Softmax Regression Design for Stochastic Computing Based Deep Convolutional Neural Networks

Zihao Yuan[1], Ji Li[1], Zhe Li[2], Caiwen Ding[2], Ao Ren[2], Bo Yuan[3], Qinru Qiu[2], Jeffrey Draper[1,4], and Yanzhi Wang[2]

[1]University of Southern California, Los Angeles, USA ({zihaoyua, jli724}@usc.edu)

[2]Syracuse University, Syracuse, USA ({zli89, aren, cading, qiqiu, ywang393}@syr.edu)

[3]City University of New York, New York, USA (byuan@ccny.cuny.edu)

[4]Information Sciences Institute, Marina Del Rey, USA (draper@isi.edu)

ABSTRACT

Recently, Deep Convolutional Neural Networks (DCNNs) have made tremendous advances, achieving close to or even better accuracy than human-level perception in various tasks. Stochastic Computing (SC), as an alternate to the conventional binary computing paradigm, has the potential to enable massively parallel and highly scalable hardware implementations of DCNNs. In this paper, we design and optimize the SC based Softmax Regression function. Experiment results show that compared with a binary SR, the proposed SC-SR under longer bit stream can reach the same level of accuracy with the improvement of 295X, 62X, 2617X in terms of power, area and energy, respectively. Binary SR is suggested for future DCNNs with short bit stream length input whereas SC-SR is recommended for longer bit stream.

1. INTRODUCTION

Nowadays, Deep Convolutional Neural Network (DCNN) is the dominant approach for classification and detection tasks for images, video, speech as well as audio [1]. DCNNs implement the backpropagation algorithm, which points out the parameters that should be updated. These parameters are used to compute the representation in each layer from the output of the previous layer. Clearly, the huge amount of computation power of DCNNs prevents their widespread applications in wearable and Internet of Things (IoT) devices [2, 3, 4].

Compared to the studies using conventional binary arithmetic computing, Stochastic Computing (SC) is a fascinating solution to the above issues due to its superior performance in terms of area and power consumption as well as high tolerance to soft errors [5, 6, 7]. SC represents a number by the probability of 1s in a random bit stream. Many complex arithmetic operations can be implemented with very simple hardware logic in the SC framework, which alleviates the extensive computation complexity [6, 8]. On this account, a mass of research efforts have been put into designing neural networks using SC [2, 6, 8, 9]. Both of the recent designs [2, 9] successfully implement the SC-based neuron cells and the layerwise structure of neural networks. Nevertheless, there is no existing design flow for Softmax Regression (SR) function after the fully-connected layer for DCNNs. SR is the generalization of logistic regression function when multiple categories need to be classified. It is one of the most significant part in deep learning networks due to the fact that it directly affects the final result.

In this paper, we first propose a SC-based Softmax Regression function block. The design parameters are optimized in order to achieve best performance for accuracy. After that we conduct a comprehensive comparison between binary ASIC SR function and SC-based SR function under different input sizes and stochastic bit stream lengths. Moreover, we further construct and investigate the network performances between the conventional binary SR design and the proposed SC-based SR design in a practical DCNN.

GLSVLSI '17, May 10–12, 2017, Banff, AB, Canada.

© 2017 Copyright held by the owner/author(s).

ACM ISBN 978-1-4503-4972-7/17/05…$15.00.

DOI: http://dx.doi.org/10.1145/3060403.3060467

2. DCNN ARCHITECTURE AND SOFTMAX REGRESSION FUNCTION

2.1 Deep Convolutional Neural Network

A general DCNN architecture consists of a stack of convolutional layers, pooling layers, and fully connected layers. A convolutional layer is associated with a set of learnable filters (or kernels), and common patterns in local regions of inputs are extracted by convolving this kind of filter over the inputs [10]. A feature map is built to store the convolution result. After that, a subsampling step is applied to aggregate statistics of these features in the pooling layer for the sake of reducing the dimensions of data and alleviating over-fitting issues. In addition, a nonlinear activation function is applied here to generate the output of the layer [11]. After several convolutional and pooling layers, the high-level reasoning fully connected layer is applied in order to further aggregate the local information learned in previous layers. After that, a Softmax Regression function should be applied for classification.

2.2 Stochastic Computing (SC)

Stochastic computing is a technology that represents a numeric value $x \in [0, 1]$ by counting the number of 1s in a bit stream, e.g., the value of a 4-bit sequence "0100" is $x = P(X = 1) = 0.25$. In addition to this unipolar format, another widely used format is bipolar format. In this coding scheme, a value $x \in [-1, 1]$ is processed by $P(X = 1) = \frac{x+1}{2}$. With SC, addition, multiplication, and division can be implemented using significantly smaller circuits, compared to the conventional binary arithmetic [6] as shown in Figure 1 (a), (b) and (c).

To be more specific, multiplications are executed using XNOR gates in bipolar format. The stochastic number of C is calculated as $c = 2P(C) - 1 = 4P(A)P(B) - 2P(A) - 2P(B) + 1 = (2P(A) - 1)(2P(B) - 1) = a \times b$. Multiplexers (MUXes) are used to processed addition in SC [6]. In order to achieve a better accuracy with little deficit in terms of power, area and energy, we adopt the Approximate Parallel Counter (APC) proposed in [12]

Division can only be accomplished in an approximate form in the stochastic number representation schemes [6]. Given input X and Y, output $Q = \frac{Y}{X}$ is represented as

$$Q = -\alpha(XQ - Y) \tag{1}$$

where α is a positive parameter which controls the rate change for the counter state. A SC-based unipolar division circuit is implemented by adopting the gradient descent technique with a saturated counter as an integrator. Division is implemented by incrementing the counter when Y is 1 and decrementing the counter when both X and Q are 1s.

2.3 Softmax Regression (SR) Function

Softmax Regression (SR) is a generalization of logistic regression for the sake of classifying multiple mutually exclusive classes. SR is placed after fully connected layer in order to assign probabilities to an object being one of several different things. SR is composed of two parts, i.e., summation and softmax. Summation is used to add up the pixel intensities. It is quite similar to normal neuron cell operation except the activation function. Given input x, output Z in class i is calculated as

$$Z_i = \sum_j W_{i,j} x_j + b_i \tag{2}$$

where $W_{i,j}$ is the weight and b_i stands for extra parameter called bias of class i. These parameters are adjusted during the backpropagation process. The subsequent softmax step acts like an activation function which changes the linear function into different nonlinear shapes. In this scenario, the summation result is shaping into a probability distribution function over different classes. Given input x, output P for class i is defined as

$$P_i = \frac{exp(x_i)}{\sum_j exp(x_j)} \tag{3}$$

The exponential function means little increase in input x_i will result in dramatically growth in result $exp(x_i)$ and indeed increase the probability in class i. This enables SR to distinguish among different categories and select the most similar result.

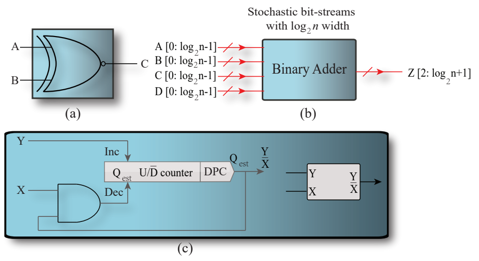

Figure 1: Stochastic computing for neuron design: (a) XNOR gate for bipolar multiplication, (b) binary adder, and (c) unipolar division.

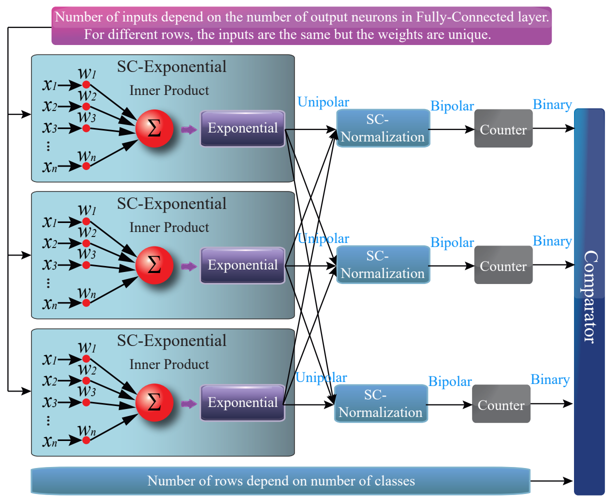
Number of inputs depend on the number of output neurons in Fully-Connected layer.
For different rows, the inputs are the same but the weights are unique.
SC-Exponential
Inner Product
x1 x2 x3 xn
w1 w2 w3 wn
Σ
Exponential
Unipolar
SC-Normalization
Bipolar
Counter
Binary
Comparator
Number of rows depend on number of classes

Figure 2: Structure for SC based Softmax Regression function.

Table 1: Naming Conventions in a SC based SR

m	the length of bit stream
q	number of classes
n	input size: the number of input bit streams (or the number of input and weight pairs) in each convolution block
x_i^j	the i-th input bit of the j-th convolution block ($i \in [0, n-1]$, $j \in [0, q-1]$)
w_i^j	the i-th weight bit of the j-th convolution block ($i \in [0, n-1]$, $j \in [0, q-1]$)
t_j	the sum calculated by the j-th convolution block, which is a $log_2 n$-bit binary number

3. SC-SOFTMAX REGRESSION FUNCTION DESIGN

3.1 Overall Structure

The proposed structure of SC-SR is shown in Figure 2, which is composed of SC-exponential, SC-normalization, SC-comparator and counter blocks. Bipolar encoding scheme is employed. The proposed SC-SR design adopt XNOR gates and APCs for addition and multiplication (same as convolution), respectively. Note that the outputs of APCs are binary. In addition, SC-exponential accomplishes a binary input to unipolar stochastic bit stream conversion. After that, the SC-normalization step converts the unipolar output from exponential block to bipolar stochastic bit stream. For the convenience of discussions, we follow the naming conventions in Table I.

3.2 SC-exponential

The author in [2] use a saturated up/down counter to implement a binary hyperbolic tangent function. We adopt the saturated counter idea and implant in our design. Algorithm 1 presents the proposed SC-exponential function where step 8-11 correspond to convolution using XNOR gates and APCs and step 12-16 are the adopted saturated counter. The counted binary number generated by APC is taken by the saturated up/down counter which represents the amount of increase and decrease. Besides, we also use a history shift register H[0:α-1] in order to count the last α bits of output stochastic bit stream. By using the sum of the α bits, which is denoted by δ, we can predict the next stochastic output bit. To be more specific, if the sum of last α bits falls in range of [0.3α, 0.4α], the output is predicted as 0 because the possibility of this value being large is small. On the other hand, if it falls into [0.6α, 0.7α], this output is predicted to be a 1. Another reason for this is that the normalization block in our

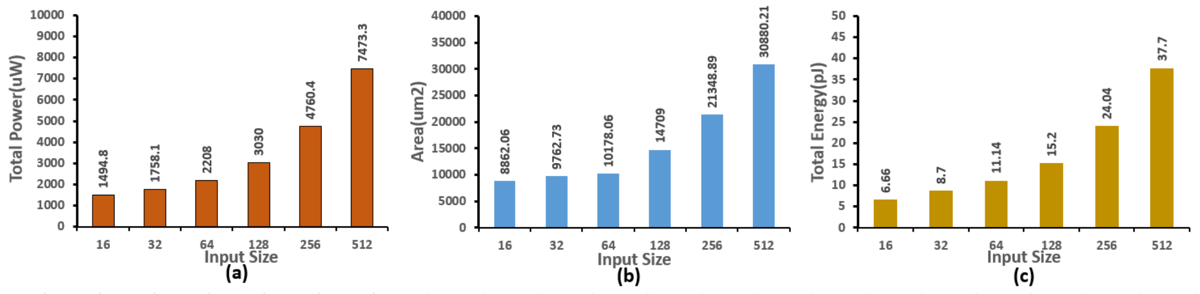

Figure 3: Input size versus (a) total power, (b) area, and (c) total energy for the proposed SC-SR.

proposed design is created by adopting the unipolar division method in [6]. Note that the outputs of our SC-exponential block are in unipolar encoding scheme.

Algorithm 1: Designated Softmax Regression SC-Exponential $(m, n, x_i^j, w_i^j, \alpha, e)$

```
    input  : $q$ indicates number of classes
    input  : $\alpha$ is the number of registers in the temporary array
    input  : $e$ is internal FSM state number
    output: $z_k$ is the $k$-th stochastic output bit for the SC-exponential
1   $S_{max} \leftarrow e - 1$ ;                                   /* max state */
2   $S \leftarrow \frac{e}{2}$ ;                                   /* current state */
3   $r \leftarrow \alpha - 1$ ;                                    /* r is an iterator */
4   $H[0 : \alpha - 1] \leftarrow 0$ ;                             /* initialize history array */
5   $\delta \leftarrow \frac{\alpha}{2}$ ;                         /* initialize shadow counter */
6   for $k \leftarrow 1$ to $m$ do
7       for $i \leftarrow 1$ to $n$ do
8           $p_i^j = x_i^j \odot w_i^j$ ;                          /* XNOR multiplication */
9       end
10      $t_j = 2 \cdot \sum_{i=0}^{n} p_i^j - n$ ;                 /* APC addition */
11      $S \leftarrow S + t_j$
12      if $S < 0$ then                                            /* saturated counter */
13          $S \leftarrow 0$
14      else if $S > S_{max}$ then
15          $S \leftarrow S_{max}$
16      end
17      if $S < \frac{n}{4}$ then                                  /* output logic */
18          if $0.4 \times \alpha > \delta > 0.3 \times \alpha$ then
19              $z_k \leftarrow 0$
20          else if $S > \frac{S}{2}$ then
21              $z_k \leftarrow 1$
22          else
23              $z_k \leftarrow 0$
24          end
25      else
26          if $0.7 \times \alpha > \delta > 0.6 \times \alpha$ then
27              $z_k \leftarrow 1$
28          else if $S > \frac{S}{2}$ then
29              $z_k \leftarrow 1$
30          else
31              $z_k \leftarrow 0$
32          end
33      end
34  end
35  while $r \geq 1$ do
36      $H[r] \leftarrow H[r - 1]$;                                /* update the history array */
37      $r \leftarrow r - 1$
38  end
39  $H[0] \leftarrow z_k$
40  $\delta \leftarrow \sum_{r=0}^{\alpha-1} H[r]$;                /* update the shadow counter */
41  $r \leftarrow \alpha - 1$
```

3.3 SC-normalization

Since the output from previous SC-exponential block is in unipolar encoding format. In Algorithm 2, We adopt the unipolar division circuit from [6]. Step5-17 correspond to the summation of all the previous outputs and determine the divisor value by using an AND gate. We create designated SC-exponential and SC-normalization for each class. Therefore, the dividend is just the output of previous SC-exponential block for corresponding class. We have 4 different conditions to control the amount of increase and decrease in the implanted saturated counter as shown in step19-30. Since the output is a non-negative bipolar stochastic number, a history shift register H[0:α-1] is also implemented in this algorithm in order to eliminate negative value.

Table 2: Network Accuracy

	input size	bit stream	validation(%)	test(%)
software	256	-	99.03	99.10
	512	-	98.96	99.04
binary	256	-	99.04	99.10
	512	-	98.96	99.07
SC	256	16	59.04	59.75
		32	78.26	77.23
		64	98.80	98.91
		256	99.02	99.08
		1024	99.00	99.14
	512	16	19.72	19.53
		32	98.85	99.00
		64	98.04	98.29
		256	98.93	99.09
		1024	98.78	99.06

4. EXPERIMENTAL RESULTS

4.1 Performance analysis for SC-SR

For SC-SR, the accuracy depends on the bit stream length m and input size n. Hence, in order to consider the aforementioned factors, we create and analyze the inaccuracy of SC-SR using LeNet-5 [13] classification method based on the wide range of input size and bit stream length as shown in Figure 4. The corresponding hardware costs of proposed SC-SR are shown in Figure 3 (a), (b) and (c). The SRs and DCNNs are synthesized in Synopsys Design Compiler with the 45nm Nangate Library [14] using Verilog. Note that the inaccuracy here is calculated by comparing with the software results. It is obviously that SC-SR will be less accurate provided that the input size increases and it will be precision if the bit stream length increases. we further test the proposed design under different number of classes using AlexNet [15] classification method, the classification accuracies for different input size and bit stream length are all 100 % which means there is no accuracy degradation.

4.2 Comparison with Binary ASIC SR

We further compare the performance of the proposed SC-based Softmax Regresion block with the binary ASIC hardware SR. The input is set to 800 and the number of classes

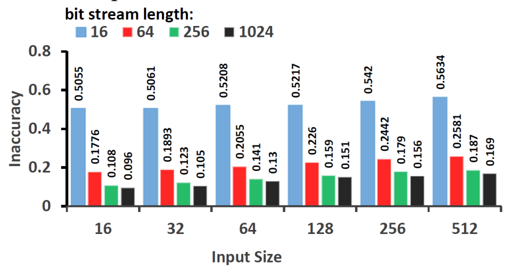

Figure 4: Input size versus absolute inaccuracy under different bit stream lengths for SC-SR.

Algorithm 2: Designated Softmax Regression SC-Normalization (m, x_j, α, e)

```
    input : $f$ is rate change for the FSM
    input : $x_j$ is the input bit of the $j$-th exponential block
    input : $e$ is number of different classes
    output: $z_k^j$ is the $k$-th stochastic output bit of class $j$ for the
            SC-normalization
 1  Algorithm 1 step 1-4
 2  $\delta \leftarrow 0$ ;                                    /* initialize shadow counter */
 3  for $k \leftarrow 1$ to $m$ do
 4      $p = \sum_{j=1}^{e} x_j$ ;       /* summation of output bits from exponential blocks */
 5      if $p > \frac{e}{2}$ then                             /* nomralize summation result */
 6          $h \leftarrow 1$
 7      else
 8          $h \leftarrow 0$
 9      end
10      if $S > \frac{e}{2}$ then                                                /* Divisor */
11          $X \leftarrow h$
12      else
13          $X \leftarrow 0$
14      end
15      $Y = x_j$;                                                              /* Dividend */
16      if $X == 1 \&\& Y == 1$ then                                    /* Next state logic */
17          $S = S - p + f \times e$
18      end
19      if $X == 0 \&\& Y == 1$ then
20          $S = S + f \times e$
21      end
22      if $X == 1 \&\& Y == 0$ then
23          $S = S - p$
24      end
25      if $X == 0 \&\& Y == 0$ then
26          $S = S$
27      end
28      if $S < 0$ then                                                /* saturated counter */
29          $S \leftarrow 0$
30      else if $S > S_{max}$ then
31          $S \leftarrow S_{max}$
32      end
33      if $\delta < \frac{\alpha}{2}$ then                     /* compensate for negative value */
34          $z_k^j \leftarrow 1$
35      else
36          if $0.7 \times \alpha > \delta > 0.6 \times \alpha$ then                  /* output logic */
37              $z_k^j \leftarrow 1$
38          else if $S > \frac{e}{2}$ then
39              $z_k^j \leftarrow 1$
40          else
41              $z_k^j \leftarrow 0$
42          end
43      end
44  end
45  Algorithm 1 step 35-41
```

Table 3: Performance Comparison with 8 Bit Fixed Point Binary Design when $n = 800$ and $q = 10$

	SC-800bits	binary 800bits	improvement
dynamic power(uW)	10981	3503800	319X
leakage power(uW)	1078	58986	55X
total power(uW)	12058	3562100	295X
area(um2)	50083	3094968	62X
delay(ns)	5.05	44.74	8.8X
energy(pJ)	61	159368	2617X

is set to 10. The binary exponential function is built using LUTs, whereas the normalization block is built using divider. Clearly, the number of bits in fixed-point numbers affect both the hardware cost and accuracy. To make the comparison fair, we adopt minimum fixed point (8 bit) that yields a DCNN network accuracy that is almost identical to the software DCNN (with < 0.0003 difference in network test error). Table 3 shows the performance comparison between binary SR and SC-SR. Compared with binary ASIC SR, the proposed SC-SR achieves up to 295X, 62X, and 2617X improvement in terms of power, energy and area, respectively, indicating significant hardware savings.

4.3 DCNN Accuracy Evaluation

To evaluate the network accuracy, we construct a LeNet-5 DCNN, which is a widely-used DCNN structure, by replacing the software Softmax function with the proposed SC-SR as well as binary SR. We evaluate two SR configurations, i.e., the LeNet-5 with configurations of 784-11520-2880-3200-800-256-10 and 784-11520-2880-3200-800-512-10. The DCNNs are evaluated using the MNIST handwritten digit image dataset [16]. we apply the same amount of training time in software for these two DCNN architectures. Table 2 summarizes the accuracy of DCNNs using SC-SR and binary SR. One can observe that with a long input bit stream length ($m >= 64$), SC-SR reaches the same precision level as binary SR and software SR. As we discuss above, compared to binary SR, SC-SR has better performance in terms of power, area and energy. Hence, binary SR is suggested for future DCNNs when bit stream length is short (e.g., $m = 64$), whereas SC-SR is recommended for long bit stream length DCNNs.

5. CONCLUSION

In this paper, we present a novel SC based Softmax Regresion function design. We test the proposed SC-SR under different input size and bit stream length as well as output classes. In addition, we implant the proposed design into LeNet-5 DCNN. Experimental results on the MNIST dataset demonstrate that compared to the binary SR, the proposed SC-SR under long bit stream length input were able to significantly reduce the area, power and energy footprint with nearly no accuracy degradation.

6. REFERENCES

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

[2] 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 *Proceedings of the 53rd Annual Design Automation Conference*. ACM, 2016, p. 124.

[3] J. Li, A. Ren, Z. Li, C. Ding, B. Yuan, Q. Qiu, and Y. Wang, "Towards acceleration of deep convolutional neural networks using stochastic computing," in *Asia and South Pacific Design Automation Conference (ASP-DAC). IEEE*, 2017.

[4] Y. Xue, J. Li, S. Nazarian, and P. Bogdan, "Fundamental challenges towards making iot a reachable reality: A model-centric investigation," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 2017.

[5] J. Li and J. Draper, "Accelerating soft-error-rate (ser) estimation in the presence of single event transients," in *Proceedings of the 53rd Annual Design Automation Conference*. ACM, 2016, p. 55.

[6] 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.

[7] J. Li and J. Draper, "Joint soft-error-rate (ser) estimation for combinational logic and sequential elements," in *VLSI (ISVLSI), 2016 IEEE Computer Society Annual Symposium on*, 2016, pp. 737–742.

[8] 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 *Proceeding of IEEE International Conference on Computer Design*, 2016.

[9] A. Ren, J. Li, Z. Li, C. Ding, X. Qian, Q. Qiu, B. Yuan, and Y. Wang, "Sc-dcnn: Highly-scalable deep convolutional neural network using stochastic computing," *arXiv preprint arXiv:1611.05939*, 2016.

[10] Z. Li, A. Ren, J. Li, Q. Qiu, B. Yuan, J. Draper, and Y. Wang, "Structural design optimization for deep convolutional neural networks using stochastic computing."

[11] J. Li, Z. Yuan, Z. Li, C. Ding, A. Ren, Q. Qiu, J. Draper, and Y. Wang, "Hardware-driven nonlinear activation for stochastic computing based deep convolutional neural networks," in *Neural Networks (IJCNN), 2017 International Joint Conference on*. IEEE, 2017.

[12] K. , J. Lee, and K. Choi, "Approximate de-randomizer for stochastic circuits," *Proc. ISOCC*, 2015.

[13] 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.

[14] Nangate 45nm Open Library, Nangate Inc., 2009. [Online]. Available: http://www.nangate.com/

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012.

[16] L. Deng, "The mnist database of handwritten digit images for machine learning research," *IEEE Signal Processing Magazine*, vol. 29, no. 6.