Enhancement of a Stochastic-Computing Morphological Neural Network through Approximate Adders

Christiam F. Frasser[†], Tingting Zhang[⁕], Bowen Liu[⁕], Joan Font-Rosselló [†*], Lluc Crespí-Castañer[†], Alejandro Morán[†], Vincent Canals[†*], Miquel Roca[‡†*], Jie Han[⁕] and Josep L. Rosselló[‡†*] .

***Abstract*—Stochastic Computing has proven to be an extremely suitable hardware design approach for Morphological Neural Networks (MNNs) due to its ease of implementing maxima, minima, and products using simple logic gates that perform bitwise operations. This study enhances the design of MNNs by incorporating stochastic and approximate computing techniques. This approach results in a significant reduction in the required hardware resources for adders, which are the most area-consuming components of MNNs. Experimental results demonstrate a minimal decrease in accuracy, along with reductions in hardware resources, and improvements in speed and energy efficiency compared to previous studies. The proposed methodology is validated through implementation on a field-programmable gate array.**

I. Introduction

ARTIFICIAL Neural Networks (ANNs) are widely regarded as one of the primary methodologies in modern machine learning due to their remarkable adaptability across various problem domains [1]. However, their extensive adoption is hindered by significant computational demands, making them less feasible for deployment in systems constrained by power consumption and hardware resources [2], [3]. This limitation is particularly pertinent in edge devices, such as those within the Internet of Things (IoT) framework, which often face restrictions in both power and hardware resources [4].

The efficiency of ANNs relies on optimizing specialized building blocks designed for complex multiplicative accumulation operations, commonly known as MAC operations [5], [6]. However, these operations incur substantial energy consumption and introduce noticeable latency, posing challenges in real-time applications.

To address these limitations and meet the demand for intelligent autonomous systems, researchers have explored alternative computing techniques diverging from conventional ANN paradigms. These techniques include stochastic computing [7], photonic information processing [8], and the use of memresistive and/or superconductive devices [9], [10], as well as new ANN paradigms such as Morphological Neural Networks (MNNs) [11].

MNNs have gained attention for their departure from using MAC blocks, opting for simpler building blocks performing the addition (accumulation) of maximum and minimum operations (through ACC&MAX-MIN blocks). This results in a significant reduction in computational cost, without the need for nonlinear functions when combining MAC and ACC&MAX-MIN blocks [12].

Recent advancements in software frameworks, network architectures, and training methodologies have led to progress in hardware implementations of these alternative ANN architectures. Traditional binary logic is primarily used for implementing accumulator blocks, while stochastic logic is prevalent for implementing maximum, minimum, and MAC functions [12].

This study proposes integrating stochastic computing and approximate adders into non-stochastic circuitry for MNN hardware implementation. This synergistic approach enhances normalized operation speed relative to the clock frequency and significantly improves energy efficiency compared to existing literature, representing a notable advancement in MNN hardware design.

The paper is structured as follows: Section II introduces the basic principles for MNN implementation, Section III describes the experimental suite and results, and Section IV presents the main conclusions and future work.

II. Methods

In this section, we first review the basic principles for the MNN implementation proposed in [12], and then we expose the stochastic and approximate computing techniques applied to the circuitry in order to maximize energy efficiency.

† Electronics Engineering Group at Industrial Engineering and Construction Department, University of Balearic Islands, Ctra. Valldemossa Km 7.5, Palma de Mallorca 07122, Spain.

⁕ Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB T6G 1H9, Canada.

‡ Institut d'Investigació en Inteligència Artificial de les Illes Balears (IAIB), Palma de Mallorca, Spain.

∗ Balearic Islands Health Research Institute (IdISBa), Palma de Mallorca, Spain.

E-mail of corresponding author: {j.rossello@uib.es}

This work was partially supported by the Ministerio de Ciencia e Innovación and by European Union NextGenerationEU/PTR. under grant contracts PID2020-120075RB-I00 and PDC2021-121847-I00. Grant PID2020-120075RB-I00 funded by MCIN/AEI/10.13039/501100011033. Grant PDC2021-121847-I00 funded by MCIN/AEI/10.13039/501100011033 by the European Union NextGenerationEU/PRTR. The work at the University of Alberta was supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada (Project Numbers: RES0048688, RES0051374 and RES0054326) and Alberta Innovates (Project Number: RES0053965). T. Zhang was supported by a Ph.D. scholarship from the China Scholarship Council (CSC).

A. The single-layer morphological neural network

A single-layer MNN consists of two stacked layers: a hidden layer comprising M morphological units and an output layer with K linear neurons. Within the hidden layer, the neurons are divided into two groups: the upper $M/2$ neurons compute the maximum of the inputs, while the lower $M/2$ neurons compute the minimum. The inputs are represented by a single vector $\mathbf{x}$ of dimension Q, denoted as a column vector. The hidden layer is described using a vector $\mathbf{h}$, which accounts for the intermediate results of each hidden layer neuron and comprises M components, such that $\mathbf{h} = \{h_j\}_{M\times 1}$.

For the initialization of the layer, the MNN adopts an architecture similar to that described in [13], employing two processes: dilation and erosion. These processes are implemented using coefficients defined within a matrix $\mathbf{\Omega}_{M\times Q}$. Each component of the intermediate vector $\mathbf{h}_{M\times 1}$ is therefore generated using the expression:

$$h_j = \begin{cases} max_k\left(\omega_{jk} + x_k\right) & j \leq M/2 \\ min_k\left(\omega_{jk} + x_k\right) & j > M/2 \end{cases} \tag{1}$$

In the provided context, the index j spans all integer values from 1 to M, representing the neurons in the hidden layer, while the index k covers all possible input components, denoted by values from 1 to Q. Parameters ω_{jk} are the components of the weight matrix $\mathbf{\Omega}$.

The output vector $\mathbf{y}$, is composed of K components and is representing the desired output of the neural network. A single component of this vector can be expressed as follows:

$$y_i = \sum_{j=1}^{M} u_{ij} h_j, \tag{2}$$

In this context, u_{ij} represents the corresponding weight, with the index i ranging from 1 to K.

To facilitate effective training of the MNN, backpropagation algorithms can be utilized to derive the appropriate weights u_{ij} and ω_{jk}.

B. Natural pruning of MNNs

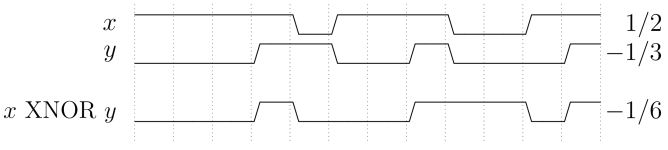
A natural pruning strategy stems from the unique min-max dependency of MNNs. Each morphological neuron selects only one input $\omega_{jl} + x_l$ from the Q possible inputs as the output. This means that, apart from the selected lth component $(\omega_{jl} + x_l)$, the remaining $Q - 1$ inputs do not contribute to the output of the jth neuron. Furthermore, after the network has been trained on the entire training set, there will be some weights ω_{jk} that never contribute to any output. As a result, these weights can be safely discarded or pruned. The significant aspect of eliminating redundant weights ω_{jk} is that their removal does not affect the performance or test accuracy of the network in any way. Pruning stands out as one of the most potent and appealing features of MNNs.

C. Stochastic computing

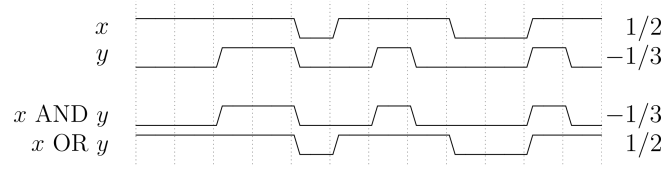
Stochastic computing (SC) is an unconventional approach renowned for its potential as an energy-efficient solution in image processing and the hardware implementation of different machine-learning methods [7], [14]. Among the various feasible stochastic encodings, bipolar coding stands out as a prominently utilized methodology. A bipolar signal, denoted as x, is derived from a binary sequence consisting of N_0 zeroes with a weight of $-1/(N_0 + N_1)$ and N_1 ones with a weight of $+1/(N_0 + N_1)$. This formulation yields $x = \frac{N_1 - N_0}{N_1 + N_0}$. As a consequence, an SC bipolar signal is confined within the interval $[-1, 1]$.

Operating on individual bit streams enables the implementation of product, maximum, or minimum operations using a single logic gate [15]. For example, in Fig. 1a, the stochastic product of two uncorrelated bipolar bit streams x and y is computed using an XNOR gate. Specifically, this subfigure illustrates an example where input bitstreams x have an activation probability of $3/4$ and y have an activation probability of $1/3$, corresponding to the values $1/2$ and $-1/3$ in bipolar coding, respectively. Thus, the activation probability of the output is $5/12$, corresponding to a bipolar quantity equal to the product of the input values $(-1/6)$. On the other hand, in scenarios where both x and y signals are generated by the same source of randomness (i.e., correlated signals), the max and min functions can be achieved using an OR gate for maximum and an AND gate for minimum, respectively. These operations are depicted in Fig. 1b. In the example, the activation rates for x and y are $3/4$ and $5/12$, respectively. Thus, the collision ratio between both (correlated) signals is the minimum $(5/12)$, and the disjunction is the maximum $(3/4)$.

Therefore, SC emerges as a promising choice for implementing the product and max-min functions, which are extensively used in MNNs. This study demonstrates the effectiveness of a hybrid architecture where activation functions are absent, additions are performed in 2's complement codification, while max-min and multiplications are executed through SC. The incorporation of SC circuitry contributes to minimizing power consumption and optimizing hardware resources, making it a compelling approach for efficient MNN implementations.



(a) XNOR with uncorrelated bit streams.



(b) AND and OR with correlated bit streams.

Figure 1. Examples of stochastic bit stream operations. (a) Bipolar stochastic computing product using uncorrelated inputs of an XNOR gate. (b) Stochastic computing minimum and maximum using correlated input bit streams for AND and OR gates, respectively.

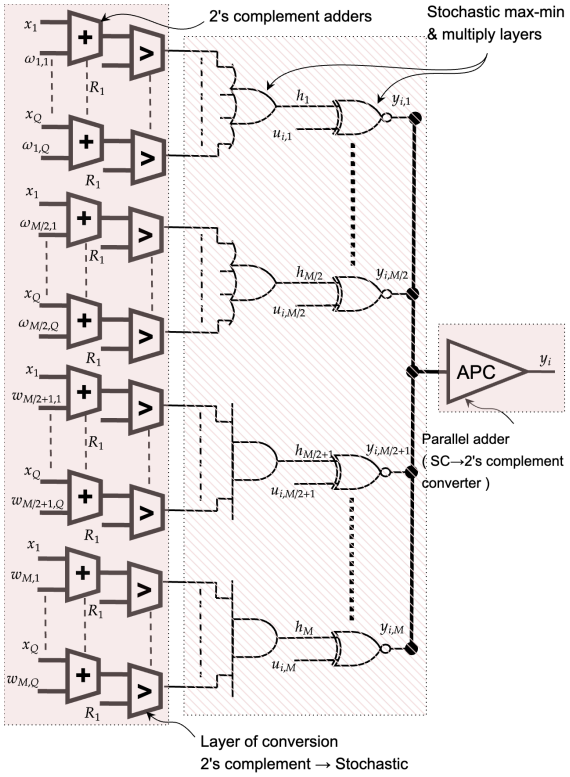


Figure 2. Example of an SC-MNN with a single output (y_i) and a single morphological layer with M nodes. We utilize adders, binary comparators and Accumulative Parallel Counters (APCs) employing classical 2's complement computations (shaded area), whereas single logic gates operate using bipolar stochastic codification (hatched area).

D. Hardware implementation of MNNs using stochastic computing

To attain an efficient hardware implementation of MNNs, a digital circuitry approach was adopted, combining classic 2's complement (C2) and stochastic bipolar (SCB) codifications. The conversion between C2 and SCB signals involves utilizing two random number generators that provide two random numbers, R_1 for the data path and R_2 for the weights (u_{ij}). In this design, C2 adders handle the necessary additions for MNN implementation, while the maximum, minimum, and product functions are executed using simple gates in the SCB codification. The final overall addition at the conclusion of MNN operations is carried out through an Accumulated Parallel Adder (APC), as illustrated in Fig. 2. Notably, a high degree of compactness is achieved by employing correlated maximum and minimum functions with OR and AND gates, along with XNOR gates for the multiplication of processed data with the uncorrelated u_{ij} weight parameters.

E. Hardware optimization using approximate adders

Approximate computing has garnered significant attention in recent years for its capability to decrease power, area, and timing overheads while maintaining acceptable accuracy in very-large-scale integration (VLSI) designs, particularly for arithmetic computing units [16]. Neural networks are commonly perceived as error-resilient, rendering them suitable for the deployment of approximate computing designs. In the SC-MNN structure, each morphological layer necessitates parallel addition operations in C2, which often dominate the circuit's area and power consumption. Therefore, it becomes advantageous to substitute conventional adders with approximate designs in order to mitigate these challenges.

Approximate adders can be categorized into four main types: speculative adders, segmented adders, carry select adders, and those employing approximate full adders [17]. For designs with relatively small bit widths, approximate adders constructed with approximate full adders are typically preferred due to their simplicity in implementation and moderate accuracy loss. In the design of the lower-part-OR adder (LOA), the input operands are divided into two parts: the least significant bits (LSBs) and the most significant bits (MSBs) [18]. Figure 3 illustrates an LOA that adds two n-bit C2 numbers, $A(n-1:0)$ and $B(n-1:0)$, to produce the sum, $S(n-1:0)$, and a carry out bit, C_{out}. The LSBs of the addition, $S(l-1:0)$ ($l \leq n$), are computed using multiple OR gates and one AND gate. The AND gate, with $A(l)$ and $B(l)$ as inputs, generates the carry C_{in}, which is then fed into the accurate adder on the left to calculate the MSBs in the sum, namely $S(n-1:l)$, and a carry out bit, C_{out}.

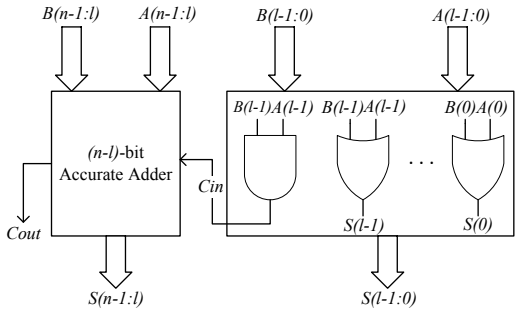


Figure 3. Design of the lower-part-OR adder (LOA) with l approximate bits [18].

In conventional adders for the LSBs, multiple gates are typically required for implementation, leading to higher hardware complexity. However, in the lower-part-OR adder (LOA), this complexity is significantly reduced to just one gate. This reduction in complexity is beneficial as it simplifies the overall circuit design. Additionally, the critical path of a multiple-bit LOA is shortened since the carry no longer needs to propagate through the LSBs. This relaxation of timing constraints contributes to improved circuit performance. It's important to note that using OR gates to generate the LSBs in the sum may introduce errors, resulting in a higher chance of erroneous results. However, the impact of approximation in the LOA on relative accuracy is less pronounced when the inputs are large. Therefore, while there may be some loss of accuracy, it may be acceptable depending on the application, especially when dealing with large input values.

III. Experiments and Results

Two distinct experiments were conducted to validate the significance of our contributions: one using the IRIS dataset [19] and the other employing the MNIST dataset [20].

Table I
ACCURACY RESULTS WHEN VARYING THE NUMBER OF APPROXIMATE BITS (l) IN THE LOA.

Dataset	Approximate bits (l)	Accuracy (%)
IRIS	0	96.00
	1	96.00
	2	93.33
	3	93.33
	4	85.33
	5	69.33
MNIST	0	97.25
	1	97.21
	2	97.20
	3	97.17
	4	97.29
	5	96.81
	6	95.94

Table II
FPGA IMPLEMENTATION FOR THE IRIS DATASET (75 TRAIN/75 TEST).

Metric	[21]	[12]	This work
ML Method	SOM	MNN	MNN
Test Accuracy (%)	**100**	96	93.3
Throughput (KIPS)	48.8	**1563**	**1563**
Power (mW)	21.5	7.81	**3.26**
Energy Efficiency (Inf/μJ)	2.2	200	**479**

Both experiments involved validation on field-programmable gate arrays (FPGAs).

For the IRIS dataset experiment, we utilized the Cyclone V 5CSEBA6U23I7S FPGA, while for the MNIST dataset experiment, we employed the Arria10 10AX115H1F34I1SG FPGA. The selection of FPGAs was made to validate the outcomes through physical hardware validation, thus enhancing the credibility of the results beyond mere simulation-based validation.

A. IRIS dataset

To start, the IRIS dataset was divided into two halves: 75 samples for training and 75 for testing. The MNN architecture comprises a single hidden layer with four intermediate neurons, each with a precision of 5 bits ($n = 5$). Then, we conducted experiments to approximate various numbers of parameter l in the LOA circuit (see Fig. 3). Table I presents the accuracy results for the different scenarios.

Subsequently, we selected the 3-bit approximate design from Table I given that it is the higher precision before accuracy plummets. Our 5-bit (2+3) approach was compared with other existing works in the literature that address the IRIS dataset in hardware. In Table II, we present the performance results for the FPGA implementation of the proposed model.

As observed, the proposed method achieves the best performance and energy efficiency compared to previous implementations [12], [21].

All of these benefits are attributed to the optimization of hardware adders using approximate computing techniques.

B. MNIST dataset

The proposed unconventional approach was validated using the MNIST dataset, which primarily consists of images. A total of 60,000 images from the dataset were used for training, while the remaining 10,000 images were reserved for testing.

Table III
COMPARISON BETWEEN MNNS WITH APPROXIMATE OR EXACT ADDERS

Metric	(Exact)	(Approximate)	Difference (%)
Accuracy (%)	97.25	96.81	0.45
ALMs	170,464	76,322	55.22
Comb. ALUT	267,348	116,314	56.5
Registers	5,498	5,468	0.54
Power (mW)	2,596	1,986	23.5

In this experiment, we employed an architecture comprising a single hidden layer with 200 intermediate neurons with 8-bit precision. Following a similar procedure to the IRIS dataset experiment, we carried out experiments aimed at determining the optimum number of bits l of the LOA circuit before accuracy sharply declines, as shown in Table I.

In Table III, we show the differences between an MNN that uses exact adders and the proposed MNN. As can be seen, significant benefits are obtained in terms of hardware resource reduction (over 50%) as well as power reduction (by 23.5%). The hardware reduction is in terms of Adaptive Logic Modules (ALMs) and Adaptive Look-Up Tables (ALUTs). At the same time, the overall accuracy of the network remains at very similar values (losing less than half a percentage point in accuracy). In addition to this comparison, which demonstrates the advantages of using approximate adders, we perform a comparison of the MNN design in relation to other similar works present in the literature. In this way, we compare the proposed design with other unconventional implementations such as Binary Convolutional Neural Networks (BCNNs) [22] and Stochastic Computing-based Convolutional Neural Networks (SCCNN) [23], [24]. All of these approaches were implemented using high-end FPGAs.

Table IV illustrates the comparison results. The *Performance* parameter, usually expressed in inferences per second in Neural Networks, has been normalized to the operation frequency to facilitate comparison among methodologies operating at different clock frequencies. The SCMNN implementation demonstrates a twofold increase in speed and performance compared to the other best non-conventional hardware implementation [24], while achieving a 21x improvement in energy efficiency.

In comparison to another SCMNN implementation [12], the proposed design performs the same task with 76% of the power and 44% of the required area, while also exhibiting higher energy efficiency (68K more inferences per Joule). These significant advantages are obtained with only a slight decrease in accuracy, amounting to a 0.44% penalty.

Figure 4 visually illustrates the comparison of normalized throughput and energy efficiency against other works using the data shown in Table IV. We also compared with reference [12], that is a similar MNN hardware implementation but without the use of approximate adders.

It is evident that the proposed design (red triangle) stands out as the most efficient one.

Table IV
COMPARISON WITH OTHER FPGA IMPLEMENTATIONS USING NON-CONVENTIONAL COMPUTING FOR MNIST.

Metric	[22]	[23]	[24]	This work
Year	2019	2020	2022	2024
Architecture	BCNN	SCCNN	SCCNN	SCMNN
Activation/Weight Precision (bits)	1/1	9/9	8/8	8/8
Test Accuracy (%)	**98.91**	98.13	97.58	96.81
FPGA Platform	Zynq ZC706	Zynq XC7Z020	Arria10 GX1150	Arria10 GX1150
Frequency (MHz)	120	60	150	150
Throughput (KIPS)	23	0.166	294	**590**
Performance (Inf/s/MHz)	191.7	2.77	1961	**3933**
Power (W)	0.63	**0.1**	21	1.99
Energy Efficiency (Inf/J)	36508	1666	14006	**296482**
Logic used (LUT or ALM) ($\times 10^3$)	29	28	343	76
DSP (blocks)	-	0	0	0
Memory (Mbits)	1.06	1.73	0.00	0.00

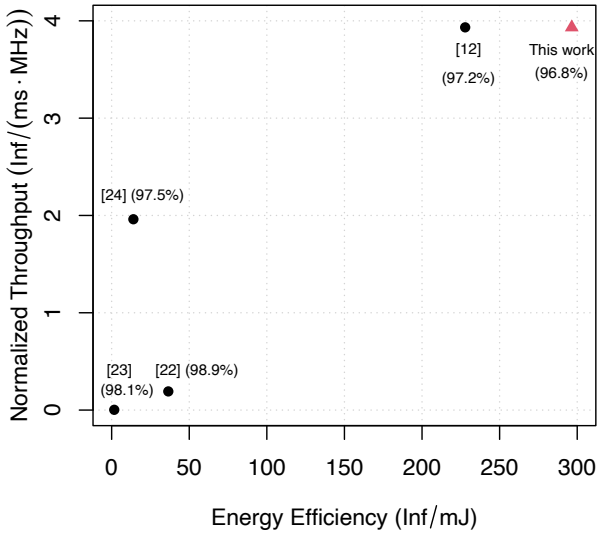


Figure 4. FPGA comparison for the normalized throughput and energy efficiency when addressing the MNIST benchmark.

IV. CONCLUSIONS AND FUTURE WORK

Stochastic Computing is particularly well-suited for MNNs due to its inherent simplicity in implementing operations such as maxima, minima, and products - key morphological operations - through basic logic gates like AND, OR, or XNOR gates, which perform bitwise operations on stochastic bitstreams. This work enhances a hybrid MNN design that combines conventional 2's complement logic with stochastic computing by replacing traditional adders with approximate ones.

Compared to the design in reference [12], the use of approximate adders instead of traditional adders in the initial stage has led to significant improvements in performance, energy efficiency, area reduction, power consumption, and hardware resource utilization, with only minimal accuracy penalty. This enhancement has been validated for both the IRIS and MNIST datasets in FPGA implementations, through comparisons with six previously published works. Consequently, the proposed design emerges as the top choice in terms of energy efficiency while maintaining its superiority in normalized throughput.

Looking ahead, the research team plans to develop a straightforward MNN Very Large-Scale Integration (VLSI) design using TSMC 180nm CMOS technology. The circuit will feature a total of 25,408 synapses, specifically optimized for handwritten number recognition. This initiative serves as an initial proof of concept for the proposed technology, enabling precise measurements of processing speed and energy efficiency.

REFERENCES

[1] Y. Lecun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[2] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proceedings of the IEEE*, vol. 105, no. 12, pp. 2295–2329, 2017.

[3] Q. Zhang, M. Zhang, T. Chen, Z. Sun, Y. Ma, and B. Yu, "Recent advances in convolutional neural network acceleration," *Neurocomputing*, vol. 323, pp. 37–51, 2019.

[4] S. Thouti, N. Venu, D. R. Rinku, A. Arora, and N. Rajeswaran, "Investigation on identify the multiple issues in IoT devices using convolutional neural network," *Measurement: Sensors*, vol. 24, p. 100509, 2022.

[5] V. Camus, L. Mei, C. Enz, and M. Verhelst, "Review and benchmarking of precision-scalable multiply-accumulate unit architectures for embedded neural-network processing," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 9, no. 4, pp. 697–711, 2019.

[6] Z. Niu, T. Zhang, H. Jiang, B. F. Cockburn, L. Liu, and J. Han, "Hardware-efficient logarithmic floating-point multipliers for error-tolerant applications," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 71, no. 1, pp. 209–222, 2024.

[7] A. Alaghi, W. Qian, and J. Hayes, "The promise and challenge of stochastic computing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 8, pp. 1515–1531, 2018.

[8] D. Brunner, M. Soriano, C. Mirasso, and I. Fischer, "Parallel photonic information processing at gigabyte per second data rates using transient states," *Nature Communications*, vol. 4, 2013.

[9] F. Pan, S. Gao, C. Chen, C. Song, and F. Zeng, "Recent progress in resistive random access memories: Materials, switching mechanisms, and performance," *Materials Science and Engineering R: Reports*, vol. 83, no. 1, pp. 1–59, 2014.

[10] J. Salmilehto, F. Deppe, M. Di Ventra, M. Sanz, and E. Solano, "Quantum memristors with superconducting circuits," *Scientific Reports*, vol. 7, 2017.

[11] G. X. Ritter and P. Sussner, "An introduction to morphological neural networks," in *Proceedings of 13th International Conference on Pattern Recognition*, vol. 4, Aug 1996, pp. 709–717 vol.4.

[12] J. L. Rosselló, J. Font-Rosselló, C. F. Frasser, A. Morán, E. S. Skibinsky-Gitlin, V. Canals, and M. Roca, "Highly optimized hardware morphological neural network through stochastic computing and tropical pruning," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 13, no. 1, pp. 249–256, 2023.

[13] N. Dimitriadis and P. Maragos, "Advances in the training, pruning and enforcement of shape constraints of morphological neural networks using tropical algebra," Nov. 2020, arXiv:2011.07643.

[14] V. Canals, A. Morro, and J. L. Rosselló, "Stochastic-based pattern-recognition analysis," *Pattern Recognition Letters*, vol. 31, no. 15, pp. 2353–2356, 2010.

[15] S. Liu, J. L. Rosselló, S. Liu, X. Tang, J. Font-Rosselló, C. F. Frasser, W. Qian, J. Han, P. Reviriego, and F. Lombardi, "From multipliers to integrators: A survey of stochastic computing primitives," *IEEE Transactions on Nanotechnology*, vol. 23, pp. 238–249, 2024.

[16] J. Han and M. Orshansky, "Approximate computing: An emerging paradigm for energy-efficient design," in *2013 18th IEEE European Test Symposium (ETS)*. IEEE, 2013, pp. 1–6.

[17] H. Jiang, F. J. H. Santiago, H. Mo, L. Liu, and J. Han, "Approximate arithmetic circuits: A survey, characterization, and recent applications," *Proceedings of the IEEE*, vol. 108, no. 12, pp. 2108–2135, 2020.

[18] H. R. Mahdiani, A. Ahmadi, S. M. Fakhraie, and C. Lucas, "Bio-inspired imprecise computational blocks for efficient VLSI implementation of soft-computing applications," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 57, no. 4, pp. 850–862, 2009.

[19] "IRIS," 2018. [Online]. Available: https://dx.doi.org/10.21227/rz7n-kj20

[20] Y. LeCun and C. Cortes, "The MNIST database of handwritten digits," http://yann.lecun.com/exdb/mnist/

[21] A. Morán, J. Rosselló, M. Roca, and V. Canals, "SoC kohonen maps based on stochastic computing," in *Proc. Proceedings of the International Joint Conference on Neural Networks*, Glasgow, UK, 2020.

[22] B. Liu, S. Chen, Y. Kang, and F. Wu, "An energy-efficient systolic pipeline architecture for binary convolutional neural network," in *2019 IEEE 13th International Conference on ASIC (ASICON)*. IEEE, 2019, pp. 1–4.

[23] P. K. Muthappa, F. Neugebauer, I. Polian, and J. P. Hayes, "Hardware-based fast real-time image classification with stochastic computing," in *2020 IEEE 38th International Conference on Computer Design (ICCD)*. IEEE, 2020, pp. 340–347.

[24] C. F. Frasser, P. Linares-Serrano, I. D. de los Ríos, A. Morán, E. S. Skibinsky-Gitlin, J. Font-Rosselló, V. Canals, M. Roca, T. Serrano-Gotarredona, and J. L. Rosselló, "Fully parallel stochastic computing hardware implementation of convolutional neural networks for edge computing applications," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–11, 2022.