ISSN 2303-4521
Original Research
Periodicals of Engineering and Natural Sciences
Vol. 10, No. 2, March 2022, pp.101-108

Mathematical simulation of memristive for classification in machine learning

Ammar A. Hasan[1] *, Nada A.Z. Abdullah[2], Amenah D. Abbood[2]
[1] Department of Computer Engineering - University of Baghdad
[2] Department of Computer Science - University of Baghdad

ABSTRACT

Over the last few years, neuromorphic computation has been a widely researched topic. One of the neuromorphic computation elements is the memristor. The memristor is a high density, analogue memory storage, and compliance with Ohm's law for minor potential changes. Memristive behaviour imitates synaptic behaviour. It is a nanotechnology that can reduce power consumption, improve synaptic modeling, and reduce data transmission processes. The purpose of this paper is to investigate a customized mathematical model for machine learning algorithms. This model uses a computing paradigm that differs from standard Von-Neumann architectures, and it has the potential to reduce power consumption and increasing performance while doing specialized jobs when compared to regular computers. Classification is one of the most interesting fields in machine learning to classify features patterns by using a specific algorithm. In this study, a classifier based memristive is used with an adaptive spike encoder for input data. We run this algorithm based on Anti-Hebbian and Hebbian learning rules. These investigations employed two of datasets, including breast cancer Wisconsin and Gaussian mixture model datasets. The results indicate that the performance of our algorithm that has been used based on memristive is reasonably close to the optimal solution.

Keywords: Memristive, machine learning, Anti-Hebbian and Hebbian learning, classification, artificial synapses.

Corresponding Author:

Ammar A. Hasan
Department of Computer Engineering
University of Baghdad
Baghdad, Iraq
Email: mr.ammaradel@coeng.uobaghdad.edu.iq

1. Introduction

Memristors is the fourth component of the expected principal behaviour of the passive device along with resistor, capacitor and inductor [1-2]. It is firstly introduced in 1971 by Chua [1], to reflect the relationship of resistance with memory. A varying resistance is used for storing data in different states [3-5]. Memristive devices are currently being developed to use, as a functional block of multilevel storage, in many common applications, such as neuromorphic computing networks[6–9] and functional logic devices[10–13]. In neuromorphic networks and system, the memristive material is widely used as a bio-material in many applications of the neuromorphic systems to provide both memory and processing functions[14-16]. The purpose for that it is being used as a resistive material with the memory function of electric charge. In other words, the amount of the electric charge at any time provides the value of its passing current and then it can be defined as, [14] and [17]:

$$M(q) = \frac{df(q)}{d(q)}, \; i = \frac{q}{t} \tag{1}$$

Where f is the magnetic flux, q is the electric charge, i is the current and t is a given time.

When the current flows through the device in one direction, the resistance of the material can rise up, or it might decrease if the current flows in the other direction. This helps to implement a physical neural network with one or more memristors-like synapse to connect neuron nodes together for forming physical multi-layer in a network. Each node within a given layer connects one or more inputs with one or more outputs via synaptic weights which modified by threshold values associated with their varying resistances between pre/post-synapses neuron signals as illustrated in Figure 1. The basic mechanism of synaptic plasticity is described by Hebbian scientific theory.

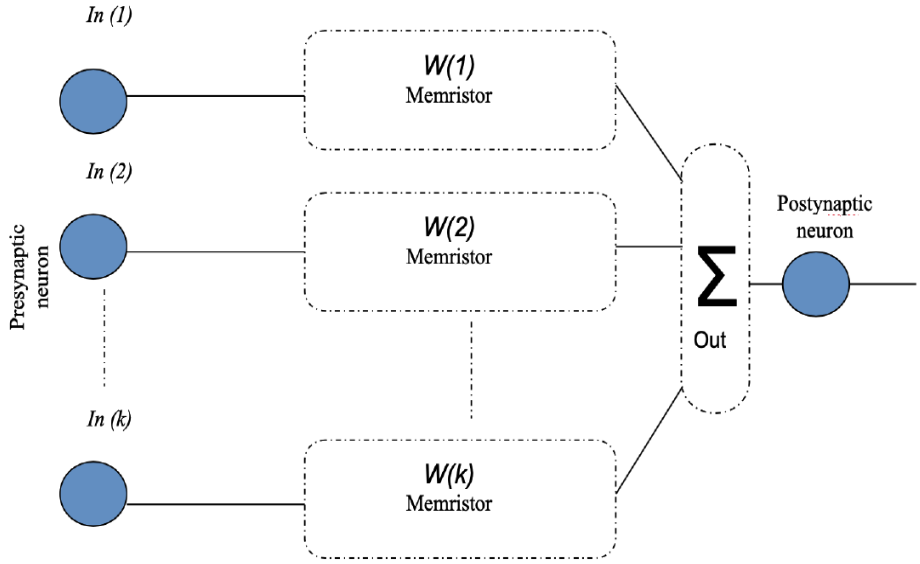

Figure 1. A pre-description of a perceptron network, the postsynaptic neuron is an output summation of the k synapses connected by k input

Anti-Hebbian and Hebbian may be used to describe the particular adaptation of learning rule by which synaptic plasticity can be self-adapted [18].

In order to use machine learning more efficiently, the input data is represented here as a spike (bundle wires) with each input spike corresponding to a wire. Depending on the input data representation, some of the input spikes to the classifier are active (1) and others are not active (0). Although spike encoding is not necessary for the classifier, it can become efficient in several points. Firstly, in the case of chip design, it is difficult to move analogue voltages. Secondly, if the number of active line synapses is low, then the computation will become efficient because only active line synapses are adapted. Thirdly, it is a digital code and it is easy to interpret what these digits mean.

The remainder of the paper is outlined as follows: the following section describes the essential background and related studies. The next section demonstrates the design methodology of using memristive circuits in AHaH classification. Following that, Spike encoding description as an input data encoder is described by a case study.

2. Background

Resistive switching devices such as phase-change memories and memristors are all been introduced as an alternative technology to traditional CMOS and TTL devices that have typical hardware implementations based on von Neumann architecture [10], [19] and [20]. The resistive devices have the ability to produce different levels of non-volatile resistance for determining their memory states. Therefore, such resistive devices, like memristors, have potential concepts to use in neuromorphic (brain-like) computing technologies for producing the functions of synapses and neurons [7], [21]–[26], i.e. building non-volatile connections between neurons. Despite the fact that the memristive devices have been studies over many years with some limitations, a real establishing was discovered in 2008 [8]. Next, crossbar arrays of memristors were introduced and then developed as building blocks of memory circuits [27]–[34]. Then, memristors were trained to use as spike-like circuits [14-27]. After this, Learning and memory were thought to begin with artificial synaptic plasticity which provided excellent dependability, scalability, and low energy consumption, similar to natural synapses.

Nugent and Timothy [18] published a basic memristive device model based on Anti Hebbian and Hebbian (AHaH) learning rules in 2014, in which synaptic weights are represented by two memristors linked serially with the same polarities. To mimic Anti Hebbian and Hebbian (AHaH) circuit nodes, these memristors are required. Although there are numerous memristive device models that model specific devices, this sort of memristor simulated a huge number of them [18].

3. Method

A memristive can be construct synapse which used in classification to adapt the values of the weights and the adaption property leads to reduce the communication between the memory and processing units, thus the power consumption is reduced. Nugent and Timothy [18] derive equations to update weight and these equations are derived from thermodynamic law to form a function model as illustrated in the following equations.

$$\Delta w = -\beta y + \alpha sgn(y) + \eta - (1-\delta)w \quad (2)$$

$$\Delta b = -\beta y + \eta - (1-\delta)b \quad (3)$$

where w represents a synapse. α and β represent read and write periods respectively.
η is the noise variable (normal Gaussian) and δ is a decay variable.

Equation (2) is used in the case of unsupervised feedback while in the case of supervised the following equation is used:

$$\Delta w = -\beta y + \alpha sgn(s) + \eta - (1-\delta)w \quad (4)$$

But in the first, initialize weight for each of the active input and the bias.

$$w = 0.1 \text{ x } \text{randomGaussian()} \quad (5)$$

Then find the output (y) by the summation of the weight for all the active input and the bias.

$$y = \sum_{i=0}^{M-1} w_i \quad (6)$$

After that, the weights are modified by applying Equations (2) and (4) and find new weights.

4. An Adaptive Spike encoding (Input data encoding)

In order to use Nugent and Timothy [18] classifier in machine learning, the input data to this classifier is represented as a spike (bundle wires). Each input spike represents a wire. Some of the input spikes to the classifier are active (1) and the remaining is not active (0), depending on the input data representation. There is number of steps necessary to convert input data into spikes, see Figure 2.

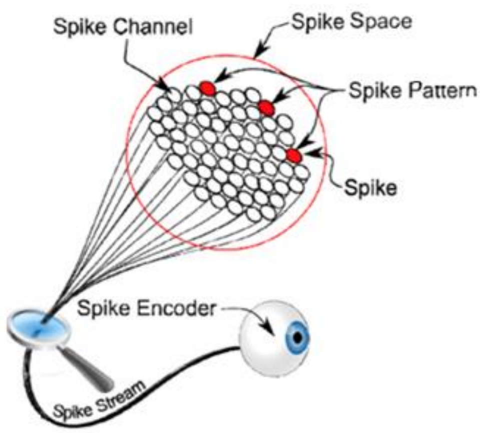

Figure 2. Spike representation (digital representation) for the input data [35].

Spike pattern represents indexes for active spike input to the classifier. Spike space is the maximum index which can be used to encode the input data. Spike encoder converts the input data into spike encode.

Spike encoder: Converts input data into spike code by using a binary tree with anti-Hebbian learning. It is adaptive and the spike encoding will change over time. One encoder is created for each input feature. Firstly, recursion is used to divide the tree into the high and low bin, and a unique number is assigned for each node in the tree, as illustrated in Figure 3, where the number of nodes depends on the number of depths.

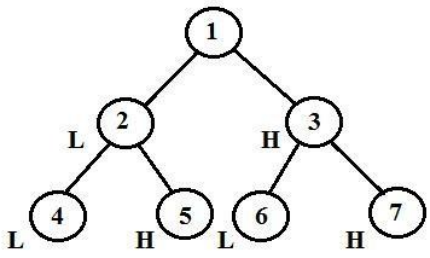

Figure 3. Split each node into high and low bin

Secondly, the effect of the bias is computed on the input by the following equation:

$$y = x + b \tag{7}$$

Where, x is the input feature data, and b is the bias. The adaptive average subtraction is Anti-Hebbian learning, and that is why this particular encoder is used. The bias adapts its value depending on the input feature value, which has an effect on the next input pattern for the same feature.

$$b = b - (lr \times y) \tag{8}$$

Where, *lr* is the learning rate.

Finally, the routing path for the input feature is determined depending on y value, if it is negative, then it goes left and if it is positive then goes right and adds 1: the process is repeated recursively until a null is reached. As a result, the path consists of the nodes which pass from the root to the leave of the tree, and this path represents a spike encode.

Spike converter: It gives a unique index for each node in the path for each feature until the maximum index is reached which represents the maximum number of the input spike for the classifier.

$$s = f \times n + bias \tag{9}$$

Where *s* is the maximum input spikes, *f* is the number of features and *n* is the number of nodes in the tree.
Spike pattern: It is the indices for the active spike for the classifier.

$$p = f \times node_p + bias \tag{10}$$

Where *p* is the spike pattern space, *f* is the number of features and node *p* is the number of nodes within the routing path. For example, if the number of features is 2 and the path for each feature is represented by 3 nodes, then the spike pattern space is 7 because the value for the bias is 1 for each pattern.

Because an adaptive method is used, the same input will produce slightly different spike outputs over time as the encoder adapts to the signal. If its adaptation is affecting the classification, then the learning rate is decreased and the number of training epochs is increased.

5. Performance measures

Five metrics were used to assess the performance of our method. Accuracy, sensitivity, specificity, precision, and F-measure are examples of these metrics. They are among the most common measures used to evaluate machine learning performance. They are calculated using the four numbers true positive (T_P), true negative (T_N), false positive (F_P), and false negative (F_N). The T_P and T_N stand for the number of situations that are properly categorized as positive and negative classes respectively. The F_P is the number of situations in the negative class that were erroneously categorized as positive class, as well as the F_N is the number of situations that are erroneously categorized as negative and those that are in a positive class.

The proportion of correctly categorized situations is referred to as classification accuracy, and it is computed as follows:

$$Accuracy = \frac{T_P + T_N}{T_P + T_N + F_P + F_N} \tag{11}$$

The percentage of all positive samples that are correctly identified is referred to as sensitivity, and it is computed as follows:

$$Sensitivity = \frac{T_P}{T_P + F_N} \tag{12}$$

The percentage of negative samples that are accurately recognized as negative is measured by specificity (not having the breast cancer). The following formula is used to compute it:

$$Specificity = \frac{T_N}{T_N + F_P} \tag{13}$$

Precision is sometimes called the positive predictive value (PPV). It's the proportion of samples properly identified as positive among those that have been classified, and it's computed like this:

$$Precision = \frac{T_P}{T_P + F_P} \tag{14}$$

F-measure is sometimes referred to as *F-score*. It has a range of 0 to 1 and expresses the balance between accuracy and sensitivity. There are no true positives with a value of 0, and there are no false negatives or positives with a value of 1. The following formula may be used to determine the F-measure:

$$F - score = \frac{2*Precision * Sensitivity}{Precision + Sensitivity} \tag{15}$$

6. Experiment results

To illustrate how classifier rule derives from the memristive simulation is work in the machine learning applications, a breast cancer Wisconsin and Gaussian mixture model data set are used here. The classifier consists of two parts: training part and test part. In this case, a number of synapses are connected to form classifier and the number of synapses is determined by the number of input spikes and bias from Equation (10). The order of spikes in a spike pattern should not matter. It is a "set" and is used to co-active respective synapses. The adaptive spike encoding will change over time to track the statistics of the data. It is a decision tree that is performing an (exponential running) average subtraction at each node. Over time, it will adapt to the signal. It is likely that the same input will produce slightly different spike outputs over time as the encoder adapts to the signal. If its adaptation is affecting the classification, then the learning rate would lower and increase the number of training epochs.

Breast Cancer Wisconsin: breast cancer Wisconsin is used to classify if the cell is benign or malignant [18]. In this data set, there are 9 features for this application as illustrated in the following list:

1. Clump Thickness.
2. Uniformity of Cell Size.
3. Uniformity of Cell Shape.
4. Marginal Adhesion.
5. Single Epithelial Cell Size.
6. Bare Nuclei.
7. Bland Chromatin.
8. Normal Nucleoli.
9. Mitoses.

In this case, the maximum number of input spikes is 64 (Equation 10) where f=9 and n=7 for the tree with depth=3 and 1 bias for each feature). The range values for all the above features are between 1 and 10. Here one of the input training patterns is 327 (10, 3, 3, 1, 2, 10, 7, 6, and 1) for the above features respectively, is converted into the following code by spike encoder: 34, 0, 35, 1, 32, 33, 3, 38, 4, 39, 6, 8, 42, 10, 41, 11, 46, 13, 14, 15, 45, 18, 21, 20, 23, 25, 27and 26. These numbers of the input lines are active for the classifier. In the test part, also the input is converted into spike and 183 patterns are tested. Here unsupervised signal (0) is used because there is no supervised signal, then the output y is computed (Equation 6). In the training part, 500 patterns are trained and each pattern has 9 features of the cell. The class of the cell is either 2 (benign) or 4 (malignant. Then, the output y is computed (Equation 6).

Finally, a threshold is used to determine the output class, either class 2 or class 4. We get 95.14%, 95.5%, 86%, 98%, 0.96 for the accuracy, sensitivity, specificity, precision and F-measure respectively. When the threshold is increased, the precision is increased while sensitivity will drop.

Gaussian mixture model data set: Gmm is used here because there are two advantages. Firstly, the input data can be represented by two-dimensional pictures. Secondly, the bayes error rate and the decision boundary are known. In this study, 250 patterns are used for the training process and then used 50 *60 patterns for the test process. The range of x-Axis between -1.5, 1.5 and the range of y between -0.2, 1.6 and test data is 1000. Figure 4 shows that the result of the classification output for 1000 test patterns and also there are 250 patterns for training part. The red and blue colours represent class 0 and class 1 respectively. The true positive is 0.946 and the threshold is 0.2.

The most important properties of the function simulation, it is less computation than the circuit simulation and both of them give converges in the simulation results.

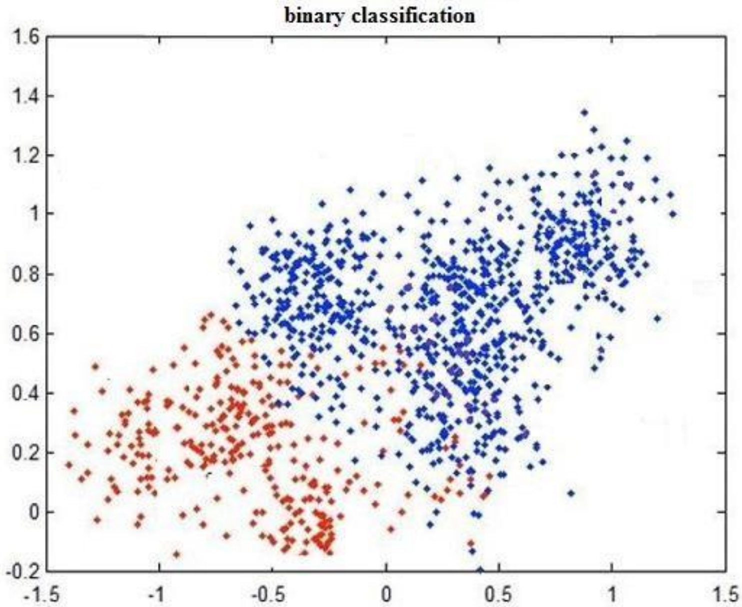

Figure 4. The output of classifier for 50 points in the x-Axis and 60 points in the y-Axis

7. Conclusions

In this paper, we presented a classifier based on minimum memristive devices for general purpose computing and machine learning through the implementation of neuromorphic learning rules based on memristive nano-devices. The memristor works with changing resistance depends on the current ow through it. The synapse weights have been built from two memristors, and an array of these synapses is used to build the classifier, which could be used for machine learning applications with minimal computational overheads and cost, and give efficient results in classification. Also, using spike encoder to convert the input data into spike encoding leads to more efficiency in the applications of the machine learning. There are different applications of machine learning; for example, supervised and unsupervised classification and clustering based on a memristive nano-device, which could be used in the storage and the computations.

8. References

[1] L. Chua, “Memristor-the missing circuit element,” *IEEE Trans. circuit theory*, vol. 18, no. 5, pp. 507–519, 1971.

[2] R. B. Leighton and M. Sands, *The Feynman lectures on physics*. Addison-Wesley Boston, MA, USA, 1965.

[3] J. P. Strachan, A. C. Torrezan, G. Medeiros-Ribeiro, and R. S. Williams, “Measuring the switching dynamics and energy efficiency of tantalum oxide memristors,” *Nanotechnology*, vol. 22, no. 50, p. 505402, 2011.

[4] F. Miao *et al.*, “Anatomy of a nanoscale conduction channel reveals the mechanism of a high-performance memristor,” *Adv. Mater.*, vol. 23, no. 47, pp. 5633–5640, 2011.

[5] A. A. H. Al-Shahrablee, “Reconfigurable three-terminal logic devices using phase-change materials,” 2018.

[6] K. He, X. Zhang, S. Ren, and J. Sun, “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification,” in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

[7] S. H. Jo, T. Chang, I. Ebong, B. B. Bhadviya, P. Mazumder, and W. Lu, “Nanoscale memristor device as synapse in neuromorphic systems,” *Nano Lett.*, vol. 10, no. 4, pp. 1297–1301, 2010.

[8] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, “The missing memristor found,” *Nature*, vol. 453, no. 7191, pp. 80–83, 2008.

[9] J. J. Yang, M. D. Pickett, X. Li, D. A. A. Ohlberg, D. R. Stewart, and R. S. Williams, “Memristive switching mechanism for metal/oxide/metal nanodevices,” *Nat. Nanotechnol.*, vol. 3, no. 7, pp. 429–433, 2008.

[10] J. Borghetti *et al.*, “A hybrid nanomemristor/transistor logic circuit capable of self-programming,” *Proc. Natl. Acad. Sci.*, vol. 106, no. 6, pp. 1699–1703, 2009.

[11] E. Lehtonen and M. Laiho, “Stateful implication logic with memristors,” in *2009 IEEE/ACM International Symposium on Nanoscale Architectures*, 2009, pp. 33–36.

[12] J. Borghetti, G. S. Snider, P. J. Kuekes, J. J. Yang, D. R. Stewart, and R. S. Williams, “‘Memristive’switches enable ‘stateful’logic operations via material implication,” *Nature*, vol. 464, no. 7290, pp. 873–876, 2010.

[13] S. Kvatinsky, A. Kolodny, U. C. Weiser, and E. G. Friedman, “Memristor-based IMPLY logic design procedure,” in *2011 IEEE 29th International Conference on Computer Design (ICCD)*, 2011, pp. 142–147.

[14] M. D. Pickett, G. Medeiros-Ribeiro, and R. S. Williams, “A scalable neuristor built with Mott memristors,” *Nat. Mater.*, vol. 12, no. 2, pp. 114–117, 2013.

[15] A. Mehonic and A. J. Kenyon, “Emulating the electrical activity of the neuron using a silicon oxide RRAM cell,” *Front. Neurosci.*, vol. 10, p. 57, 2016.

[16] T. Tuma, A. Pantazi, M. Le Gallo, A. Sebastian, and E. Eleftheriou, “Stochastic phase-change neurons,” *Nat. Nanotechnol.*, vol. 11, no. 8, pp. 693–699, 2016.

[17] B. Suresh *et al.*, “Simulation of integrate-and-fire neuron circuits using HfO 2-based ferroelectric field effect transistors,” in *2019 26th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, 2019, pp. 229–232.

[18] M. A. Nugent and T. W. Molter, "AHaH computing--from metastable switches to attractors to machine learning," *PLoS One*, vol. 9, no. 2, p. e85175, 2014.
[19] M. Klimo and O. Such, "Memristors can implement fuzzy logic," *arXiv Prepr. arXiv1110.2074*, 2011.
[20] D. Acharyya, A. Hazra, and P. Bhattacharyya, "A journey towards reliability improvement of TiO_2 based resistive random access memory: a review," *Microelectron. Reliab.*, vol. 54, no. 3, pp. 541–560, 2014.
[21] B. Linares-Barranco and T. Serrano-Gotarredona, "Memristance can explain spike-time-dependent-plasticity in neural synapses," *Nat. Preced.*, p. 1, 2009.
[22] L. Chua, V. Sbitnev, and H. Kim, "Hodgkin--Huxley axon is made of memristors," *Int. J. Bifurc. Chaos*, vol. 22, no. 03, p. 1230011, 2012.
[23] L. Chua, V. Sbitnev, and H. Kim, "Neurons are poised near the edge of chaos," *Int. J. Bifurc. Chaos*, vol. 22, no. 04, p. 1250098, 2012.
[24] E. Gale, B. de Lacy Costello, and A. Adamatzky, "Is spiking logic the route to memristor-based computers?," in *2013 IEEE 20th international conference on electronics, circuits, and systems (ICECS)*, 2013, pp. 297–300.
[25] H. A. Alatabi and A. R. Abbas, "Sentiment analysis in social media using machine learning techniques," *Iraqi J. Sci.*, pp. 193–201, 2020.
[26] K. R. AL-Rawi and S. K. AL-Rawi, "Smart doctor: Performance of supervised ART-I artificial neural network for breast cancer diagnoses," *Iraqi J. Sci.*, pp. 2385–2394, 2020.
[27] W. Lu, K.-H. Kim, T. Chang, and S. Gaba, "Two-terminal resistive switches (memristors) for memory and logic applications," in *16th Asia and South Pacific Design Automation Conference (ASP-DAC 2011)*, 2011, pp. 217–223.
[28] C. Yakopcic, T. M. Taha, and R. Hasan, "Hybrid crossbar architecture for a memristor based memory," in *NAECON 2014-IEEE National Aerospace and Electronics Conference*, 2014, pp. 237–242.
[29] R. Hasan, T. M. Taha, and C. Yakopcic, "On-chip training of memristor crossbar based multi-layer neural networks," *Microelectronics J.*, vol. 66, pp. 31–40, 2017.
[30] S. Pi *et al.*, "Memristor crossbars with 4.5 terabits-per-inch-square density and two nanometer dimension," *arXiv Prepr. arXiv1804.09848*, 2018.
[31] Y. Kim *et al.*, "Memristor crossbar array for binarized neural networks," *AIP Adv.*, vol. 9, no. 4, p. 45131, 2019.
[32] Y. Li and K.-W. Ang, "Hardware Implementation of Neuromorphic Computing Using Large-Scale Memristor Crossbar Arrays," *Adv. Intell. Syst.*, vol. 3, no. 1, p. 2000137, 2021.
[33] A. D. Abbood, A. A. Hasan, and B. A. Attea, "The Effects of Conductance on Metastable Switches in Memristive Devices Based on Anti-Hebbian and Hebbian (AHaH) Learning Rules," *Iraqi J. Sci.*, pp. 3724–3732, 2021.
[34] E. Gale, B. de L. Costello, and A. Adamatzky, "Observation, characterization and modeling of memristor current spikes," *arXiv Prepr. arXiv1302.0771*, 2013.
[35] M. A. Nugent and T. W. Molter, "Thermodynamic-RAM technology stack," *Int. J. Parallel, Emergent Distrib. Syst.*, vol. 33, no. 4, pp. 430–444, 2018.