ストカスティック計算に基づくニューラルネットワークにおけるシグモイド関数の演算精度に関する解析

可児 冬弥[1,a)]　瀬戸 信明[2]　市原 英行[1,b)]　岩垣 剛[1,c)]　井上 智生[1,d)]

概要：ニューラルネットワーク（NN）における演算素子であるニューロンを，ストカスティックコンピューティング（SC）で実現する研究がなされている．活性化関数の 1 つであるシグモイド関数は，SC において線形有限状態機械（線形 FSM）で実現できるが，演算精度に関する解析が十分になされていない．本研究では，SC によるシグモイド関数のハードウェア実装を対象に，線形 FSM の状態数，ストカスティック数長（ビット系列長），入力ストカスティック数の無作為性（randomness）による影響と関係性を明らかにする．また，NN を SC で実装した際のニューロンの演算精度と認識精度に関する考察を行う．

キーワード：近似計算，線形有限状態機械，双曲線正接関数，カイ二乗値．

On the Accuracy of Sigmoid Functions in Stochastic-Computing-Based Neural Networks

Kani Toya[1,a)]　Seto Nobuaki[2]　Ichihara Hideyuki[1,b)]　Iwagaki Tsuyoshi[1,c)]　Inoue Tomoo[1,d)]

Abstract: Stochastic computing (SC) is an approximate computation with probablities. SC-based neural networks (NNs) recently draw attention as an efficient implementation of NNs. Sigmoid functions, which are used as an activation functions of neurons in NNs, can be implemented with linear finite state machines (linear FSMs) in terms of stochastic computing. However, the effect of the accuracy of the sigmoid functions on NNs is not sufficiently discussed. In this study, we focus on a hardware implementation of SC-based sigmoid functions and clarify the effect of three parameters of the hardware implementation: the number of states of linear FSMs, the length of the input stochastic numbers (or input binary sequences), and the randomness of the input stochastic numbers. Furthermore, we discuss the calculation (or recognition) accuracy of NNs with SC-based sigmoid functions.

Keywords: Approximate computing, linear finite state machines, hyperbolic tangent functions, and Chi-square values.

1. はじめに

ストカスティックコンピューティング（SC）は，ストカスティック数と呼ばれるビット系列における 1 の発生確率によって数値を表現し，確率的に演算を行う計算手法（Approximate Computing）である．SC は 1950 年頃に J. von Neumann によって提案され，2000 年頃に再注目された後，2010 年頃から関連論文が爆発的に増えている．現在 SC は，その省面積性などからニューラルネットワーク（NN）などの機械学習，脳型コンピュータ，LDPC などの符号化，デジタルフィルタ，画像処理や音声処理などへの適用が注目されている [1]．

SC では，線形有限状態機械（線形 FSM）を用いることで，指数関数やシグモイド関数などの関数を実現することができる [2], [3], [4], [5], [6]．特に，シグモイド関数は NN

[1] 広島市立大学大学院情報科学研究科
[2] 広島市立大学情報科学部
[a)] kani@cd.info.hiroshima-cu.ac.jp
[b)] ichihara@hiroshima-cu.ac.jp
[c)] iwagaki@hiroshima-cu.ac.jp
[d)] tomoo@hiroshima-cu.ac.jp

のニューロンの活性化関数に用いられるため，低面積なニューロンを実装するための 1 つの手法として線形 FSM を用いた SC 回路が利用されている [5], [6].

文献 [3] では，線形 FSM を順序回路で実現した SC 回路でシグモイド関数が近似できる原理と，その計算機実験結果が示されている．しかしながら，示されている実験結果は限定的であり，実験の前提条件も明らかにされていない．また，文献 [6] では，SC における線形 FSM を用いた近似シグモイド関数によってニューロンを実装しているが，演算精度に関する議論が十分にされていない．SC の効果を活かした最適な NN を設計するためには，線形 FSM によって実現された近似シグモイド関数が NN の認識精度に与える影響を明らかにする必要がある．

本研究では，文献 [2], [3], [4], [6] で提案されている線形 FSM によるシグモイド関数のハードウェア実装法に着目する．SC によるシグモイド関数のハードウェア実装を対象に計算機実験を行い，その演算精度が NN の認識精度に与える影響について考察を行う．一般に，演算精度は，線形 FSM の状態数，ストカスティック数の長さ（ビット系列長），入力ストカスティック数の無作為性（randomness）に影響を受けると考えられる．Verilog シミュレータを用いた実験結果から，(1) 入力ストカスティック数の無作為性が最も演算精度に影響を与えること，(2) ニューロン内で SC 実装を用いたときの演算精度の特徴，(3) 無作為性が高い場合のシグモイド関数を計算する SC 回路の特徴を明らかにする． また，このシグモイド関数を計算する SC 回路を用いたニューロン実装 [6] において，SC に基づくニューロンにおける演算誤差が，NN の認識精度に与える影響を明らかにする．

2. ストカスティック計算に基づくニューラルネットワーク

2.1 ストカスティック数と演算

ストカスティックコンピューティング（SC）は，**ストカスティック数** *(SN: Stochastic Number)* と呼ばれるビット系列の 1 の確率で数値の大きさを表現する．SN には，ユニポーラ（単極）表現とバイポーラ（双極）表現が存在する．数値 x を表現した長さ L のストカスティック数を $SN_x = sn_{x1}, sn_{x2}, \ldots, sn_{xL}$ と表現する．また，ストカスティック数 SN_x の 1 の存在確率を $P(SN_x)$ と記述すると，ユニポーラ表現では $x = P(SN_x)$，バイポーラ表現では $x = 2 \times P(SN_x) - 1$ という計算式により，それぞれ [0,1], [−1,1] の区間で数値 x を表現できる．例えば，00100010 という 8 ビットの SN は，8 ビット中に”1”が 2 つ存在しているため，ユニポーラ表現では 10 進数の $0.25(= \frac{2}{8})$ を，バイポーラ表現では 10 進数の $-0.5(= 2 \times \frac{2}{8} - 1)$ を表現している．一般的に，ビット系列長 L が長いほど表現したい数値を精度よく表現できる可能性が高まる（丸め誤差が小さくなる）.

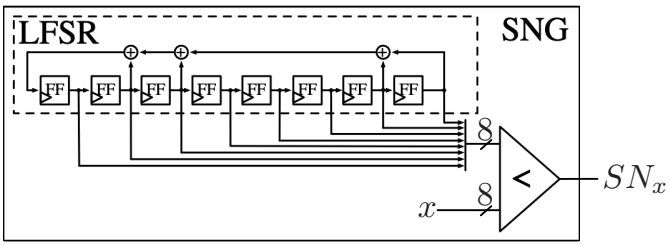

図 1 8 ビット LFSR を用いた SNG
Fig. 1 SNG with 8-bit LFSR.

2 進数から SN への変換は，SNG と呼ばれる回路で行う．SNG は乱数列を生成する乱数生成器と比較器からなり，乱数生成器の出力と変換したい 2 進数を比較器で比較することで行う．8 ビットの 2 進数を SN に変換するための SNG を図 1 に示す．一般的に乱数生成器には，線形フィードバックシフトレジスタ（LFSR : Linear Feedback Shift Register）がよく利用される．n 個の記憶素子を持つ LFSR は一般的に $2^n - 1$ の周期で乱数列を生成できる．図中では LFSR は 8 ビットの疑似ランダム数を毎サイクル出力するように設計されており（系列の周期 L は $2^8 - 1$），LFSR の出力値と SN に変換したい 2 進数の値（図中では x）を比較器により比較することにより，入力 x のほうが大きければ比較器の出力は 1 を，小さければ 0 を出力する．一方，SN から 2 進数への変換は，系列中の 1 の数を数えればよいため，一般的にはカウンタ（CNT）で行う．

SN を用いた基本的な演算は簡単な論理演算やシンプルな有限状態機械で行えることが知られている．ここでは，ユニポーラ表現における乗算と加算を紹介する．

(1) 乗算

ユニポーラ表現における 2 つの SN SN_x, SN_y の乗算は，図 2(a) に示すように，各ビットの論理積（AND 演算）で近似計算できる．$x \times y = P(SN_x \wedge SN_y) = P(sn_{x1} \wedge sn_{y1}, sn_{x2} \wedge sn_{y2}, \ldots, sn_{xL} \wedge sn_{yL})$．例えば，8 ビットの SN で表現された $0.25 = P(SN_{0.25}) = P(10001000)$ と $0.5 = P(SN_{0.5}) = P(01111000)$ の積は，2 つのストカスティック数の論理積 $SN_{0.25} \wedge SN_{0.5}$ を演算することにより $P(SN_{0.25} \wedge SN_{0.5}) = P(00001000) = 0.125$ となる．

(2) 加算

ユニポーラ表現における加算は，図 2(b) に示すように，，論理和（OR 演算）を用いるものと，マルチプレクサ（MUX）を用いたものがある．2 つのストカスティック数 SN_x, SN_y の各ビットの論理和は，$P(SN_x) + P(SN_y) - P(SN_x) \times P(SN_y)$ の確率を表現する SN となり，$P(SN_x) \times P(SN_y)$ が誤差となる．この誤差は，入力 SN である $P(SN_x)$ や $P(SN_y)$ が大きくなると無視できなくなる．また，MUX を用いた場合は，$P(SN_c)P(SN_x) + (1 - P(SN_c))P(SN_y)$ となる（x が MUX の 1 側に接続されている場合）．ここで，SN_c

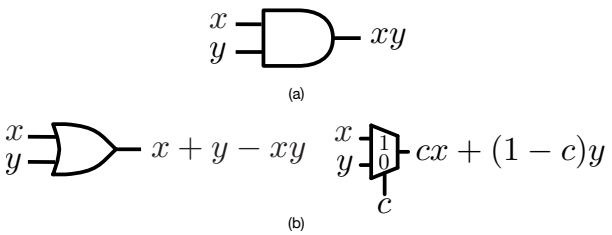

図 2　SC における乗算，加算
Fig. 2　multiplication and addition in SC.

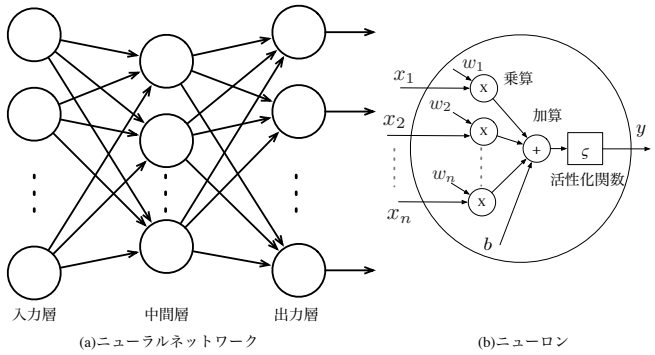

図 3　全結合型 2 層 NN とニューロン
Fig. 3　fully-connected two-layered NN and neuron.

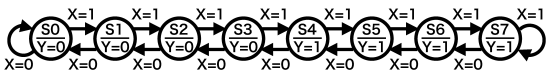

図 4　SC による近似シグモイド関数を実現する線形 FSM
Fig. 4　Linear FSM for approximate sigmoid function in SC.

は MUX の制御入力に与えられる SN である．このように，MUX を用いた場合は重み付きの加算となる．なお，$P(SN_x)+P(SN_y)$ となるような誤差や重み付きのない SC 加算は提案されていない．

2.2　ストカスティック計算に基づくニューラルネットワーク

ニューラルネットワーク（NN）は，ニューロンと呼ばれる演算素子からなる人間の脳を模した数理モデルである．全結合型の 2 層 NN の例を図 3 に示す．図 3(a) はネットワークの全体像を，図 3(b) はニューロンをそれぞれ示している．NN は入力層，中間層，出力層で構成されており，2 層 NN の場合中間層は 1 つとなる．

ニューロンは式 (1) のように，入力値 $x_1, x_2, ..., x_n$ と重み $w_1, w_2, ..., w_n$ の積およびバイアスと呼ばれる定数項 b を合計し，その結果を入力とした活性化関数の出力値 y を出力とする．活性化関数には式 (2) に示すような，シグモイド関数 $\varsigma_\alpha(x)$ などが用いられる．α は関数の傾きを決めるパラメータである．

$$y = \varsigma_\alpha(\sum_{i=1}^{n} x_i w_i + b) \tag{1}$$

$$\varsigma_\alpha(x) = \frac{1}{1+\exp(-\alpha x)} = \frac{\tanh(\frac{\alpha x}{2})+1}{2} \tag{2}$$

NN は多数のニューロンによって構成されるため，ニューロンの演算の一部または全てを SC で実現することによる低面積，低消費電力な NN の実現を試みる研究がされている [5], [6], [7].

これまでに提案されている SC に基づくニューロン実装 [5], [6], [7] の概要を表 1 に示す．ニューラルネットワークにおける重み w は一般的に正負の値をとるため，基本的にはバイポーラ表現を用い，そのため乗算は XNOR ゲートで行う [5]．一方，MUX による加算のスケーリングなどを考慮し，重みの絶対値表現を用いたユニポーラ表現による実装も存在する [6], [7]．この場合，乗算は AND ゲートで行う．加算は，SC による演算誤差を考慮してカウンタで行い 2 進数に戻すもの [5], [7] と，SN のまま SC で行うもの [6] が存在する．文献 [6] では，OR ゲートを用いて加算を行っている．活性化関数については，線形 FSM を用いて SC で行うもの [5], [6] と，文献 [7] のように省略するものもある．低面積性や耐故障性などの SC の利点を活かすには，文献 [6] のような，入力から出力までのすべてが SC で実装されていることが求められる．

3.　ストカスティック計算に基づくニューラルネットワークにおけるシグモイド関数の演算精度

本節では，3.1 節で線形 FSM を用いたシグモイド SC 回路について説明し，3.2 節でこの回路における演算精度に影響を与えるパラメータを挙げる．3.3 節では演算精度の解析に用いる，このシグモイド SC 回路を用いたニューロン [6] について述べる．

3.1　線形有限状態機械を用いたシグモイド関数実装

SC では，図 4 のような線形 FSM を用いることで，シグモイド関数，指数関数，双曲線正接（tanh）関数などの 1 入力関数を表現できる [2], [3], [4]．状態は入力 X が 0 ならば左，1 ならば右へ遷移する．この線形 FSM は，出力 Y が現状態だけで決まるムーア型であり，1 サイクル毎に 1 ビット入力 X が与えられ，1 ビット出力 Y を出力する同期式順序回路で実現できる．表 2 には例として，図 4 の線形 FSM に入力 X として SN 10110001（バイポーラ表現で 0）が与えられたとき，出力 Y として SN 10111000（ユニポーラ表現で 0.5）が出力されることを示している（初期状態は S_3 である）.

状態数 N の線形 FSM において，状態 S_0 から $S_{\frac{N}{2}-1}$ の出力 Y を 0 に，状態 $S_{\frac{N}{2}}$ から状態 S_{N-1} の出力 Y を 1 に設定すると，十分に SN 長 L が長い SN が与えられた場合，出力は

表 1　これまでに提案されている SC に基づくニューロン
Table 1　Neurons based on SC.

	乗算	加算	活性化関数	特徴
[5]	XNOR	近似並列カウンタ	線形 FSM（近似 ReLu）	SC による ReLu 関数の実装
[6]	AND	OR	線形 FSM（近似シグモイド）	すべて SC による実装 重みの正負分離
[7]	AND	並列カウンタ	-	重み操作による活性化関数の省略 重みの正負分離

表 2　線形 FSM を用いた回路の動作例
Table 2　Behavioral example of linear FSM.

時刻	入力 X	次状態	出力 Y
t_0	1	S_4	1
t_1	0	S_3	0
t_2	1	S_4	1
t_3	1	S_5	1
t_4	0	S_4	1
t_5	0	S_3	0
t_6	0	S_2	0
t_7	1	S_3	0

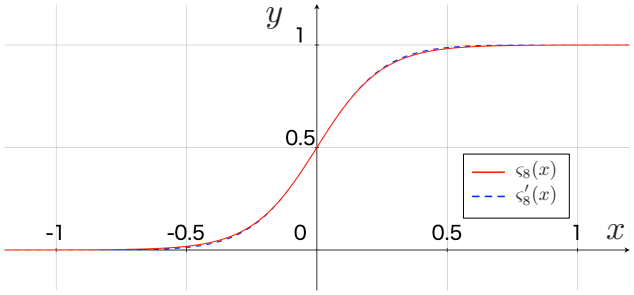

図 5　シグモイド関数 $\varsigma_8(x)$ と近似シグモイド関数 $\varsigma'_8(x)$
Fig. 5　Sigmoid function $\varsigma_8(x)$ and approximate sigmoid function $\varsigma'_8(x)$.

$$y = \varsigma'_N(x) = \sum_{i=N/2}^{N-1} \frac{(\frac{x}{1-x})^i}{\sum_{j=0}^{N-1}(\frac{x}{1-x})^j} \simeq \varsigma_N(x) \qquad (3)$$

となり，式 (2) に示したような，x を入力とするゲイン $\alpha = N$ のシグモイド関数 $\varsigma_\alpha(x)$ を近似的に実現することができる [2], [3]．このとき，線形 FSM の入力 X はバイポーラ表現であり，出力 Y はユニポーラ表現となっている．

図 4 の線形 FSM は $N = 8$ のため，$\varsigma_8(x)$ を近似計算できる．図 5 に $\varsigma_8(x)$ と，この線形 FSM で実現される $\varsigma'_8(x)$ を示す．本研究では，シグモイド関数を計算するための線形 FSM による順序回路を「シグモイド SC 回路」と呼ぶ．

3.2　演算精度に影響を与えるパラメータ

一般的に，線形 FSM を用いた SC 回路における演算精度は，(1) 線形 FSM の状態数 N，(2) ストカスティック数の長さ L，(3) 入力ストカスティック数の無作為性（randomness）R の影響を受けると考えられる．

(1) 線形 FSM の状態数 N

状態数 N は，3.1 で述べたように，近似シグモイド関数

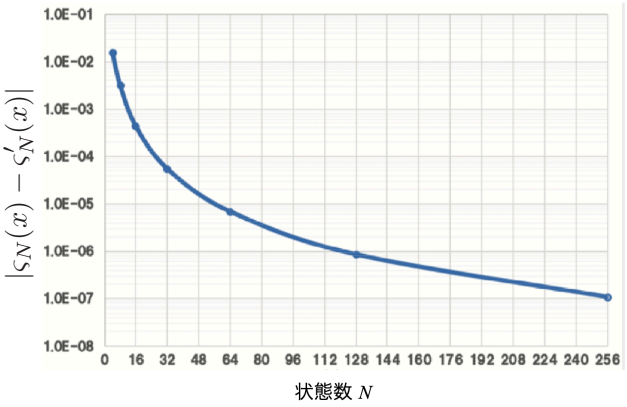

図 6　状態数 N に対する近似誤差 $|\varsigma_N(x) - \varsigma'_N(x)|$
Fig. 6　Approximate error $|\varsigma_N(x) - \varsigma'_N(x)|$ for the number N of states.

$\varsigma'_N(x)$ のゲインを決めるパラメータである．状態数 N の線形 FSM で表現できる $\varsigma'_N(x)$（式 (3)）とシグモイド関数 $\varsigma_N(x)$（式 (2)）の差（近似誤差）を図 6 に示す．この図から，N が小さいほど，近似誤差は大きくなることがわかる．例えば，$N = 8$ でのシグモイド関数 $\varsigma_8(x)$ との近似誤差は 3.0×10^{-3} 程度にまで大きくなる．

このように，状態数 N は直接的に演算精度を決定するわけではないが，近似誤差が異なるという点で間接的に影響する．例えば，同じ長さ L のストカスティック数でも N が大きいときと小さいときでは演算精度が異なると考えられる．

(2) ストカスティック数（SN）の長さ L

SN の長さ L は，2.1 で述べたように SC 演算の丸め誤差に影響するだけでなく，確率的に直接演算誤差に影響を与える．つまり，L を大きくすれば，丸め誤差の現象と，大数の法則により線形 FSM の演算結果は $\varsigma'_N(x)$ に近づく．

(3) 入力ストカスティック数の無作為性 R

入力 SN の無作為性（randomness）R とは，SN 内の 0 と 1 の出現の仕方に法則性（規則性）がなく，予測が不可能であることを意味する．ただしここでは，擬似乱数を用いて SN を生成しているため，真の意味で無作為性があるわけではなく（法則性はあり，予測は可能である），無作為性があるように見えるかどうかを意味している．無作為性が高いほど線形 FSM は期待値に近い動作をすると考えられる．

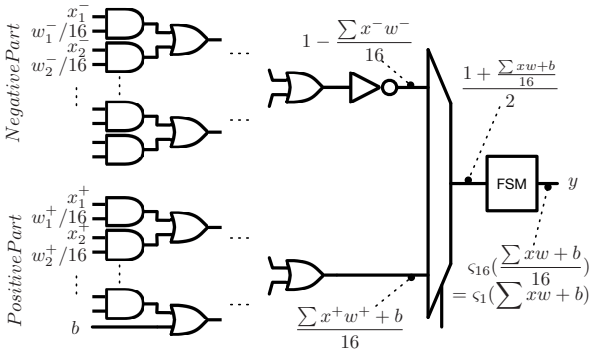

図 7 SC に基づくニューロン [6]
Fig. 7 SC-based neuron[6].

本研究では，無作為性 R を，シリアルテストによって得られたカイ二乗値を用いて定義する．シリアルテストとは，SN を長さ m で重なり o $(o < m)$ を持ったブロックに分け，それぞれのブロックの観測度数を数え，観測度数と期待度数の差からカイ二乗値

$$\chi^2 = \sum \frac{(\text{観測度数} - \text{期待度数})^2}{\text{期待度数}} \tag{4}$$

を求めることで，無作為性があるかどうかを調べる検定である [8]．つまり，$R = \chi^2$ となり，R の値が小さいほど無作為性の高いビット系列となる．

例として，1 の発生確率が $P_x = \frac{3}{9}$ である 9 ビットの 2 つの SN 001001001, 000000111 に対してブロック長 $m = 2$, 重なり $o = 1$ のときの無作為性 R を説明する．9 ビットの SN を $m = 2, o = 1$ のブロックに分割すると，8 つのブロックに分かれる．ブロックの種類は 00, 01, 10, 11 の 4 通りである．1 の発生確率は $\frac{3}{9}$ なので，0 の発生確率は $1-\frac{3}{9}$ となり，それぞれの期待度数は（00 の期待度数）$= 8\times(1-\frac{3}{9})\times(1-\frac{3}{9}) = \frac{32}{9}$，（01 の期待度数）$= 8 \times (1 - \frac{3}{9}) \times \frac{3}{9} = \frac{16}{9}$，（10 の期待度数）$= 8 \times \frac{3}{9} \times (1 - \frac{3}{9}) = \frac{16}{9}$，（11 の期待度数）$= 8 \times \frac{3}{9} \times \frac{3}{9} = \frac{8}{9}$ となる．001001001 は 00, 01, 10, 00, 01, 10, 00, 01 の 8 つのブロックに分かれるため，カイ二乗値は $\chi^2 = \frac{(3-\frac{32}{9})^2}{\frac{32}{9}} + \frac{(3-\frac{16}{9})^2}{\frac{16}{9}} + \frac{(2-\frac{16}{9})^2}{\frac{16}{9}} + \frac{(0-\frac{8}{9})^2}{\frac{8}{9}} \simeq 1.84$ となる．一方， 000000111 は 00, 00, 00, 00, 00, 01, 11, 11 の 8 つのブロックに分かれるため，カイ二乗値は $\chi^2 = \frac{(5-\frac{32}{9})^2}{\frac{32}{9}} + \frac{(1-\frac{16}{9})^2}{\frac{16}{9}} + \frac{(0-\frac{16}{9})^2}{\frac{16}{9}} + \frac{(2-\frac{8}{9})^2}{\frac{8}{9}} \simeq 4.09$ となる．この値は 001001001 のカイ二乗値よりも大きいため，000000111 のほうが無作為性の低い SN だということがわかる．

3.3 SC に基づくニューロン実装

シグモイド SC 回路の演算精度に関する議論を行う，文献 [6] の SC に基づくニューロン（SC ニューロン）の実装を考える．この SC ニューロンのアーキテクチャを図 7 に示す．図 7 は定数項 b が正のときの例である．

この SC ニューロンは，学習済みの重みの符号に基づく正負分離によるユニポーラ表現の積和実装である．w_i^+ は正の重み，w_j^- は負の重みを意味する．x_i^+ と x_j^- はそれぞれの重みに対応する入力であり，すべて正の値である．AND ゲートにより入力 x と重み w の乗算を行い，OR ゲートにより加算を $NegativePart$ と $PositivePart$ のそれぞれで行っている．その後，$NegativePart$ の積和結果は NOT ゲートより論理反転される．これらを入力とする制御信号の 1 の確率が 1/2 である MUX の出力は $\frac{1+\frac{\sum xw+b}{16}}{2}$ となる．この SN はバイポーラ表現の積和結果を表している．この積和結果の SN を入力とする線形 FSM によりシグモイド関数の演算を行い，ニューロンの出力とする．

また，この SC ニューロンでは演算誤差を小さくするための工夫がなされている．2.1 で説明したとおり，OR ゲートによる加算は近似的な加算であり，OR ゲートの入力が大きな値になると近似誤差は大きくなる．そこで，学習済みの重みをあらかじめ 1/16 することにより，OR ゲートによる加算の演算誤差を小さくしている．このとき，積和結果も 1/16 された値になるため，16 状態の線形 FSM（ゲイン α が 16 であるシグモイド関数を表現している）を用いることによりこれを打ち消している．

4. 実験的考察

本節では，線形 FSM を用いたシグモイド SC 回路に着目した 2 つの実験を行う．4.1 節では，シグモイド SC 回路の演算精度について，4.2 節ではシグモイド SC 回路を用いた SC ニューロンの演算精度と NN の認識精度の関係についてそれぞれ実験的考察を行う．

4.1 シグモイド SC 回路での実験

3.2 で述べた 3 つのパラメータを変更した計算機実験を行うことで，パラメータ間の関係性や定量的な誤差の評価を行った．シグモイド関数 SC 回路の入力 x は，集合 X ={$-256/256, -254/256, -252/256, \ldots, 254/256, 256/256$} の要素とする．各入力ごとの試行回数は 100 回とし，試行回数 i 回目の出力を $y_i(x)$, $i \in \{1, 2, \ldots, 100\}$ と表す．これの平均値 $\overline{y(x)} = \frac{1}{100}\sum_{i=1}^{100} y_i(x)$ とシグモイド関数 $\varsigma_N(x)$ との差を演算誤差とする．

本実験では，図 8 に示すように，シグモイド SC 回路が利用される 2 種類の状況を想定する．1 つは，(a) シグモイド SC 回路単体での演算精度の評価を行うために，SC 回路の入出力にそれぞれ SNG と CNT を繋いだ回路であり，もう 1 つは (b) ニューロン内でシグモイド SC 回路が利用される状況であり，シグモイド SC 回路と SNG の間に積和回路が挿入された回路である．(a) では，無作為性を変更するために，SNG の乱数生成源が (a-1) メルセンヌ・ツイスタ（MT）[9] であるものと，(a-2) LFSR であるものを想定する．シグモイド SC 回路は Verilog HDL にて記述

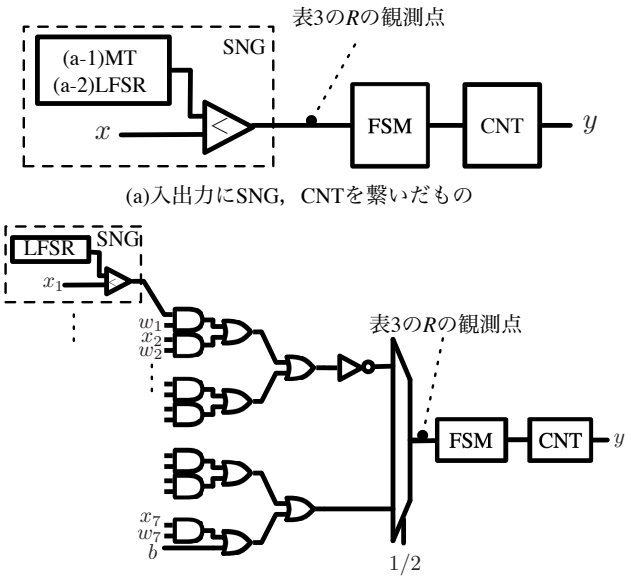

図 8 2 つの実験回路.
Fig. 8 Two circuits.

表 3 SN の無作為性 R（カイ二乗値）の平均値
Table 3 Randomness R (average).

(a-1) LFSR	(a-2) MT	(b) ニューロン内
118.71	6.70	7.82

し，Verilog シミュレーター Icarus Verilog [10] を用いてシミュレーションを行った．

これらの実験回路でパラメータを変化させた実験結果からわかったことを述べる．

(1) 入力 SN の無作為性の影響が大きい

表 3 に $L = 256$ のときの，(a-1) SNG の乱数源に LFSR を用いた場合，(a-2) MT を用いた場合に生成される SN の R（カイ二乗値）の平均値を示す．表 3 から，(a-1) より (a-2) のほうがカイ二乗値が非常に小さいので，入力 SN の無作為性が高いことがわかる．

図 9 にシグモイド関数と各回路での演算結果（平均値）を示す．状態数 N は 8，SN 長 L は 256，512，1024 の結果を示している．図 9(a-1) のように入力 SN の無作為性が高い場合は $N = \alpha$ のシグモイド関数を精度よく近似できるが，(a-2) のように無作為性が低い場合は L を大きくしても，シグモイドとの誤差が大きい傾向がある．よって入力 SN の無作為性が，演算精度に最も影響を与えることがわかる．LFSR はストカスティックコンピューティングで最もよく使われている乱数源であるが，(a) のように単体でシグモイド SC 回路を利用するときに LFSR を用いる場合は，必要な演算精度が得られているか十分に注意する必要がある．

(2) ニューロン内で SC 実装を用いたときの演算精度の特徴

ニューロン内で SC 実装を用いたときの結果もまた図 9 に示している．この図から，ニューロン内で SC 実装を用いたときは，ニューロンの入力 SNG において LFSR を乱数源に用いているにも関わらず，演算精度が高いことがわかる．また，表 3 にはシグモイド SC 回路の入力部分の SN を観測した結果を示している．この表からわかるように，シグモイド SC 回路の前段にある積和 SC 回路により，線形 FSM の入力 SN の無作為性が向上し，その結果演算精度が向上していると考えられる．

(3) 無作為性が高い場合の特徴

表 4 には無作為性が十分に高い (a-1) において，状態数 N と SN 長 L を変化したときの，誤差の平均値（err）と標準偏差の平均値（sd）を示している．それぞれ，以下の式で求めた．

$$err = \frac{1}{|X|}\sum_{x\in X}|\overline{y(x)} - \varsigma_N(x)|$$

$$sd = \frac{1}{|X|}\sum_{x\in X}\sqrt{\frac{1}{100}\sum_{i=1}^{100}(y_i(x) - \overline{y(x)})^2}$$

また，表 4 の最後の列には，図 6 に示した $|\varsigma'_N(x) - \varsigma_N(x)|$ の値を示している．表 4 より，$|\varsigma'_N(x) - \varsigma_N(x)|$ で求まる近似誤差は，状態数 N が大きくなるにつれて減少しているにもかかわらず，同じ L でみたときの誤差および標準偏差は N が大きくなるにつれて増加していることがわかる．このことから，N の大きいシグモイド SC 回路が N の小さいシグモイド SC 回路と同等程度の演算精度を得るには，十分に長い L を確保する必要があるといえる．

4.2 ニューラルネットワークでの実験

層構成 784-100-10 の全結合型 2 層 NN において，代表的な画像認識のデータセットである MNIST データセット [11] を用いて Python で記述し学習を行った．MNIST データセットは 0 から 9 までの手書き数字画像のデータセットである．画像は 8 ビットのグレースケールで幅，高さともに 28 画素である．学習の結果，テストデータの認識精度が 0.979 となる重みを獲得した．

この重みを用いて，3.3 で述べた SC ニューロンによってこの NN を実装した．4.1 と同様に Verilog HDL および Icarus Verilog を用いてシミュレーションを行った．入力となるテストデータは 1000 枚の画像データを用いている．線形 FSM の状態数は 8，16，32 とし，それぞれすべての重みを 1/8，1/16，1/32 として実装した．実験結果を表 5 に示す．この表は，中間層における AND と OR ゲートによる積和結果の演算誤差，線形 FSM の出力（ニューロンの出力）の演算誤差および NN の認識精度から構成されている．演算誤差は，観測点に CNT を繋いだシミュレーション結果と Python による計算結果の平均絶対誤差である．

表 5 より，状態数 $N = 8$ のときは，L が小さいとき，ニューロン単位（線形 FSM の出力）での演算誤差が比較

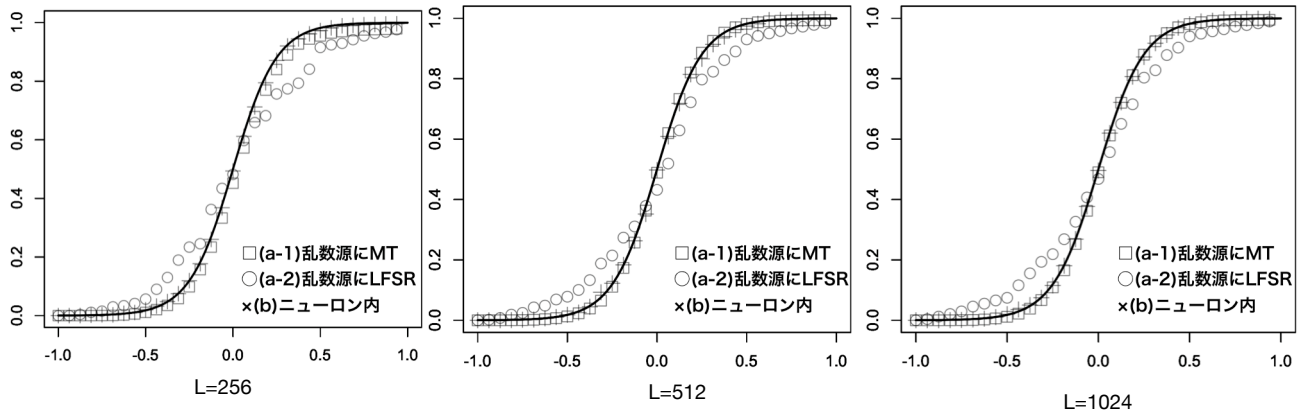

図 9 シグモイド関数と各回路の演算結果（平均値）．$N = 8$，$L = 256, 512, 1024$

Fig. 9 Sigmoid functions and calculation results (average) of approximate sigmoid functions for three circuits. $N = 8$, $L = 256, 512, 1024$

表 4 乱数源 MT(a-1) において N と L を変えたときの誤差 (err) と標準偏差 (sd) の平均値 ($\times 10^{-2}$)

Table 4 Average err and sd for MT (a-1).

	$L = 256$		$L = 512$		$L = 1024$		$L = 2048$		$L = 4096$		$\lvert\varsigma_N(x) - \varsigma'_N(x)\rvert$
N	err	sd	err	sd	err	sd	err	sd	err	sd	err
8	0.86	3.10	0.45	2.09	0.39	1.36	0.26	0.97	0.03	0.72	0.31
16	1.06	3.47	0.45	2.50	0.23	1.60	0.12	1.08	0.06	0.77	0.04
32	1.81	3.51	0.92	2.79	0.50	1.82	0.23	1.11	0.13	0.79	0.005
64	2.70	3.57	1.68	2.88	1.00	2.09	0.44	1.27	0.25	0.88	0.0007
256	3.58	3.57	2.45	2.69	1.75	2.17	1.07	1.37	0.69	0.99	0.00001

的小さく，認識精度は他の N の実装と比べて高いことがわかる．一方，$N = 16, 32$ のときは，L が小さいときは，重みを小さくしたことにより積和の演算誤差は小さいものの，線形 FSM の出力の演算誤差は大きくなり，結果として認識精度が低下している．よって，高い認識精度を得るためには L を 512 以上にする必要があることがわかる．

5. まとめ

本研究では，線形 FSM によるシグモイド関数のハードウェア実装法 [2], [3], [4], [6] に注目し，演算精度に関する考察を行った．実験によって，十分長いストカスティック長においても，線形 FSM の入力ビット系列の無作為性が低い場合は，文献 [3] で示されているような $\varsigma_N(x) \simeq \varsigma'_N(x)$ が成り立たないほどの演算誤差が生まれてしまうことがわかった．ただし，乱数源にメルセンヌツイスターを用いることや，ニューロンのような SNG と線形 FSM の間に何かしらの演算装置が存在する場合，無作為性が高くなり回避できることがわかった．一方，ニューロン内であっても，状態数 N の大きい線形 FSM において十分な演算精度を得るには，長いストカスティック長が必要であることがわかった．ニューラルネットワークにおける実験からも，状態数 N の小さい線形 FSM を用いて実装したものが短いストカスティック長のときに認識精度が高く，ストカスティック長が長い場合は，状態数 N の大きい線形 FSM を用いた実装の認識精度が高くなることがわかった．今後は，明らかになった 3 つの要素の影響と関係性に基づいて，最適なニューロンおよび NN の設計方法を考察する予定である．

謝辞 本研究は JSPS 科研費 JP16K00080 及び 19K11882 の助成を受けたものである．

参考文献

[1] A. Alaghi and J. P. Hayes: *Survey of Stochastic Computing*, Trans. Embed. Comput. Syst., vol. 12, no. 2, pp. 1–19, May 2013.

[2] B. D. Brown and H. C. Card: *Stochastic Neural Computation I: Computational Elements*, IEEE Trans. on Computers, vol. 50, pp. 891–905, Sep. 2001.

[3] Peng Li, Weikang Qian, Marc D. Riedel, Kia Bazargan, David J. Lilja: *The Synthesis of Linear Finite State Machine-Based Stochastic Computation Elements*, Proc. of 17th Asia and South Pacific Design Automation Conference, pp. 757–762, Jan. 2012.

[4] Peng Li, David J. Lilja, Weikang Qian, Kia Bazargan, Marc Riedel: *The synthesis of Complex Arithmetic Computation on Stochastic Bit Streams Using Sequential Logic*, Proc. of 2012 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), pp. 480–487, Nov. 2012.

[5] Z. Li et al.: *HEIF: Highly Efficient Stochastic*

表 5　SC による演算誤差と NN の認識精度

Table 5　Calculation erros and recognition accuracy of NNs

	$L = 64$			$L = 128$			$L = 256$			$L = 512$			$L = 1024$		
N	積和	FSM	認識精度	積和	FSM	認識精度	積和	FSM	認識精度	積和	FSM	認識精度	積和	FSM	認識精度
8	0.189	0.178	0.789	0.168	0.150	0.848	0.161	0.138	0.893	0.156	0.135	0.871	0.155	0.140	0.844
16	0.122	0.290	0.426	0.085	0.185	0.749	0.064	0.133	0.903	0.057	0.108	0.934	0.051	0.105	0.929
32	0.077	0.368	0.094	0.060	0.293	0.169	0.041	0.208	0.650	0.028	0.144	0.934	0.022	0.118	0.924

Computing-Based Inference Framework for Deep Neural Networks, IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, vol. 38, no. 8, pp. 1543-1556, Aug. 2019.

[6] B. Li, M. H. Najafi, D. Lilja: *An FPGA implementation of a Restricted Boltzmann Machine classifier using stochastic bit streams*, IEEE Proc. ASAP, pp. 68-69, 2015.

[7] B. Li, Y. Qin, B. Yuan, D. Lilja: *Neural Network Classifiers using Stochastic Computing with a Hardware-Oriented Approximate Activation Function*, IEEE Proc. ICCD, pp. 97-104, 2017.

[8] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, A. Heckert, J. Dray, S. Vo: *A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications*, Special Publication 800-22, Revision 1a, National Institute of Standards and Technology, 2010.

[9] M. Matsumoto and T. Nishimura, Mersenne twister: *A 623-dimensionally equidistributed uniform pseudorandom number generator*, ACM Trans. on Modeling and Computer Simulations, 1998.

[10] 入手先 ⟨http://iverilog.icarus.com⟩

[11] 入手先 ⟨http://yann.lecun.com/exdb/mnist/⟩