Universal Approximation Property and Equivalence of Stochastic Computing-Based Neural Networks and Binary Neural Networks

Yanzhi Wang,[1] Zheng Zhan,[2] Liang Zhao,[3] Jian Tang,[2] Siyue Wang,[1] Jiayu Li,[2] Bo Yuan,[4] Wujie Wen,[5] Xue Lin[1]

[1]Department of Electrical and Computer Engineering, Northeastern University, Boston, MA 02115
[2]Department of Electrical Engineering and Computer Science, Syracuse University, Syracuse, NY 13244
[3]Department of Mathematics and Computer Science, Lehman College of CUNY, Bronx, NY 10468
[4]Department of Electrical and Computer Engineering, Rutgers University, Piscataway, NJ 08854
[5]Department of Electrical and Computer Engineering, Florida International University, Miami, FL 33199

Abstract

Large-scale deep neural networks are both memory and computation-intensive, thereby posing stringent requirements on the computing platforms. Hardware accelerations of deep neural networks have been extensively investigated. Specific forms of binary neural networks (BNNs) and stochastic computing-based neural networks (SCNNs) are particularly appealing to hardware implementations since they can be implemented almost entirely with binary operations.

Despite the obvious advantages in hardware implementation, these approximate computing techniques are questioned by researchers in terms of accuracy and universal applicability. Also it is important to understand the relative pros and cons of SCNNs and BNNs in theory and in actual hardware implementations. In order to address these concerns, in this paper we prove that the "ideal" SCNNs and BNNs satisfy the universal approximation property with probability 1 (due to the stochastic behavior), which is a new angle from the original approximation property. The proof is conducted by first proving the property for SCNNs from the strong law of large numbers, and then using SCNNs as a "bridge" to prove for BNNs. Besides the universal approximation property, we also derive an appropriate bound for bit length M in order to provide insights for the actual neural network implementations. Based on the universal approximation property, we further prove that SCNNs and BNNs exhibit the same energy complexity. In other words, they have the same asymptotic energy consumption with the growth of network size. We also provide a detailed analysis of the pros and cons of SCNNs and BNNs for hardware implementations and conclude that SCNNs are more suitable.

Introduction

Large-scale neural networks are both memory-intensive and computation-intensive, thereby posing stringent requirements on the computing platforms when deploying those large-scale neural network models on memory-constrained and energy-constrained embedded devices. In order to overcome these limitations, the hardware accelerations of deep neural networks have been extensively investigated in both industry and academia (Mahajan et al. 2016; Zhao et al. 2017b; Umuroglu et al. 2017; Han et al. 2016; Chen et al. 2014; Moons et al. 2017). These hardware accelerations are based on FPGA and ASIC devices and can achieve a significant improvement on energy efficiency, along with small form factor, compared with traditional CPU or GPU based computing of deep neural networks. Both characteristics are critical for the battery-powered embedded and autonomous systems.

Hardware systems, including FPGAs and ASICs, have much higher peak performance for binary operations compared to floating point ones. Besides, it is also desirable to reduce the model size of deep neural network such that the whole model can be stored using on-chip memory, thereby reducing the timing and energy overheads of off-chip storage and communications. As a result, the Binary Neural Networks (BNNs), proposed by (Courbariaux, Bengio, and David 2015), are particularly appealing since they can be implemented almost entirely with binary operations, with the potential to attain performance in the tera-operations per second (TOPS) range on FPGAs or ASICs.

Besides BNNs, reference work (Ren et al. 2017; Yu et al. 2017; Kyounghoon Kim 2015; Merolla et al. 2014; Li et al. 2018; Neftci 2016; Andreou and Chatzis 2016) have also proposed to utilize the hardware-oriented Stochastic Computing (SC) technique for developing (large-scale) deep neural networks, i.e., SCNNs. The SC technique represents a number using the portion of 1's in a bit sequence. Many key operations in neural networks, such as multiplications and additions, can be implemented in a single gate in SC. For example, multiplication of two stochastic numbers can be implemented using a single AND gate or XNOR gate (depending on unipolar or bipolar representations). It enables the efficient implementation of deep neural networks with extremely small hardware footprint.

The BNNs and SCNNs are essentially alike: Both rely on binary operations and very simple calculations in hardware such as AND, XNOR gates, multiplexers and counters. For their distinctions, SCNNs "stretch" in the temporal domain and use a bit sequence (stochastic number) to approximate a real number, whereas BNNs "span" in the spatial domain and require more input and hidden neurons to maintain the desired accuracy.

Despite the obvious advantages in hardware implementation, these approximate computing techniques are questioned by researchers in terms of accuracy. Will SCNNs and BNNs be accurate for any types of neural networks and applications? More specifically, conventional neural networks with at least one hidden layer satisfy the *universal approximation property* (Csáji 2001) in that they can approximate an arbitrary continuous or measurable function given enough number of neurons in the hidden layer. Will SCNNs and BNNs satisfy such property as well? Finally, what are the relative pros and cons of SCNNs and BNNs in theory, and at the hardware level?

In this paper we aim to answer the above questions. We consider the "ideal" SCNNs and BNNs that are independent of specific hardware implementations. As the key contribution of this paper, we prove that SCNNs and BNNs satisfy the universal approximation property with probability 1 (due to the stochastic behavior in these networks), which is a new angle from the original approximation property. The proof is conducted by first proving the property for SCNNs from the strong law of large numbers, and then using SCNNs as a "bridge" to prove for BNNs. This is because it is difficult to directly prove the property for BNNs, as BNNs represent functions with discrete (binary) input values instead of continuous ones. Besides the universal approximation property, we also derive an appropriate bound for bit length M in order to provide insights for the actual neural network implementations.

Based on the universal approximation property, we further prove that SCNNs and BNNs exhibit the same energy complexity. In other words, they have the same asymptotic energy consumption with the growing of network size. We also provide a detailed analysis of the pros and cons of SCNNs and BNNs for hardware implementations and conclude that SCNNs are more suitable for hardware. It is also possible to strike a desirable balance between SCNNs and BNNs to derive the best-suited implementation for a specified hardware.

Background and Related Work

Stochastic Computing and SCNNs

Stochastic computing (SC) is a paradigm that represents a number, named *stochastic number*, by counting the number of ones in a bit-stream. For example, the bit-stream 0100110100 contains four ones in a ten-bit stream, thus it represents $x = P(X = 1) = 4/10 = 0.4$. In the bit-stream, each bit is independent and identically distributed (i.i.d.) which can be generated in hardware using stochastic number generators (SNGs). Obviously, the length of the bit-streams can significantly affect the calculation accuracy in SC (Brown and Card 2001). In addition to this unipolar encoding format, SC can also represent numbers in the range of $[-1, 1]$ using the bipolar encoding format. In this scenario, a real number x is processed by $P(X = 1) = (x + 1)/2$. Thus 0.4 can be represented by 1011011101, as $P(X = 1) = (0.4 + 1)/2 = 7/10$. -0.5 can be represented by 10010000, as it shown in figure 1(b), with $P(X = 1) = (-0.5 + 1)/2 = 2/8$.

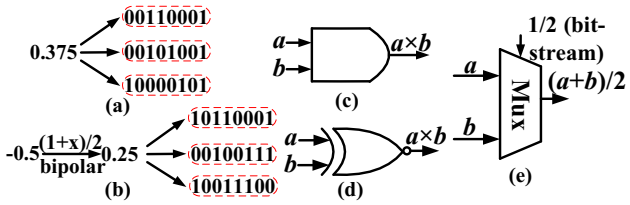

Figure 1: (a) Unipolar encoding format and (b) bipolar encoding format. (c) AND gate for unipolar multiplication. (d) XNOR gate for bipolar multiplication. (e) MUX gate for addition.

Compared to conventional computing, the major advantage of stochastic computing is the significantly lower hardware cost for a large category of arithmetic calculations. A summary of the basic computing components in SC, such as multiplication and addition, is shown in Figure 1. As an illustrative example, a unipolar multiplication can be performed by a single AND gate since $P(A \cdot B = 1) = P(A = 1)P(B = 1)$ (assuming independence), and a bipolar multiplication is performed by a single XNOR gate since $c = 2P(C = 1) - 1 = 2(P(A = 1)P(B = 1) + P(A = 0)P(B = 0)) - 1 = (2P(A = 1) - 1)(2P(B = 1) - 1) = ab$.

Besides multiplications and additions, SC-based activation functions are also developed (Li et al. 2017a; 2017b). As a result, SC has become an interesting and promising approach to implement large-scale neural networks (Yuan et al. 2017; Yu et al. 2017; Li et al. 2017c; Kyounghoon Kim 2015) with high performance/energy efficiency and minor accuracy degradation.

We note that the goal of SCNN in approximating using stochastic computation is very different from dropout (or other stochastic techniques). SCNN aims to facilitate hardware implementations, and by transforming binary numbers and weights to stochastic ones, it enables efficient implementation with extremely small hardware footprint. This is different from dropout which aims to enhance the generality and robustness.

Binary Neural Networks (BNNs)

BNNs use binary weights, i.e., weights that are constrained to only two possible values (not necessarily 0 and 1) (Courbariaux, Bengio, and David 2015). BNNs also have great potential to facilitate consumer applications on low-power devices and embedded systems. (Zhao et al. 2017b; Umuroglu et al. 2017) have implemented BNNs in FPGAs with high performance and modest power consumption.

BNNs constrain the weights to either $+1$ or -1 during the forward propagation process. As a result, many multiply-accumulate operations are replaced by simple additions (and subtractions) using single gates. This results in a huge gain in hardware resource efficiency, as fixed-point adders/accumulators are much less expensive both in area and energy than fixed-point multiply-accumulators (David, Kalach, and Tittley 2007).

The real-valued weights are transformed into the two possible values through the following stochastical binarization operation:

$$w_{\mathrm{B}} = \begin{cases} +1 & \text{with probability} \quad p = \sigma(w) \\ -1 & \text{with probability} \quad 1-p \end{cases} \tag{1}$$

where σ is the *hard sigmoid* function:

$$\sigma(x) = \mathrm{clip}(\frac{x+1}{2}, 0, 1) = \max(0, \min(1, \frac{x+1}{2})) \tag{2}$$

A hard sigmoid rather than the soft version is used because it is far less computationally expensive.

At training time, BNNs randomly pick one of two values for each weight, for each minibatch, for both the forward and backward propagation phases of backpropagation. However, the stochastic gradient descent (SGD) update is accumulated in a real-valued variable storing the parameter to average the noise for keeping sufficient resolution. Moreover, binarization process adds some noise into the model, which provides a form of generalization to address the over-fitting problem.

Universal Approximation Property

For feedforward neural networks with one hidden layer, (Cybenko 1989) and (Hornik, Stinchcombe, and White 1989) have proved separately the universal approximation property, which guarantees that for any given continuous function or measurable function and any error bound $\epsilon > 0$, there always exists a single-hidden layer neural network that approximates the function within ϵ integrated error. Besides the approximation property itself, it is also desirable to cast a limit on the maximum amount of neurons. In this direction, (Barron 1993) showed that feedforward networks with one layer of sigmoidal nonlinearities achieve an integrated squared error with order of $\mathrm{O}(1/n)$, where n is the number of neurons.

More recently, several interesting results were published on the approximation capabilities of deep neural networks or neural networks using structured matrices. (Delalleau and Bengio 2011) have shown that there exists certain functions that can be approximated by three-layer neural networks with a polynomial amount of neurons, while two-layer neural networks require exponentially larger amount to achieve the same error. (Montufar et al. 2014) and (Telgarsky 2016) have shown the exponential increase of linear regions as neural networks grow deeper. (Liang and Srikant 2016) proved that with $\log(1/\epsilon)$ layers, the neural network can achieve the error bound ϵ for any continuous function with $\mathrm{O}(polylog(\epsilon))$ parameters in each layer. Recently, (Zhao et al. 2017a) have proved that neural networks represented in structured, low displacement rank matrices preserve the universal approximation property. These recent research have sparked the research interests on the theoretical properties of neural networks with simplifications/approximations which are suitable for high-efficiency hardware implementations.

Neural Network of Interests and SCNNs

Our problem statement follows the flow of reference work (Zhao et al. 2017a) for investigating the universal approximation property. Let I_n denote the n-dimensional unit cube, $[0,1]^n$. The space of continuous functions on I_n is denoted by $C(I_n)$. A feedforward neural network with N units of neurons arranged in a single hidden layer is denoted by a function $G : \mathbb{R}^n \to \mathbb{R}$, satisfying the form

$$G(\boldsymbol{x}) = \sum_{i=1}^{N} \alpha_i \sigma(\boldsymbol{w}_i^{\mathsf{T}} \boldsymbol{x} + b_i) \tag{3}$$

where $\boldsymbol{w}_i$, $\boldsymbol{x} \in \mathbb{R}^n$, α_i, $b_i \in \mathbb{R}$, and σ is a nonlinear sigmoidal activation function. The $\boldsymbol{w}_i$ denotes weights associated with hidden neuron i and is applied to input $\boldsymbol{x}$. α_i denotes the i-th weight of output neuron, and is applied to the output of i-th neuron in the hidden layer. b_i is the bias of unit i.

Definition 1. *A sigmoidal activation function $\sigma : \mathbb{R} \to \mathbb{R}$ satisfies*

$$\sigma(t) \to \begin{cases} 1 & as \quad t \to \infty \\ 0 & as \quad t \to -\infty \end{cases}$$

Definition 2. *Starting from the neural network of interests, we define an SCNN satisfying the form:*

$$G_{SC,M}(\boldsymbol{x}_{SC,M}) = \sum_{i=1}^{N} \alpha_i \sigma(\boldsymbol{w}_{SC,M,i}^{\mathsf{T}} \boldsymbol{x}_{SC,M} + b_{SC,M,i}) \tag{4}$$

where each element j in $\boldsymbol{w}_{SC,M,i}^{\mathsf{T}}$ is denoted by $w_{SC,M,i}^{j}$, and each element in $\boldsymbol{x}_{SC,M}$ is denoted by $x_{SC,M}^{j}$. $w_{SC,M,i}^{j}$, $x_{SC,M}^{j}$, and $b_{SC,M,i}$ are stochastic numbers represented by M-bit streams, as approximations of w_i^j, x^j, and b_i, respectively. These bit-streams are independent in each bit and $\boldsymbol{w}_{SC,M,i}^{\mathsf{T}}$, $\boldsymbol{x}_{SC,M}$, and $b_{SC,M,i}$ will converge to $\boldsymbol{w}_i^{\mathsf{T}}$, $\boldsymbol{x}$, and b_i as $M \to \infty$, respectively. The computation in $\boldsymbol{w}_{SC,M,i}^{\mathsf{T}} \boldsymbol{x}_{SC,M} + b_{SC,M,i}$ follows the SC rules described before.

In the above definitions we focus on an "ideal" SCNN that assumes accurate activation and output layer calculation (which is reasonable because the output layer size is typically very small). The SCNN of interest, as illustrated in Figure 2, does not depend on specific hardware implementations that may be different in practice. We also do not specify any limitation on the weight and input ranges because they can be effectively dealt with by pre-scaling techniques.

The Universal Approximation Property of SCNNs and BNNs

In this section, we prove that SCNNs and BNNs satisfy the universal approximation property with probability 1, which is a new angle from the original universal approximation property. More specifically, we first prove the property for SCNNs and then use SCNNs as a "bridge" to prove for BNNs. This two-step proof is due to the fact that directly proving the property for BNNs is difficult, as BNNs represent functions with binary input values.

The proof is on single hidden layer, as inherited from the original universal approximation property. For multi-layer networks, the universal approximation still holds as a natural extension (because the previous layers can be considered as mapping).

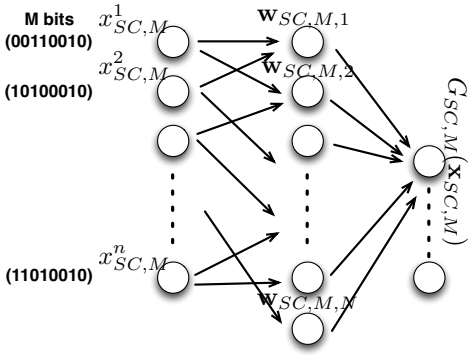

Figure 2: The structure of SCNN of interest.

Universal Approximation Property of SCNN

In this section we will prove a lemma on the closeness of stochastic approximation for the inputs of each neuron, a lemma on the closeness of approximations for the outputs, and finally extend the universal approximation theorem from (Cybenko 1989) to SCNNs.

Lemma 1. *As the bit-stream length $M \to \infty$, the stochastic number $\boldsymbol{w}^{\mathsf{T}}_{SC,M,i}\boldsymbol{x}_{SC,M} + b_{SC,M,i}$ converges to $\boldsymbol{w}_i^{\mathsf{T}}\boldsymbol{x} + b_i$ almost surely.*

Proof. Let Ω be the sample space of all bit-streams generated to represent elements in $\boldsymbol{w}_i^{\mathsf{T}}$, $\boldsymbol{x}$, and b_i. For each instance $\omega \in \Omega$, use notations $\boldsymbol{w}^{\mathsf{T}}_{\text{SC},M,i}(\omega)$, $\boldsymbol{x}_{\text{SC},M}(\omega)$, and $b_{\text{SC},M,i}(\omega)$ to represent stochastic numbers (or vectors) calculated from the corresponding M-bit streams associated with ω. Moreover, define three constant random variables representing the target real values, namely for each $i \in \{1, ..., N\}$,

$$\begin{cases} \boldsymbol{w}_i^{\mathsf{T}}(\omega) \equiv \boldsymbol{w}_i^{\mathsf{T}}, \forall \omega \in \Omega, \\ \boldsymbol{x}(\omega) \equiv \boldsymbol{x}, \forall \omega \in \Omega, \\ b_i(\omega) \equiv b_i, \forall \omega \in \Omega, \end{cases} \tag{5}$$

We shall prove that for every $\omega \in \Omega$:

$$\begin{aligned} \lim_{M\to\infty} \boldsymbol{w}^{\mathsf{T}}_{\text{SC},M,i}(\omega) \cdot \boldsymbol{x}_{\text{SC},M}(\omega) + b_{\text{SC},M,i}(\omega) = \\ \boldsymbol{w}_i^{\mathsf{T}}(\omega) \cdot \boldsymbol{x}(\omega) + b_i(\omega). \end{aligned} \tag{6}$$

From the construction of the random variables, we have that for each i and j

$$\begin{aligned} \lim_{M\to\infty} w^j_{\text{SC},M,i}(\omega) =& w_i^j, \\ \lim_{M\to\infty} x^j_{\text{SC},M}(\omega) =& x^j, \\ \lim_{M\to\infty} b_{\text{SC},M,i}(\omega) =& b_i. \end{aligned}$$

Therefore, these exists $M_{min}(\omega)$ such that for all $M \geq M_{min}(\omega)$ and all $\epsilon > 0$, we have

$$\begin{aligned} \left|w^j_{\text{SC},M,i}(\omega)x^j_{\text{SC},M}(\omega) - w_i^j x^j\right| <& \epsilon' \\ \left|b_{\text{SC},M,i}(\omega) - b_i\right| <& \epsilon', \end{aligned}$$

where $\epsilon' = \frac{1}{n+1}\epsilon$. Use an argument of triangle inequality to show

$$\left|\boldsymbol{w}^{\mathsf{T}}_{\text{SC},M,i}(\omega)\cdot\boldsymbol{x}_{\text{SC},M}(\omega) + b_{\text{SC},M,i}(\omega) - \boldsymbol{w}_i^{\mathsf{T}}\boldsymbol{x} - b_i\right| < \epsilon \tag{7}$$

Since ϵ can be arbitrarily small, it implies

$$\lim_{M\to\infty} \boldsymbol{w}^{\mathsf{T}}_{\text{SC},M,i}(\omega) \cdot \boldsymbol{x}_{\text{SC},M}(\omega) + b_{\text{SC},M,i}(\omega) = \boldsymbol{w}_i^{\mathsf{T}}\boldsymbol{x} + b_i. \tag{8}$$

Since this is true for every $\omega \in \Omega$, we conclude that

$$\begin{aligned} P\big(\{\omega \in \Omega : \lim_{M\to\infty} \boldsymbol{w}^{\mathsf{T}}_{\text{SC},M,i}(\omega) \cdot \boldsymbol{x}_{\text{SC},M}(\omega) + b_{\text{SC},M,i}(\omega) = \\ \boldsymbol{w}_i^{\mathsf{T}}\boldsymbol{x} + b_i\}\big) = 1. \end{aligned} \tag{9}$$

In other words, we proved that as $M \to \infty$, the stochastic number $\boldsymbol{w}^{\mathsf{T}}_{\text{SC},M,i}(\omega) \cdot \boldsymbol{x}_{\text{SC},M}(\omega) + b_{\text{SC},M,i}(\omega)$ almost surely converges to $\boldsymbol{w}_i^{\mathsf{T}}\boldsymbol{x} + b_i$. □

Lemma 2. *If the sigmodial function $\sigma(t)$ has bounded derivative, then the stochastic number $\sigma(\boldsymbol{w}^{\mathsf{T}}_{SC,M,i}\boldsymbol{x}_{SC,M} + b_{SC,M,i})$ almost surely converge to the real value $\sigma(\boldsymbol{w}_i^{\mathsf{T}}\boldsymbol{x} + b_i)$ as the bit-stream length $M \to \infty$, .*

Proof. We have the following inequalities:

$$\begin{aligned} &\left|\sigma(\boldsymbol{w}^{\mathsf{T}}_{\text{SC},M,i}\boldsymbol{x}_{\text{SC},M} + b_{\text{SC},M,i}) - \sigma(\boldsymbol{w}_i^{\mathsf{T}}\boldsymbol{x} + b_i)\right| \\ &\leq \max_t \left|\sigma'(t) \cdot |\boldsymbol{w}^{\mathsf{T}}_{\text{SC},M,i}\boldsymbol{x}_{\text{SC},M} + b_{\text{SC},M,i} - \boldsymbol{w}_i^{\mathsf{T}}\boldsymbol{x} - b_i|\right| \\ &\leq \left(\max_t \left|\sigma'(t)\right|\right) \cdot \left|\boldsymbol{w}^{\mathsf{T}}_{\text{SC},M,i}\boldsymbol{x}_{\text{SC},M} + b_{\text{SC},M,i} - \boldsymbol{w}_i^{\mathsf{T}}\boldsymbol{x} - b_i\right| \end{aligned} \tag{10}$$

For the currently utilized activation functions, including sigmoid, tanh (hyperbolic tangent), ReLU functions, there is an upper bound on the derivatives. The maximum absolute value of the derivatives is often 1. Then, from the above Lemma 1 about the almost sure convergence of $\boldsymbol{w}^{\mathsf{T}}_{\text{SC},M,i}\boldsymbol{x}_{\text{SC},M} + b_{\text{SC},M,i}$ to $\boldsymbol{w}_i^{\mathsf{T}}\boldsymbol{x} + b_i$, we arrive at the almost sure convergence of $\sigma(\boldsymbol{w}^{\mathsf{T}}_{\text{SC},M,i}\boldsymbol{x}_{\text{SC},M} + b_{\text{SC},M,i})$. □

Based on the above lemmas and the original universal approximation theorem, we arrive at the following universal approximation theorem for SCNNs.

Theorem 1. *(Universal Approximation Theorem for SCNNs). For any continuous function $f(\boldsymbol{x})$ defined on I_n and any $\epsilon > 0$, we define an event that there exists an SCNN function $G_{SC,M}(\boldsymbol{x}_{SC,M})$ in the form of Eqn. (4) that satisfies*

$$\lim_{M\to\infty} |G_{SC,M}(\boldsymbol{x}_{SC,M}) - f(\boldsymbol{x})| < \epsilon. \tag{11}$$

This event is satisfied almost surely (with probability 1).

Proof. From the universal approximation theorem stated in (Cybenko 1989), we know that there exists a function $G(x)$ representing a deterministic neural network such that $|G(x) - f(x)| < \epsilon/2$ for all $x \in I_n$. For each positive integer M define $G_{\text{SC},M}(x)$ as the SCNN function obtained by replacing each parameter of $G(x)$ with its M-bit stochastic representation. Then we have

$$\begin{aligned} &|G_{\text{SC},M}(\boldsymbol{x}_{\text{SC},M}) - f(\boldsymbol{x})| \\ &= |G_{\text{SC},M}(\boldsymbol{x}_{\text{SC},M}) - G(\boldsymbol{x}) + G(\boldsymbol{x}) - f(\boldsymbol{x})| \\ &\leq |G_{\text{SC},M}(\boldsymbol{x}_{\text{SC},M}) - G(\boldsymbol{x})| + |G(\boldsymbol{x}) - f(\boldsymbol{x})| \end{aligned} \tag{12}$$

Applying Lemma 1 and 2, we can bound the first term as

$$
\begin{aligned}
&|G_{\text{SC},M}(\boldsymbol{x}_{\text{SC},M}) - G(\boldsymbol{x})| = \\
&\Bigg| \sum_{i=1}^{N} \alpha_i \sigma(\boldsymbol{w}_{\text{SC},M,i}^{\mathsf{T}} \boldsymbol{x}_{\text{SC},M} + b_{\text{SC},M,i}) - \\
&\sum_{i=1}^{N} \alpha_i \sigma(\boldsymbol{w}_i^{\mathsf{T}} \boldsymbol{x} + b_i) \Bigg| \\
&\leq \sum_{i=1}^{N} \left| \alpha_i \left[\sigma(\boldsymbol{w}_{\text{SC},M,i}^{\mathsf{T}} \boldsymbol{x}_{\text{SC},M} + b_{\text{SC},M,i}) - \sigma(\boldsymbol{w}_i^{\mathsf{T}} \boldsymbol{x} + b_i) \right] \right| \\
&\leq \sum_{i=1}^{N} |\alpha_i| \left| \sigma(\boldsymbol{w}_{\text{SC},M,i}^{\mathsf{T}} \boldsymbol{x}_{\text{SC},M} + b_{\text{SC},M,i}) - \sigma(\boldsymbol{w}_i^{\mathsf{T}} \boldsymbol{x} + b_i) \right|
\end{aligned}
\tag{13}
$$

where N is the size of the hidden layer in the neural network represented by $G(\boldsymbol{x})$, and α_i is the i-th weight in the output layer.

Deriving from Lemma 2, we know that for $\frac{\epsilon}{2\sum_{i=1}^{N} \alpha_i} > 0$, with probability 1 there exists M_{min} such that

$$
\left| \sigma(\boldsymbol{w}_{\text{SC},M,i}^{\mathsf{T}} \boldsymbol{x}_{\text{SC},M} + b_{\text{SC},M,i}) - \sigma(\boldsymbol{w}_i^{\mathsf{T}} \boldsymbol{x} + b_i) \right| < \frac{\epsilon}{2\sum_{i=1}^{N} \alpha_i} \tag{14}
$$

for $M \geq M_{min}$. Incorporating into Eqn. (13) we have $|G_{\text{SC},M}(\boldsymbol{x}_{\text{SC},M}) - G(\boldsymbol{x})| < \frac{\epsilon}{2}$. Further incorporating into Eqn. (12) we have $|G_{\text{SC},M}(\boldsymbol{x}_{\text{SC},M}) - f(\boldsymbol{x})| < \epsilon$ for $M \geq M_{min}$. Thereby we have formally proved that universal approximation theorem holds with probability 1 for SC-NNs. □

Besides the universal approximation property, it is also critical to derive an appropriate bound for bit length M in order to provide insights for the actual neural network implementations. The next theorem gives an explicit bound on the bit length for close approximation with high probability.

Theorem 2. *For the SCNN function $G_{SC,M}$ in Theorem 1, let M be any integer that satisfies*

$$
M > \frac{(n+1)^2 \cdot N^2}{\epsilon^2 \delta}. \tag{15}
$$

Then with probability at least $1 - \delta$, $|G_{SC,M}(\boldsymbol{x}_{SC,M}) - f(\boldsymbol{x})| < \epsilon$ holds for all $x \in I_n$.

Proof. Different from the above proof based on the strong law of large numbers (almost sure convergence), deriving bounds is more related to the weak law (convergence in probability). As the former case will naturally ensure the latter, we have the following convergence in probability property: For any $\epsilon, \delta > 0$, there exists M_{min}^{δ}, such that for any $M \geq M_{min}^{\delta}$, we have

$$
Pr\Big\{ |G_{\text{SC},M}(\boldsymbol{x}_{\text{SC},M}) - f(\boldsymbol{x})| < \epsilon \Big\} > 1 - \delta \tag{16}
$$

Based on a reverse order of the above proof of universal approximation, the above inequality is satisfied when we have

$$
Pr\Big\{ |G_{\text{SC},M}(\boldsymbol{x}_{\text{SC},M}) - G(\boldsymbol{x})| < \frac{\epsilon}{2} \Big\} > 1 - \delta \tag{17}
$$

Furthermore, the above inequality is satisfied when we have

$$
\begin{aligned}
&Pr\Bigg\{ \left| w_{\text{SC},M,i}^{j} x_{\text{SC},M}^{j} - w_i^j x^j \right| < \frac{\epsilon}{2(n+1) \cdot \sum_i \alpha_i} \Bigg\} \\
&> 1 - \delta
\end{aligned}
\tag{18}
$$

As each bit in stochastic number $w_{\text{SC},M,i}^{j} x_{\text{SC},M}^{j}$ satisfies a binary distribution with expectation $w_i^j x^j$, the maximum variance is $\frac{1}{4}$. Due to i.i.d. property, the maximum variance (σ^2) of $w_{\text{SC},M,i}^{j} x_{\text{SC},M}^{j}$ is $\frac{1}{4M}$. According to the Chebyshev's inequality $Pr\big(\|X - \mu\| \geq k\sigma\big) \leq \frac{1}{k^2}$, we let $\frac{1}{k^2} = \delta$ and obtain

$$
\frac{1}{2\sqrt{\delta M}} = \frac{\epsilon}{2(n+1) \cdot \sum_i \alpha_i} \tag{19}
$$

Then we derive an upper bound of M_{min}^{δ} as

$$
M_{min}^{\delta} \leq \frac{(n+1)^2 \cdot \left(\sum_i \alpha_i \right)^2}{\epsilon^2 \delta} \leq \frac{(n+1)^2 \cdot N^2}{\epsilon^2 \delta} \tag{20}
$$

□

As indicated in the universal approximation theory, any continuous or measurable function in the domain $[0, 1]^n$ can be approximated, and this domain matches the input domain of stochastic computing. As a result this bound is a general bound with broad applicability. We conducted multiple experiments and all have validated this statement. We demonstrate two experiments shown in (a), (b) in Figure 3. It shows the bound vs. Monte Carlo experiments results, on the bit-sequence length M as a function of neuron number N in the hidden layer. (a) is result on linear function and (b) on sinusoidal function. The other parameters $\epsilon = 0.2$ and $\delta = 0.1$.

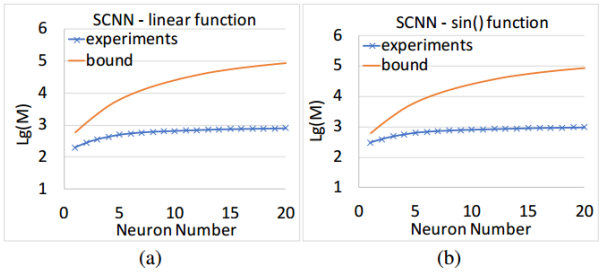

Figure 3: Illustration of the effect of bound on two types of functions.

We can show that (i) the bound is the same for different functions and is a general bound; (ii) the bound is indeed an effective upper bound on both types of functions (and also in the other types of functions in our experiments); (iii) the bound is tighter with smaller number of hidden neurons and when the approximation function has higher degree of non-linearity.

Universal Approximation of BNNs and Equivalence between SCNNs and BNNs

In this section we start from the formal definition of BNNs of interests and then state the universal approximation property. Similar to the definition of SCNNs, here we focus on an "ideal" BNN that is independent of actual BNN implementations. An illustration is shown in Figure 4.

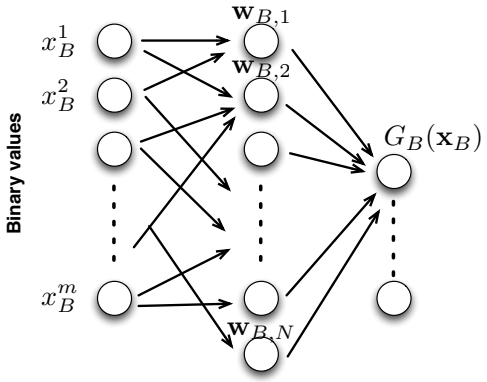

Figure 4: The structure of BNN of interest.

Definition 3. *A BNN of interest is defined as a function $G_B(\boldsymbol{x}_B)$, satisfying:*

$$G_B(\boldsymbol{x}_B) = \sum_{i=1}^{N} \alpha_i \sigma(\boldsymbol{w}_{B,i}^{\mathsf{T}} \boldsymbol{x}_B + b_{B,i}) \tag{21}$$

where the input vector $\boldsymbol{x}_B$ and weight vector $\boldsymbol{w}_{B,i}$ for each i represent vectors of binary values. Let m denote the dimensionality in these two vectors (dimension of inputs). $b_{B,i}$ is a binary bias value. The computation in $\boldsymbol{w}_{B,i}^{\mathsf{T}} \boldsymbol{x}_B + b_{B,i}$ follows the BNN rules as described before. Similar to SCNNs, we also consider here accurate activation and output layer calculation. This is reasonable and also applied in BNN deployments because the output layer size is typically very small.

The Equivalence of SCNNs and BNNs: The BNNs can be transformed into SCNNs, and vice versa. We illustrate the former case as an example. Let M denote the length of stochastic number and the number of inputs in SCNN becomes $n = \dfrac{m}{M}$. Then the first input stochastic number $x_{\text{SC}}^1 = \boldsymbol{x}_{\text{B}}[1 : M]$ (i.e., the first M bits in $\boldsymbol{x}_{\text{B}}$), the second input stochastic number $x_{\text{SC}}^2 = \boldsymbol{x}_{\text{B}}[M + 1 : 2M]$, and so on. This also applies to the weight stochastic numbers. The bias stochastic number $b_{\text{SC},i}$ can be a sign extension of $b_{\text{B},i}$. In this way the BNN is transformed into SCNN described in Definition 2. The transformation from SCNN to BNN is similar.

Because of the universal approximation property of SCNNs and the equivalence of BNNs, we arrive at the universal approximation for BNNs as well.

Theorem 3. *(Universal Approximation Theorem for BNNs). For any continuous function $f(\boldsymbol{x})$ defined on I_n, $\epsilon > 0$, we define an event that there exists an BNN function $G_B(\boldsymbol{x}_B)$ in the form of Eqn. (21) that satisfies*

$$\lim_{m \to \infty} |G_B(\boldsymbol{x}_B) - f(\boldsymbol{x})| < \epsilon. \tag{22}$$

This event is satisfied almost surely (with probability 1).

Proof. Apply Theorem 1 to obtain a close approximation of $f(x)$ with SCNN functions, then build a BNN function that closely approximations the SCNN function. □

The equivalence in SCNNs and BNNs also leads to the same bound, defined as the total number of input bits $m = n \cdot M$ required to achieve universal approximation. The reasoning is using proof by contradiction. Suppose that SCNNs have a lower bound, i.e., $n \cdot M_{min} < m_{min}$. Then there exists an SCNN with n inputs each with M_{min} bits satisfying the universal approximation property. From the above equivalance analysis we can construct a BNN with $M_{min} \cdot n$ input bits that also achieves such property, which is smaller and thus in contradiction with the bound m_{min}. And vice versa.

Energy Complexity and Hardware Design Implications

Energy Complexity Analysis

The *energy complexity*, as defined and described in (Martin 2001; Khude, Kumar, and Karnik 2005), specifies the asymptotic energy consumption with the growth of neural network size. It can be perceived as a multiplication of the time complexity and parallelism degree, and therefore is important for hardware implementations and evaluations. As an example, when the input size (number of bits) is n, a ripple carry adder has an energy complexity of O(n) whereas a multiplier has energy complexity of O(n^2). On the other hand, both of their time complexity is O(n). The reason is because the ripple carry adder is a sequential computation whereas the multiplier is a parallel computation.

Next we provide an analysis on the energy complexity of the key calculation in $\boldsymbol{w}_{\text{SC},M,i}^{\mathsf{T}} \boldsymbol{x}_{\text{SC},M} + b_{\text{SC},M,i}$ in SCNNs and $\boldsymbol{w}_{\text{B},\boldsymbol{i}}^{\mathsf{T}} \boldsymbol{x}_{\text{B}} + b_{\text{B},i}$ in BNNs. From the above equivalence analysis, we have $m = n \cdot M$ and $M \geq M_{min}$ for satisfying the universal approximation property. According to the hardware implementation details in SCNN and BNN, the multiplication of two bits has energy complexity of O(1), then the multiplication of two stochastic numbers has energy complexity of O(M). The addition of a set of n stochastic numbers has energy complexity of O(nM) using simple calculation units like multiplexers or energy complexity O($n \log n \cdot M$) using more accurate accumulation units like the approximate parallel counter (APC) (Kyounghoon Kim 2015). As a result, the overall energy complexity in $\boldsymbol{w}_{\text{SC},M,i}^{\mathsf{T}} \boldsymbol{x}_{\text{SC},M} + b_{\text{SC},M,i}$ is O(nM) (for less accurate results) or O($n \log n \cdot M$) (for more accurate results). For the whole layer with N neurons, the overall energy complexity is $n \cdot M \cdot N$ or $n \log n \cdot M \cdot N$. The energy complexity for BNNs with $m = n \cdot M$ is the same due to the equivalence.

In fact, one of the key of this work is to show that SCNN and BNN are equivalent, in both functionality and energy complexity. This will shed some light on the neural network implementation because many related work think them not equivalent. As a result this work will draw the attention and provide some guideline about actual implementations.

Hardware Design Implications

Despite the same energy complexity, the actual hardware implementations of SCNNs and BNNs are different. As discussed before, SCNNs "stretch" in the temporal domain whereas BNNs span in the spatial domain. This is in fact the most important advantage of SCNNs. For BNN actual implementations, there is often an imbalance between the input I/O size and the computation requirement. The total computation requirement (please refer to the energy complexity discussion) is low, but the input requirement is huge even compared with conventional neural networks. This makes actual BNN implementations I/O bound systems, as in actual hardware tapeouts the I/O clock frequency is much lower compared with the computation clock frequency. In other words, the advantage of low and simple computation in BNNs is often not fully exploited in actual deployments (Zhao et al. 2017b; Umuroglu et al. 2017). This limitation can be effectively mitigated by SCNNs, because the spatial requirement is effectively traded-off with the temporal requirement. In this aspect SCNNs can use lower I/O account and thereby more effective usage of hardware computation and memory storage resources compared with BNN counterparts, thereby becoming more suitable for hardware implementations. Of course, the most desirable hardware design is platform-dependent, and will be an effective tradeoff between SCNN and BNN.

On the other hand, BNNs are more heavily optimized in literature compared with SCNNs. Especially, many research work (Courbariaux, Bengio, and David 2015; Hubara et al. 2016) are dedicated for effective training methods for BNNs making efficient usage of randomization techniques. On the other hand, the research on SCNNs are mainly from the hardware aspect (Ren et al. 2017; Yu et al. 2017; Li et al. 2017c). For training these work use a straightforward way of transforming directly (every input and weight) from conventional neural networks to stochastic numbers. As a result, it will be effective to take advantage of the training methods for BNNs, transform into SCNNs that are more suitable for hardware implementations using the method described in the equivalence analysis. In this way, we can effectively exploit the advantage while hiding weakness in both SCNNs and BNNs.

Conclusion

SCNNs and BNNs are low-complexity variants of deep neural networks that are particularly suitable for hardware implementations. In this paper, we conduct theoretical analysis and comparison between SCNNs and BNNs in terms of universal approximation property, energy complexity, and suitability for hardware implementations. More specifically, we prove that the "ideal" SCNNs and BNNs satisfy the universal approximation property with probability 1. The proof is conducted by first proving the property for SCNNs from the strong law of large numbers, and then using SCNNs as a "bridge" to prove for BNNs. Besides the universal approximation property, we also derived an appropriate bound for bit length M in order to provide insights for the actual neural network implementations. Based on the universal approximation property, we further prove that SCNNs and BNNs exhibit the same energy complexity. In other words, they have the same asymptotic energy consumption with the growing of network size. We also provide a detailed analysis of the pros and cons of SCNNs and BNNs for hardware implementations and present a way of effectively exploiting the advantage of each type while hiding the weakness.

Acknowledgments

This work is partly supported by the National Science Foundation (CNS-1704662, CCF-1733701, and CNS-1739748).

References

Andreou, A. S., and Chatzis, S. P. 2016. Software defect prediction using doubly stochastic poisson processes driven by stochastic belief networks. *Journal of Systems and Software* 122:72–82.

Barron, A. R. 1993. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information theory* 39(3):930–945.

Brown, B. D., and Card, H. C. 2001. Stochastic neural computation. i. computational elements. *IEEE Transactions on computers* 50(9):891–905.

Chen, Y.; Luo, T.; Liu, S.; Zhang, S.; He, L.; Wang, J.; Li, L.; Chen, T.; Xu, Z.; Sun, N.; et al. 2014. Dadiannao: A machine-learning supercomputer. In *Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture*, 609–622. IEEE Computer Society.

Courbariaux, M.; Bengio, Y.; and David, J.-P. 2015. Binaryconnect: Training deep neural networks with binary weights during propagations. In *Advances in neural information processing systems*, 3123–3131.

Csáji, B. C. 2001. Approximation with artificial neural networks. *Faculty of Sciences, Etvs Lornd University, Hungary* 24:48.

Cybenko, G. 1989. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems* 2(4):303–314.

David, J. P.; Kalach, K.; and Tittley, N. 2007. Hardware complexity of modular multiplication and exponentiation. *IEEE Transactions on Computers* 56(10).

Delalleau, O., and Bengio, Y. 2011. Shallow vs. deep sum-product networks. In *Advances in Neural Information Processing Systems*, 666–674.

Han, S.; Liu, X.; Mao, H.; Pu, J.; Pedram, A.; Horowitz, M. A.; and Dally, W. J. 2016. Eie: efficient inference engine on compressed deep neural network. In *Proceedings of the 43rd International Symposium on Computer Architecture*, 243–254. IEEE Press.

Hornik, K.; Stinchcombe, M.; and White, H. 1989. Multilayer feedforward networks are universal approximators. *Neural networks* 2(5):359–366.

Hubara, I.; Courbariaux, M.; Soudry, D.; El-Yaniv, R.; and Bengio, Y. 2016. Quantized neural networks: Training neural networks with low precision weights and activations. *arXiv preprint arXiv:1609.07061*.

Khude, N.; Kumar, A.; and Karnik, A. 2005. Time and energy complexity of distributed computation in wireless sensor networks. In *INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings IEEE*, volume 4, 2625–2637. IEEE.

Kyounghoon Kim, Jongeun Lee, K. C. 2015. Approximate de-randomizer for stochastic circuits. In *SoC Design Conference (ISOCC), 2015 International SoC Design Conference*. IEEE.

Li, H.; Wei, T.; Ren, A.; Zhu, Q.; and Wang, Y. 2017a. Deep reinforcement learning: Framework, applications, and embedded implementations. *arXiv preprint arXiv:1710.03792*.

Li, J.; Yuan, Z.; Li, Z.; Ding, C.; Ren, A.; Qiu, Q.; Draper, J.; and Wang, Y. 2017b. Hardware-driven nonlinear activation for stochastic computing based deep convolutional neural networks. In *Neural Networks (IJCNN), 2017 International Joint Conference on Neural Networks*, 1230–1236. IEEE.

Li, Z.; Ren, A.; Li, J.; Qiu, Q.; Yuan, B.; Draper, J.; and Wang, Y. 2017c. Structural design optimization for deep convolutional neural networks using stochastic computing. In *Proceedings of the Conference on Design, Automation & Test in Europe*, 250–253. European Design and Automation Association.

Li, B.; Najafi, M. H.; Yuan, B.; and Lilja, D. J. 2018. Quantized neural networks with new stochastic multipliers. In *2018 19th International Symposium on Quality Electronic Design (ISQED)*, 376–382. IEEE.

Liang, S., and Srikant, R. 2016. Why deep neural networks for function approximation? *arXiv preprint arXiv:1610.04161*.

Mahajan, D.; Park, J.; Amaro, E.; Sharma, H.; Yazdanbakhsh, A.; Kim, J. K.; and Esmaeilzadeh, H. 2016. Tabla: A unified template-based framework for accelerating statistical machine learning. In *High Performance Computer Architecture (HPCA), 2016 IEEE International Symposium on*, 14–26. IEEE.

Martin, A. J. 2001. Towards an energy complexity of computation. *Information Processing Letters* 77(2-4):181–187.

Merolla, P. A.; Arthur, J. V.; Alvarez-Icaza, R.; Cassidy, A. S.; Sawada, J.; Akopyan, F.; Jackson, B. L.; Imam, N.; Guo, C.; Nakamura, Y.; et al. 2014. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science* 345(6197):668–673.

Montufar, G. F.; Pascanu, R.; Cho, K.; and Bengio, Y. 2014. On the number of linear regions of deep neural networks. In *Advances in neural information processing systems*, 2924–2932.

Moons, B.; Uytterhoeven, R.; Dehaene, W.; and Verhelst, M. 2017. 14.5 envision: A 0.26-to-10tops/w subword-parallel dynamic-voltage-accuracy-frequency-scalable convolutional neural network processor in 28nm fdsoi. In *Solid-State Circuits Conference (ISSCC), 2017 IEEE International*, 246–247. IEEE.

Neftci, E. 2016. Stochastic neuromorphic learning machines for weakly labeled data. In *Computer Design (ICCD), 2016 IEEE 34th International Conference on*, 670–673. IEEE.

Ren, A.; Li, Z.; Ding, C.; Qiu, Q.; Wang, Y.; Li, J.; Qian, X.; and Yuan, B. 2017. Sc-dcnn: highly-scalable deep convolutional neural network using stochastic computing. In *Proceedings of the Twenty-Second International Conference on Architectural Support for Programming Languages and Operating Systems*, 405–418. ACM.

Telgarsky, M. 2016. Benefits of depth in neural networks. *arXiv preprint arXiv:1602.04485*.

Umuroglu, Y.; Fraser, N. J.; Gambardella, G.; Blott, M.; Leong, P.; Jahre, M.; and Vissers, K. 2017. Finn: A framework for fast, scalable binarized neural network inference. In *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 65–74. ACM.

Yu, J.; Kim, K.; Lee, J.; and Choi, K. 2017. Accurate and efficient stochastic computing hardware for convolutional neural networks. In *Computer Design (ICCD), 2017 IEEE International Conference on Computer Design*, 105–112. IEEE.

Yuan, Z.; Li, J.; Li, Z.; Ding, C.; Ren, A.; Yuan, B.; Qiu, Q.; Draper, J.; and Wang, Y. 2017. Softmax regression design for stochastic computing based deep convolutional neural networks. In *Proceedings of the on Great Lakes Symposium on VLSI 2017*, 467–470. ACM.

Zhao, L.; Liao, S.; Wang, Y.; Tang, J.; and Yuan, B. 2017a. Theoretical properties for neural networks with weight matrices of low displacement rank. *arXiv preprint arXiv:1703.00144*.

Zhao, R.; Song, W.; Zhang, W.; Xing, T.; Lin, J.-H.; Srivastava, M.; Gupta, R.; and Zhang, Z. 2017b. Accelerating binarized convolutional neural networks with software-programmable fpgas. In *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 15–24. ACM.