October 2024).
35. Jo, Y.; Woo, D.Y.; Noh, G.; Park, E.; Kim, M.J.; Sung, Y.W.; Lee, D.K.; Park, J.; Kim, J.; Jeong, Y.; et al. Hardware Implementation of Network Connectivity Relationships Using 2D hBN-Based Artificial Neuron and Synaptic Devices. *Adv. Funct. Mater.* **2023**, *34*, 2309058. [CrossRef]
36. Bao, H.; Zhou, H.; Li, J.; Pei, H.; Tian, J.; Yang, L.; Ren, S.; Tong, S.; Li, Y.; He, Y.; et al. Toward memristive in-memory computing: Principles and applications. *Front. Optoelectron.* **2022**, *15*, 23. [CrossRef]
37. Wang, F.; Zhang, T.; Dou, C.; Shi, Y.; Pan, L. Neuromorphic Devices, Circuits, and Their Applications in Flexible Electronics. *IEEE J. Flex. Electron.* **2024**, *3*, 42–56. [CrossRef]
38. Radhakrishnan, A.; Palliyalil, J.; Babu, S.; Dorzhigulov, A.; James, A. PyMem: A Graphical User Interface Tool for Neuro-Memristive Hardware-Software Co-design. *IEEE Open J. Ind. Electron. Soc.* **2024**, *5*, 81–90. [CrossRef]
39. Kiani, F.; Yin, J.; Wang, Z.; Yang, J.J.; Xia, Q. A fully hardware-based memristive multilayer neural network. *Sci. Adv.* **2021**, *7*, eabj4801. [CrossRef] [PubMed]
40. TOP500 LIST—JUNE 2023. 2024. Available online: https://www.top500.org/lists/top500/list/2023/11/ (accessed on 3 October 2024).
41. Cueto-Mendoza, E.; Kelleher, J. A Framework for Measuring the Training Efficiency of a Neural Architecture. *Artif. Intell. Rev.* **2024**, *57*, 349. [CrossRef] [PubMed]
42. Patterson, D.; Gonzalez, J.; Holzle, U.; Le, Q.; Liang, C.; Munguia, L.-M.; Rothchild, D.; So, D.R.; Texier, M.; Dean, J. The Carbon Footprint of Machine Learning Training Will Plateau, Then Shrink. *Computer* **2022**, *55*, 18–28. [CrossRef]
43. Desislavov, R.; Plumed, F.; Hernandez-Orallo, J. Trends in AI inference energy consumption: Beyond the performance-vs-parameter laws of deep learning. *Sustain. Comput. Inform. Syst.* **2023**, *38*, 100857. [CrossRef]
44. Schwartz, R.; Dodge, J.; Smith, N.A.; Etzioni, O. Green AI. *arXiv* **2019**, arXiv:1907.10597v3. [CrossRef]
45. Strubell, E.; Ganesh, A.; McCallum, A. Energy and policy considerations for deep learning in NLP. *arXiv* **2019**, arXiv:1906.02243.
46. Jurj, S.L.; Rotar, R.; Opritoiu, F.; Vladutiu, M. Efficient Implementation of a Self-sufficient Solar-Powered Real-Time Deep Learning-Based System. In Proceedings of the 21st EANN (Engineering Applications of Neural Networks) 2020 Conference, Halkidiki, Greece, 5–7 June 2020; Springer: Cham, Switzerland, 2020; Volume 2, pp. 1–12. [CrossRef]
47. Jurj, S.L.; Opritoiu, F.; Vladutiu, M. Environmentally-Friendly Metrics for Evaluating the Performance of Deep Learning Models and Systems. In *Neural Information Processing*; Yang, H., Pasupa, K., Leung, A.C.S., Kwok, J.T., Chan, J.H., King, I., Eds.; Springer: Cham, Switzerland, 2020; Volume 12534, pp. 1–20. [CrossRef]
48. Verdecchia, R.; Sallou, J.; Cruz, L. A systematic review of Green AI. *WIREs Data Min. Knowl. Discov.* **2023**, *13*, e1507. [CrossRef]
49. Luo, T.; Wong, W.-F.; Goh, R.; Do, T.; Chen, Z.; Li, H.; Jiang, W.; Yau, W. Achieving Green AI with Energy-Efficient Deep Learning Using Neuromorphic Computing. *Commun. ACM* **2023**, *66*, 52–57. [CrossRef]
50. Dahiya, A.S.; Shakthivel, D.; Kumaresan, Y.; Zumeit, A.; Christou, A.; Dahiya, R. High-performance printed electronics based on inorganic semiconducting nano to chip scale structures. *Nano Converg.* **2020**, *7*, 33. [CrossRef]
51. Wiklund, J.; Karakoç, A.; Palko, T.; Yiğitler, H.; Ruttik, K.; Jäntti, R.; Paltakari, J. A Review on Printed Electronics: Fabrication Methods, Inks, Substrates, Applications and Environmental Impacts. *J. Manuf. Mater. Process.* **2021**, *5*, 89. [CrossRef]
52. Tan, H.W.; Choong, Y.; Kuo, C.-N.; Low, H.Y.; Chua, C. 3D Printed Electronics: Processes, Materials and Future Trends. *Prog. Mater. Sci.* **2022**, *127*, 100945. [CrossRef]
53. Ali, S.; Khan, S.; Khan, A.; Bermak, A. Memristor Fabrication Through Printing Technologies: A Review. *IEEE Access* **2021**, *9*, 95970–95985. [CrossRef]
54. Khan, Y.; Thielens, A.; Muin, S.; Ting, J.; Baumbauer, C.; Arias, A. A New Frontier of Printed Electronics: Flexible Hybrid Electronics. *Adv. Mater.* **2019**, *32*, 1905279. [CrossRef] [PubMed]
55. Lin, R.; Shi, G.; Qiao, F.; Wang, C.; Wu, S. Research progress and applications of memristor emulator circuits. *Microelectron. J.* **2023**, *133*, 105702. [CrossRef]
56. Strutwolf, J.; Chen, Y.; Ullrich, J.; Dehnert, M.; Hübler, A.C. Memristive devices based on mass printed organic resistive switching layers. *Appl. Phys. A* **2021**, *127*, 709. [CrossRef]
57. Jurj, S.L.; Nouri, S.B.; Strutwolf, J. SNNtrainer3D: Training Spiking Neural Networks Using a User-Friendly Application with 3D Architecture Visualization Capabilities. *Appl. Sci.* **2024**, *14*, 5752. [CrossRef]
58. Eshraghian, J.; Ward, M.; Neftci, E.; Wang, X.; Lenz, G.; Dwivedi, G.; Bennamoun, M.; Jeong, D.S.; Lu, W. Training Spiking Neural Networks Using Lessons From Deep Learning. *Proc. IEEE* **2023**, *111*, 1016–1054. [CrossRef]
59. Yamazaki, K.; Vo-Ho, V.K.; Bulsara, D.; Le, N. Spiking Neural Networks and Their Applications: A Review. *Brain Sci.* **2022**, *12*, 863. [CrossRef]

60. Pietrzak, P.; Szczęsny, S.; Huderek, D.; Przyborowski, Ł. Overview of Spiking Neural Network Learning Approaches and Their Computational Complexities. *Sensors* **2023**, *23*, 3037. [CrossRef]
61. An, H.; Al-Mamun, M.S.; Orlowski, M.K.; Yi, Y. A Three-dimensional (3D) Memristive Spiking Neural Network (M-SNN) System. In Proceedings of the 2021 22nd International Symposium on Quality Electronic Design (ISQED), Santa Clara, CA, USA, 7–9 April 2021; pp. 337–342. [CrossRef]
62. Sanaullah, K.S.; Rückert, U.; Jungeblut, T. Exploring spiking neural networks: A comprehensive analysis of mathematical models and applications. *Front. Comput. Neurosci.* **2023**, *17*, 1215824. [CrossRef]
63. Lemaire, E.; Cordone, L.; Castagnetti, A.; Novac, P.E.; Courtois, J.; Miramond, B. An Analytical Estimation of Spiking Neural Networks Energy Efficiency. In *Neural Information Processing*; Tanveer, M., Agarwal, S., Ozawa, S., Ekbal, A., Jatowt, A., Eds.; Springer: Cham, Switzerland, 2023; Volume 13623, pp. 1–11. [CrossRef]
64. Khacef, L.; Klein, P.; Cartiglia, M.; Rubino, A.; Indiveri, G.; Chicca, E. Spike-based local synaptic plasticity: A survey of computational models and neuromorphic circuits. *Neuromorphic Comput. Eng.* **2023**, *3*, 042001. [CrossRef]
65. Dampfhoffer, M.; Mesquida, T.; Valentian, A.; Anghel, L. Are SNNs really more energy-efficient than ANNs? An in-depth hardware-aware study. *IEEE Trans. Emerg. Top. Comput. Intell.* **2022**, *2022*, 731–741. [CrossRef]
66. Ottati, F.; Gao, C.; Chen, Q.; Brignone, G.; Casu, M.R.; Eshraghian, J.K.; Lavagno, L. To Spike or Not To Spike: A Digital Hardware Perspective on Deep Learning Acceleration. *IEEE J. Emerg. Sel. Top. Circuits Syst.* **2023**, *13*, 1015–1025. [CrossRef]
67. Huang, J.; Serb, A.; Stathopoulos, S.; Prodromakis, T. Text Classification in Memristor-based Spiking Neural Networks. *arXiv* **2022**, arXiv:2022.12345. [CrossRef]
68. Song, S.; Miller, K.; Abbott, L. Competitive Hebbian learning through spike-timing-dependent synaptic plasticity. *Nat. Neurosci.* **2000**, *3*, 919–926. [CrossRef]
69. Siddique, A.; Vai, M.I.; Pun, S.H. A low cost neuromorphic learning engine based on a high performance supervised SNN learning algorithm. *Sci. Rep.* **2023**, *13*, 6280. [CrossRef]
70. Chen, L.; Xiong, X.; Liu, J. A Survey of Intelligent Chip Design Research Based on Spiking Neural Networks. *IEEE Access* **2022**, *10*, 89663–89686. [CrossRef]
71. Ganguly, C.; Bezugam, S.S.; Abs, E.; Payvand, M.; Dey, S.; Suri, M. Spike frequency adaptation: Bridging neural models and neuromorphic applications. *Commun. Eng.* **2024**, *3*, 22. [CrossRef]
72. Aguirre, F.; Sebastian, A.; Le Gallo, M.; Song, W.; Wang, T.; Yang, J.J.; Lu, W.; Chang, M.-F.; Ielmini, D.; Yang, Y.; et al. Hardware implementation of memristor-based artificial neural networks. *Nat. Commun.* **2024**, *15*, 1974. [CrossRef]
73. Saxena, V. A Compact CMOS Memristor Emulator Circuit and its Applications. In Proceedings of the 2018 IEEE 61st International Midwest Symposium on Circuits and Systems (MWSCAS), Windsor, ON, Canada, 5–8 August 2018; pp. 190–193. [CrossRef]
74. Vohra, S.K.; Thomas, S.A.; Sakare, M.; Das, D.M. CMOS Circuit Implementation of Spiking Neural Network for Pattern Recognition Using On-chip Unsupervised STDP Learning. *arXiv* **2022**, arXiv:2204.04430.
75. Jaafar, A.; Gee, A.; Kemp, N. Printed and Flexible Organic and Inorganic Memristor Devices for Non-volatile Memory Applications. *J. Phys. D Appl. Phys.* **2023**, *56*, 503002. [CrossRef]
76. Hu, H.; Scholz, A.; Dolle, C.; Zintler, A.; Quintilla, A.; Liu, Y.; Tang, Y.; Breitung, B.; Marques, G.C.; Eggeler, Y.; et al. Inkjet-Printed Tungsten Oxide Memristor Displaying Non-Volatile Memory and Neuromorphic Properties. *Adv. Funct. Mater.* **2023**, *34*, 2302290. [CrossRef]
77. Singaraju, S.A.; Weller, D.D.; Gspann, T.S.; Aghassi-Hagmann, J.; Tahoori, M.B. Artificial Neurons on Flexible Substrates: A Fully Printed Approach for Neuromorphic Sensing. *Sensors* **2022**, *22*, 4000. [CrossRef]
78. Xu, W.; Wang, J.; Yan, X. Advances in Memristor-Based Neural Networks. *Front. Nanotechnol.* **2021**, *3*, 645995. [CrossRef]
79. Cao, Z.; Sun, B.; Zhou, G.; Mao, S.; Zhu, S.; Zhang, J.; Ke, C.; Zhao, Y.; Shao, J. Memristor-based neural networks: A bridge from device to artificial intelligence. *Nanoscale Horiz.* **2023**, *8*, 716–745. [CrossRef]
80. Gökgöz, B.; Gül, F.; Aydın, T. An overview memristor based hardware accelerators for deep neural network. *Concurr. Comput. Pract. Exp.* **2024**, *36*, e7997. [CrossRef]
81. Duan, X.; Cao, Z.; Gao, K.; Yan, W.; Sun, S.; Zhou, G.; Wu, Z.; Ren, F.; Sun, B. Memristor-Based Neuromorphic Chips. *Adv. Mater.* **2024**, *36*, 2310704. [CrossRef]
82. Bian, J.; Liu, Z.; Tao, Y.; Wang, Z.; Zhao, X.; Lin, Y.; Xu, H.; Liu, Y. Advances in Memristor Based Artificial Neuron Fabrication-Materials, Models, and Applications. *Int. J. Extrem. Manuf.* **2023**, *6*, 012002. [CrossRef]
83. Eshraghian, J.K.; Wang, X.; Lu, W.D. Memristor-Based Binarized Spiking Neural Networks: Challenges and applications. *IEEE Nanotechnol. Mag.* **2022**, *16*, 14–23. [CrossRef]
84. Gupta, J.; Koppad, D. A survey on Memristor and CMOS based Spiking Neural Networks. In Proceedings of the 2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA), Coimbatore, India, 15–17 July 2020; pp. 1052–1058. [CrossRef]
85. Zhang, X.; Wang, X.; Ge, Z.; Li, Z.; Wu, M.; Borah, S. A Novel Memristive Neural Network Circuit and Its Application in Character Recognition. *Micromachines* **2022**, *13*, 2074. [CrossRef]
86. Routhu, G.; Singh, N.P.; Raja, S.; Reddy, E.S.K. Investigation of Memristor-Based Neural Networks on Pattern Recognition. *Eng. Proc.* **2023**, *34*, 9. [CrossRef]
87. Wang, L.; Meng, Q.; Wang, H.; Jiang, J.; Wan, X.; Liu, X.; Lian, X.; Cai, Z. Digital image processing realized by memristor-based technologies. *Discov. Nano* **2023**, *18*, 120. [CrossRef]

88. Oh, S.; An, J.; Min, K.-S. Area-Efficient Mapping of Convolutional Neural Networks to Memristor Crossbars Using Sub-Image Partitioning. *Micromachines* **2023**, *14*, 309. [CrossRef]
89. Dalgaty, T.; Moro, F.; Demirağ, Y.; De Pra, A.; Indiveri, G.; Vianello, E.; Payvand, M. Mosaic: In-memory computing and routing for small-world spike-based neuromorphic systems. *Nat. Commun.* **2024**, *15*, 142. [CrossRef]
90. Jebali, F.; Majumdar, A.; Turck, C.; Harabi, K.-E.; Faye, M.-C.; Muhr, E.; Walder, J.-P.; Bilousov, O.; Michaud, A.; Vianello, E.; et al. Powering AI at the edge: A robust, memristor-based binarized neural network with near-memory computing and miniaturized solar cell. *Nat. Commun.* **2024**, *15*, 741. [CrossRef]
91. Zhang, W.; Yao, P.; Gao, B.; Liu, Q.; Wu, D.; Zhang, Q.; Li, Y.; Qin, Q.; Li, J.; Zhu, Z.; et al. Edge learning using a fully integrated neuro-inspired memristor chip. *Science* **2023**, *381*, 1205–1211. [CrossRef]
92. Lin, P.; Li, C.; Wang, Z.; Li, Y.; Jiang, H.; Song, W.; Rao, M.; Zhuo, Y.; Upadhyay, N.K.; Barnell, M.; et al. Three-dimensional memristor circuits as complex neural networks. *Nat. Electron.* **2020**, *3*, 225–232. [CrossRef]
93. Lin, P.; Xia, Q. Three-dimensional hybrid circuits: The future of neuromorphic computing hardware. *Nano Express* **2021**, *2*, 031003. [CrossRef]
94. Thomas, S.A.; Vohra, S.K.; Kushwaha, S.; Sharma, R.; Das, D.M. Modeling and Analysis of CMOS-based Folded Memristive Crossbar Array for 3D Neuromorphic Integrated Circuits. In Proceedings of the 2023 IEEE 73rd Electronic Components and Technology Conference (ECTC), Orlando, FL, USA, 30 May–2 June 2023; pp. 960–966. [CrossRef]
95. An, H.; Al-Mamun, M.S.; Orlowski, M.K.; Liu, L.; Yi, Y. Three-Dimensional Neuromorphic Computing System With Two-Layer and Low-Variation Memristive Synapses. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2022**, *41*, 400–409. [CrossRef]
96. Salvaire, F. PySpice. Available online: https://pyspice.fabrice-salvaire.fr/releases/v1.4/installation.html (accessed on 3 October 2024).
97. Kim, M. Implementation of MNIST Dataset Learning Using Analog Circuit. *arXiv* **2023**, arXiv:2308.16307.
98. Mladenov, V.; Tsenov, G.; Kirilov, S. Memristor-Based Neural Network Implementation with Adjustable Synaptic Weights in LT-SPICE. In Proceedings of the 2023 International Conference Automatics and Informatics (ICAI), Varna, Bulgaria, 5–7 October 2023; pp. 403–408. [CrossRef]
99. Mladenov, V.; Kirilov, S. A Memristor Neural Network Based on Simple Logarithmic-Sigmoidal Transfer Function with MOS Transistors. *Electronics* **2024**, *13*, 893. [CrossRef]
100. Panagiotopoulou, C.; Michailidis, A.; Noulis, T.; Siozios, K.; Siskos, S. A Python Framework for System-on-Chip Power Integrity Simulation. In Proceedings of the 2023 IEEE International Conference on Industrial Technology (ICIT), Orlando, FL, USA, 4–6 April 2023; pp. 1–5. [CrossRef]
101. Popov, S.; Hinov, N. Comparative Analysis of Software Environments for Computer Modeling in Electrical and Electronic Engineering. In Proceedings of the 2023 International Conference on Information Technologies (InfoTech), Varna, Bulgaria, 20–21 September 2023; pp. 1–6. [CrossRef]
102. Campbell, K. Self-Directed Channel Memristor for High Temperature Operation. *Microelectron. J.* **2016**, *59*, 10–14. [CrossRef]
103. Gandolfi, D.; Benatti, L.; Zanotti, T.; Boiani, G.M.; Bigiani, A.; Puglisi, F.M.; Mapelli, J. Information Transfer in Neuronal Circuits: From Biological Neurons to Neuromorphic Electronics. *Intell. Comput.* **2024**, *3*, 59. [CrossRef]
104. Chen, J.; Xiao, M.; Chen, Z.; Khan, S.; Ghosh, S.; Macadam, N.; Chen, Z.; Zhou, B.; Yun, G.; Wilk, K.; et al. Inkjet-Printed High-Yield, Reconfigurable, and Recyclable Memristors on Paper. *arXiv* **2023**, arXiv:2312.16501.
105. Ngspice, the Open-Source Spice Circuit Simulator. Available online: https://ngspice.sourceforge.io/ (accessed on 3 October 2024).
106. Jurj, S.L. Efficient Implementation of Spiking Neural Networks for Inference using Ex-Situ Training. *IEEE Access* **2024**, *12*, 135014–135027. [CrossRef]
107. Jurj, S.L. SPICE-Based In-Situ Memristive Learning Method for Pattern Classification Using Spiking Neural Networks. Available online: https://github.com/jurjsorinliviu/In-Situ-Memristive-Learning-Method-for-Pattern-Classification (accessed on 3 October 2024).
108. Martins, P.; Pereira, N.; Lima, A.C.; Garcia, A.; Mendes-Filipe, C.; Policia, R.; Correia, V.; Lanceros-Mendez, S. Advances in Printing and Electronics: From Engagement to Commitment. *Adv. Funct. Mater.* **2023**, *33*, 2213744. [CrossRef]
109. Mack, C. *Fundamental Principles of Optical Lithography: The Science of Microfabrication*; Wiley: Hoboken, NJ, USA, 2007. [CrossRef]
110. Chau, R. Process and Packaging Innovations for Moore's Law Continuation and Beyond. In Proceedings of the 2019 IEEE International Electron Devices Meeting (IEDM), San Francisco, CA, USA, 7–11 December 2019; pp. 1.1.1–1.1.6. [CrossRef]
111. Wang, X.; Chen, D.; Li, D.; Kou, C.; Yang, Y. The Development and Progress of Multi-Physics Simulation Design for TSV-Based 3D Integrated System. *Symmetry* **2023**, *15*, 418. [CrossRef]
112. Lee, T.K.; Yang, H.; Dutta, I. Damage Mechanisms in Through-Silicon Vias Due to Thermal Exposure and Electromigration. *J. Electron. Mater.* **2023**, *53*, 1–9. [CrossRef]
113. Wang, T.-Y.; Meng, J.-L.; Chen, L.; Zhu, H.; Sun, Q.; Ding, S.; Bao, W.; Zhang, D.W. Flexible 3D memristor array for binary storage and multi-states neuromorphic computing applications. *InfoMat* **2021**, *3*, 212–221. [CrossRef]
114. Kurshan, E.; Li, H.; Seok, M.; Xie, Y. A Case for 3D Integrated System Design for Neuromorphic Computing & AI Applications. *arXiv* **2021**, arXiv:2103.04852. Available online: https://arxiv.org/abs/2103.04852 (accessed on 12 August 2024).
115. An, H.; Bai, K.; Yi, Y. The Roadmap to Realize Memristive Three-Dimensional Neuromorphic Computing System. *Adv. Memristor Neural Netw.-Model. Appl.* **2018**, 25–44. [CrossRef]

116. Gonzalez, H.A.; Huang, J.; Kelber, F.; Nazeer, K.K.; Langer, T.; Liu, C.; Mayr, C. Spinnaker2: A large-scale neuromorphic system for event-based and Asynchronous Machine Learning. *arXiv* **2024**, arXiv:2401.04491.
117. Han, J.-K.; Lee, J.-W.; Kim, Y.; Kim, Y.B.; Yun, S.-Y.; Lee, S.-W.; Yu, J.-M.; Lee, K.J.; Myung, H.; Choi, Y.-K. 3D Neuromorphic Hardware with Single Thin-Film Transistor Synapses Over Single Thin-Body Transistor Neurons by Monolithic Vertical Integration. *Adv. Sci.* **2023**, *10*, 2302380. [CrossRef]
118. Huo, Q.; Yang, Y.; Wang, Y.; Lei, D.; Fu, X.; Ren, Q.; Xu, X.; Luo, Q.; Xing, G.; Chen, C.; et al. A computing-in-memory macro based on three-dimensional resistive random-access memory. *Nat. Electron.* **2022**, *5*, 469–477. [CrossRef]
119. Jayachandran, D.; Pendurthi, R.; Sadaf, M.U.K.; Sakib, N.U.; Pannone, A.; Chen, C.; Han, Y.; Trainor, N.; Kumari, S.; Mc Knight, T.V.; et al. Three-dimensional integration of two-dimensional field-effect transistors. *Nature* **2024**, *625*, 276–281. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.