Processing big-data with Memristive Technologies: Splitting the Hyperplane Efficiently

A. Serb, G. Papandroulidakis, A. Khiat, and T. Prodromakis
Nanoelectronics and Nanotechnology Research Group, Electronics and Computer Science Department,
University of Southampton, Southampton, SO17 1BJ, UK
a.serb@soton.ac.uk

***Abstract*—An important cornerstone of data processing is the ability to efficiently capture structure in data. This entails treating the input space as a hyperplane that needs partitioning. We argue that several modern electronic systems can be understood as carrying out such partitionings: from standard logic gates to Artificial Neural Networks (ANNs). More recently, memristive technologies equipped such systems with the benefit of continuous tunability directly in hardware, thus rendering these reconfigurable in a power and space efficient manner. Here, we demonstrate several proof-of-concept examples where memristors enable circuits optimised to carry out different flavours of the fundamental task of splitting the hyperplane. These include threshold logic and receptive field based classifiers that are presented within the context of a unified perspective.**

Keywords—memristor, Metal Oxide RRAM, Artificial Neural Networks, Threshold Logic Gates, Template Pixel, texel, Clusterer, Fuzzy Gate

I. Introduction

The world we live in is full of structure and regularity as dictated by the laws of physics. This structure is captured every day by biological neural networks and artificial computing machines that in their respective ways encode and exploit patterns. The neural network is a prime example demonstrating how regularities in some, typically high-dimensional, input space (e.g. the space defined by visual or auditory inputs) can be learned by an appropriate architecture underpinned by a suitable learning rule. Similarly, even in the realm of simple, multi-D binary spaces, patterns are regularly made use of as exemplified by logic gates and other combinatorial circuits.

The ubiquity of pattern exploitation is illustrated in Fig. 1. Whether the input space is digitised, continuous, or fixed-resolution quantised, many of the fundamental computational primitives used today essentially perform the same function: split their input space hyperplanes into different domains. Typically this involves two domains mapping for outputing = 0 or 1 respectively. Computational elements as diverse as logic gates and McCulloch-Pitts neurons [1] (esp. if using a step activation function) fit this description.

Countless implementations of computational elements have been proposed and used over time, ranging from the commercially used logic gates to artificial neurons employing Complementary Metal-Oxide Silicon (CMOS) technologies [2]. However, attempting to move from binary input spaces towards continuous ones typically involves heavy use of multiplication operations. This cannot be carried out efficiently under modern electronics constraints for power, area and accuracy efficiency, be that digital or analogue multiplication.

Recently, emerging device technologies, such as RRAM (also known as memristor [3]), have garnered attention due to their unique properties, offering: continuously tuneable resistance[4], non-volatility [5] and fabrication in ultra-dense crossbar arrays [6]. In particular, their ability to assume and hold an analogue resistive state, i.e. act as ultra-compact analogue memory, renders them ideal candidates for the implementation of analogue weights. Input signal-weight multiplication can then be carried out effortlessly by applying a voltage across the device and measuring the resulting current. These properties have led to numerous proposals for using emerging devices of all flavours (PCM, RRAM etc.) in dot product multiplier blocks [7].

In this work we present an array of proposed memristor-based computational module implementations, and relate them to the general principle of 'splitting a hyperplane' efficiently. Specifically, in section II we detail the conceptual connection of logic gates and neurons of various types to the general framework of splitting hyperplanes. In section III we present memristor-based circuit implementations of the computational units examined in II and provide details on their functionality. Finally, in section IV, we discuss future prospects for applications in real-time big-data processing.

II. Theoretical background

Data classification is a fundamental processing operation that is the cornerstone of most machine learning applications. This may be implemented in a number of specific forms, but ultimately it involves the separation of an input space into sub-domains.

A. Linear separation

The idea of splitting an input space hyperplane in different ways is shown in Fig. 1. In 1a an AND gate splits a 2D binary space consisting of four distinct points into a domain consisting of three points (00, 01, 10 mapping to output 0) and another domain containing the remaining point (11 mapping to output 1). In 1d a typical 2-input neuron uses its weights and activation function threshold to separate its input space into two domains by means of a demarcating straight line. Notably, the AND gate can be interpreted as a special case of a neural network where an appropriate threshold line linearly separates the input space. The same applies for the OR gate, shown in 1b.

This fundamental concept generalises to higher dimensional input spaces, where the familiar 2-input gates become threshold logic gates (TLGs). TLGs have a discrete space to separate and act as majority gates able to perform linearly separable functions of quantized space. In practice this

means that they split a hypercube along a hyperplane that cuts every facet of the hypercube at an angle of 45º, at locations determined by the value of the thresholds. Depending on the threshold level the TLG will toggle state when 1/N, 2/N, or in general M out of N inputs are concurrently active, without the ability to discriminate which particular inputs are active/ inactive. Notably, the operation of TLGs can be directly related to highly restricted neural networks, where all the input weights are the same and the designer retains control of the threshold level, as TLGs were indeed originally conceived [1]. The property of linear separability is maintained by TLGs.

Introducing weights at the inputs of TLGs turns them into restricted ANNs by allowing the tilting of the TLG threshold plane much like the hyperplane separator in an ANN. The input space remains binary, but the contribution of each input is moderated by a weight. Unlocking further flexibility can essentially upgrade TLGs to full artificial neural networks (ANNs). Specifically, if the input space is now permitted to be continuous (or even above-binary fixed-resolution) we obtain a typical perceptron with a step activation function, as illustrated in 1d. Linear separability is thus maintained, and if the weights are constrained to positive values (e.g. within the typical interval [0,1]), the slope of the line can only take negative values as per 1d.

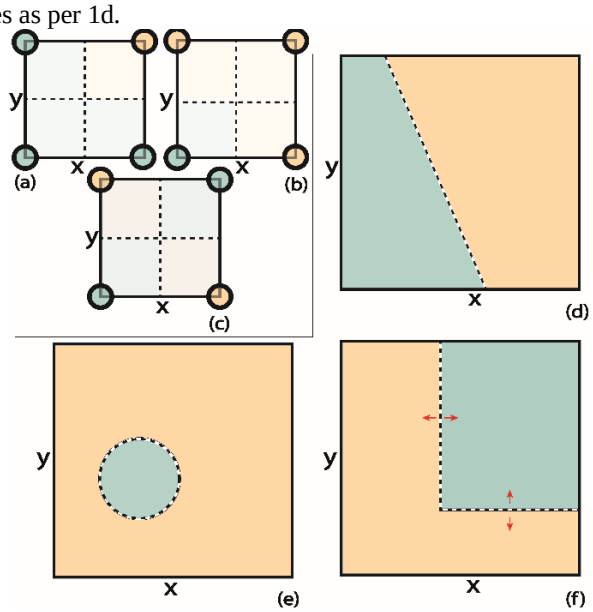

Fig. 1. Examples of ways of splitting 2D input spaces into output 0 (green) and 1(orange). (a-c) Binary input space. (a) AND gate, (b) OR gate, (c) XOR gate. (d-f) Continuous input space. (d) Linear separation, (e) receptive field separation, (f) double-threshold separation, in this cse implementing the NAND equation $\overline{A \wedge B}$.

B. *Beyond linear separation – receptive fields*

Computational units, in principle, need not separate their input spaces using hyperplanes. This is exemplified both in nature, through neurons featuring receptive fields, which might look as in Fig. 1e, and in digital logic by the use of the non-linearly separable XOR gate (Fig. 1c) and its higher dimensional generalisations.

The XOR gate (and its XNOR and higher dimensional 'exactly M out of N' or 'between K and M out of N' counterparts) are not arbitrary look-up tables (LUTs), but rather exploit symmetries in the input space. Similar to TLGs, they cannot discriminate between individual inputs. Nonetheless, even if particular symmetries are not exploited, digital circuits, built to pick specific patterns in an arbitrary LUT manner, can also be conceived. They differ qualitatively from TLGs (and generalised XORs) in that they are neither constrained by linear separability, nor by symmetry. What is common to these approaches, however, is that specific areas of the input space are picked and mapped directly to the desired output (e.g. a digital 1), rather than splitting their input spaces with hyperplanes.

Transitioning to a continuous input space, the same concept generalises quite naturally to receptive fields. They may be as simple as picking up a band of auditory frequencies (selecting a –weighted- range out of a 1D input space) or as complicated as Gabor filter-like patterns in retinal cells [8]. In principle, receptive fields in a continuous input space, just like LUTs in binary spaces, can, but need not necessarily exploit symmetries. They are there to scan for specific input patterns, i.e. essentially perform template matching, a form of clustering, much like may be found in the first layers of a typical convolutional neural network (CNN) [9].

Naturally, receptive field and hyperplane-based space partitioning are closely interrelated, it being possible to express one as a synthesis of elements of the other. In other words, a receptive field can be synthesized out of hyperplane separators and a hyperplane partition can be synthesized out of appropriate combinations of suitable receptive fields. This can be achieved within the context of multi-layered ANNs, or equivalently combinatorial circuits using cascades of multiple gates or LUTs for binary spaces.

C. *Between linear separation and receptive fields*

An important element of linear space separation, as implemented in ANNs, is dot product multiplication. Within the context of data classification as input space splitting, we propose an alternative to hyperplane-based partitioning that requires no explicit multiplication. This is illustrated in 1f, where two thresholds (one for each input dimension) combine in an OR fashion in order to split the space in a 'L' shape. In turn, classification into output 1 and 0 can be carried out by two thresholding/comparisons and a simple logical operation, thus obviating the need for multiplication that could be an inherently computationally intensive operation.

This can be interpreted as a receptive field, or as an approximation of a linear, hyperplane cut. In that vein it can also be interpreted as sitting between traditional neurons (separation of a continuous input space using a simple shape – but not a single hyperplane) and logic gates (space splitting expressible as N thresholding plus one logic operations).

III. Implementation Examples

Countless approaches exist for implementing plane splitting in hardware. These range from gates and digital system (e.g. FPGA) approximations to such analogue blocks as transistor-based fuzzy gates [10]. In this work we focus on a set of approaches exploiting memristive technologies as developed in our group. Our purpose is to relate memristor-based circuit design approaches to the quest for accelerator hardware specialized for AI-relevant computation.

A. *Threshold Logic Gates*

The introduction of traditional threshold logic gates (TLGs) was the result of continuous progress in the area of TL circuit implementations towards the development of a brain-like computer. Many different TLG circuit designs exist, with differential input current-mode TLGs as a front-running candidate due to their low power and high performance operation, as well as the compatibility with existing nanoscale conventional computing architectures [11], [12]. The circuit implementation of Current-Mode TLGs [13] consists of two parts: the differential and the sensor parts. The part 1 receives the input and threshold vectors at corresponding N-input differential pair networks. Part 2is the thresholding element of the circuit, which detects the difference in currents flowing through the input and threshold branches of part 1 and amplifies them to a digital answer indicating which is higher.

Fig. 2a shows a memristor-enhanced current-mode TLG design. The differential branches consist of parallel memristively source-degenerated pMOS transistors (1T1M) [14].Memristors are employed as the analogue weights at the input/bias binary signals. Using memristors has the advantage of introducing highly localised, continuously tuneable, minimal front-end footprint and low-voltage operated memory into the TLG, thus providing a decisive advantage in the implementation of memory-heavy ANN accelerators [15].

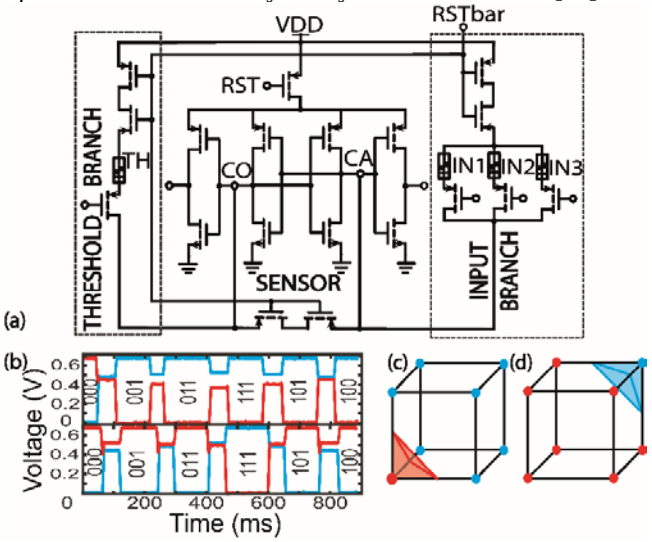

Fig. 2. Circuit schematic (2a) and measured results of the sensor part outputs regarding the OR/NOR and AND/NAND functions (2b) for the upper and lower trace, respectively. The schematics in (2c) and (2d) showcase the 3D space split for the AND and OR threshold function, respectively.

The measured results (Fig. 2b) are extracted from a 3-element input vector and 1-element threshold setting TLG circuit, where discrete resistors were used to emulate appropriately programmed RRAM devices. The upper trace of Fig. 2b shows the response for the {IN1, IN2, IN3; TH} = {22kΩ, 22kΩ, 22kΩ; 22kΩ} weight configuration which functions as a 3-input OR. The blue waveform is the measured response for the canonical output (CA) while the red waveform is the naturally provided complementary output (CO) by the bi-stable latching sensor part. Similarly the lower trace of the Fig. 2b presents the results of a 3-input AND/NAND TLG resulting from changing the threshold weight (weight config.: {22kΩ, 22kΩ, 22kΩ; 7kΩ}).

B. *The template pixel (texel)*

Whereas receptive fields tend to be implemented using computationally intensive and multiplication-heavy convolutional layers in ANNs, memristor-enhanced circuits can directly implement receptive fields in hardware. An example of such circuit is shown in Fig. 3a [16].The first inverter maps each input voltage V_{IN} to some voltage V_{MID}; a mapping influenced by the values of the memristors inside the inverter. The 2nd inverter, is essentially scanning for a particular V_{MID} value for which its transistors M3 and M4 will be simultaneously maximally conductive. At that preferred voltage V_{pk}, the current through the output transistor M6 is also maximised. As a result, the whole circuit uses memristors to map specific, programmable values of V_{IN} to a maximum current I_{out}. This is shown experimentally in Fig. 3b, where the input voltage is swept between the power supplies (blue) as V_{MID} (red) and I_{out} (green) are monitored. We observe that values of V_{IN} within a narrow band of V_{pk} will also lead to high I_{out}, therefore, the system implements a 1D receptive field with the selectivity profile shown in Fig 3b. The precise range of values falling within or outside the receptive field is also determined by the chosen I_{out} threshold level.

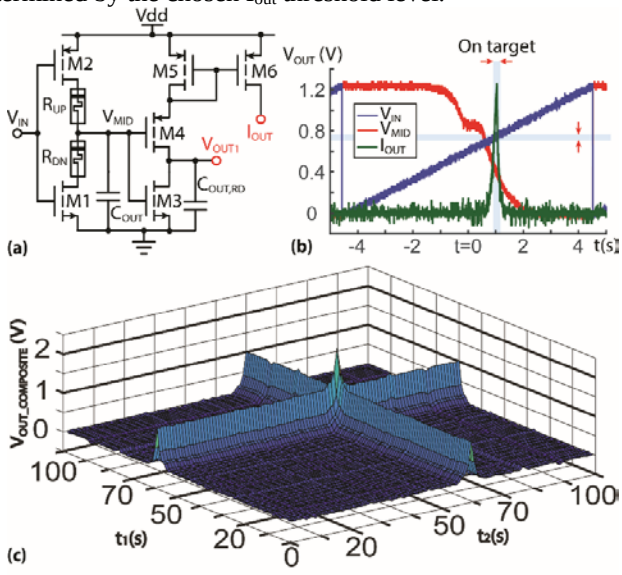

Fig. 3. Receptive field-based input plane-splitting using memristive circuits. (a) Circuits schematic showing memristor enhanced inverter (stage 1) and read-out circuit (stage 2). In this case the output quantity is I_{OUT}. (b) Measured input signals from circuit in (a). I_{OUT} peaks in a narrow range of input voltages around 0.7V and is indirectly measured as voltage dropped across a 1MΩ resistor. (c) The result of linearly summing I_{OUT} from two identical texels (equivalent voltage drop down 1MΩ resistor – simulated extrapolation of data from (b)). R_{UP}=10.5kΩ, R_{DN}=11.5kΩ. Adapted from [17].

Combining the outputs of multiple texels leads to higher dimensional receptive fields, the specific manner of interaction between texel outputs will determining the rules of composition. Summing I_{out} currents, for example, will lead to fields as shown in Fig. 3c. This is a straightforward current summation; easily implementable and avoiding any multiplication, which is another option for higher D receptive field composition. There is no reason why other designs featuring more complicated circuits cannot be used to implement more flexible or complex receptive fields.

C. Fuzzy Gates

The texel circuit is an enhancement of the classical inverter. Separation of an input plane in 'L' shapes as shown in Fig. 1f can be achieved in hardware using an equivalent memristor-based enhancement of higher level gates, such as the NAND, shown in Fig. 4a and explored in [17]. Observing how a memristor-enhanced NAND maps its 2D input space to analogue outputs (Fig. 4b) we note that: a) choosing a fixed voltage threshold will always cut the plane into a 'L' shape and b) the input/output relationship bears some resemblance to a fuzzy gate. Hence we dub the enhanced NAND as a 'fuzzy NAND' implementing a non-standard flavour of fuzzy logic determined by the electrical characteristics of the components involved. We note that once a threshold is introduced, the fuzzy gate implements the logic equation $\overline{(A > A_{th}) \wedge (B > B_{th})}$, where A, B are the inputs and A_{th}, B_{th} their respective thresholds. This is opposed to the corresponding McCulloch Pitts arithmetic equation $Aw_A + Bw_B > \theta$, where w_A, w_B are the weights corresponding to inputs A and B and θ the threshold value separating an output 0 from an output 1.

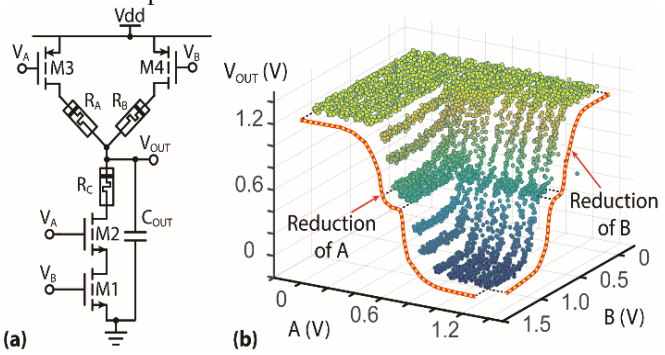

Fig. 4. Double threshold-based plane-splitting using memristive soft gates. (a) Soft NAND circuit topology. (b) Measured soft NAND behaviour. Setting an output voltage threshold at an intermediate voltage level will split the plane according to the double-threshold methodology shown in Fig. 1f. Side note: The reduction of the NAND to soft inverters as marked on the figure are essentially activation functions along each dimension (outside scope of current paper). R_A= 3.5kΩ, R_B=0.5kΩ, R_C=4kΩ. Figure modified from [17].

Reconfiguring a fuzzy gate mainly involves the appropriate tuning of memristors R_A and R_B,which independently control the shape of the fuzzy input/output mapping along each input space dimension respectively. Therefore, setting the precise thresholds A_{th}, B_{th} is controllable in a straightforward manner that required no complicated calculations of any interaction effects between R_A and R_B. In principle, if the input signal ranges are mapped to a subset of the full power supply and the memristive devices are programmed appropriately, A_{th} and B_{th} may move entirely outside the valid input space shown in Fig. 1f, thus reducing the system to a (vertical or horizontal) plane separator. In any other case, the inverter 'L' shape continues to enforce a fundamentally NAND structure, as recognised by the fact that the (0,1), (1,0) and (1,1) corners will always lead to output 1 and corner (0,0) to output 0.Similar observations apply to the fuzzy NOR and the basic principle of enhancing gates with memristors generalises naturally to higher dimensionality gates (allocate 1 memristor to each fuzzy gate input, as was done with R_A and R_B in the 2-input case).

Finally, a basic difference between the full texel circuit and the multiple input fuzzy gates is that the latter need no output stage and communicate directly in a voltage in/voltage out manner, thus allowing in principle for natural chaining of said gates.

IV. Discussion & Conclusions

In this work we are discussing data classification within the context of splitting an input hyperplane space and relate some significant methods of plane splitting to corresponding memristor-based circuit implementations. The common aspect in all circuits presented, is the use of memristors as analogue tuning elements within what would in absentia of these devices be a purely digital, standard circuit. We thus exemplify how intimately intertwining memristors with existing electronic technologies can begin to transform a design strategy that has worked well for our fully digital world, into an approach more directly optimised for data classification (plane splitting).

In this work, we have deliberately restricted ourselves to systems that split the input space via clear-cut binary decisions. Naturally, a new degree of freedom is unlocked if we move away from the simple McCulloch-Pitts-like step activation function and into different functions such as ReLU [18], sigmoids [19] etc. However, this would require the development of (ideally very simple) read-out structures that modularly attach to the output of the circuits shown in Figs. 2-4 and transform it using the desired activation function.

Furthermore, we noted that in principle, given enough layers in the plane-splitting network, input space slicing can be achieved by using either traditional hyperplane, or receptive field or perhaps 'L'-shaped splitting techniques (or even a combination of the above). The choice of splitting strategy has significant impact on the methodologies that must be used for successful learning of the structure of any given input space. Traditional neural networks make use of e.g. error gradients for powering back-propagation-based approaches [20] and hence the importance of activation functions. It is an interesting and complex subject of further work to see exactly how learning in receptive field/L-shape-based networks might function and how this might compare to traditional ANNs.

In summary, we envision this approach to be key in developing next-generation electronic platforms that can meet modern societal requirements and constraints. On one side, real-time clustering and classification becomes increasingly important across a wide spread of applications that have in common the generation of big-data from a variety of sensory nodes. On the other side, these "computation substrates" must be designed within stringent power/area budgets for meeting the demanding specifications of emerging applications, e.g. autonomous vehicles and/or embedded monitoring platforms.

References

[1] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bull. Math. Biophys.*, vol. 5, no. 4, pp. 115–133, 1943.

[2] C. Bartolozzi and G. Indiveri, “Synaptic Dynamics in Analog VLSI,” *Neural Comput.*, vol. 19, no. 10, pp. 2581–2603, 2007.

[3] L. Chua, “If it’s pinched it’s a memristor,” *Memristors Memristive Syst.*, vol. 9781461490, pp. 17–90, 2014.

[4] S. Stathopoulos *et al.*, “Multibit memory operation of metal-oxide bi-layer memristors.”

[5] L. Qingjiang, A. Khiat, I. Salaoru, C. Papavassiliou, X. Hui, and T. Prodromakis, “Memory Impedance in TiO2 based Metal-Insulator-Metal Devices,” *Sci. Rep.*, vol. 4, no. 1, p. 4522, 2015.

[6] A. Khiat, P. Ayliffe, and T. Prodromakis, “High Density Crossbar Arrays with Sub- 15 nm Single Cells via Liftoff Process Only,” *Sci. Rep.*, vol. 6, no. 1, p. 32614, 2016.

[7] P. M. Sheridan, F. Cai, C. Du, W. Ma, Z. Zhang, and W. D. Lu, “Sparse coding with memristor networks,” *Nat. Nanotechnol.*, no. May, 2017.

[8] I. Hayashi and J. R. Williamson, “A Formulation of Receptive Field Type Input Layer for TAM Network Using Gabor Function,” 2004.

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” *Adv. Neural Inf. Process. Syst.*, pp. 1–9, 2012.

[10] V. Catania and M. Russo, “Design of basic hardware gates for efficient fuzzy computing,” *Proc. Fourth Int. Conf. Microelectron. Neural Networks Fuzzy Syst.*, pp. 100–109, 1994.

[11] C. B. Dara, T. Haniotakis, and S. Tragoudas, “Delay Analysis for Current Mode Threshold Logic Gate Designs,” vol. 25, no. 3, pp. 1063–1071, 2017.

[12] S. Leshner, N. Kulkarni, S. Vrudhula, and K. Berezowski, “Design of a robust, high performance standard cell threshold logic family for DSM technology,” *Proc. Int. Conf. Microelectron. ICM*, no. Icm, pp. 52–55, 2010.

[13] S. Bobba and N. Hajj, “Current-Mode Threshold Logic Gates,” pp. 4–9, 2000.

[14] C. B. Dara, T. Haniotakis, and S. Tragoudas, “Low Power and High Speed Current-Mode Memristor-Based TLGs,” *IEEE*, pp. 89–94, 2013.

[15] A. Kumar Maan, D. Anirudhan Jayadevi, A. Pappachen James, and S. Member, “A Survey of Memristive Threshold Logic Circuits,” *IEEE Trans. Neural Networks Learn. Syst.*, 2016.

[16] A. Serb, C. Papavassiliou, and T. Prodromakis, “A memristor-CMOS hybrid architecture concept for on-line template matching,” *Proc. - IEEE Int. Symp. Circuits Syst.*, pp. 1–4, 2017.

[17] A. Serb, A. Khiat, and T. Prodromakis, “Charge-based computing with analogue reconfigurable gates,” 2017.

[18] K. Hara, D. Saito, and H. Shouno, “Analysis of function of rectified linear unit used in deep learning,” *Proc. Int. Jt. Conf. Neural Networks*, vol. 2015–Septe, 2015.

[19] C. H. Tsai, Y. T. Chih, W. H. Wong, and C. Y. Lee, “A Hardware-Efficient Sigmoid Function with Adjustable Precision for a Neural Network System,” *IEEE Trans. Circuits Syst. II Express Briefs*, vol. 62, no. 11, pp. 1073–1077, 2015.

[20] R. Lister and J. V Stone, “Error functions, error signals, and conjugate gradient back propagation,” *Artif. Neural Networks Conf.*, no. 409, pp. 26–28, 1995.