SCFNNs: Stochastic Computing Fuzzy Neural Networks

Majid Farhadi, Witold Pedrycz, *Life Fellow, IEEE*, and Jie Han, *Senior Member, IEEE*

***Abstract*—The resource intensity of fuzzy neural networks (FNNs) may hinder their ubiquity, particularly in resource-constrained hardware platforms such as edge devices in Internet of Things (IoT) applications. In this letter, stochastic computing fuzzy neural networks (SCFNNs) are proposed as hardware-efficient FNNs to mitigate the resource demand. In SCFNNs, stochastic computing (SC) is tailored to implement FNNs, rather than using conventional floating-point (FP) and fixed-point (FxP) representations. By leveraging the compatibility of SC and fuzzy arithmetic, range normalization as required by conventional SC is obviated. Experiments on Iris, Wine, and Breast Cancer datasets, comparing FP, FxP, and SC implementations, manifest the advantage of the SC design in terms of hardware efficiency and performance in most cases, without degradation in accuracy.**

***Index Terms*—Stochastic Computing (SC), Fuzzy Neural Networks (FNNs)**

I. Introduction

FUZZY neural networks (FNNs) [1] are interpretable architectures by incorporating fuzzy sets (FSs) [2] into neural networks (NNs) [3]. Interpretability is essential in critical applications, where inspection of the internal mechanism of a system is obligatory for validation and verification [4].

However, hardware platforms with limited computational resources, e.g., edge devices in Internet of Things (IoT) [5] applications, may not be able to implement FNNs without special considerations.

Stochastic computing (SC) [6], on the other hand, is established to offer hardware efficiency at the cost of approximate computation. FNNs are tolerant of inexact computation, and approximate arithmetic can even lead to improved accuracy in some scenarios, due to the regularization with noise [7].

In this letter, stochastic computing fuzzy neural networks (SCFNNs) are proposed as hardware-efficient FNNs, suitable for resource-constrained critical applications such as medical diagnosis on portable devices. As the main contribution, SC is leveraged to efficiently implement FNNs, instead of using floating-point (FP) [8] and fixed-point (FxP) representations. Particularly, two essential modules are designed, including 1) an inference architecture constructed only of *AND* and *OR* logic gates, and 2) a novel training circuit tailored for fuzzy arithmetic, comprising two innovative operators for clipped addition and subtraction. In the integration of SC into FNNs, range normalization as required by conventional SC is lifted, since all the operations in FNNs are bounded to the unit interval (i.e., [0, 1]).

Although the acceleration of conventional NNs has been extensively investigated, the intersection of interpretability and hardware efficiency of NNs is largely unexplored. More specifically, there is no prior work on accelerating FNNs.

For evaluation, the effectiveness of the proposed methodology is verified by contrasting the training and inference results obtained for FP, FxP, and SC implementations, comprising accuracy, hardware efficiency, and performance, for Iris [9], Wine [10], and Breast Cancer [11] datasets.

The rest of the letter is structured as follows. Section II provides the prerequisites. Section III describes the proposed methodology in detail. Section IV presents the experimental results. Section V concludes the letter.

II. Preliminaries

A. Fuzzy Sets

FSs are felicitous when the criteria for membership in a class of objects are not strictly defined (e.g., a class of all real numbers close to 5). As an extension to sets, where the membership degrees are Boolean, the degree of membership in an FS lies within the unit interval. In other words, if $f_A(x)$ is the membership degree of x in an FS A, the value of $f_A(x)$ belongs to $[0, 1]$, instead of $\{0, 1\}$.

Primary set operations are complementation, intersection, and union, respectively realized by logical *NOT*, *AND*, and *OR* operations on Boolean membership degrees. For FSs, while complementation is defined as $1-f_A(x)$, t-norm and t-conorm operations implement intersection and union [12]. Based on the definition, t-norm and t-conorm operations, denoted by t and s, satisfy commutativity, monotonicity, and associativity properties, as well as the boundary conditions stated as

$$f_A(x)\ t\ 0 = 0, \qquad f_A(x)\ t\ 1 = f_A(x), \tag{1}$$

$$f_A(x)\ s\ 0 = f_A(x), \qquad f_A(x)\ s\ 1 = 1. \tag{2}$$

Specifically, minimum and product operations are common examples of t-norms, as maximum and probabilistic sum (i.e., $c = a + b - ab$) operations are of t-conorms.

B. Fuzzy Neural Networks

FNNs are interpretable 2-layered logic-oriented architectures. One category of neurons in FNNs is aggregative neurons, classified into *AND* and *OR* neurons. If x and w are input and weight vectors of dimension n, employing t-norm and t-conorm operations, *AND* and *OR* neurons realize

$$y = \mathit{AND}(x; w) = T_{i=1}^{n}(w_i\ s\ x_i), \tag{3}$$

$$y = \mathit{OR}(x; w) = S_{i=1}^{n}(w_i\ t\ x_i). \tag{4}$$

In fact, *AND* and *OR* neurons are generalizations of their corresponding logic gates, equipped with weights, which define a weighted intersection and union of the inputs.

FNNs are composed of either a layer of *AND* neurons followed by a layer of *OR* neurons or organized in the reverse order. Inspired by the Shannon theorem [13], stating that logic

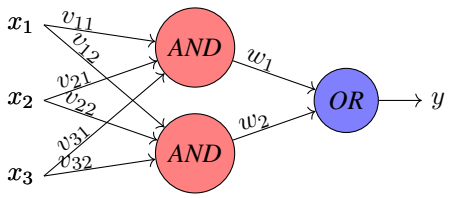

Fig. 1: An SOP-inspired FNN employing t-norm and t-conorm operations.

circuits built of either a layer of *AND* gates followed by a layer of *OR* gates, as a sum of products (SOPs), or reverse, as a product of sums (POSs), can implement any Boolean function (assuming the input complements are provided), FNNs are also universal approximators [14]. For instance, Fig. 1 represents an SOP-inspired FNN, implementing

$$y = S^2_{j=1}(w_j \; t \; T^3_{i=1}(v_{ij} \; s \; x_i)). \tag{5}$$

In the context of supervised learning [15], assume that the training dataset consists of N pairs of input-output sample vectors (x_k, t_k), y_k is the output vector of the underlying FNN fed by x_k, and Q is the sum of the squared error (SSE) loss function. The gradient descent algorithm [16] is employed to train the architecture and find the optimum weights, for which the gradient of the loss function with respect to each weight is required. For example, with respect to the weight w_j, it is

$$\frac{\partial Q}{\partial w_j} = -2 \sum_{k=1}^{N} (t_k - y_k) \frac{\partial y_k}{\partial w_j}. \tag{6}$$

Then, the corresponding weight is updated by

$$w_j \leftarrow w_j - \alpha \frac{\partial Q}{\partial w_j}, \tag{7}$$

where α is a positive learning rate. As the weights must be retained within the unit interval, clipping is performed as

$$clip(w_j) = \begin{cases} 0, & w_j < 0 \\ w_j, & 0 \leq w_j \leq 1 \\ 1, & w_j > 1 \end{cases}, \tag{8}$$

if they are updated to values outside of the unit interval.

FNNs are interpretable in such a way that the whole architecture can be translated into a logic statement. The equivalent logic description of the architecture depicted in Fig. 1 is

$$\begin{gathered} y = \left(x_{1|v_{11}} \; AND \; x_{2|v_{21}} \; AND \; x_{3|v_{31}}\right)_{|w_1} \\ OR \\ \left(x_{1|v_{12}} \; AND \; x_{2|v_{22}} \; AND \; x_{3|v_{32}}\right)_{|w_2}. \end{gathered} \tag{9}$$

In this notation, subscripts indicate the weights that determine the relevance or significance of the corresponding connection. For *AND* neurons, weights closer to 0, and for *OR* neurons, weights closer to 1 imply a higher relevance.

C. Stochastic Computing

In SC, numbers are encoded as the frequency of uniformly distributed 1s in a bit stream (e.g., 10100101 represents 0.5). Unlike binary representation, in which bits have different weights, SC exploits unweighted sequences of bits. As a consequence, each bit is processed individually and independently, eliminating the need for carry bits as in the binary format. Hence, basic SC arithmetic circuits can operate in parallel.

SC arithmetic is based on probability calculus, in which quantities, as the probability of events, are encapsulated within the unit interval. Matching the intersection and union operations for sets, *AND* and *OR* gates respectively compute the intersection and union probabilities of the SC encoded events.

Nevertheless, as the range of conventional SC numbers is limited to the unit interval, range normalization is inevitable for operations such as addition [17]. In this case, the result is scaled, which reduces the resolution.

III. Stochastic Computing Fuzzy Neural Networks

A. Fuzzification

Fuzzification (fuzzy encoding) [18] is the process of converting numeric data into fuzzy membership degrees. Fuzzy C-Means (FCM) [19] is a clustering algorithm used for data-driven fuzzification. Key parameters of FCM are the number of clusters, fuzzification coefficient, and distance function. Since this work is applied to classification datasets, the number of clusters is set to the number of class labels. To reduce computational complexity, the fuzzification coefficient is set to 1.01, resulting in Boolean membership degrees, without degradation in clustering accuracy. Therefore, each sample is converted into a one-hot vector, specifying its membership degrees to the clusters. The selection of an appropriate distance function depends on the underlying dataset and requires meticulous tuning. This preprocessing is performed by a software program, leveraging the skfuzzy package [20]. As for the SC implementation, all-zero or all-one sequences are generated for the Boolean membership degrees.

B. Inference

Utilizing *AND* and *OR* gates in SC to carry out t-norm and t-conorm operations has been investigated [21]. If the input sequences are generated independently, *AND* and *OR* gates perform the product t-norm and the probabilistic sum t-conorm. Likewise, in the case of perfect correlation, the minimum t-norm and the maximum t-conorm are realized.

One step further, by adjusting the correlation between the inputs, *AND* and *OR* gates are generic t-norm and t-conorm operators, irrespective of the implemented function, because of the compliance with the commutativity, monotonicity, and associativity properties along with the corresponding boundary conditions in (1) and (2). Employing *AND* and *OR* gates to carry out t-norm and t-conorm operations, Fig. 1 transforms into Fig. 2, replicated across all bits.

Particular to this implementation, because of all-zero and all-one sequences, the minimum and product t-norms, and the maximum and probabilistic sum t-conorms are identical.

As an alternative approach, since the inputs are Boolean in this particular case, *OR* gates in the first layer can be replaced with multiplexers, realizing the same function.

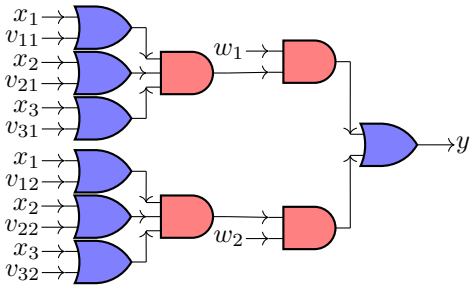

Fig. 2: A 1-bit slice of the inference circuit with AND and OR gates to respectively perform t-norm and t-conorm operations.

C. Training

The training algorithm is altered into stochastic gradient descent (SGD) [22], in which weights are updated on a per-sample basis. The rationale behind this modification is that computing a large gradient over all samples is unnecessary because if a large gradient causes the weights to be updated to values outside of the unit interval, clipping is performed at the end. Moreover, due to the limited range of SC numbers, incapable of representing large values, this alteration is better suited for SC arithmetic. Substituting (6) into (7) for the kth sample results in the weight update specified as

$$w_j \leftarrow w_j + 2\alpha(t_k - y_k)\frac{\partial y_k}{\partial w_j}. \tag{10}$$

The term $t^k - y^k$ in (10) can be negative or positive, which is difficult to handle in SC. Instead, by splitting (10), the weight update can be realized by consecutively applying

$$w_j \leftarrow w_j + 2\alpha t_k \frac{\partial y_k}{\partial w_j}, \tag{11}$$

$$w_j \leftarrow w_j - 2\alpha y_k \frac{\partial y_k}{\partial w_j}. \tag{12}$$

For the SC implementation of (11) and (12), using a fixed SC sequence representing the constant 2α renders training of the weights less effective because only the bits at positions where the SC sequence of 2α is 1 are updated, while the other bits remain intact. Repeated regeneration of the SC sequence representing 2α also imposes extra latency. To address this, a linear feedback shift register (LFSR) [23] is concurrently utilized to probabilistically shuffle the positions of the 1s in the SC sequence of 2α (e.g., 1000, 0100, 0010, and 0001 all represent 0.25). Furthermore, *AND* gates are employed for multiplication as in conventional SC.

Instead of employing conventional SC arithmetic circuits for addition and subtraction followed by clipping, a novel customized design is proposed. If A and B are two stochastic sequences encoding numbers a and b, our proposal is to utilize $A \vee B$ and $A \wedge \neg B$ to respectively emulate $clip(a+b)$ and $clip(a-b)$. The rationale is that $clip(a+b)$ is larger than (or at least equal to) both a and b, which can be emulated by $A \vee B$. Likewise, $clip(a-b)$ is smaller than (or at most equal to) a and decreases as b increases, which can be emulated by $A \wedge \neg B$. These operations match the addition in Boolean algebra and the subtraction in set theory, respectively.

Employing these novel operators, Fig. 3 depicts the proposed circuit design for a 1-bit slice. When processing each sample, the select (*sel*) input of the multiplexer takes both values to perform (11) and (12) successively.

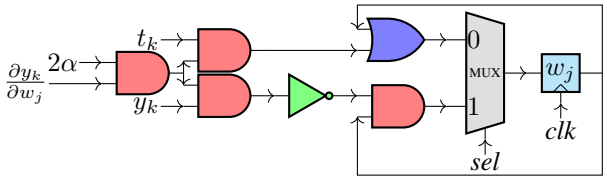

Fig. 3: A 1-bit slice of the training circuit with a flip-flop (FF) to store the corresponding weight, a multiplexer (MUX) to schedule the two updates, AND, NOT, and OR gates for multiplications, clipped addition, and subtraction.

IV. Experimental Results

Iris, Wine, and Breast Cancer datasets are considered to demonstrate the efficiency of the proposed methodology. The Iris dataset comprises 3 classes, each with 50 samples, described by 4 attributes, where each class represents a type of iris plant. The Wine dataset consists of 59, 71, and 48 samples of 3 different types of wine, captured by 13 variables. The Breast Cancer dataset splits into 2 categories, 212 malignant and 357 benign samples, expressed by 9 features.

As for the distance function used in the FCM algorithm, the Canberra distance is exploited for the Iris and Breast Cancer datasets, while for the Wine dataset, the standardized Euclidean distance is exploited.

The proposed SC design is compared against the FP and FxP implementations. The overall classification accuracy is bounded by the clustering accuracy of the FCM algorithm. That is, if the underlying FNN performs perfectly, the overall classification accuracy is determined by the clustering accuracy of the FCM algorithm. Based on observations, using half-precision FP format (FP16), Q8.8 (8 bits for the integer part and 8 bits for the fractional part) FxP format, and 16-bit SC representation (SC16), the underlying FNN can match the clustering accuracy of the FCM algorithm. Lower precisions for both the FxP and SC formats cause accuracy degradation. It is worth noting that the SC design operates in parallel, not sequentially. By choosing the SOP-inspired architecture, while the number of *OR* neurons must equal the number of class labels, the number of *AND* neurons is a tunable hyperparameter. To match the clustering accuracy of the FCM algorithm, twice the number of *OR* neurons is required for *AND* neurons in the FP and FxP implementations, whereas the same number suffices in the SC implementation. The learning rate, more precisely 2α in (10), is 64^{-1} in the FP and FxP implementations, and 16^{-1} in the SC implementation. Larger learning rates destabilize training and degrade the accuracy. The feedback polynomial of the LFSR employed to generate the learning rate in the SC implementation is x^4+x^3+1, with an initial state of 1. Regarding the number of training epochs, it is observed that the FP and FxP implementations require 8 epochs, while 1 epoch suffices for the SC implementation, possibly due to the larger learning rate.

In the experiments, the xaau15p-ffvb676-1-i field-programmable gate array (FPGA) from the XA Artix UltraScale+ family and the Vivado synthesis tool are used. Table I reports the yielded accuracy averaged over 10 runs, utilization, and performance summaries for each dataset.

TABLE I: Experimental Results.

Dataset	Task Accuracy	Format	LUTs	FFs	DSPs	N_{Cycles}	T_{Min} (ns)	Power (W)
Iris	Training 94.64%	FP16	37,614	83,226	543	994,736	**3**	3.400
		Q8.8	3,803	2,600	126	43,912	9	0.324
		SC16	**1,769**	**1,454**	**0**	**2,912**	**3**	**0.267**
	Inference 96.84%	FP16	8,374	19,670	129	7,676	**2**	1.303
		Q8.8	783	871	18	**418**	8	0.253
		SC16	**440**	**552**	**0**	912	3	**0.251**
Wine	Training 96.39%	FP16	38,133	83,349	543	1,181,216	**3**	3.403
		Q8.8	3,887	2,730	126	52,144	9	0.324
		SC16	**1,922**	**1,587**	**0**	**3,458**	**3**	**0.270**
	Inference 97.11%	FP16	8,434	19,616	129	9,090	**2**	1.360
		Q8.8	812	893	18	**495**	8	0.255
		SC16	**498**	**574**	**0**	1,080	3	**0.251**
Breast Cancer	Training 91.15%	FP16	18,332	39,209	246	1,898,432	**3**	1.733
		Q8.8	2,851	2,678	60	78,392	9	0.287
		SC16	**1,814**	**2,153**	**0**	**5,964**	**3**	**0.284**
	Inference 92.80%	FP16	4,412	9,785	62	20,592	**2**	0.800
		Q8.8	694	686	12	**1,001**	8	0.252
		SC16	**556**	**539**	**0**	3,003	3	**0.249**

LUTs: look-up tables; FFs: flip-flops; DSPs: digital signal processors.
N_{Cycles}: number of clock cycles; T_{Min}: minimum clock period.
Q8.8: 8-bit integer, 8-bit fractional FxP format.

The experimental results showcase the superior efficiency and performance of the proposed SC design against the FP and FxP implementations, in every aspect except the execution time in only one case. To be concise, the following compares the SC implementation against the best of the FP and FxP implementations, reporting the minimum and maximum gain for each metric across all datasets. Firstly, the number of look-up tables (LUTs) is reduced by 36% to 53% in training, and 19% to 43% in inference. Secondly, the number of flip-flops (FFs) is trimmed by 19% to 44% in training, and 21% to 36% in inference. Notably, the FP and FxP implementations require a remarkable number of digital signal processors (DSPs), whereas the proposed SC design is DSP-free. In terms of execution time, calculated as the number of clock cycles multiplied by the minimum clock period ($N_{Cycles} \times T_{Min}$), training is accelerated by more than 97% for all the datasets. However, for inference, while more than 18% speed-up is achieved for the Iris and Wine datasets, for the Breast Cancer dataset of 2 classes, there is about 13% degradation, which is the only drawback observed. This implies that the number of class labels must exceed 2 to offset the 16 extra clock cycles imposed by the SC-to-binary conversion required in inference. Lastly, power consumption is decreased by 1% to 17% in training, and about 1% in inference. If the gain in power consumption is considered negligible, energy consumption, calculated as execution time multiplied by power consumption, is effectively proportional to execution time.

V. Conclusions

In this letter, SCFNNs are introduced as hardware-friendly FNNs, suitable for resource-restricted critical applications. As the confinement of fuzzy arithmetic within the unit interval eliminates the need for range normalization, resolution remains undiminished, yielding relatively short SC sequences.

Since the overall classification accuracy is bounded by the clustering accuracy of the FCM algorithm, the scale of the datasets that the entire system can handle is dictated by the FCM algorithm. To push this boundary and tackle large real-world datasets, future research can employ preprocessing such as dimensionality reduction techniques, rather than inputting the FCM algorithm with unprocessed raw data. As the FCM algorithm is not the primary focus of this research, the datasets selected for the experiments are small-scale to serve as a proof-of-concept validation of the proposed SCFNNs.

Acknowledgments

This work 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).

References

[1] K. Hirota and W. Pedrycz, "Fuzzy logic neural networks: Design and computations," in *[Proceedings] 1991 IEEE International Joint Conference on Neural Networks*, pp. 152–157, IEEE, 1991.

[2] L. A. Zadeh, "Fuzzy sets," *Information and control*, vol. 8, no. 3, pp. 338–353, 1965.

[3] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, pp. 115–133, 1943.

[4] A. Hart and J. Wyatt, "Evaluating black-boxes as medical decision aids: issues arising from a study of neural networks," *Medical informatics*, vol. 15, no. 3, pp. 229–236, 1990.

[5] K. Ashton *et al.*, "That 'internet of things' thing," *RFID journal*, vol. 22, no. 7, pp. 97–114, 2009.

[6] B. R. Gaines, "Stochastic computing," in *Proceedings of the April 18-20, 1967, spring joint computer conference*, pp. 149–156, 1967.

[7] G. E. Hinton and D. Van Camp, "Keeping the neural networks simple by minimizing the description length of the weights," in *Proceedings of the sixth annual conference on Computational learning theory*, pp. 5–13, 1993.

[8] W. Kahan, "Ieee standard 754 for binary floating-point arithmetic," *Lecture Notes on the Status of IEEE*, vol. 754, no. 94720-1776, p. 11, 1996.

[9] R. A. Fisher, "Iris." UCI Machine Learning Repository, 1936. DOI: https://doi.org/10.24432/C56C76.

[10] S. Aeberhard and M. Forina, "Wine." UCI Machine Learning Repository, 1992. DOI: https://doi.org/10.24432/C5PC7J.

[11] W. Wolberg, O. Mangasarian, N. Street, and W. Street, "Breast Cancer Wisconsin (Diagnostic)." UCI Machine Learning Repository, 1993. DOI: https://doi.org/10.24432/C5DW2B.

[12] M. M. Gupta and J. Qi, "Theory of t-norms and fuzzy inference methods," *Fuzzy sets and systems*, vol. 40, no. 3, pp. 431–450, 1991.

[13] C. E. Shannon, "A symbolic analysis of relay and switching circuits," *Electrical Engineering*, vol. 57, no. 12, pp. 713–723, 1938.

[14] R. Hecht-Nielsen, "Kolmogorov's mapping neural network existence theorem," in *Proceedings of the international conference on Neural Networks*, vol. 3, pp. 11–14, IEEE press New York, NY, USA, 1987.

[15] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain.," *Psychological review*, vol. 65, no. 6, p. 386, 1958.

[16] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[17] B. D. Brown and H. C. Card, "Stochastic neural computation. i. computational elements," *IEEE Transactions on computers*, vol. 50, no. 9, pp. 891–905, 2001.

[18] L. A. Zadeh, "Outline of a new approach to the analysis of complex systems and decision processes," *IEEE Transactions on systems, Man, and Cybernetics*, no. 1, pp. 28–44, 1973.

[19] J. C. Bezdek, R. Ehrlich, and W. Full, "Fcm: The fuzzy c-means clustering algorithm," *Computers & geosciences*, vol. 10, no. 2-3, pp. 191–203, 1984.

[20] "Module: cluster — skfuzzy v0.3 docs." https://scikit-fuzzy.readthedocs.io/en/latest/api/skfuzzy.cluster.html. Accessed: 2026-02-04.

[21] Á. Odry, V. L. Tadic, and P. Odry, "A stochastic logic-based fuzzy logic controller: First experimental results of a novel architecture," *IEEE Access*, vol. 9, pp. 29895–29920, 2021.

[22] H. Robbins and S. Monro, "A stochastic approximation method," *The annals of mathematical statistics*, pp. 400–407, 1951.

[23] W. W. Peterson and D. T. Brown, "Cyclic codes for error detection," *Proceedings of the IRE*, vol. 49, no. 1, pp. 228–235, 1961.