An Energy-Efficient Stochastic Computational Deep Belief Network

Yidong Liu
ECE Department
University of Alberta
Edmonton, AB, Canada
Email: yidong1@ualberta.ca

Yanzhi Wang
EECS Department
Syracuse University
Syracuse, NY, USA
Email: ywang393@syr.edu

Fabrizio Lombardi
ECE Department
Northeastern University
Boston, MA, USA
Email: lombardi@ece.neu.edu

Jie Han
ECE Department
University of Alberta
Edmonton, AB, Canada
Email: jhan8@ualberta.ca

***Abstract*—Deep neural networks (DNNs) are effective machine learning models to solve a large class of recognition problems, including the classification of nonlinearly separable patterns. The applications of DNNs are, however, limited by the large size and high energy consumption of the networks. Recently, stochastic computation (SC) has been considered to implement DNNs to reduce the hardware cost. However, it requires a large number of random number generators (RNGs) that lower the energy efficiency of the network. To overcome these limitations, we propose the design of an energy-efficient deep belief network (DBN) based on stochastic computation. An approximate SC activation unit (A-SCAU) is designed to implement different types of activation functions in the neurons. The A-SCAU is immune to signal correlations, so the RNGs can be shared among all neurons in the same layer with no accuracy loss. The area and energy of the proposed design are 5.27% and 3.31% (or 26.55% and 29.89%) of a 32-bit floating-point (or an 8-bit fixed-point) implementation. It is shown that the proposed SC-DBN design achieves a higher classification accuracy compared to the fixed-point implementation. The accuracy is only lower by 0.12% than the floating-point design at a similar computation speed, but with a significantly lower energy consumption.**

***Index Terms*—stochastic computing, deep belief network, rectifier linear unit, cognitive computing.**

I. Introduction

As a type of deep neural networks (DNNs), a deep belief network (DBN) substantially improves the performance of conventional artificial neural networks such as a multilayer perceptron [1]. A DBN can perform unsupervised learning and solve nonlinearly separable pattern recognition problems such as the classification of objects [2], speech [3] and hand written characters [4]. However, the size of a DBN increases rapidly with the complexity of a problem, so it inevitably results in a large hardware and a high energy consumption. Hence, it is difficult to implement a machine learning algorithm using a DBN on a resource-limited system such as a mobile device or an embedded system. It has become imperative to develop efficient hardware design for implementing a DBN at a small circuit area and low power consumption.

The recent resurgence of stochastic computing (SC) provides such an opportunity [5] [6]: an SC circuit reduces the hardware footprint of many fundamental arithmetic circuits, such as adders, subtractors [7] [8] and multipliers [9] [10]. The hyperbolic tangent ($tanh$) and exponential functions can be implemented by linear finite state machines (LFSMs) [11]. Recently, SC designs have been utilized to implement radial basis function neural networks [12], a multi-layer perceptron [13], a convolutional neural network [14], a DBN [15] and other types of DNNs [16] [17]. In these designs, the neural networks are pre-trained to perform the nonlinear classification in hardware.

In spite of the simple SC circuits, stochastic number generators (SNGs), consisting of random number generators (RNGs) and comparators, incur a large area and high power consumption [16] [17], thus reducing the energy efficiency of an SC design. Moreover, because different types of activation functions are needed for various requirements in the learning process, the performance of SC-based DNNs is limited as it is difficult to reconfigure the activation function without re-implementing the design.

In this paper, a stochastic computational DBN (SC-DBN) is proposed to overcome the above limitations. The Modified National Institute of Standards and Technology (MNIST) dataset is used for the evaluation of the proposed design. An approximate SC activation unit (A-SCAU) is proposed to implement different types of activation functions such as the sigmoid, the rectifier linear and the pure line functions. The SC-DBN achieves a smaller area, lower power and energy consumption with a similar accuracy and computation speed compared to conventional floating- and fixed-point implementations.

II. Review

A. The structure of DBNs

A DBN consists of one input layer, multiple hidden layers and one output layer. Assume that the number of neurons in layer $l-1$ and l are M and E, w_{ij}^l denotes the weight between neuron j in layer $l-1$ and neuron i in layer l. The output signal of neuron i in layer l at epoch t, $y_i^l(t)$, is given by [18]

$$y_i^l(t) = \varphi(\sum_{j=1}^{M} y_j^{l-1}(t) \cdot w_{ij}),\ i = 1, 2, ..., E, \qquad (1)$$

where $\varphi(\cdot)$ is the activation function. One of the most widely used activation function is the sigmoid function [18], defined as

978-3-9819263-0-9/DATE18/©2018 EDAA

$$\varphi(x) = \frac{1}{1 + exp(-x)}. \quad (2)$$

The rectifier linear function (ReLU) [19] is another useful activation function, defined as

$$\varphi(x) = min(1, max(0, x)). \quad (3)$$

The pure line function [18] is defined as

$$\varphi(x) = min(1, max(-1, x)). \quad (4)$$

B. Stochastic logic elements

In SC, assume that there are a 1's in a random binary bit stream with a length of b bits; the bit stream encodes the value of a/b within $[0, 1]$ in the unipolar representation or $(2a - b)/b$ within $[-1, 1]$ in the bipolar representation [5] [6]. Some fundamental computational elements can be implemented by simple circuits. For example, a bipolar multiplier is implemented by an XNOR gate and an adder is implemented by a multiplexer with the select signal encoding a probability of 0.5 [7]. Compared to conventional binary designs, the area and power consumption of these simple stochastic circuits are significantly smaller.

III. Design of the SC-DBN

In the SC-DBN, an approximate SC activation unit (A-SCAU) that is immune to signal correlations, is proposed to implement different types of activation functions.

A. Neuron design

To implement the forward propagation algorithm, a neuron requires four components in the SC-DBN: two SNG arrays for the input signals and the layer weights, a bipolar SC multiplier array implemented by XNOR gates and an A-SCAU array (Fig. 1). The SNG arrays convert the binary input signals and the layer weights to stochastic sequences. Each signal in a D-dimensional input is converted into q parallel stochastic sequences to reduce latency. In a conventional SC design, the sharing of RNGs causes correlations between the output sequences, thus reducing the computation accuracy. In the SC-DBN, however, an RNG is shared to produce stochastic sequences for the inputs and a total of q RNGs are required for a parallelization of q levels. Therefore, the number of RNGs is changed to $1/D$ of those required in a conventional design with the same level of parallelization but no sharing structure.

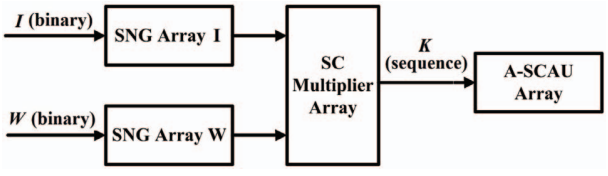

Fig. 1. Design of the neuron. $\boldsymbol{I}$ is the input signal and $\boldsymbol{W}$ is the layer weight. $\boldsymbol{K}$ is the output of the SC multiplier array.

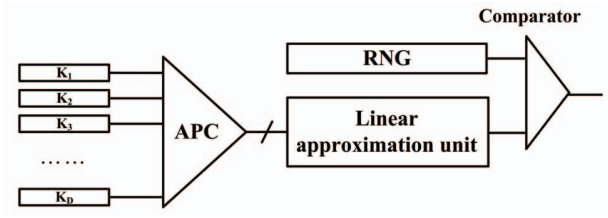

Fig. 2. Design of the A-SCAU.

B. Reconfigurability of the A-SCAU

The A-SCAU consists of an accumulative parallel counter (APC), a linear approximation unit (LAU), an RNG and a comparator (Fig. 2). The A-SCAU implements a linear function to approximate different activation functions such as (2), (3) and (4). The linear function of the A-SCAU has the generalized form of

$$\psi(x) = min(1, max(p, \frac{1}{r}x + s)), \quad (5)$$

where p, r and s are parameters that can be configured to implement different activation functions.

For the sigmoid function (2), for example, the output range is $[0, +1]$. If $x = 0$, $\psi(x) = \varphi(x) = 1/2$. Therefore, p and s are set to 0 and $1/2$, respectively, and a search is conducted to find the optimal value of r. Assume r varies in $[+2, +10]$ with a step size of 0.01, $r = 5.27$ leads to the minimum mean squared error (MSE), 6.16×10^{-4}, between $\psi(x)$ and $\varphi(x)$. The value of r is set to 4 to simplify the hardware implementation. As a result, (2) is approximated by

$$\psi(x) = min(1, max(0, \frac{1}{4}x + \frac{1}{2})). \quad (6)$$

Thus, the sigmoid function is approximated by using the configuration $p = 0, r = 4$ and $s = 1/2$ in the A-SCAU.

The rectifier linear function (3) can be directly implemented by the A-SCAU with the configuration $p = 0, r = 1$ and $s = 0$. The pure line function (4) is implemented by the configuration $p = -1$, $r = 1$ and $s = 0$.

Note that the A-SCAU implements an approximate model of the sigmoid function but accurate models of the rectifier linear and pure line functions. Fig. 3 shows the simulation results of the A-SCAU with different configurations. The range of the input signal x is set to $[-10, 10]$. With a sequence length of 4096 bits, the MSEs of the implementations are 1.1×10^{-3}, 6.1×10^{-4} and 8.9×10^{-4} for the sigmoid, ReLU and pure line functions.

C. Immune-to-correlation feature

As the core component in the A-SCAU, the LAU computes the output of the A-SCAU using a binary circuit (Fig. 4). As a result, the accuracy of the LAU is not affected by the correlations in the stochastic sequences.

In the A-SCAU, each input is implemented by a parallelization of q levels. For a D-dimensional input $\boldsymbol{K}$ encoded in

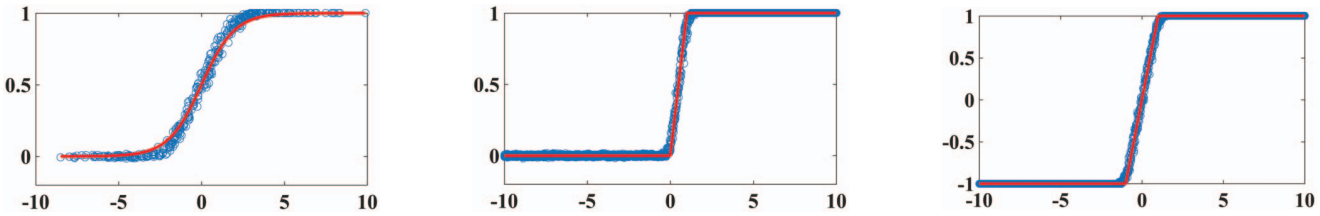

Fig. 3. The simulation results of the A-SCAU for (a) the sigmoid function, (b) the ReLU function and (c) the pure line function.

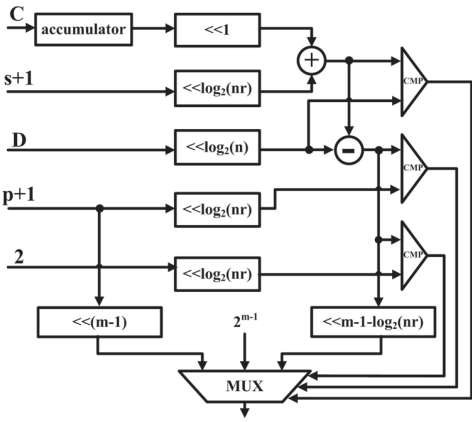

Fig. 4. Circuit design of the LAU. CMP: comparator. <<: shift register. The width of the output signal is set to m.

stochastic sequences, the APC converts every qD-bit input combination into a binary vector of m bits ($m = \lceil log_2(qD) \rceil$).

For n-bit stochastic sequences, the APC outputs n m-bit binary integers in series (denoted by **C** in Fig. 4). Then, the LAU accumulates n cycles of the output from the APC. Note that the range of $\psi(x)$ is $[0, 1]$ for the sigmoid and ReLU functions, and $[-1, 1]$ for the pure line function. The LAU converts the accumulated value into an integer for the desired $\psi(x)$, and a stochastic sequence is generated by the RNG and comparator as the output of the A-SCAU (Fig. 2). The circuit design of the LAU is shown in Fig. 4. The sequence length n and the parameter r are set to values in powers of 2, so the multipliers and dividers are simplified to shift registers.

Note that the output of the LAU is only determined by the number of 1's computed by the APC in the input sequences, regardless of the bit correlations. Therefore, the computation accuracy is not affected by the correlations due to the sharing of RNGs in the circuit.

This immune-to-correlation feature of the A-SCAU makes it possible to dramatically reduce the number of RNGs in the circuit. In the simulation, sequences for 10-dimensional input signals are generated by shared RNGs but different comparators. The parallelization is set to 16× and sequence length is set to 256 bits, so in total $256 \times 16 = 4096$ bits for each input. The simulation results of the A-SCAU and the bounded random walking based $tanh$ ($Btanh$) [16] based sigmoid functions are shown in Fig. 5. As can be seen, the

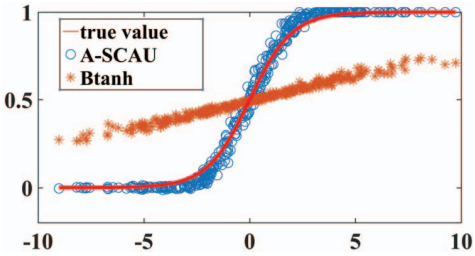

Fig. 5. Simulation results of the A-SCAU and the $Btanh$ based sigmoid functions. Both use shared RNGs.

$Btanh$ circuit does not produce correct results, whereas the A-SCAU achieves a good accuracy. The RNGs are shared among parallel A-SCAU components without loss of computation accuracy. Therefore, the RNGs can not only be shared among the signals in a single neuron, but also among all neurons in the same layer.

IV. Evaluation

A. Accuracy

The SC-DBN is evaluated on the MNIST dataset [20]. The samples are grayscale images with 28×28 pixels of 10 different hand written characters labeled as '0' to '9'. The structure of the network is optimized by the pruning algorithm [21], consisting of one input layer with 784 neurons, two hidden layers with 100 and 200 neurons, and one output layer with 10 neurons. An 8-bit fixed-point and a 32-bit floating-point implementation with the same configuration are also evaluated on the dataset.

The classification error rates of the different implementations are shown in Table I. It can be seen that the classification accuracy improves rapidly when the sequence length is under 256 bits, increasing from 89.9% (by 32 bits) to 98.9% (by 128 bits). Using 64-bit sequences (×16 parallel processes), the proposed design achieves a higher accuracy than the results in the literature [15] [16] [17] [22]. Note that most of the designs in the literature require larger latency than the proposed design (from 1024 bits to 4096 bits) except for the integral stochastic implementation [17]. Moreover, with a 256-bit sequence length, the SC-DBN achieves a higher classification accuracy than an 8-bit fixed-point implementation, which is only 0.12% lower than a 32-bit floating-point implementation.

TABLE I. Accuracy Comparison

Network	sequence length (bit)	accuracy (%)
SC-DBN (16× parallelization)	32	89.90
	64	97.78
	128	**98.90**
	256	**99.15**
8-bit fixed point	–	98.10
32-bit floating-point	–	99.27
integral stochastic NN [17]	64	97.73
SC DNN [16]	1024	97.59
FPGA-RBM [22]	1024	94.28
FPGA-DBN [15]	4096	94.10

TABLE II. Hardware Efficiency Comparison

	SC_DBN	Fixed-point (8 bits)	Floating-point (32 bits)
area (μm^2)	23062.61	86875.05	437767.22
power (mW)	1.12	4.01	24.86
frequency (MHz)	134.7	167.3	159.7
cycle (/sample)	256	296	412
latency (μs/sample)	1.90	1.77	2.58
energy (nJ/sample)	2.12	7.10	64.14

B. Hardware efficiency

ASIC implementations of the DBNs are assessed in area, power and energy consumption using VHDL synthesized by the Synopsys Design Compiler with ST's 28 nm technology library. The sequence length of the SC-DBN is set to 256 bits with $16\times$ parallelization. As shown in Table II, the simulation results indicate that the SC-DBN requires the smallest area, lowest power and energy consumption among the different implementations. It takes 5.27%, 4.49% and 3.31% of the area, power and energy consumption of the 32-bit floating-point implementation. These figures of merit are 26.55%, 27.82% and 29.89% when compared to the 8-bit fixed-point implementation. The SNGs take 16.92% and 19.31% of the circuit area and power consumption of the SC-DBN, and achieve approximately reductions of $1.5\times$ in area and $2\times$ in energy consumption when compared to the design in [16]. Overall, the proposed design is $2\times$ smaller in area and $2.5\times$ lower in energy consumption compared to the design in [16].

In spite of the latency challenge for SC [6] [23], the SC-DBN achieves a similar performance with $16\times$ parallelization compared to conventional binary designs, thus significantly reducing the latency of an SC circuit.

V. Conclusion

In this paper, an energy-efficient stochastic computational deep belief network (SC-DBN) is proposed to implement the forward propagation process for inference. An approximate SC activation unit (A-SCAU) is reconfigurable to implement different activation functions. It also leverages the shared use of RNGs among neurons in the same layer, so significantly smaller area and lower energy consumption are obtained for the proposed design. The classification accuracy of the SC-DBN is higher than a fixed-point implementation and slightly lower than a floating-point implementation. Compared to the conventional binary implementations, however, the proposed design achieves significantly smaller area, lower power and energy consumption with a similar processing speed.

References

[1] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[2] C. Szegedy, A. Toshev, and D. Erhan, "Deep neural networks for object detection," in *Advances in Neural Information Processing Systems*, pp. 2553–2561, 2013.

[3] G. E. Dahl, D. Yu, L. Deng, and A. Acero, "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 1, pp. 30–42, 2012.

[4] D. Ciregan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," in *IEEE Conf. on. Computer Vision and Pattern Recognition (CVPR)*, pp. 3642–3649, 2012.

[5] B. R. Gaines *et al.*, "Stochastic computing systems," *Advances in information systems science*, vol. 2, no. 2, pp. 37–172, 1969.

[6] A. Alaghi and J. P. Hayes, "Survey of stochastic computing," *ACM TECS*, vol. 12, no. 2s, p. 92, 2013.

[7] 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.

[8] B. D. Brown and H. C. Card, "Stochastic neural computation. II. soft competitive learning," *IEEE Transactions on Computers*, vol. 50, no. 9, pp. 906–920, 2001.

[9] J. P. Hayes, "Introduction to stochastic computing and its challenges," in *DAC*, p. 59, 2015.

[10] A. Alaghi and J. P. Hayes, "Dimension reduction in statistical simulation of digital circuits," in *Proceedings of the Symposium on Theory of Modeling & Simulation: DEVS Integrative M&S Symposium*, pp. 1–8, 2015.

[11] P. Li, W. Qian, M. D. Riedel, K. Bazargan, and D. J. Lilja, "The synthesis of linear finite state machine-based stochastic computational elements," in *IEEE ASP-DAC*, pp. 757–762, 2012.

[12] Y. Ji, F. Ran, C. Ma, and D. J. Lilja, "A hardware implementation of a radial basis function neural network using stochastic logic," in *DATE*, pp. 880–883, 2015.

[13] J. L. Rosselló, V. Canals, and A. Morro, "Probabilistic-based neural network implementation," in *IEEE IJCNN*, pp. 1–7, 2012.

[14] Z. Li, A. Ren, J. Li, Q. Qiu, Y. Wang, and B. Yuan, "Dscnn: hardware-oriented optimization for stochastic computing based deep convolutional neural networks," in *IEEE ICCD*, pp. 678–681, 2016.

[15] K. Sanni, G. Garreau, J. L. Molin, and A. G. Andreou, "FPGA implementation of a deep belief network architecture for character recognition using stochastic computation," in *IEEE CISS*, pp. 1–5, 2015.

[16] K. Kim, J. Kim, J. Yu, J. Seo, J. Lee, and K. Choi, "Dynamic energy-accuracy trade-off using stochastic computing in deep neural networks," in *DAC*, p. 124, 2016.

[17] A. Ardakani, F. Leduc-Primeau, N. Onizawa, T. Hanyu, and W. J. Gross, "VLSI implementation of deep neural network using integral stochastic computing," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2017.

[18] S. S. Haykin, *Neural networks and learning machines*, vol. 3. Pearson Upper Saddle River, NJ, USA, 2009.

[19] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pp. 315–323, 2011.

[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] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *Advances in Neural Information Processing Systems*, pp. 1135–1143, 2015.

[22] B. Li, M. H. Najafi, and D. J. Lilja, "An FPGA implementation of a restricted boltzmann machine classifier using stochastic bit streams," in *IEEE Conf. on. Application-specific Systems, Architectures and Processors (ASAP)*, pp. 68–69, 2015.

[23] R. Wang, J. Han, B. F. Cockburn, and D. G. Elliott, "Stochastic circuit design and performance evaluation of vector quantization for different error measures," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 10, pp. 3169–3183, 2016.