Independent Component Analysis using RRAMs

Mohammed E. Fouda, *Student Member, IEEE,*, Emre Neftci, Ahmed Eltawil, *Senior Member, IEEE,* and Fadi Kurdahi, *Fellow, IEEE*

***Abstract*—Resistive Random-Access Memories (RRAM)s are considered a promising candidate for neuromorphic circuits and systems. In the letter, we investigate using TiO_2 RRAMs to solve blind source separation problem through Independent Component Analysis (ICA) for the first time. ICA has numerous uses including feature extraction. We deploy a local, unsupervised learning algorithm (error-gated Hebbian rule) to extract the independent components. The online evaluation of the weights during the training is studied taking into consideration the asymmetric nonlinear weight update behavior. The effects of the device variability are considered in the results. Finally, an example of de-mixing two Laplacian signals is given to demonstrate the efficacy of the approach.**

***Index Terms*—RRAMs, Memristor, ICA, Unsupervised Learning, Brain-inspired Learning.**

I. Introduction

The Internet of Things (IoT) market is growing exponentially and it is expected to have around 50 billion connected devices generating around 500 zettabytes of data per year by 2019 [1]. As a result, almost all IoT applications need a system to analyze patterns in this data, detect certain types of events and take decisions. Component analysis techniques [2] are promising candidates to perform these tasks especially with using local and efficient learning rules, that enable online learning, rather than the conventional techniques that require massive matrix operations. Additionally, by using hardware-based matrix-vector multiplication accelerators, a significant improvement in both performance and power can be achieved enabling handling the massive data generated from IoT [3].

Independent component analysis (ICA) is a very powerful tool to solve the cocktail party problem (blind source separation), feature extraction (sparse coding) and can be utilized in many applications such as de-noising images, Electroencephalograms (EEG) signals, and telecommunications [4]. An illustrative example of the cocktail party problem is given in the supplementary materials. ICA consists of finding mutually independent and non-Gaussian hidden factors (components), $\mathbf{s}$, that form a set of signals or measurements, $\mathbf{x}$. This problem can be mathematically described for linearly mixed components as follows:

$$\mathbf{x} = \mathbf{A}\mathbf{s} \tag{1}$$

where A is the mixing matrix. Both $\mathbf{A}$ and $\mathbf{s}$ are unknowns. In order to find the independent components (sources), the problem can be formulated as $\mathbf{u} = \mathbf{w}\mathbf{x}$ where $\mathbf{x}$ is the mixed signals (inputs of ICA algorithm), $\mathbf{w}$ is the weight matrix (demixing matrix), $\mathbf{u}$ is the outputs of ICA algorithm (independent components). ICA's strength lies in utilizing the mutual statistical independence of components to separate the sources.

Resistive crossbar structures are considered a key enabler to hardware based neuromorphic acceleration due to their natural ability to do matrix-vector multiplication which is the basic operation for neural network acceleration (e.g. Multiply and Add). Crossbar arrays can perform matrix-vector multiplication in a single step as compared to m^2 steps for digital realizations, where m is the length of the vector. Furthermore, each RRAM occupies a very small area $(4F^2)$, where F is the feature size, and operates as a nonvolatile continuous weight. In the digital realization, each weight is stored in at least a 32 bits register to have continuous weight which requires 38 transistors per bit. In addition, $m \times (m-1)$ multiply and accumulate (MAC) blocks are needed, increasing the power budget of the overall system. A comparative example of the network that was used for the De-mixing problem is given in the supplementary materials. RRAMs offer features such as high density, low power consumption, high endurance, high retention, high speed switching and 3D stack-ability in addition to ease of programming. Recently, RRAM-based neural network architectures have been deployed in brain-inspired computing applications [5] such as dimensionality reduction through PCA [6], sparse coding [7], reservoir computing [8] and image processing [9].

Recently, Isomura and Toyoizumi have proposed an interesting biological plausible learning rule called *Error-Gated Hebbian Rule (EGHR)* [10] that enables local and efficient learning to find the independent components. But, the expensive part in this algorithm is the matrix-vector multiplication which can calculated using RRAMs-based crossbar array offering a low power and efficient solution. In this letter, a RRAM-based hardware realization for ICA is investigated using EGHR for online evaluation of the weights. We demonstrate that this learning rule is capable of local and efficient learning, even when taking into consideration the RRAM non-idealities, the asymmetric nonlinear conductance, and device variability.

This letter is organized as follows: Section II discusses the conductance update model which includes the device non-idealities and variations. Then, the proposed ICA learning algorithm is introduced in Section III. Section IV presents the results of the proposed algorithm. Finally, the conclusion and future work are given.

II. Conductance Update Model

Several RRAM devices demonstrating promising synaptic behaviors are characterized by nonlinear and asymmetric update dynamics, which are considered as major limitations to their large-scale deployment in neural networks [11]. Applying the standard ICA without taking into the considera-

M E Fouda, A M. Eltawil, and Fadi Kurdahi are with Dept. of Electrical Engineering and Computer Science, University of California–Irvine, Irvine, CA 92697-2625 USA. Emre Neftci is with Department of Cognitive Sciences, UC Irvine, Irvine, CA 92697-2625. E-mail: foudam@uci.edu

tion the device non-idealities (see the example given in the supplementary material), results in loss of convergance to the independent components. Thus, a closed form model for the device nonlinearity should be derived and added to the ICA algorithm to guarantee the convergence to ICs.

The asymmetric nonlinear behavior of the RRAMs can be quantitatively modeled as the difference between the potentiation and depression conductance update and linear (ideal) conductance as shown in Fig. 2a. The asymmetric nonlinearity of the RRAM's conductance update can be fitted to the following model

$$G(t) = \begin{cases} G_{max} - \beta_P e^{-\alpha_1 \phi(t)} & v(t) > 0 \\ G_{min} + \beta_D e^{-\alpha_1 \phi(t)} & v(t) < 0 \end{cases} \quad (2)$$

where G_{max} and G_{min} are the maximum and minimum conductances respectively, $\alpha_1, \alpha_2, \beta_P$ and β_D are fitting coefficients. β_P and β_D are related to the difference between G_{max} and G_{min} and $\phi(t)$ is the time integral of the applied voltage.

Updating the RRAM conductance is commonly performed through positive/negative programming pulses for potentiation/ depression with pulse width T and constant programming voltage V_p. As a result, the discrete values of the flux are $\phi(t = nT) = V_p nT$ where n is the number of applied pulses. This technique provides precise and accurate weight updates. For $t = n\Delta T$, and substituting back into eq (2), the potentiation and depression conductances are given as

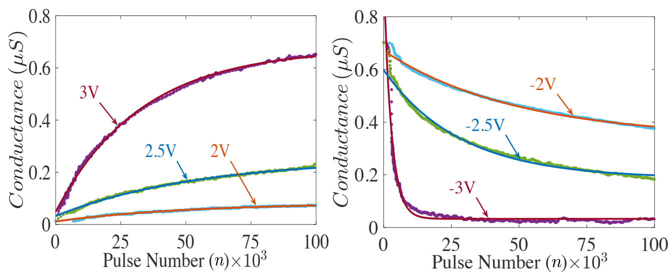
$$G_{LTP} = G_{max} - \beta_P e^{-\alpha_P n}, \text{ and} \quad (3)$$

$$G_{LTD} = G_{min} + \beta_D e^{-\alpha_D n}, \quad (4)$$

respectively, where n is the pulse number, $\alpha_P = |V_p|\alpha_1 T$ and $\alpha_D = |V_p|\alpha_2 T$.

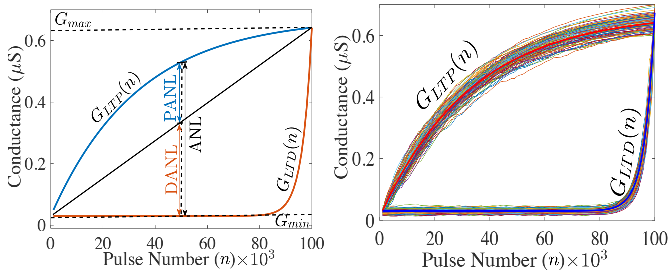
One way to quantify the device potentiation and depression asymmetry and linearity is the asymmetric nonlinearity factors to compare between the devices. The effect of these factors are reflected in the coefficients $\alpha_P, \alpha_D, \beta_P$ and β_D which are used for the training. The potentiation asymmetric nonlinearity (PANL) factor and depression asymmetric nonlinearity (DANL) are defined as $PANL = G_{LTP}(N/2)/DR - 0.5$ and $DANL = 0.5 - G_{LTD}(N/2)/DR$, respectively, where $DR = G_{max} - G_{min}$ is the RRAM's dynamic range and N is is the total number of pulses to fully potentiate the device. $PANL$ and $DANL$ are between $[0, 0.5]$. The sum of both potentiation and depression asymmetric nonlinearities represents the total asymmetric nonlinearity (ANL).

The synaptic device that will be used in this work is a non-filamentary TiO_x based RRAM with a precision measured to 6 bits by Park et al. in [12]. The $Mo/TiO_x/TiN$ device was fabricated based on a redox reaction at Mo/Tio_x interface which forms conducting MoO_x. This type of interface based switching devices exhibits good switching variability across the entire wafer and guarantee reproducibility [12]. The proposed model was fitted and parameters were extracted for the three programming cases $\{\pm 2V, \pm 2.5V, \text{ and } \pm 3V\}$. For this work, only $V_p = \pm 3V$ cases will be considered since it has the widest switching range. The extracted potentiation parameters are $G_{max} = 674nS, \beta_P = 626.8nS$ and $\alpha_P = 30.58mV^{-1}s^{-1}$ with 9.07 RMSE. On the other hand, the



(a) (b)

Fig. 1: RRAM's conductance update (a) long term potentiation (b) long term depression.



(a) (b)

Fig. 2: Non-idealities of the RRAM:(a) asymmeteric nonlinear weight update (b) device Variations

extracted depression parameters are $G_{min} = 32.95nS, \beta_D = 921.9nS$ and $\alpha_P = 353.4mV^{-1}s^{-1}$ with 23.7 RMSE. Figure 1 shows the curve fitted model on the top of the reported conductance for both potentiation and depression scenarios. This device has $PANL = 0.32$ and $DANL = 0.45$ with $ANL = 0.77$. Figure 2b shows the conductance variations of multiple devices during the potentiation and depression cycles with $\pm 3V$ programming pulses. The model parameters are sampled from Gaussian sources with 25% tolerance (Variance/mean) for α, and 1% and 5% tolerances for the maximum and minimum conductances, respectively. The effect of the variation in the parameter β is considered inside the variations of α. β should be modeled as lognormal variable to have a monotonic increasing or decreasing conductance update. Thus, the second term of the conductance update has log Gaussian variable, which is e^z, multiplied by $e^{\alpha n}$ where z and α are Gaussian variables . The sum of two Gaussian random variables is a Gaussian random variable. Thus, the effect of variation of β and α can be included in either one of them.

In neural networks, both positive (excitatory) and negative (inhibitory) connections are required. However, the RRAM conductance is positive by definition which creates only the excitatory connections. In order to create the inhibitory connections, there are two weight realization techniques; 1) by using two RRAMs per weight [13] or 2) by using one RRAM as weight in addition to one reference RRAM having a conductance set to $G_r = (G_{max} + G_{min})/2$ [14]. The first realization technique has double the dynamic range $w \in [-DR, DR]$ making it more immune to variability at a cost of double area, double power consumption during reading and additional programming operations. On the other hand, the second technique has one RRAM device, meaning that $w \in [-DR/2, DR/2]$ making it more prone to variability but

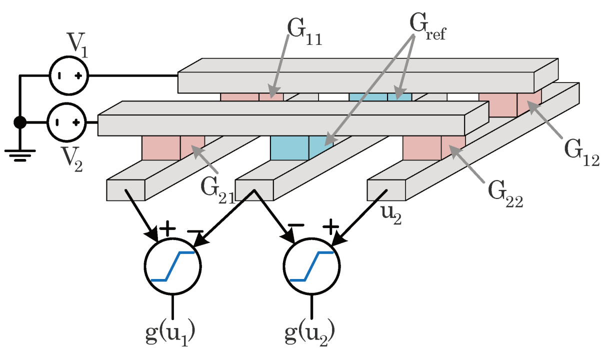


Fig. 3: Illustration of used RRAM crossbar array to realize the ICA preceptron network.

the overall area is minimal especially for large networks, less power consumption, and easier to program (programing only one RRAM per weight). In this work, we consider the second weight realization technique, and show that the algorithm converges despite the reduced dynamic range compared to the first realization.

III. Proposed RRAM's learning method

Due to the importance of blind source separation problem, many algorithms and learning rules have been proposed to find the independent components such as minimizing the mutual information or maximum likelihood estimation [4] and Bell-Sejnowski or Amari [10], etc. Recently, Isomura and Toyoizumi have proposed an interesting biological plausible learning rule called *Error-Gated Hebbian Rule (EGHR)* inspired from the standard Hebbian rule [10]. In their work, the authors proved mathematically and numerically the efficiency of EGHR to achieve ICA. The EGHR learning rule can be written as

$$\dot{\mathbf{w}} = \eta\langle(E_o - E(\mathbf{u}))\, g(\mathbf{u})\mathbf{x}^T\rangle \tag{5}$$

where η is the learning rate, $\langle\cdot\rangle$ is the expectation over the ensemble (training samples), $g(u_i)x_j$ is the Hebbian learning rule, $g(u_i)$, x_j are the postsynaptic and presynaptic terms of the neuron, respectively, $(E_o - E(\mathbf{u}))$ is the global error signal which consists of E_o which is a constant, and $E(\mathbf{u})$ which is the surprise or reward that guides the learning. The cost function of EGHR is defined as $\mathcal{L} = c\frac{1}{2}\langle(E_0 - E(\mathbf{u}))^2\rangle$. It was proven mathematically and numerically that this learning rule is robust, stable and its equilibrium point is proportional to the inverse of the mixture matrix, *i.e.* the solution of ICA. However, there are some conditions that should be satisfied; 1) $g(u_i)$ is a monotonically increasing odd function, and 2) $E(\mathbf{u})$ is a convex function. In order to satisfy these conditions, $g(u_i)$ is chosen to be either a hypertangent function or hard hypertangent function and $E(\mathbf{u})$ is chosen as modulus function.

ICA assumes that the sources are linearly mixed using a mixture matrix $\mathbf{A}$. The final weight matrix, $\mathbf{w}$, should converge to $c\mathbf{A}^{-1}$ which is still a valid solution since c is a scaling factor. If the sources have known distributions, the optimal E_0 can be calculated as discussed in [10]. On the other hand, if there is no knowledge about source distributions, Isomura and Toyoizumi proved that for any $E_0 > 0$, there is a positive c that gives the equilibrium point of EGHR regardless of the input source nature (unknown distributions) since the relation between E_0 and c is a monotonically increasing function [10]. Thus, E_0 can be used as a tuning knob to the learning rule to achieve the solution.

As previously discussed, the weights are enclosed between $[-\frac{DR}{2}, \frac{DR}{2}]$, thus E_0 must be chosen to keep the weights in this range. With unknown sources, one can iterate over E_0 until a solution is obtained. We start with an initial point (i.e. $E_o = 1$.) then keep decreasing it until the expectation of $E(\mathbf{u})$ becomes constant.

To have the resistive devices behave as EGHR, the change in each weight must be proportional to the change in the RRAM's conductance, $\Delta\mathbf{G} \propto \Delta\mathbf{w}$. To achieve this, the asymmetric nonlinear behavior of potentiation and depression should be included in developing the learning rule. We first calculate the change in the weights for both potentiation and depression cases taking into effect the asymmetric nonlinearity of the RRAM model. In general, the change in the LTP's conductance due to applying Δn is

$$\begin{aligned}\Delta G_{LTP} &= G(n+\Delta n) - G(n)\\ &= (G_{max} - G(n))\big(1 - e^{-\alpha_P \Delta n}\big)\end{aligned} \tag{6}$$

where $G(n)$ is the previous conductance. Similarly, the change in the LTD conductance due to applying Δn is $\Delta G_{LTD} = (G(n) - G_{min})(e^{-\alpha_D \Delta n} - 1)$. Clearly, the relation between the rate of change in conductance and Δn is an injective function. Thus, the number of pulses to cause ΔG_{LTP} and ΔG_{LTD} are

$$\Delta n_{LTP} = -\frac{1}{\alpha_P}\ln\left(1 - \frac{\Delta G_{LTP}}{G_{max} - G(n)}\right), \text{ and} \tag{7}$$

$$\Delta n_{LTD} = -\frac{1}{\alpha_D}\ln\left(\frac{\Delta G_{LTD}}{G(n) - G_{min}} + 1\right), \tag{8}$$

respectively. After learning, $\Delta\mathbf{G}$ goes to $\mathbf{0}$. As a result, $\Delta\mathbf{n}$ goes to zero as well. Equations (7) and (8) are nonlinear functions which are hardware expensive to implement. Thus, both can be linearized as $\ln(1-x) \approx -x(1+0.5x) \approx -x$ and $\ln(1+x) \approx x(1-0.5x) \approx x$ for $x \ll 1$. As previously discussed, we can replace $\Delta G_{i,j}$ by $\eta'\Delta w_{i,j}$, where η' is the scaled learning rate, and $\Delta w_{i,j}$ is given by EGHR. Thus, the final equations for potentiation and depression pulses can be written as follows:

$$\Delta n_{i,j}|_{LTP} \approx \frac{1}{\alpha_P}\left(\frac{\eta'(E_o - E(\mathbf{u}))g(u_j)x_i}{G_{max} - G_{i,j}(n)}\right), \text{ and} \tag{9}$$

$$\Delta n_{i,j}|_{LTD} = -\frac{1}{\alpha_D}\left(\frac{\eta'(E_o - E(\mathbf{u}))g(u_j)x_i}{G_{i,j}(n) - G_{min}}\right), \tag{10}$$

respectively.

By programming the RRAMs using the previous equations, the circuit behaves as required and compensates for the asymmetric nonlinearity of the devices. The proposed training algorithm is shown in Algorithm 1. We chose to initialize the weights with identity because we assume initially that the inputs are the independent components themselves and unmixed which means u=x, thus W=I. The inference part is represented in lines 5-7 however lines 8-9 represent the learning part. Line 5 in the algorithm is implemented using crossbar array shown in Fig.3. Lines 6-7 can be implemented using either CMOS circuitry or off the shelf components such as operational amplifiers similar to those discussed in [13]. The training part, can be implemented either using analog or digital circuits.

Algorithm 1 Proposed Training Algorithm.

```
1:  Set $E_o = 1$
2:  while $\langle E(\mathbf{u}) \rangle$ become constant do
3:      Initialize RRAMs' weights to Identity
4:      for $x \in$ the training set do
5:          $\mathbf{u} = (\mathbf{G} - \mathbf{G}_r)\mathbf{x}$
6:          $\mathbf{g} = hard\,tanh(b\mathbf{u})$
7:          $E = \sum_i |\mathbf{g}(u_i)|$
8:          $\Delta\mathbf{W} = \eta(E_o - E)\mathbf{g}\mathbf{x}^T$
9:          $\Delta\mathbf{P}$=round($\mathbf{f}(\mathbf{\Delta W})$)
10:     end for
11:     Decrease $E_o$
12: end while
```

IV. De-mixing Example and Results

As a test bench for the proposed technique, we consider two Laplacian random variables that are generated and mixed using a mixture matrix which is set to a rotation matrix $\mathbf{A} = (\cos\theta, -\sin\theta; \sin\theta, \cos\theta)$ with $\theta = \pi/6$. The circuit has been implemented as shown in Fig.3. The system weights are trained using 5×10^6 samples with 5×10^{-9} learning rate and hard $tanh$ activation function (g) with $b = 2 \times 10^6\Omega$. On the other hand, RRAMs are programmed using $1\mu s$ pulses with $\pm 3V$ using the aforementioned conductance parameters. The variability of model parameters are considered with different Gaussian distribution for each device to consider device to device variability as well using the aforementioned parameters.

Figure 4 shows the results of the online learning of independent components of the mixed signals. Figure 4a shows the weights evolution during the training. The weights $\mathbf{W} = \mathbf{G} - Gr$ are initialized by the identity matrix (no knowledge about the mixture matrix) where the conductance matrix is $G = [G_{max}, -G_r; G_r, G_{max}]$. After learning, the weight matrix is $W = [0.2034, 0.1133; -0.11, 0.17]\mu S$ and $WA = 0.233 \times [1, -0.0155; -0.043, 0.8733]\mu S$ which is approaching identity. Clearly, there are some oscillations in the weights around the final solution after 10^4 samples because of the continuous on line learning and the devices variations. Figure 4c shows the evolution of pulses for programming each weight and the evolution of the global surprise signal E is shown in Fig.4b. A visual representation of the signals before mixing, after mixing and after training is shown in Fig. 4d which depicts the similarity between the source and output signals.

Endurance and retention of RRAMs are very important measures especially for online learning case. It is required to guarantee that the algorithm converges after a number of update cycles much smaller than the endurance of the used device. The recent fabricated devices have good endurance and retention values typically around 10^8 and 4 years, respectively [15]. The total number of programming pulses for each weight are $\{2.96, 1.4, 0.64, 2.73\}$ million pulses which

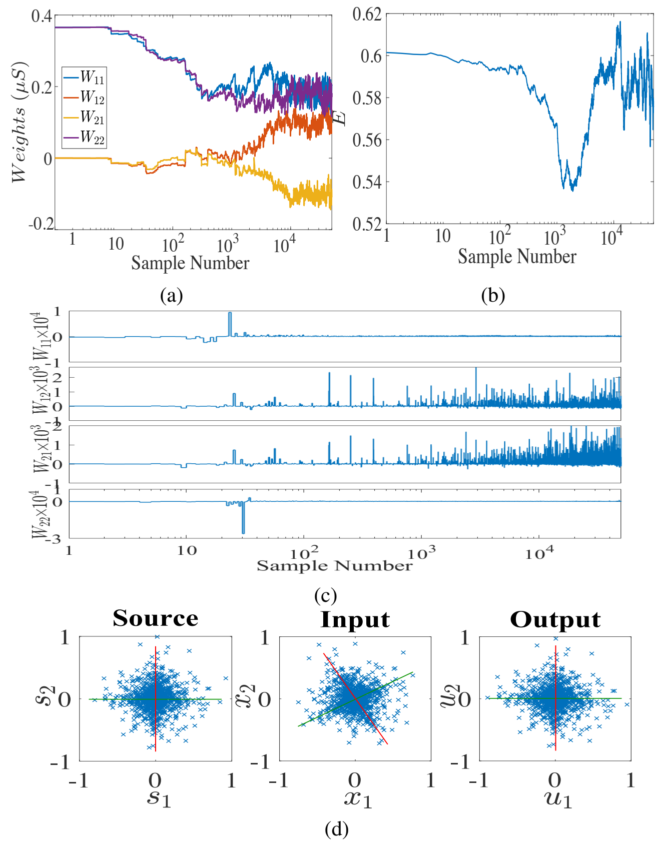


Fig. 4: The online training results versus training time (a) Evolution of the weights, (b) surprise Energy function, (c) Required programming pulses for each weight, and (d) Visual results of the input and the output.

are less than the endurance. One way to decrease the number of programming pulses is by updating the weights using batch-based updates which will decrease the variations in the weights as well.

V. Conclusion and Future Work

The realization of ICA using RRAMs is introduced taking into consideration the asymmetric nonlinearity behavior of the devices and the variations. The closed form learning rule is introduced and is applied to de-mixing two Laplacian signals. The proposed algorithm showed good performance and converges to the independent components even with the existence of the device variability. In the future work, the algorithm and circuitry needed for on-chip learning will be applied to extract image features that could be used as an alternative to sparse coding.

Acknowledgment

This work was partly supported by the National Science Foundation under grant 1640081, and the Nanoelectronics Research Corporation (NERC), a wholly-owned subsidiary of the Semiconductor Research Corporation (SRC), through Extremely Energy Efficient Collective Electronics (EXCEL), an SRC-NRI Nanoelectronics Research Initiative under Research Task ID 2698.003.

References

[1] “Idc worldwide internet of things forecast update, 2016-2020: December 2016,” in *Doc # US42082716*.

[2] D. H. Hoang and H. D. Nguyen, “A pca-based method for iot network traffic anomaly detection,” in *Advanced Communication Technology (ICACT), 2018 20th International Conference on*. IEEE, 2018, pp. 381–386.

[3] S. Joshi, C. Kim, S. Ha, Y. M. Chi, and G. Cauwenberghs, “21.7 2pj/mac 14b 8× 8 linear transform mixed-signal spatial filter in 65nm cmos with 84db interference suppression,” in *Solid-State Circuits Conference (ISSCC), 2017 IEEE International*. IEEE, 2017, pp. 364–365.

[4] A. e. a. Hyvärinen, *Independent component analysis*. John Wiley & Sons, 2004, vol. 46.

[5] A. M. Hassan, C. Yang, C. Liu, H. H. Li, and Y. Chen, “Hybrid spiking-based multi-layered self-learning neuromorphic system based on memristor crossbar arrays,” in *2017 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2017, pp. 776–781.

[6] S. e. a. Choi, “Experimental demonstration of feature extraction and dimensionality reduction using memristor networks,” *Nano letters*, vol. 17, no. 5, pp. 3113–3118, 2017.

[7] P. M. Sheridan, F. Cai, C. Du, W. Ma, Z. Zhang, and W. D. Lu, “Sparse coding with memristor networks,” *Nature nanotechnology*, vol. 12, no. 8, p. 784, 2017.

[8] C. e. a. Du, “Reservoir computing using dynamic memristors for temporal information processing,” *Nature communications*, vol. 8, p. 2204, 2017.

[9] C. Li and et al., “Analogue signal and image processing with large memristor crossbars,” *Nature Electronics*, vol. 1, no. 1, p. 52, 2018.

[10] T. Isomura and T. Toyoizumi, “A local learning rule for independent component analysis,” *Scientific reports*, vol. 6, p. 28073, 2016.

[11] S. Yu, “Neuro-inspired computing with emerging nonvolatile memorys,” *Proceedings of the IEEE*, vol. 106, no. 2, pp. 260–285, 2018.

[12] J. Park and et al., “Tio x-based rram synapse with 64-levels of conductance and symmetric conductance change by adopting a hybrid pulse scheme for neuromorphic computing,” *IEEE Electron Device Letters*, vol. 37, no. 12, pp. 1559–1562, 2016.

[13] M. Prezioso and et al., “Training and operation of an integrated neuromorphic network based on metal-oxide memristors,” *Nature*, vol. 521, no. 7550, p. 61, 2015.

[14] C.-C. Chang and et al., “Mitigating asymmetric nonlinear weight update effects in hardware neural network based on analog resistive synapse,” *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 2017.

[15] C. Nail and et al., “Understanding rram endurance, retention and window margin trade-off using experimental results and simulations,” in *Electron Devices Meeting (IEDM), 2016 IEEE International*. IEEE, 2016, pp. 4–5.