2016 International Symposium on Nonlinear Theory and Its Applications,
NOLTA2016, Yugawara, Japan, November 27th-30th, 2016

Efficient Implementation of Boltzmann Machine using Asynchronous Network of Cellular Automaton-based Neurons

Takashi Matsubara† and Kuniaki Uehara†

†Graduate School of System informatics, Kobe University
1-1 Rokko-dai, Nada, Kobe, Hyogo 657–8501, Japan.
Email: matsubara@phoenix.kobe-u.ac.jp, uehara@kobe-u.ac.jp

Abstract—Artificial neural networks with stochastic state transitions, such as Boltzmann machines, have excelled other machine learning approaches in various benchmark tasks. They however require implementation of nonlinear continuous functions and generation of numerous pseudo random numbers, resulting in increase in computational resources. This study proposes a novel implementation method of Boltzmann machine using asynchronous network of cellular automaton-based neurons. The proposed approach requires much less computational resources than traditional implementation approaches since it does not require both the nonlinear continuous functions.

1. Introduction

Artificial neural networks having deep architecture recently have achieved state-of-the-art results in various benchmark tasks (see [1, 2, 3] for review). Some of these successes depend on Boltzmann machines [4, 5]. They are artificial neural networks consisting of bidirectionally connected units with stochastic transitions; they can approximate a given probability distribution by appropriate learning algorithm [6]. As the previous studies mentioned, it requires repeated Gibbs sampling to learn and obtain a distribution [5]. In practice, the Boltzmann machines also require generation of numerous pseudo random numbers for Gibbs sampling and computation of nonlinear probability functions. The architecture of the cutting-edge artificial neural network becomes larger and larger [7] and becomes requiring a dedicated hardware [8, 9].

On the other hand, more biologically plausible neural network model, spiking neural network, also attracts attention as an alternative artificial neural network [10, 11]. Some studies modified spiking neural network models to act as Bolzmann machines [12, 13, 14]. They employed stochastic state transitions or strong noise induction to implement the stochastic units, and lack a perspective of computational efficiency. These studies however imply that a spiking neuron has a potential to act as a stochastic unit.

Recently, an alternative modeling and implementation approach for spiking neural network has been investigated; the nonlinear dynamics of a neuron is modeled as an asynchronous cellular automaton and is implemented as an asynchronous sequential logic circuit [15, 16, 17, 11]. These models have achieved better results in task of reproducing dynamics of mammalian nervous system and required less computational resources than traditional artificial neural networks. Following these previous studies, this paper proposes a type of the hardware-oriented spiking neural networks. Empirical evaluation demonstrates that the proposed model acts as Bolzmann machine and approximates a give probability distribution well despite that it requires less computational resources.

2. Asynchronous Network of Cellular Automaton-based Neuron

This study introduces a type of *cellular automaton-based neuron models* (ab. CANs) [15, 16, 17, 11]. The dynamic of the CAN in this paper is similar to those of the previously proposed versions but not the same. A CAN is denoted by an index k. The CAN k has a state V_k, which is restricted to the range of $[0, 1)$. The state V_k can be regarded as a *membrane potential* from a neuron model viewpoint. The CAN k accepts a periodic *internal clock* $C_k(t)$ expressed as

$$C_k(t) = \begin{cases} 1 & \text{if } (t - \theta_k) \pmod{1/f_k^{\mathrm{C}}} = 0, \\ 0 & \text{otherwise,} \end{cases}$$

where f_k^{C} is the internal clock frequency, θ_k is its initial phase, and $t \in [0, \infty)$ is the continuous time. The CAN also accepts multiple external binary inputs $S_l(t) \in \{0, 1\}$ They can be regarded as *pre-synaptic action potentials* from a neuron model viewpoint. The accepted inputs generate the following signal U_k;

$$U_k(t) = \sum_l G_{k,l} S_l(t). \tag{1}$$

where $G_{k,l}$ can be regarded as a *synaptic weight* from the pre-synaptic action potential $S_l(t)$ to the CAN k, and $G_{k,l} > 0$ ($G_{k,l} < 0$) implies excitation (inhibition). At the rising edge of the internal clock $Ck(t)$, the membrane potential V_k is updated as

$$V_k(t^+) = \begin{cases} V_k(t) - c_k + U_k(t) + \xi(t) & \text{if } C_k(t) = 1, \\ V_k(t) & \text{otherwise,} \end{cases}$$

where the variable t^+ denotes the moment just after t, i.e., $t^+ = \lim_{\epsilon \to +0} t + \epsilon$, the parameter c_k represents a leak current, $\xi(t)$ has a noise term. When the membrane potential V_k reaches or exceeds 1.0, the membrane potential V_k is immediately reset and the CAN k generates an output $Y_k(t) = 1$ as

$$V_k(t^+) = \begin{cases} V_k(t) \pmod 1 & \text{if } V_k(t) \geq 1, \\ V_k(t) & \text{otherwise,} \end{cases}$$

and

$$Y_k(t^+) = \begin{cases} 1 & \text{if } V_k(t) \geq 1, \\ 0 & \text{if } C(t) = 1, \\ Y_k(t) & \text{otherwise.} \end{cases}$$

An asynchronous recurrent networks of CANs (ab. ANCAN) [11] consists of n CANs. A CAN l is connected to another CAN k via a synaptic weight $G_{k,l}$. An action potential $Y_l(t) = 1$ generated by the CAN l is delivered to the CAN n_l^{h} and is accepted as a pre-synaptic action potential $S_l(t) = 1$, i.e.,

$$S_l(t) = Y_l(t). \tag{2}$$

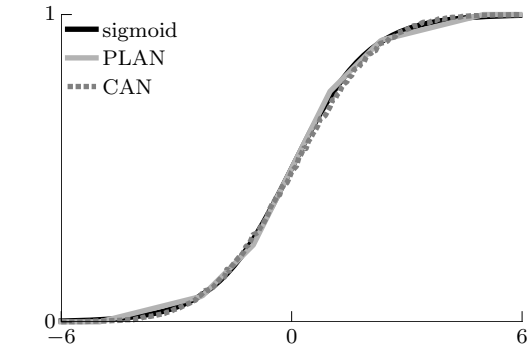
2.1. Conversion from Boltzmann Machine

The detailed description of dynamics of the Bolzmann machine is outside of scope of this paper and therefore is omitted. This paper focuses on a Bernoulli restricted Bolzmann machine, consisting of n_v visible units and n_h hidden units; they have bias terms $\boldsymbol{b}_v$ and $\boldsymbol{b}_h$ and are connected via synaptic weights $W_{k,l}$. Each unit has a binary state; 0 or 1. The target ANCAN was constructed with n_v CANs corresponding to the visible units and n_h CANs corresponding to the hidden units. The leak current c_k corresponding to a unit k is set to $-b_k$. The synaptic weights $G_{k,l}$ are set to the corresponding synaptic weights $W_{k,l}$. The internal clock frequency f_k^{C} is randomly chosen from a uniform distribution $\mathcal{U}(1, 2)$. For comparison, the synchronous version of the ANCAN, called SNCAN, is also prepared; internal clock frequency is uniformly 1 and the initial phase θ_k of a visible unit is 0 and that of a hidden unit is 0.5. For mimicking sigmoid function $\sigma(u) = (1 + \exp(-u))^{-1}$, the noise term ξ is set to follow a normal distribution $\mathcal{N}(\frac{2}{3}, 3)$.

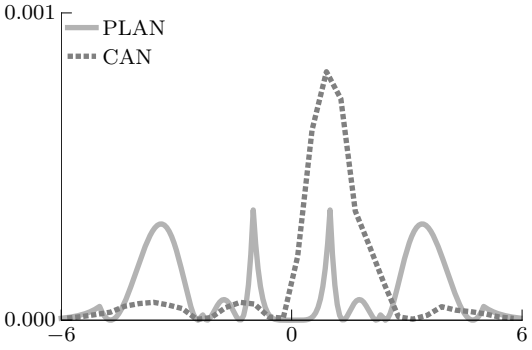
3. Results

3.1. Activation Function

When a CAN accepts the fixed inputs $U_k(t) = x$ and the probability of action potential is denoted as $y = P(Y = 1)$, the empirical relationship between x and y is depicted in Fig. 1 (a). For comparison, the sigmoid function $y = (1 - \exp(-x))^{-1}$ and its piecewise linear approximation called



(a)



(b)

Figure 1: (a) The CAN and the other activation functions. (b) The squared difference from the sigmoid function.

PLAN [18] are also depicted; PLAN is expressed as

$$y = \begin{cases} 1 & \text{if } x \geq 5 \\ 0.03125|x| + 0.84375 & \text{if } 5 > |x| \geq 2.375 \\ 0.125|x| + 0.625 & \text{if } 2.375 > |x| \geq 1 \\ 0.25|x| + 0.5 & \text{if } 1 > |x| \\ 0 & \text{if } 5 \geq x \end{cases}$$

The PLAN is one of the most efficient approximation methods for implementation on a sequential logic circuit. While the equation contains multiplications, they can be implemented by shift register and logical operators. Fig. 1 (b) shows the squared differences of the proposed CAN and the PLAN from the sigmoid function, implying that the proposed CAN and the PLAN are almost comparable. The proposed CAN is a good approximation of the sigmoid function.

3.2. Approximation of Boltzmann Machine

The Bernoulli restricted Bolzmann machine was prepared, where the number of neurons was set to $n_v = n_h = n$, and the synaptic weights $W_{k,l}$ and the bias term b_k were initialized to the samples from the normal distribution $\mathcal{N}(0, (n+1)^{-1})$. The Boltzmann machine was implemented

in several ways. The Boltzmann machine was implemented on with a general purpose computer with 64-bit floating point numbers, the sigmoid function, and the Mersenne twister psuedo-random number generator and was used for the original Boltzmann machine. The other Bolzmann machines were implemented with fixed-point numbers of scaling factor 2^8. The activation functions were the sigmoid function, the PLAN, and the proposed CAN. For the sigmoid function and the PLAN, the M-sequence random number generator (ab. M-seq. RNG) of 16-bit was used to generate uniform distribution $\mathcal{U}(0, 1)$. For the proposed CAN, the M-seq. RNG of 16-bit was also used to generate the noise term $\hat{\xi}$ following binomial distribution $\mathcal{B}(12, 0.5)$, where $\hat{\xi} + \frac{2}{3} - 6$ is a good approximation of the noise term ξ following a normal distribution $\mathcal{N}(\frac{2}{3}, 3)$. The M-seq. RNG was updated according to the internal clock with the frequency of 1.

After the Bolzmann machines were initialized and the neurons were updated repeatedly, the Bolzmann machines reached a stationary distribution. In this paper, the outputs of the first five visible neurons and their distribution were focused. The Kullback-Leibler divergence D_{KL} was used to measure the similarity between the stationary distributions of the original Bolzmann machine and the implemented Bolzmann machine. The smaller Kullback-Leibler divergence D_{KL} implies the better accuracy of the approximation. The number of neurons was set to $n = 5,\ 20,\ 100$. When $n \leq 20$, the stationary distribution of the original Bolzmann machine was theoretically obtained. Otherwise the empirical distribution of 10^7 samples from the original Bolzmann machine was used. The average Kullback-Leibler divergences D_{KL} obtained from 10 trials is shown in Fig. 2 and is summarized in Table 1. When $n = 5$, the sigmoid function demonstrated the best performance. The PLAN and the ANCAN were comparable, while the SNCAN had a worse performance. When $n = 20$ and $n = 100$, the ACAN got performance comparable to the sigmoid function and excelled the PLAN. Remarkably, the Kullback-Leibler divergence D_{KL} of the ANCAN kept decreasing after sampling 10^6 times, while that of the PLAN converged before sampling 10^5 times.

3.3. Implementation

The Bolzmann machines were also implemented on an field programmable gate array (FPGA) device with the fixed-point numbers with scaling factor 2^8. Xilinx FPGA Kintex-7 XC7K325T-2FFG900C mounted on the Kintex-7 FPGA KC705 Evaluation Kit [19] was used. A bitstream file for the FPGA configuration was generated by the Xilinx design software environment ISE 14.7. The corresponding implementation cost, i.e., the number of the occupied slices on the FPGA devices, are also shown in Table 1. Straight-forward implementation of the sigmoid function is trouble some and thus is omitted. When $n = 5$, the ANCAN and the SNCAN require computational resources less than half that of the PLAN. When $n = 20$, they reduced the implementation cost by 30 %.

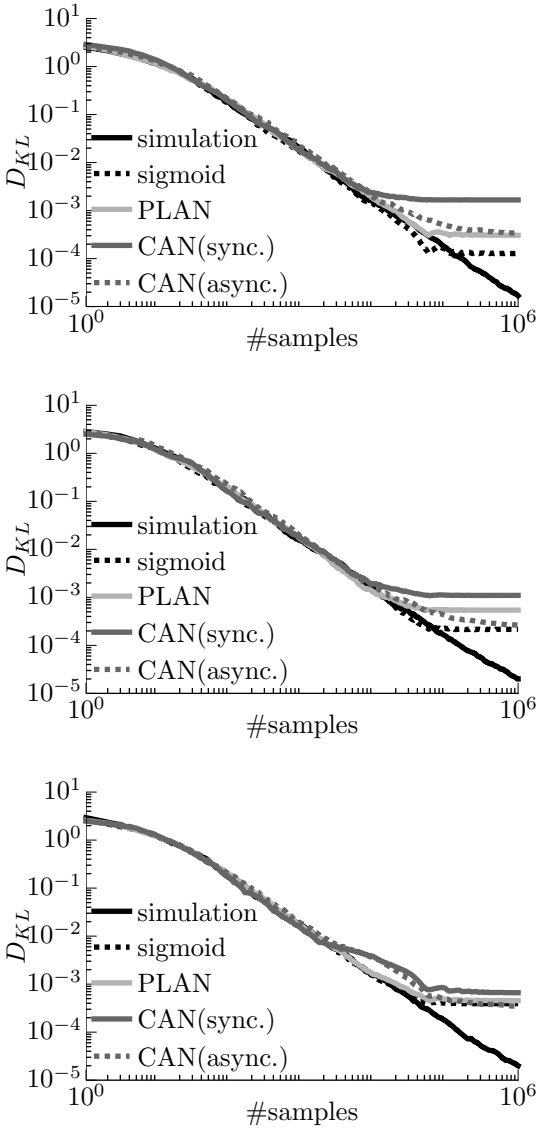


Figure 2: Kullback-Leibler divergence D_{KL}. Simulation implies implementation on a general purpose computer with 64-bit floating point numbers, the sigmoid function, and the Mersenne twister psuedo-random number generator. The other results are obtained from implementation with fixed-point numbers of scaling factor 2^8, the corresponding activation function, and the M-sequence random number generator (ab. M-seq. RNG) of 16-bit.

4. Discussion

These results suggest that the ANCAN is a better approximation and requires much less computational resources when compared to the Bolzmann machines with the sigmoid function or the PLAN function. This study

Table 1: Comparison between the Boltzmann Machines.

function	bit length	RNG	D_{KL}			Implementation	
			$n = 5$	$n = 20$	$n = 100$	$n = 5$	$n = 20$
sigmoid	64-bit float.	Mersenne twister	$< 1.68 \times 10^{-5}$	2.00×10^{-5}	$< 2.00 \times 10^{-5}$	–	–
sigmoid	8-bit fixed	M-seq. RNG of 16bit	1.28×10^{-4}	2.15×10^{-4}	3.92×10^{-4}	–	–
PLAN	8-bit fixed	M-seq. RNG of 16bit	3.06×10^{-4}	5.43×10^{-4}	4.54×10^{-4}	1033	8470
CAN (sync.)	8-bit fixed	M-seq. RNG of 16bit	1.67×10^{-3}	1.10×10^{-3}	6.63×10^{-4}	438	5683
CAN (async.)	8-bit fixed	M-seq. RNG of 16bit	3.41×10^{-4}	2.62×10^{-4}	3.49×10^{-4}	435	5974

was partially supported by the KAKENHI (16K12487 and 26280040), Kayamori Foundation of Information Science Advancement, and The Nakajima Foundation.

References

[1] Y. Bengio, "Learning Deep Architectures for AI," *Foundations and Trends in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

[2] J. Schmidhuber, "Deep Learning in Neural Networks: An Overview," *arXiv preprint arXiv:1404.7828*, vol. 61, pp. 1–66, 2014.

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

[4] R. Salakhutdinov and G. E. Hinton, "Deep Boltzmann Machines," *International Conference on Artificial Intelligence and Statics*, no. 3, pp. 448–455, 2009.

[5] R. Salakhutdinov and G. Hinton, "An Efficient Learning Procedure for Deep Boltzmann Machines," *Neural Computation*, vol. 24, no. 8, pp. 1967–2006, 2012.

[6] D. Ackley, G. E. Hinton, and T. Sejnowski, "A learning algorithm for boltzmann machines," *Cognitive science*, vol. 9, no. 1, pp. 147–169, 1985.

[7] K. He *et al.*, "Deep Residual Learning for Image Recognition," *Arxiv.Org*, vol. 7, no. 3, pp. 171–180, 2015.

[8] K. Ovtcharov *et al.*, "Accelerating Deep Convolutional Neural Networks Using Specialized Hardware," *Microsoft Research*, pp. 3–6, 2015.

[9] T. Marukame *et al.*, "Error tolerance analysis of deep learning hardware using restricted Boltzmann machine towards low-power memory implementation," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 7747, no. c, pp. 1–1, 2016.

[10] Y. Cao, Y. Chen, and D. Khosla, "Spiking Deep Convolutional Neural Networks for Energy-Efficient Object Recognition," *International Journal of Computer Vision*, no. Darpa 2011, pp. 54–66, 2014.

[11] T. Matsubara and H. Torikai, "An Asynchronous Recurrent Network of Cellular Automaton-Based Neurons and Its Reproduction of Spiking Neural Network Activities," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 27, no. 4, p. (under review), 2014.

[12] L. Buesing *et al.*, "Neural dynamics as sampling: A model for stochastic computation in recurrent networks of spiking neurons," *PLOS Computational Biology*, vol. 7, no. 11, 2011.

[13] P. O'Connor *et al.*, "Real-time classification and sensor fusion with a spiking deep belief network," *Frontiers in Neuroscience*, vol. 7, no. 7 OCT, pp. 1–13, 2013.

[14] E. Neftci *et al.*, "Event-driven contrastive divergence for spiking neuromorphic systems," *Frontiers in Neuroscience*, vol. 7, no. January, pp. 1–14, 2014.

[15] T. Matsubara, H. Torikai, and T. Hishiki, "A Generalized Rotate-and-Fire Digital Spiking Neuron Model and Its On-FPGA Learning," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 58, no. 10, pp. 677–681, 2011.

[16] T. Matsubara and H. Torikai, "Neuron-Like Responses and Bifurcations of a Generalized Asynchronous Sequential Logic Spiking Neuron Model," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E95.A, no. 8, pp. 1317–1328, 2012.

[17] ——, "Asynchronous Cellular Automaton-Based Neuron: Theoretical Analysis and On-FPGA Learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 24, no. 5, pp. 736–748, 2013.

[18] H. Amin, K. Curtis, and H.-G. B.R., "Piecewise linear approximation applied to nonlinear function of a neural network," *IEEE Proceedings of Circuits, Devices System*, vol. 144, no. 6, pp. 313–317, 1997.

[19] Xilinx Inc., "Xilinx Inc." url: http://www.xilinx.com/