Exploiting Non-idealities of Resistive Switching Memories for Efficient Machine Learning
Victor You, Amirali Amirsoleimani, Fabien Alibart, Roger Melko, Dominique Drouin, Yann Beilliard

To cite this version:

HAL Id: hal-03679671
https://hal.science/hal-03679671
Submitted on 27 May 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License
Exploiting Non-idealities of Resistive Switching Memories for Efficient Machine Learning

Victor Yon1,2*, Amirali Amirsoleimani3, Fabien Alibart2,4, Roger G. Melko5,6, Dominique Drouin1,2 and Yann Beilliard1,2

1Institut Interdisciplinaire d’Innovation Technologique (3IT), Université de Sherbrooke, Sherbrooke, QC, Canada, 2Laboratoire Nanotechnologies Nanosystèmes (LN2)—CNRS 3463, Université de Sherbrooke, Sherbrooke, QC, Canada, 3Department of Electrical Engineering and Computer Science, Lassonde School of Engineering, York University, Toronto, ON, Canada, 4Institute of Electronics, Microelectronics and Nanotechnology (IEMN), Villeneuve d’Ascq, France, 5Department of Physics and Astronomy, University of Waterloo, Waterloo, ON, Canada, 6Perimeter Institute for Theoretical Physics, Waterloo, ON, Canada

Novel computing architectures based on resistive switching memories (also known as memristors or RRAMs) have been shown to be promising approaches for tackling the energy inefficiency of deep learning and spiking neural networks. However, resistive switch technology is immature and suffers from numerous imperfections, which are often considered limitations on implementations of artificial neural networks. Nevertheless, a reasonable amount of variability can be harnessed to implement efficient probabilistic or approximate computing. This approach turns out to improve robustness, decrease overfitting and reduce energy consumption for specific applications, such as Bayesian and spiking neural networks. Thus, certain non-idealities could become opportunities if we adapt machine learning methods to the intrinsic characteristics of resistive switching memories. In this short review, we introduce some key considerations for circuit design and the most common non-idealities. We illustrate the possible benefits of stochasticity and compression with examples of well-established software methods. We then present an overview of recent neural network implementations that exploit the imperfections of resistive switching memory, and discuss the potential and limitations of these approaches.

Keywords: resistive switching memories, memristor, in-memory computing, hardware non-idealities, artificial neural networks, bayesian neural networks, probabilistic computing

INTRODUCTION

The recent success of machine learning has essentially arisen from breakthroughs in learning algorithms, data availability, and computing resources. The latter benefited from the consistent increase in the number of transistors in complementary metal-oxide-semiconductor (CMOS) microchips and the development of highly parallel and specialized hardware (Reuther et al., 2019) such as graphics processing units (GPUs) (Keckler et al., 2011) and tensor processing units (TPUs) (Jouppi et al., 2017). Nevertheless, even with such optimizations, the traditional computing architecture is inappropriate for the implementation of modern machine learning algorithms due to the intensive data transfer requirement between memory and processing units (Horowitz, 2014; Ankit et al., 2017; Sze et al., 2017). Novel architectures based on resistive switching memories (RSMs) can take advantage of non-volatile in-memory computing (Mutlu et al., 2019) to efficiently perform vector-matrix multiplications (VMMs) (Gu et al., 2015; Amirsoleimani et al.,...
2020), the most critical operation of neural networks (NNs) inference. This technology is also an excellent candidate for the implementation of membrane potential and activation functions for brain-inspired spiking neural networks (SNNs) (Xia and Yang, 2019; Wang et al., 2020a; Yang et al., 2020a; Agrawal et al., 2021). Therefore, this emerging technology offers an opportunity to tackle current limitations of traditional computing hardware, such as energy efficiency, computation speed, and integration footprint (Chen, 2020; Marković et al., 2020; Christensen et al., 2021).

However, in the context of in-memory computing, RSM technology is still in its infancy compared to the much more mature CMOS technology at the heart of traditional von Neumann computers. RSMs are subject to variability and performance issues (Adam et al., 2018; Wang et al., 2019a; Krestinskaya et al., 2019; Chakraborty et al., 2020; Zahoor et al., 2020; Zhao et al., 2020; Xi et al., 2021), which currently restrict their usage to small and noisy NN hardware implementations that can solve only simple problems such as classification of the MNIST digit database (Ambrogio et al., 2018; Hu et al., 2018; Li et al., 2018; Wang et al., 2019b; Lin et al., 2019; Liu et al., 2020b; Yao et al., 2020; Zahari et al., 2020).

Since a wide variety of resistive memory technologies are still at an early stage of development (Zahoor et al., 2020; Christensen et al., 2021), it may be that a better understanding of resistive switching mechanisms combined with improvements in fabrication processes in the future will rectify some of their non-idealities (e.g., device-to-device variability and programming nonlinearity). But, even with technological maturity, other imperfections may remain due to their inseparable relationship with the physics of the device/circuit (e.g., programming variability and interconnect resistance). Numerous methods have therefore been proposed to mitigate the effect of these non-idealities for NN applications (Chen et al., 2015; Lim et al., 2018; He et al., 2019; Liu et al., 2020a; Wang et al., 2020b; Mahmoodi et al., 2020; Pan et al., 2020; Zhang et al., 2020; Xi et al., 2021). Although a mitigation approach can significantly increase the performance of RSM-based NNs, none of these methods are able to achieve the accuracy of their software counterparts. An alternative strategy consists in harnessing device imperfections rather than fighting them, which would enable highly efficient probabilistic and approximate computing hardware. This strategy is particularly appealing since software and biological NNs already take advantage of randomness to enhance information processing (McDonnell and Ward, 2011). The goal of this article is to review the latest progress in the exploitation of hardware imperfections by RSM-based NNs.

PRIOR DESIGN CHOICES

Design choices can influence the overall performance of RSM-based systems and induce different types and amounts of non-idealities. Starting with the choice of resistive switching mechanisms (Sung et al., 2018; Xia and Yang, 2019; Zahoor et al., 2020; Christensen et al., 2021) (e.g., valence change, electrochemical metallization, phase-change, ferroelectricity, magnetoresistivity), which could all be implemented with numerous materials and fabrication processes. For example, magnetoresistive memories are known to be especially durable [$>10^{15}$ cycles (Kan et al., 2016)], whereas ferroelectric memories benefit from highly linear resistance programming (Tian et al., 2020), and valence change memories usually offer a high on/off ratio [$>10^2$ (Chen et al., 2017)]. Despite these many options, an ideal RSM that combines all of these desirable properties has yet to be discovered. Until this is found, the design of an RSM-based computing system will always involve a trade-off between non-idealities (Siegel et al., 2020).

The design of the circuit architecture is also a critical step of the system conception. RSMs are typically organized in the form of an array when they represent the synaptic weights of a NN. Therefore, the size of this array limits the dimension of the VMM operation that can be computed in a single step. Large-scale implementations have been reported (Mochida et al., 2018; Ishii et al., 2019); for example, the system designed by Ambrogio et al. (2018) contained 524 k cells and could implement 204,900 synapses of a fully connected 4-layer NN. There are at least three strategies for selecting each of these cells (Chen and Yu, 2015; Wang et al., 2019a): 1) active selectors, 1-Transistor-n-Resistor (1TnR), in which one transistor allows access to one or a group of RSMs; 2) passive selectors, 1-Selector-1-Resistor (1S1R), which is usually a diode vertically stacked with the resistive material; 3) and no selector, 1-Resistor (1R).

Finally, the peripheral circuit used to address and control each RSM must not be neglected, since it constrains system performance and impact the speed and energy efficiency (Li et al., 2016; Kadetotad et al., 2017; Amirsoleimani et al., 2020; Li and Ang, 2020). In particular, the limited resolution of the digital-to-analog converter (DAC) and the analog-to-digital converter (ADC) will restrict the input/output precision and produce a significant energy overhead. This external circuit is also used to send writing pulses using a specific programming scheme that affects the performance (Woo et al., 2016; Stathopoulos et al., 2017; Chen et al., 2020). A write-verify scheme (Papandreou et al., 2011; Yi et al., 2011; Alibart et al., 2012; Perez et al., 2017; Pan et al., 2020) typically neglects energy consumption and memory lifetime for the sake of accuracy.

NON-IDEAL CHARACTERISTICS

Device Level

Modern fabricated RSM-based systems are imperfect at many levels (Figure 1). At the device level, one of the most critical characteristics for NN applications is the resistance range (Yu, 2018), since this property constrains the possible values of the model’s parameters. This metric is often expressed as the ratio between the minimal and maximal resistance values. Moreover, by considering the smallest writing value addressable with the peripheral circuit, we can estimate the total number of intermediate resistance states in the memory. This number can vary from 2 states for binary RSMs (Wong et al., 2012; Bocquet et al., 2018; Zahari et al., 2020) to more than 128 for the most
accurate devices (Gao et al., 2015; Li et al., 2017; Ambrogio et al., 2018; Wu et al., 2018). However, to estimate the quantity of information that a RSM can store, only reproducible and distinguishable states should be considered, which reduces this number to around 64-level (equivalent to 6-bits) in the best case (Li et al., 2017; Stathopoulos et al., 2017).

Thus, even with the best control circuit, it appears to be impossible to program a resistance state under an arbitrary precision threshold, usually between 1–5% of the resistance range (Adam et al., 2018; Xia and Yang, 2019; Xi et al., 2021). This writing variability is most likely attributable to the local environment [room temperature (Abunahla et al., 2016; Bunnam et al., 2020; Roldán et al., 2021), humidity (Messerschmitt et al., 2015; Valov and Tsuruoka, 2018)] and the internal state of the RSM at the atomic scale. In valence change memories, for example, the conductive filament may break abruptly and result in a state that is more resistive than expected (Gao et al., 2009; González-Cordero et al., 2017; Wiefels et al., 2020). The reading process is also affected by the variability, and several phenomena such as random telegraph noise (Ielmini et al., 2010; Lee et al., 2011; Yekels et al., 2013; Claeyss et al., 2016) and thermal noise (Bae and Yoon, 2020) can disturb the measured resistance value and lead to inaccurate outputs.

The dynamics of the resistance programming can also pose a challenge for NN training (Sidler et al., 2016; Woo and Yu, 2018). The same writing pulse can lead to a different outcome depending on the current resistance value (programming nonlinearity) (Jacobs-Gedrim et al., 2017) and the update direction (programming asymmetry). However, if the behavior of the RSM is well characterized, its programming dynamics can be partially anticipated and taken into account during the NN training (Chang et al., 2018; Lim et al., 2018; Pan et al., 2020).

Finally, a RSM is never perfectly stable over time, data retention can vary from a few seconds (Oh et al., 2019) to more than 10 years (Wei et al., 2008), depending on the resistive technology and the local environment (Gao et al., 2011; Subheeha et al., 2016; Kang et al., 2017; Zhao et al., 2019). Moreover, the number of writing operations is also limited [up to $10^{10}$ (Yang et al., 2010)], each change in resistance will slightly degrade the characteristics of the RSM, which will decay until breakdown is eventually reached.

**Array Level**

To efficiently compute VMM, RSMs must be arranged in large arrays surrounded by CMOS-based control electronics. Although this structure takes advantage of the two terminals of this device to maximize the integration density and offer a good scaling perspective, this integration (whether in two or three dimensions) faces technical issues (Li and Ang, 2020), including the inevitable resistance of the interconnections (Mahmoodi et al., 2020). Hence, even if two RSMs have the same internal state, the resistance of the metallic lines will affect the total resistance depending on the RSMs position in the crossbar. Furthermore, in 1R arrays (and to a lesser extent in 1S1R arrays), the sneak path current could also be a concern (Cassuto et al., 2013; Chen et al., 2021b). This phenomenon occurs when the electric current follows an unexpected path, leading to corruption of the final
output. These two technical challenges motivated the development of tile-based architectures (Shafee et al., 2016; Nag et al., 2018), in which a crossbar is split into several smaller ones.

The fabrication processes of RSMs typically induce significant variability between devices, meaning that each characteristic identified in Section 3.1 may differ for each RSM in a given crossbar. In extreme cases, the resistive state may even be stuck at its maximal or minimal value. The current best fabrication techniques can approach a yield of 99% (Li et al., 2017; Ambrogio et al., 2018). Although variability and faults will strongly impact the performance of a NN in the case of ex-situ (offline) training (Boquet et al., 2021), a NN trained in-situ (online) can mitigate these imperfections to a certain extent (Alibart et al., 2013; Li et al., 2018; Wang et al., 2019b; Romero et al., 2019).

Writing and reading operations in a crossbar can also induce unwanted disturbances in the resistance state of the RSMs (Yan et al., 2017; Wang et al., 2018; Amirsoleimani et al., 2021). The writing operation is usually achieved by applying a tension $V$ to the device we want to program, where $V$ is the threshold voltage required to change the resistance value. However, the surrounding devices that share the bottom or top electrode with the target receive a tension $V/2$, which can alter the RSM state after a large number of these operations, even though the tension is below the theoretical writing threshold. The same side effect can be observed for the read operation if the reading tension is not low enough to guarantee no disturbance.

The device and array level non-idealities listed in this section impact the overall quality of the NNs. The training of these models typically relies on numerous parameter updates, computed by gradient descent, which is strongly affected by inconsistent writing operation. A NN trained on such imperfect hardware will either reach a suboptimal state or simply fail to converge to a solution. While the drift of resistance in time and the non-idealities that damage the read operation (Figure 1) will lead to inaccuracy during the inference.

**TAKING ADVANTAGE OF NON-IDEALITIES**

**Software Methods**

With 32-bits floating-point variables, noiseless computation and very large-scale integration, there are solid arguments for using traditional computers for machine learning computation. Nevertheless, this impressive accuracy becomes a curse when the parameters of the model are so numerous and precise that they are able to extract the residual variation of the training data. This overfitting issue is a real challenge for modern NNs, but fortunately, many solutions now exist to mitigate this problem. Surprisingly, the most common methods are similar to what we would consider non-idealities in the field of RSMs.

The most popular approaches are probably dropout (Hinton et al., 2012) and drop connect (Wan et al., 2013). These regularization techniques consist in randomly omitting a subset of activation units or weights during NNs training. This turns out to be very effective to prevent complex co-adaptations of the units that usually lead to overfitting. Another counterintuitive but efficient strategy is to purposely add noise at different stages of the training process (An, 1996; McDonnell and Ward, 2011; Qin and Vucinic, 2018), in particular to the gradient value (Neelakantan et al., 2015), activation functions (Gulcehre et al., 2016), model parameters (He et al., 2019), and layers input (Liu et al., 2017; Creswell et al., 2018; Rakin et al., 2018). The injection of an appropriate level of noise can improve generalization, reduce training losses, and increase the robustness of these models against adversarial attacks.

The success of quantization techniques (Guo, 2018; Mishra et al., 2020; Chen et al., 2021a) also indicate that very accurate parameters are not mandatory to implement reliable NNs. Some works have achieved compression from 32 to 16-bits without sacrificing the final accuracy (Gupta et al., 2015; Micikevicius et al., 2017), and down to 8 or 3-bits with acceptable performance loss depending on the targeted application (Holt and Baker, 1991; Anwar et al., 2015; Shin et al., 2015). This idea can be extended to binarized parameters (Courbariaux et al., 2015) and activation functions (Zhou et al., 2016), which drastically reduce the computational cost although with a significant loss of accuracy.

Although regularization methods based on stochastic processes are now well-established in the machine learning community, their implementation on von Neumann computers is very inefficient, due to the serial and deterministic nature of the hardware. In particular, the high power consumption of random number generation (Cai et al., 2018; Gross and Gaudet, 2019) with standard CMOS technologies calls for the development of novel hardware that natively implements stochasticity. In the next section, we will see that RSMs have attracted much attention over recent years in this regard.

**RSM-Based Methods**

RSM-based NNs have been successfully used on hardware to solve simple data-driven problems. The state-of-the-art 1T1R array can classify MNIST digits with >96% accuracy (Ambrogio et al., 2018; Yao et al., 2020). Those high scores are often made possible by mitigating the negative impact of non-idealities with specific weight mapping schemes and learning strategies (Chen et al., 2015; Wu et al., 2017; Gong et al., 2018; Cai et al., 2020b; Wang et al., 2020b; Pan et al., 2020; Zanotti et al., 2021), such as committee machines (Jokas et al., 2020) or the write-verify update loop procedure. This mitigating approach is promising, but does not seem to be sufficient to fill the accuracy gap between the state-of-art RSM-based NNs and their software counterparts [99.7% accuracy (Ciresan et al., 2012; Mazzia et al., 2021) for the same task]. A different strategy is therefore required to overcome this limitation, one possibility is to accept the imperfections of the hardware and take advantage of them. This can be done by imitating software regularization methods or implementing NNs that rely on stochastic mechanisms (Figure 1).
Over the last decade, the unpredictable behavior of RSMs has been demonstrated to be an efficient way to create true random number generators (Shen et al., 2021; Gaba et al., 2013; Yang et al., 2020b; Hu et al., 2016; Faria et al., 2018; Bao et al., 2020) that can be used for security purposes (Khan et al., 2021; Pang et al., 2020). For machine learning applications, the Gaussian nature of the stochastic distribution turns out to be an efficient way to implement probabilistic computing in hardware (Table 1). Indeed, for multi-level RSMs, the programming error around the targeted conductance state can be used as a regularization method if the device encodes the synaptic weights. Wang et al. (2019b) showed that a standard deviation of 10 μS of Gaussian writing noise helped to avoid overfitting for the MNIST classification task using a RSM-based convolutional neural network (CNN). In the case of binary memories, a consistent stochastic switch can be triggered by a programming writing voltage that is below the device threshold value. With such weak programming conditions, it is possible to set a RSM from a high resistive state (HRS) to a low resistive state (LRS) with a given probability of 50%. This
behavior has successfully been exploited to implement a neuron activation function (Wijesinghe et al., 2018), a stochastic learning rule (Yu et al., 2013; Payvand et al., 2019; Zahari et al., 2020), and controllable weight sampling (Yang et al., 2020b).

Some types of NNs are particularly suitable to take advantage of RSM non-idealities, and of these, Bayesian neural networks (BNNs), restricted Boltzmann machines (RBMs), and spiking neural networks (SNNs) have received special attention.

BNNs are difficult to use for real-world problems because of their prohibitive computation cost on traditional computers, which is mainly due to the expensive random sampling of parameters. Nevertheless, the uncertainty measurement provided by this NN is valuable for many applications, such as healthcare (McLachlan et al., 2020) and autonomous vehicles (McAllister et al., 2017). An array of RSMs can provide an elegant solution to this issue by exploiting the writing (Malhotra et al., 2020; Dalgaty et al., 2021a,b; Yang et al., 2020b) or the reading (Lin et al., 2019) variability to efficiently sample the network weights in parallel while computing the VMM in place at the same time.

RBMs have non-deterministic activation functions, and usually have a relatively small number of parameters, which fits well with the small and noisy RSM crossbars that are currently available. In the same way as BNNs, the probabilistic aspect of RBMs is not very desirable for CMOS chips, whereas RSMs offer new design perspectives (Kaiser et al., 2022). For example, Suri et al. (2015) suggested using the HRS and LRS variability to build a stochastic activation function an RBM and Mahmoodi et al. (2019) experimentally demonstrated the benefits of thermal noise to realize a stochastic dot product computation.

Finally, the spiking approach seems to be a promising candidate for RSM-based NNs, as they naturally adapt to variability (Maass, 2014; Neftci et al., 2016; Leugering and Pipa, 2018). SNNs share many similarities with biological NNs, which are known to rely heavily on stochastic mechanisms (Stein et al., 2005; Deco et al., 2009; Rolls and Deco, 2010; Yarom and Hounsgaard, 2011) such as Poisson process or short-term memory. In this context, RSMs are well suited to implement the synaptic weights (Yu et al., 2013; Naous et al., 2016; Payvand et al., 2019; Wang et al., 2020a) and the components of a neuron (Al-Shedivat et al., 2015; Naous et al., 2016; Wijesinghe et al., 2018; Dalgaty et al., 2019; Li et al., 2020), in particular, the membrane potential and the activation function.

**DISCUSSION**

In the early 2010’s, GPUs played an essential role in the rebirth of artificial intelligence as a research field by offering an efficient alternative to CPUs for VMM. Although RSM-based electronics have the potential to give rise to a similar hardware revolution, this transition is much more challenging since machine learning models face new constraints. Several works have shown through simulation and experimental results that harnessing the imperfections of RSMs is a viable option for tackling this problem and getting closer to the performance of software NNs.

However, this approach is subject to some limitations. While a reasonable quantity of stochasticity can improve the robustness of the model, larger amounts will be beneficial only to NNs that intrinsically rely on stochasticity, such as BNNs, RBMs, or SNNs. But, even in these cases, non-idealities must be kept under control to obtain a specific probability distribution shape or a consistent switching probability. The covariance between the resistance and the standard deviation is one example of the constraints that must be considered. To implement efficient probabilistic or approximate computing on RSMs, we may have to use unconventional approaches such as aggregating devices (Dalgaty et al., 2021a) or applying continuous writing operations (Yang et al., 2020b; Malhotra et al., 2020), which will alter the global energy efficiency and reduce the device lifetime.

Moreover, several RSM non-idealities have not yet been exploited for non-conventional computing schemes, such as device-to-device variability, sneak path current, state drift in time, or read and write disturbances. Further studies should explore novel circuit designs, encoding methods, and learning techniques benefiting from these characteristics for future hardware-based NNs.

The exploitation of non-idealities seems desirable, if not necessary, to accelerate the development of large-scale artificial NNs at state-of-the-art performance with competitive energy and area efficiency. The benefits are twofold in the case of stochastic (Dalgaty et al., 2021c) (BNNs and RBMs) or asynchronous event-based (Wang et al., 2020a; Agrawal et al., 2021) (SNNs) models, for which von Neumann CMOS computers are especially inefficient. This approach could be combined with the mitigation of harmful non-idealities, hardware-software co-design, and optimization of fabrication techniques to reach the full potential of RSM technology.

**AUTHOR CONTRIBUTIONS**

All authors conceived the review topic. VY wrote the first draft. All authors contributed to the article and approved the submitted version.

**FUNDING**

This work was financed by the EU: ERC-2017-COG project ICONOS (#GA 773228), the Natural Sciences and Engineering Research Council of Canada (NSERC) HIDATA project 506289-2017, CHIST-ERA UNICO project, the Fond de Recherche du Québec Nature et Technologies (FRQNT), and the Canada First Research Excellence Fund (CFREF). RM is supported by NSERC, the Canada Research Chair program, and the Perimeter Institute for Theoretical Physics. Research at Perimeter Institute is supported in part by the Government of Canada through the Department of Innovation, Science and Economic Development Canada and by the Province of Ontario through the Ministry of Economic Development, Job Creation and Trade.


Conflict of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher’s Note: All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.

Copyright © 2022 Yon, Amirsoleimani, Alibart, Melko, Drouin and Beillard. This is an open-access article distributed under the terms of the Creative Commons Attribution License (CC BY). The use, distribution or reproduction in other forums is permitted, provided the original author(s) and the copyright owner(s) are credited and that the original publication in this journal is cited, in accordance with accepted academic practice. No use, distribution or reproduction is permitted which does not comply with these terms.