Energy-Efficient Stochastic Computing for Convolutional Neural Networks by Using Kernel-wise Parallelism

Zaipeng Xie*, Chenyu Yuan†, Likun Li†, and Jiahao Wu†
*Key Laboratory of Water Big Data Technology of Ministry of Water Resources, Hohai University, Nanjing, China
†College of Computer and Information, Hohai University, Nanjing, China
Email: {zaipengxie, chenyu_yuan, lilikun, jiahaowu}@hhu.edu.cn

***Abstract*—Stochastic computing (SC) is a low-cost computation paradigm that can replace conventional binary arithmetic to provide a low hardware footprint with high scalability. However, since the SC bitstream length grows with the precision of the represented data, regardless of its lower power consumption, the convolutional SC-based neural networks may not be efficient in hardware area and energy. This work proposes a novel SC accelerator, PSC-Conv, to implement the convolutional layer using a new binary-interfaced stochastic computing architecture. PSC-Conv exploits kernel-wise parallelism in CNNs, reducing hardware footprint and energy consumption. Experimental results show that the proposed implementation excels among several state-of-the-art SC-based implementations regarding area and power efficiency. We also compared the implementations of three modern CNNs, including LeNet-5, MobileNet, and ResNet-50. Experimental results demonstrate that, on average, PSC-Conv can achieve $5.02\times$ speedup and 87.9% energy reduction compared with the binary implementation.**

***Index Terms*—Stochastic computing, Hardware accelerator, Convolutional Neural Networks, Kernel-wise parallelism**

I. Introduction

Recent research [1] on Neural Networks (NN) has achieved unprecedented success in many machine learning applications, where high accuracy can be achieved at the cost of a substantial computation, necessitating approaches beyond traditional computing paradigms to improve operational efficiency. Stochastic Computing (SC) [2] introduces robustness against randomness in number representation that can process data bitwise with simple circuits. A stochastic bitstream of length m represents numbers with a precision of $1/m$. Several SC-based hardware accelerators [3]–[6] have been developed for convolutional neural networks (CNNs). However, existing SC methods [7] require long streams that may negatively impact the computation latency. Binary-Interfaced Stochastic Computing (BISC) [8] was developed to build matrix-vector multipliers that can significantly reduce time consumption and energy overhead by using a special stochastic number generator (SNG). However, CNN implementations using BISC multipliers [9] may still incur extra area and energy overhead when the number of convolutional kernels grows.

The corresponding author is Zaipeng Xie. This work is supported by The Belt and Road Special Foundation of the State Key Laboratory of Hydrology-Water Resources and Hydraulic Engineering under Grant No. 2021490811.

This study proposes a novel SC-based architecture, PSC-Conv, to implement the CNN inference. PSC-Conv is based on the BISC structure, but it exploits the kernel-wise parallelism in CNNs to provide a reduced hardware footprint and energy consumption. We evaluated the performance of PSC-Conv by implementing 256 multiply–accumulate (MAC) units. Compared with the state-of-the-art SC-based approaches, PSC-Conv can achieve an average of 29.4% and 40.2% reduction in area and power. We also compared the implementations of three modern CNNs, including LeNet-5, MobileNet, and ResNet-50. Experimental results show that, on average, PSC-Conv can achieve $5.02\times$ speedup and 87.9% energy reduction compared with the conventional binary counterpart.

II. Related Work

Hardware accelerators for CNNs have become a research hotspot [10] by developing new architectures on resource-limited platforms with tight energy and hardware constraints. SC is a low-cost alternative [2] to conventional binary arithmetic for many computation-intensive tasks. SC-based CNN implementations [3]–[6], [8], [9], [11]–[15] have attracted much attention from the scientific community. Computation with SC requires multiple cycles equal to the number of bits in the stochastic bitstream. Since the length of the bitstream grows with the precision of the data being represented, regardless of its lower power consumption, the SC-based CNN accelerator may consume more energy than a conventional binary processing element. Since multiplication operations are relatively slow on general-purpose hardware and require significant resource investment, exploring efficient implementations toward architectural optimization is desired.

Ji et al. [11] proposed a fully parallel and scalable CNNs architecture to achieve a $2\times$ improvement in energy compared with the conventional fixed-point binary implementation. Hojabr et al. [12] proposed a differential MAC unit for SC-based CNNs, and it offers $1.2\times$ speedup and $2.7\times$ energy saving compared to a binary implementation. Sim et al. [9] proposed a binary-interfaced stochastic computing matrix-vector multiplier (BISC-MVM) that can reduce the area-delay product and energy consumption without significant degradation in neural network accuracy. However, as the number of convolutional kernels grows, these approaches may still incur additional area and energy overhead.

III. Proposed Kernel-wise Parallel Architecture

The BISC-MVM structure [9] was recently proposed to improve the matrix multiplication in the computation speed and energy efficiency. However, the current implementation of BISC-MVM may still incur a significant area overhead because a considerable amount of kernels are usually used in modern CNNs. For instance, in a ResNet-50 implementation [16], the convolutional kernels in a hidden layer can be categorized into 2048 groups, resulting in a total of 1,048,576 kernel matrices. Therefore, our work proposes a novel parallel architecture, PSC-Conv, to improve the area and energy efficiency by exploiting kernel-wise parallelism in CNNs.

A. BISC Architecture

BISC is an energy-efficient architecture [13] to implement multipliers that can significantly reduce the clock cycles taken in the stochastic multiplication. A BISC module consists of a stochastic number generator (SNG), counters, and an XOR gate. Fig. 1 gives an example of BISC, where the inputs are feature x and weight w, and the output is the product $x \cdot w$.

The SNG can produce stochastic bitstreams by following a deterministic bit shuffling pattern. Each SNG is equipped with a finite state machine (FSM) and a multiplexer (MUX) to implement stochastic multiplication. The MUX takes in x as the input, and its select (SEL) port is driven by an FSM. The FSM is designed so that each bit in x can be selected as the output of the MUX in a pre-defined order. Therefore, a stochastic representation of x is generated and XORed with the sign of w. Meanwhile, the absolute value of w is set as the value of the down counter (DC). The up-down counter (UDC) takes the output of the XOR gate and keeps updating for w clock cycles. The DC outputs a stop signal to the UDC after w clock cycles and produces the binary result $d = x \cdot w$.

We consider the example in Fig. 1 to see how it works. We let $w = 6/8$ and $x = -4/8$. Hence the output of the MUX is a binary bitstream γ = 0100_0100, which is then sent to the XOR gate. Since sign(w) is 0, the output of the XOR gate is a binary bitstream $\tau = \text{XOR}(\gamma, sign(w))$ = 0100_0100. The UDC takes in τ as a binary bitstream and outputs a 4-bit binary number, d. If $\tau[j] = 1$ and j is the time index, then d increases by 1; otherwise, d decreases by 1. Meanwhile, because the DC is initialized to $|w| = 2'$b0110, DC sends out a stop signal to the UDC after 6 clock cycles. The result of $w \cdot x$ is $d = -2/8$. It is worth noting that d deviated from the actual result $-3/8$ due to the use of a limited number of bits. However, since recent studies [17], [18] indicate that convolutional neural networks can exhibit some robustness towards quantization errors, the error can be negligible [9], [12], [14] when the bitwidth increases.

BISC-MVM [9] is a parallelized BISC structure that implements multiple BISC multipliers. This structure calculates $(w \cdot X)$, where $X = [x^{(0)}, x^{(1)}, \cdots, x^{(t-1)}]$ is a vector of t elements. The BISC multipliers share part of the control logic,

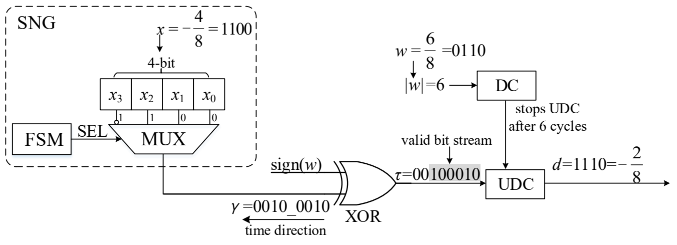


Fig. 1: Schematic of the BISC multiplier with an example.

including the FSM and the DC, as shown in Fig. 2. However, this architecture may still incur area and energy overhead in the presence of a large number of convolutional kernels.

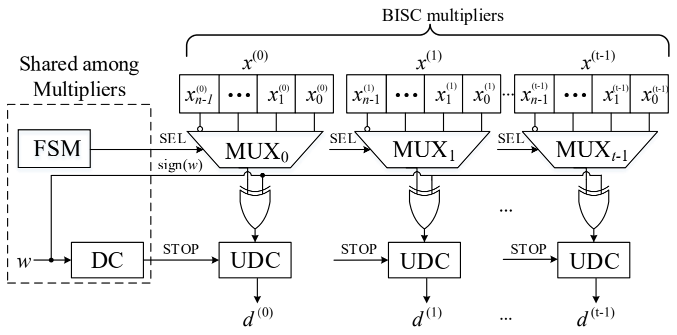


Fig. 2: Schematic of the BISC-MVM Structure.

Given a weight $\tilde{w}$ and assume $|\tilde{w}| < |w|$, we observe that the product of $\tilde{w} \cdot X$ can be produced as an intermediate result when calculating $w \cdot X$. This is because each element of X is turned into a stochastic bitstream during the BISC computation, and the product $\tilde{w} \cdot X$ is produced after $|\tilde{w}|$ cycles. Hence, based on this fact, we propose the kernel-wise optimization method, PSC-Conv, for the CNN implementation.

B. Proposed PSC-Conv Architecture

PSC-Conv is a parallel structure for implementing the convolutional layer in CNNs. In the convolutional layer, the feature extraction is performed by sliding a kernel matrix $K^{(i)} \in \mathbb{R}^{P \times Q}$ (i is the index of the kernel, P and Q denotes the number of rows and columns of the kernel matrix) over the feature map $X \in \mathbb{R}^{R \times C}$ (R and C denotes the number of rows and columns of the feature map). The convolution operation [19] can be considered as multiple MAC operations, and it can be described by

$$Y^{(i)}_{r,c} = \sum_{p=0}^{P-1} \sum_{q=0}^{Q-1} K^{(i)}_{p,q} \cdot X\left(r + p - \left\lfloor \frac{P}{2} \right\rfloor, c + q - \left\lfloor \frac{Q}{2} \right\rfloor\right), \quad (1)$$

where Y is the resulting matrix, (r, c) is the index of the resulting matrix elements, and (p, q) is the index of the weights in each kernel. The kernel matrix (usually smaller in size than the input feature map) is first multiplied with a $P \times Q$ sized block of X, and the results are accumulated by sliding across the X matrix. The convolution operation in (1) can be expanded as a summation of multiple sub-equations, and each sub-equation has the following expression:

$$\hat{Y}_{r,c} = \left[K^{(0)}_{p,q}, K^{(1)}_{p,q}, \cdots, K^{(\lambda-1)}_{p,q}\right] \cdot x, \quad (2)$$

where λ denotes the number of kernel groups in a convolutional layer, and x is a scalar element.

Hence, our PSC-Conv design only requires one BISC multiplier as shown in Fig. 3 to implement (2). PSC-Conv takes in $\max_i\{K_{p,q}^{(i)}\}$ and x as the inputs, and the result of each element in $\hat{Y}_{r,c}$ can be produced as the intermediate results. It's noted that PSC-Conv requires some extra control logic to coordinate the correct timing for collecting the results of (2).

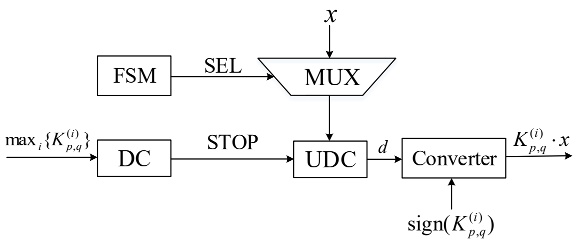


Fig. 3: Schematic of a single cell in PSC-Conv architecture.

In Fig. 3, the converter handles the signed arithmetic. It passes the result d directly to its output if the sign of $K_{p,q}^{(i)}$ is positive and outputs the two's complement of d if otherwise. Therefore, compared with the BISC-MVM structure, PSC-Conv can significantly reduce the area overhead and improve energy performance when implementing the vector-scalar multiplication in (2).

We can further improve the PSC-Conv implementation by considering the multiplication of two vectors given by

$$\left[K_{p,q}^{(0)}, K_{p,q}^{(1)}, \cdots, K_{p,q}^{(\lambda-1)}\right]\cdot X, \tag{3}$$

where $X = [X(p,q), \cdots, X(p+(R-P+1), q+(C-Q+1))]$ ($p \in [0, P-1]$ and $q \in [0, Q-1]$). We propose a parallel PSC-Conv structure, as shown in Fig. 4, that consists of size(X) BISC multipliers and each multiplier is designated for the computation in the form of (2) with the corresponding element in X. The FSM and DC in this parallel PSC-Conv structure is shared among multipliers in PSC-Conv.

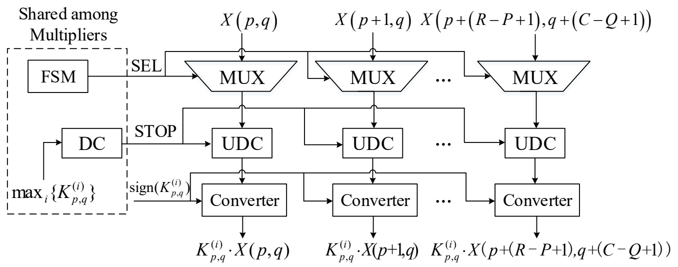
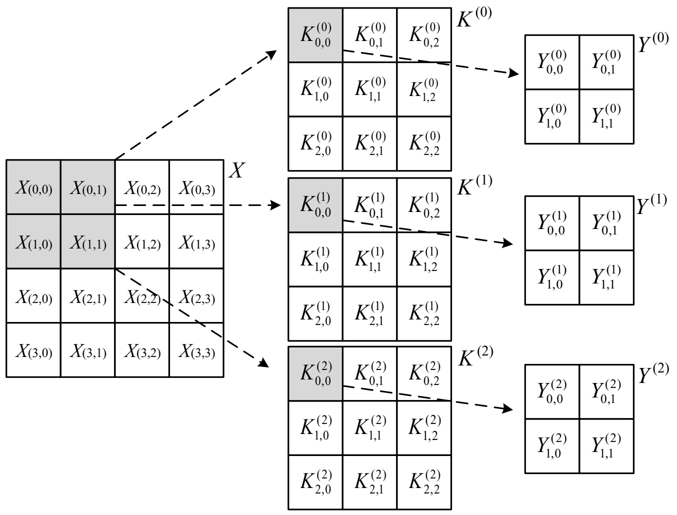


Fig. 4: Schematic of the PSC-Conv architecture.

Consider the example in Fig. 5 that computes the product for $K \cdot X$, where $K = [K_{0,0}^{(0)}, K_{0,0}^{(1)}, K_{0,0}^{(2)}]$ and $X = [X(0,0), X(0,1), X(1,0), X(1,1)]$. The overall PSC-Conv implementation only requires four BISC multipliers with some extra logic for the controller. In contrast, the BISC-MVM implementation requires twelve BISC multipliers, and the control logic is about the same size. Our PSC-Conv architecture can significantly reduce the overall area overhead, especially when the number of kernels is sizable. Consider a convolutional layer consisting of m kernel groups, and each group has n kernel matrices of size $P \times Q$. If the feature map is a $R \times C$ matrix, our PSC-Conv architecture requires a minimum of $(R-P+1) \times (C-Q+1)$ BISC multipliers for implementing a fully-parallel structure for such a convolutional layer. For example, consider a hidden layers with 2048 groups of convolutional kernels and each consists of 512 1×1 kernel matrices. If the input feature map is a 3×3 matrix, our PSC-Conv can fully parallelize the convolutional operation using 9 BISC multipliers.



$K^{(0)}$
$K^{(1)}$
$K^{(2)}$
X
$Y^{(0)}$
$Y^{(1)}$
$Y^{(2)}$
$X(0,0)$ $X(0,1)$ $X(0,2)$ $X(0,3)$
$X(1,0)$ $X(1,1)$ $X(1,2)$ $X(1,3)$
$X(2,0)$ $X(2,1)$ $X(2,2)$ $X(2,3)$
$X(3,0)$ $X(3,1)$ $X(3,2)$ $X(3,3)$
$K_{0,0}^{(i)}$ $K_{0,1}^{(i)}$ $K_{0,2}^{(i)}$
$K_{1,0}^{(i)}$ $K_{1,1}^{(i)}$ $K_{1,2}^{(i)}$
$K_{2,0}^{(i)}$ $K_{2,1}^{(i)}$ $K_{2,2}^{(i)}$
$Y_{0,0}^{(i)}$ $Y_{0,1}^{(i)}$
$Y_{1,0}^{(i)}$ $Y_{1,1}^{(i)}$

Fig. 5: Diagram of a convolutional layer with three kernels.

IV. Experimental Results

We evaluate the PSC-Conv architecture by implementing three modern CNNs: LeNet-5 [20], MobileNet [21], and ResNet-50 [16], which are evaluated using the MNIST, CIFAR-10, and ImageNet datasets. The proposed PSC-Conv is also compared to five state-of-the-art SC implementations regarding the hardware cost and power performance. The Synopsys Design Compiler G-2012.06-SP2 is used to synthesize the implementations with a 45nm gate library.

A. Network Accuracy

The accuracy of the LeNet-5, MobileNet, and ResNet-50 is evaluated by varying the bitwidth of the multiplier's inputs and outputs. Four implementations are compared against our proposed PSC-Conv as shown in Fig. 6. The floating-point design performs the best among all implementations, while the conventional SC method has the worst accuracy. The PSC-Conv method is on par with the BISC-MVM and the fixed-point binary when the bitwidth is 12 bits.

B. Hardware Cost and Performance Evaluation

We evaluate the hardware cost of 256 MAC implementations of various SC-based works, including DPS [14], SkippyNN [12], BISC-MVM [9], PHSB-NN [15], and FPSC [6].

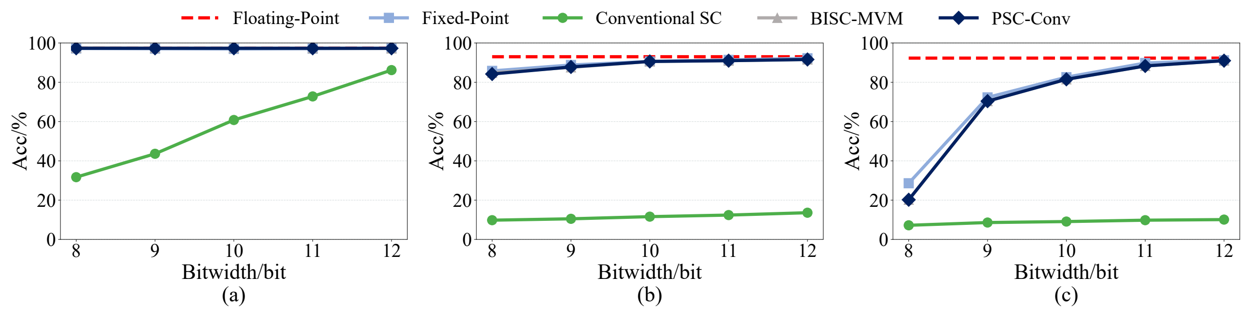


Fig. 6: Inference accuracy on (a) LeNet-5, (b) MobileNet and (c) ResNet-50.

TABLE I: Comparison With Related SC-Based Works

Work	DPS [14]	SkippyNN [12]	BISC-MVM [9]	PHSB-NN [15]	FPSC [6]	**Proposed PSC-Conv**	Binary
DataSet	ImageNet	ImageNet	CIFAR-10	MNIST	CIFAR-10	**ImageNet**	ImageNet
Accuracy/ %	82.47 (top-5)	90 (top-5)	82 (top-1)	98.9 (top-1)	81 (top-1)	**91.0 (top-5)**	92.3 (top-5)
Scope	256 MAC	256 MAC	256 MAC	256 MAC	256 MAC	**256** MAC	256 MAC
Tech.	45nm	45nm	45nm	14nm	40nm	**45nm**	45nm
Freq.(Mhz)	1000	1064	1000	400	\	**1000**	1000
Area(mm^2)	0.0971	0.0844	0.0803	0.0826	0.133	**0.0655**	0.1736
Power(mW)	\	16.452	16.177	32.873	25.52	**12.45**	106.69

TABLE II: Performance Comparisons of PSC-Conv, BISC-MVM and the binary implementation

CNN	**LeNet-5 (12-bit)**		**MobileNet (12-bit)**		**ResNet-50 (12-bit)**	
Compared with	Binary	BISC-MVM	Binary	BISC-MVM	Binary	BISC-MVM
Speedup	4.38×	1.00×	4.90×	1.00×	5.80×	1.00×
Energy Reducion	85.6%	15.4%	90.1%	18.9%	88.2%	16.2%

The results are summarized in TABLE I. Our proposed PSC-Conv can attain a top-5 accuracy of 91.0%, which is on par with that of its binary counterpart. The PSC-Conv has the best hardware area and power performance among all implementations. Compared with five state-of-the-art SC implementations, PSC-Conv can, on average, reduce the hardware area by 29.4% and improve the power performance by 40.2%. Although PHSB-NN and FPSC utilize smaller feature sizes, their results are not as favorable as those achieved by PSC-Conv using the 45nm feature size. Compared with the binary implementation, PSC-Conv reduces the hardware area and power by 62.3% and 88.3%, respectively. The parallel design in our PSC-Conv effectively leverages kernel-wise parallelism, leading to a significant improvement in the MAC implementation.

C. Implementation Efficiency of PSC-Conv

To better evaluate the efficiency, we implement the entire neural network hardware for the LeNet-5, MobileNet, and ResNet-50. TABLE II summarizes PSC-Conv's speedup and reduction in energy compared with the binary and the BISC-MVM design. Speedup measures the average improvement in the inference time. The energy reduction measures the decrease in energy consumption, which is the product of the inference time and the power. We observe that, for the implementations of LeNet-5, PSC-Conv can achieve 4.38× speedup and 85.6% energy reduction compared with the binary implementation. Similar results can be concluded for the PSC-Conv implementation of MobileNet and ResNet-50. Compared to a BISC-MVM implementation, the PSC-Conv implementations of LeNet-5, MobileNet, and ResNet-50 can achieve an average 16.8% reduction in energy consumption. However, due to the use of BISC as the core structure in our PSC-Conv, it does not exhibit substantial improvement in speed compared to BISC-MVM. In summary, our PSC-Conv excels among all implementations for the LeNet-5, MobileNet, and ResNet-50 regarding energy performance.

V. Conclusion

This work proposes a novel SC accelerator for CNNs, called PSC-Conv. PSC-Conv exploits the kernel-wise parallelism in each convolutional layer. We demonstrate that our design can effectively reduce the hardware area and energy consumption. Experimental results show that PSC-Conv outperforms the binary design and five state-of-the-art SC-based implementations in terms of area and power performance. The proposed PSC-Conv can reduce the area and power by an average of 29.4% and 40.2% for a 256 MAC array. Compared with the binary implementation, the proposed PSC-Conv can reduce 62.3% area cost and 88.3% power. We also evaluate the hardware implementations for the LeNet-5, MobileNet, and ResNet-50. Experimental results demonstrate that the proposed implementation is an energy-efficient design that can provide 5.02× speedup and 87.9% energy reduction, on average, compared with the conventional binary implementation.

References

[1] X. Bai, X. Wang, X. Liu *et al.*, “Explainable deep learning for efficient and robust pattern recognition: A survey of recent developments,” *Pattern Recognition*, vol. 120, p. 108102, 2021.

[2] Y. Liu *et al.*, “A survey of stochastic computing neural networks for machine learning applications,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 7, pp. 2809–2824, 2020.

[3] A. Ren, Z. Li, C. Ding *et al.*, “SC-DCNN: Highly-scalable deep convolutional neural network using stochastic computing,” *ACM SIGPLAN Notices*, vol. 52, no. 4, pp. 405–418, 2017.

[4] Z. Li, J. Li, A. Ren *et al.*, “HEIF: Highly efficient stochastic computing-based inference framework for deep neural networks,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, no. 8, pp. 1543–1556, 2018.

[5] B. Li, Y. Qin, B. Yuan, and D. J. Lilja, “Neural network classifiers using stochastic computing with a hardware-oriented approximate activation function,” in *2017 IEEE International Conference on Computer Design (ICCD)*. IEEE, 2017, pp. 97–104.

[6] C. F. Frasser, P. Linares-Serrano *et al.*, “Fully parallel stochastic computing hardware implementation of convolutional neural networks for edge computing applications,” *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[7] L. Sousa, “Nonconventional computer arithmetic circuits, systems and applications,” *IEEE Circuits and Systems Magazine*, vol. 21, no. 1, pp. 6–40, 2021.

[8] H. Sim, D. Nguyen *et al.*, “Scalable stochastic-computing accelerator for convolutional neural networks,” in *22nd Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 2017, pp. 696–701.

[9] H. Sim and J. Lee, “Bitstream-based neural network for scalable, efficient, and accurate deep learning hardware,” *Frontiers in Neuroscience*, vol. 14, p. 1198, 2020.

[10] S. Mittal, “A survey of FPGA-based accelerators for convolutional neural networks,” *Neural computing and applications*, vol. 32, no. 4, pp. 1109–1139, 2020.

[11] J. Li, A. Ren *et al.*, “Towards acceleration of deep convolutional neural networks using stochastic computing,” in *22nd Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 2017, pp. 115–120.

[12] R. Hojabr, K. Givaki, S. R. Tayaranian *et al.*, “SkippyNN: An embedded stochastic-computing accelerator for convolutional neural networks,” in *56th Design Automation Conference (DAC)*. IEEE, 2019, pp. 1–6.

[13] H. Sim and ohters, “A new stochastic computing multiplier with application to deep convolutional neural networks,” in *54th Design Automation Conference (DAC)*. IEEE, 2017, pp. 1–6.

[14] H. Sim *et al.*, “DPS: Dynamic precision scaling for stochastic computing-based deep neural networks,” in *55th Design Automation Conference (DAC)*. IEEE, 2018, pp. 1–6.

[15] Y. Zhang, R. Wang, X. Zhang, Y. Wang, and R. Huang, “Parallel hybrid stochastic-binary-based neural network accelerators,” *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 67, no. 12, pp. 3387–3391, 2020.

[16] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[17] N. Cheney, M. Schrimpf, and G. Kreiman, “On the robustness of convolutional neural networks to internal architecture and weight perturbations,” *arXiv preprint arXiv:1703.08245*, 2017.

[18] A. Boopathy, T.-W. Weng, P.-Y. Chen *et al.*, “CNN-Cert: An efficient framework for certifying robustness of convolutional neural networks,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 3240–3247.

[19] C. Cheng and K. K. Parhi, “Fast 2D convolution algorithms for convolutional neural networks,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 67, no. 5, pp. 1678–1691, 2020.

[20] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[21] A. G. Howard, M. Zhu, B. Chen *et al.*, “MobileNets: Efficient convolutional neural networks for mobile vision applications,” *arXiv preprint arXiv:1704.04861*, 2017.