Asynchronous network of cellular automaton-based neurons for efficient implementation of Boltzmann machines

Matsubara, Takashi

Uehara, Kuniaki

(Citation)
Nonlinear Theory and Its Applications, IEICE,9(1):24-35

(Issue Date)
2018

(Resource Type)
journal article

(Version)
Version of Record

(URL)
https://hdl.handle.net/20.500.14094/90006112

NOLTA, IEICE

Paper

Asynchronous network of cellular automaton-based neurons for efficient implementation of Boltzmann machines

Takashi Matsubara [1 a)] *and Kuniaki Uehara* [1 b)]

[1] *Graduate School of System informatics, Kobe University*
1-1 Rokko-dai, Nada, Kobe, Hyogo 657-8501, Japan

[a)] *matsubara@phoenix.kobe-u.ac.jp*
[b)] *uehara@kobe-u.ac.jp*

Received April 23, 2017; Revised July 10, 2017; Published January 1, 2018

Abstract: Artificial neural networks with stochastic state transitions and calculations, such as Boltzmann machines, have excelled over other machine learning approaches in various benchmark tasks. The networks often achieve better results than deterministic neural networks of similar sizes, but they require implementation of nonlinear continuous functions for probabilistic density functions, thus resulting in an increase in computational effort. The architecture size of cutting-edge artificial neural networks are ever-growing; therefore, they require dedicated hardware. Conversely, asynchronous cellular automaton-based neuron models have been investigated to model the highly nonlinear dynamics of biological neurons. They are special types of cellular automata and are implemented as small asynchronous sequential logic circuits. In this study, we propose a new type of asynchronous network of cellular automaton-based neuron for the efficient implementation of Boltzmann machines. Experimental comparisons demonstrate that the proposed approach achieves comparable or better performances in such benchmark tasks as image classification and generation while it requiring much less computational resources than traditional implementation approaches.

Key Words: artificial neural network, Boltzmann machine, asynchronous cellular automaton, asynchronous sequential logic

1. Introduction

Artificial neural networks with deep architecture have recently achieved state-of-the-art results in various benchmark and practical tasks (see [1–3] for a review). These successes largely depend not only on convolutional neural networks [4] but on neural networks employing Monte Carlo methods such as Boltzmann machines [5, 6] and variational autoencoders [7, 8]. Boltzmann machines are artificial neural networks consisting of bidirectionally connected units with stochastic state transitions. The Boltzmann machines can approximate a given probability distribution by using an appropriate learning algorithm [9]. They require the computation of nonlinear functions to calculate probability distributions such as logistic function and Gaussian function in addition to repeated Gibbs sampling

Nonlinear Theory and Its Applications, IEICE, vol. 9, no. 1, pp. 24–35 DOI: 10.1587/nolta.9.24

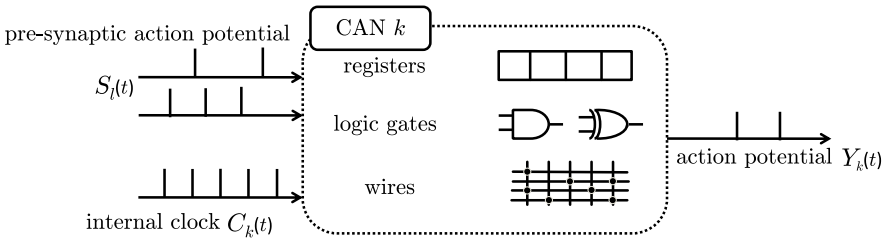


Fig. 1. Diagram of a cellar automaton-based neuron.

from the distributions [6]. The architecture size of the cutting-edge artificial neural networks are ever-growing [10]; therefore, they require dedicated hardware [11, 12]. However, many studies have investigated dedicated hardware for deterministic feedforward neural networks and often employ simpler activation functions with limited nonlinearity such as rectified linear unit. Therefore, these studies cannot be applied directly to Boltzmann machines.

Conversely, more biologically plausible neural network model called spiking neural network also attracts attention as an alternative artificial neural network [13–15]. Their applications include approximation of arbitrary functions and probability distributions as well as modeling of biological neuronal activity. Some studies have modified spiking neural network models to act as Boltzmann machines [16–18]. They employ stochastic state transitions or strong noise induction to implement the stochastic units. These studies demonstrate that although a spiking neuron has the potential to act as a stochastic unit in Boltzmann machines, they lack a perspective of computational efficiency. Recently, an alternative modeling and implementation approach called *asynchronous network of cellar automaton-based neurons* (ANCAN) has been investigated for implementing spiking neural networks [14, 19–22]. In this approach, a neuron's nonlinear dynamics is modeled as an asynchronous cellular automaton and implemented as an asynchronous sequential logic circuit. These models have achieved better results in tasks such as reproducing the nonlinear dynamics of a mammalian nervous system and have required less computational resources than traditional artificial neural networks.

In this paper, we propose a new type of ANCANs for hardware-oriented Boltzmann machines. This study implements Bernoulli-Bernoulli restricted Boltzmann machines on the proposed approach and on conventional approximation approaches [18, 23]. Experimental results confirm that the proposed ANCAN acts as a Boltzmann machine and approximates a given probability distribution. The approximation accuracy of the proposed ANCAN is comparable to or better than those of the competitive approaches; moreover, the proposed ANCAN requires much less computational resources. In addition, the restricted Boltzmann machines are implemented by using both the proposed and the competitive approaches and trained to model the joint probability distribution of the pixels and class labels of the MNIST handwritten digit database [24]. Although the classification accuracy of the proposed approach is not better than but almost comparable to the competitive approaches, the proposed approach remarkably generates a wide variety of images and achieves the best modeling accuracy measured by the log-likelihood of the pixels.

The preliminary and limited results are found in a conference paper [25].

2. Proposed neural network model

2.1 Asynchronous network of cellular automaton-based neuron

This study proposes a type of *cellular automaton-based neuron model* (ab. CAN) [14, 19–22]. The dynamics of CAN in this paper is similar to those of the previously proposed models but has significant differences. Figure 1(a) shows a diagram of a CAN that is denoted by an index k. The CAN k has an internal state V_k, which is limited to a range of $[0, 1)$. The internal 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

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

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 inputs $S_l(t) \in \{0, 1\}$ indexed by l. They can be regarded as *pre-synaptic action potentials* from the pre-synaptic neurons l. The accepted inputs generate the following signal U_k;

$$U_k(t) = \sum_l G_{k,l} S_l(t), \tag{2}$$

where $G_{k,l}$ can be regarded as a *synaptic weight* from the pre-synaptic neuron l to the CAN k, and $G_{k,l} > 0$ ($G_{k,l} < 0$) implies excitation (inhibition). According to the external inputs $S_l(t)$, the following virtual membrane potential $\tilde{V}_k$ is calculated;

$$\tilde{V}_k(t) = V_k(t) - c_k + U_k(t) + \xi(t), \tag{3}$$

where the parameter c_k represents a leak current and $\xi(t)$ is a noise term. At the rising edge of the internal clock $C_k(t)$, the membrane potential V_k is updated. When the virtual membrane potential $\tilde{V}_k$ 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 follows;

$$V_k(t^+) = \begin{cases} 0 & \text{if } C_k(t) = \uparrow \text{ and } \tilde{V}_k(t) < 0 \\ \tilde{V}_k(t) & \text{if } C_k(t) = \uparrow \text{ and } 0 \le \tilde{V}_k(t) < 1 \\ \tilde{V}_k(t) \pmod 1 & \text{if } C_k(t) = \uparrow \text{ and } 1 \le \tilde{V}_k(t) \\ V_k(t) & \text{otherwise,} \end{cases} \tag{4}$$

and

$$Y_k(t^+) = \begin{cases} 0 & \text{if } C_k(t) = \uparrow \text{ and } \tilde{V}_k(t) < 1 \\ 1 & \text{if } C_k(t) = \uparrow \text{ and } 1 \le \tilde{V}_k(t) \\ Y_k(t) & \text{otherwise,} \end{cases} \tag{5}$$

where the variable t^+ denotes the moment just after t, i.e., $t^+ = \lim_{\epsilon \to +0} t + \epsilon$, and $C_k(t) = \uparrow$ denotes the internal clock C_k at a rising edge.

An asynchronous network of CANs (ab. ANCAN) [14] consists of multiple 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.,

$$S_l(t) = Y_l(t). \tag{6}$$

In previous studies [14, 19–22], the internal states of a CAN such as membrane potential were discretized and always changed by the minimum unit, resulting in very slow time constants. This is because the purpose of the CANs and the ANCANs was modeling the dynamics of biological neural tissue, thereby suppressing the frequency of spike generation within a biologically plausible range. Conversely, since the purpose of this study is modeling a given Bernoulli distribution, the CAN is required to efficiently represent the probability from 0.0 to 1.0. Hence, we inject noise ξ into the internal states and allow them to jump by c_k so that the frequency of spike generation becomes stochastic and is able to reach 1 per unit time.

2.2 Conversion from Boltzmann machine

A detailed description of dynamics of the Boltzmann machine is omitted since it is outside the scope of this paper. This study focuses on a Bernoulli-Bernoulli restricted Boltzmann 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 $W_{k,l}$ (see Fig. 2(a)). Each unit has a binary state 0 or 1 and follows a Bernoulli distribution $\mathscr{Ber}(y)$: The probability y that the state takes a value of 1 is determined by the logistic function

$$y = (1 + \exp(-x))^{-1}, \tag{7}$$

where x corresponds to the summation U_k of all the input signals S_l multiplied by the weights $G_{k,l}$. The ANCAN was constructed with n_v CANs corresponding to the visible units and n_h CANs

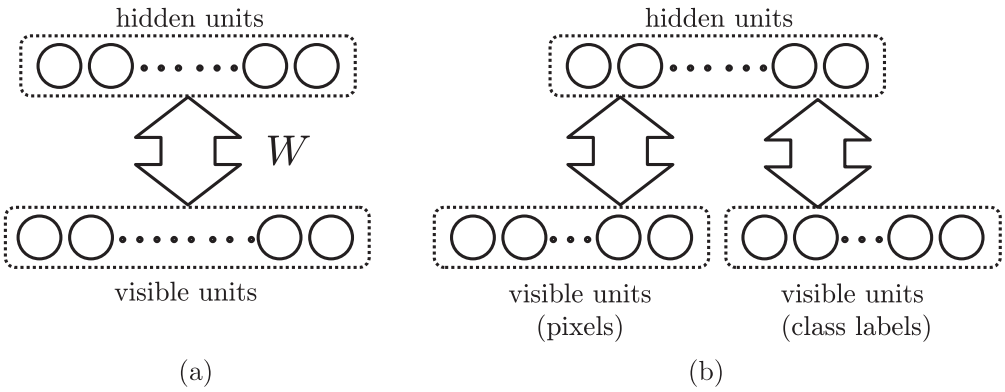


Fig. 2. (a)A diagram of a restricted Boltzmann machine and (b) a diagram of a restricted Boltzmann machine for classification. Each circle represents a unit such as a cellar automaton-based neuron. Each area enclosed by a dotted line is a layer, consisting of multiple units. The visible units in the Boltzmann machine for classification are divided into two subsets: One corresponds to the pixels of an image and the other corresponds to class labels.

corresponding to the hidden units. The leak current c_k of a unit k was set to $-b_k$. The synaptic weights $G_{k,l}$ were set to the corresponding synaptic weights $W_{k,l}$. The internal clock frequency f_k^{C} was randomly chosen from a uniform distribution $\mathcal{U}(1,2)$. We assumed that the noise term ξ follows a normal distribution $\mathcal{N}(\mu_\xi, \sigma_\xi)$ and found that $\mu_\xi = \frac{2}{3}$ and $\sigma_\xi = 3$ were suited for mimicking the logistic function. Since a straight-forward implementation of the normal distribution used for the CAN is troublesome, a binomial distribution was employed for digital circuit implementation instead of a normal distribution. For comparison, a synchronous version of the ANCAN, called SNCAN, was also prepared: All the internal clocks C_k shared a uniform frequency of 1; the initial phases θ_k of all the visible units were set to 0 and those of the hidden units were set to 0.5. Under this condition, all the visible units were updated simultaneously and all the hidden units were updated after the update of the visible units as is the case with ordinary restricted Boltzmann machines.

3. Approximation of Boltzmann machines

3.1 Activation function

The empirical probability y of action potential was calculated as

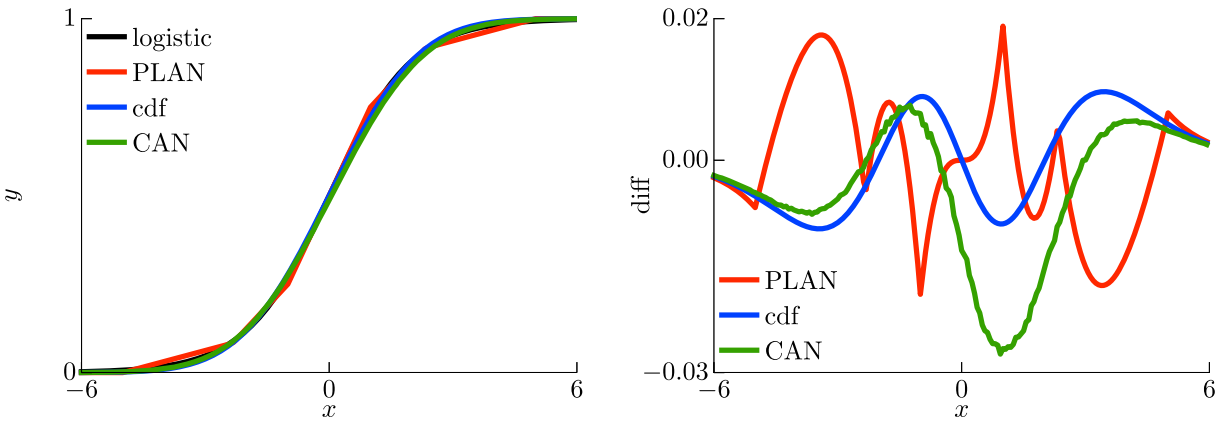
$$\begin{aligned} y &= P(Y_k = 1) \\ &= \frac{\int_t Y_k(t^+) C_k(t) dt}{\int_t C_k(t) dt}. \end{aligned} \tag{8}$$

When a CAN accepts a fixed input $U_k(t) = x$, the empirical relationship between input x and output y is depicted in Fig. 3(a). For comparison, the logistic function $y = (1 - \exp(-x))^{-1}$ and its piecewise linear approximation known as *PLAN* [23] are also depicted. PLAN is expressed as

$$y = \begin{cases} 1 & \text{if } x \geq 5 \\ 0.03125|x| + 0.84375 & \text{if } 5 > |x| \geq 2.375 \\ 0.125|x| + 0.625 & \text{if } 2.375 > |x| \geq 1 \\ 0.25|x| + 0.5 & \text{if } 1 > |x| \\ 0 & \text{if } 5 \geq x. \end{cases} \tag{9}$$

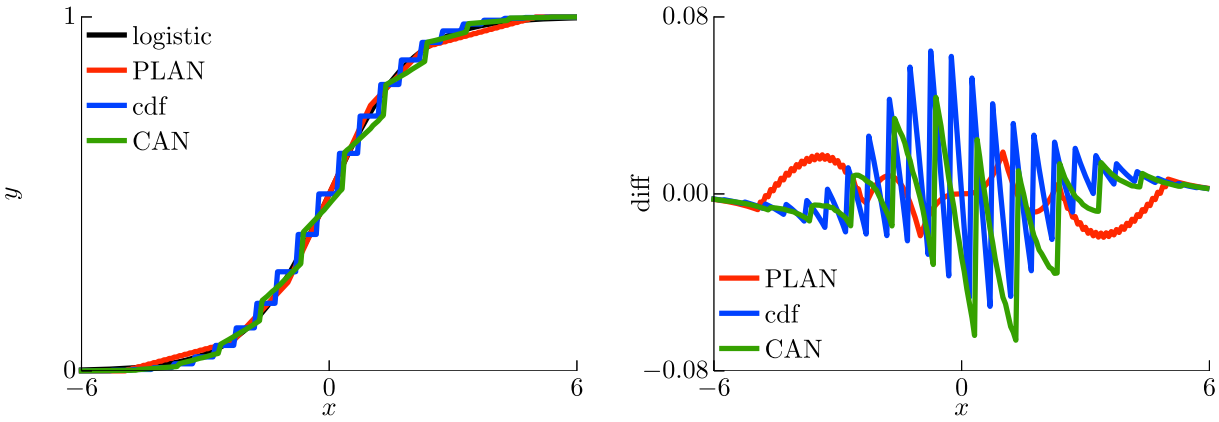
PLAN is one of the most efficient approximation methods for implementation on a sequential logic circuit. The multiplications in the equation can be implemented as shift registers and logical operators. In addition, the cumulative distribution function (ab. cdf) of a normal distribution is also depicted [18]. The cdf of a normal distribution $\mathcal{N}(\mu, \sigma^2)$ is expressed as

$$y = \frac{1}{2}\left(1 + \mathrm{erf}\left(\frac{x - \mu}{\sqrt{2\sigma^2}}\right)\right), \tag{10}$$



(a) (b)

Fig. 3. (a) CAN and the other activation functions. (b) Differences from the logistic function.



(a) (b)

Fig. 4. (a) CAN and the other activation functions after implementation. (b) Differences from the logistic function after implementation.

where $\mathrm{erf}(u)$ is the error function defined as $\frac{2}{\sqrt{\pi}}\int_0^u e^{-t^2}dt$. The cdf is known as a function similar to the logistic function [18, 26], and the probability that a sample from the normal distribution $\mathcal{N}(x, \sigma^2)$ is greater than zero is a good approximation of the logistic function. We found that the cdf with $\mu = 0$ and $\sigma = 1/0.589$ has the minimum squared error from the logistic function. Figure 3(b) shows the differences of the PLAN, the cdf-based approximation, and the proposed CAN from the logistic function that illustrates that they are almost comparable. The proposed CAN is a good approximation of the logistic function.

3.2 Digital circuit implementation

This section considers digital circuit implementations of the CAN and the other activation functions. They were implemented with fixed-point numbers with a scaling factor of 2^8. The 16-bit M-sequence random number generator (ab. M-seq. RNG) was used to generate random variables. The M-seq. RNG was updated according to the internal clock with a frequency of 1. For the logistic function and the PLAN, the M-seq. RNG generated a sample from a uniform distribution $\mathcal{U}(0, 1)$ with fixed-point numbers with a scaling factor of 2^8. On the other hand, the implementation of random variables following normal distributions used for the CAN and the cdf-based approximation is troublesome. Commonly-used approaches such as Box-Muller's method require highly nonlinear computations. Instead, we used the summation of the n_B binary random variables sampled from the M-seq. RNG, which follows a binomial distribution $\mathcal{B}i(n_B, 0.5)$. For the proposed CAN, the 16-bit M-seq. RNG

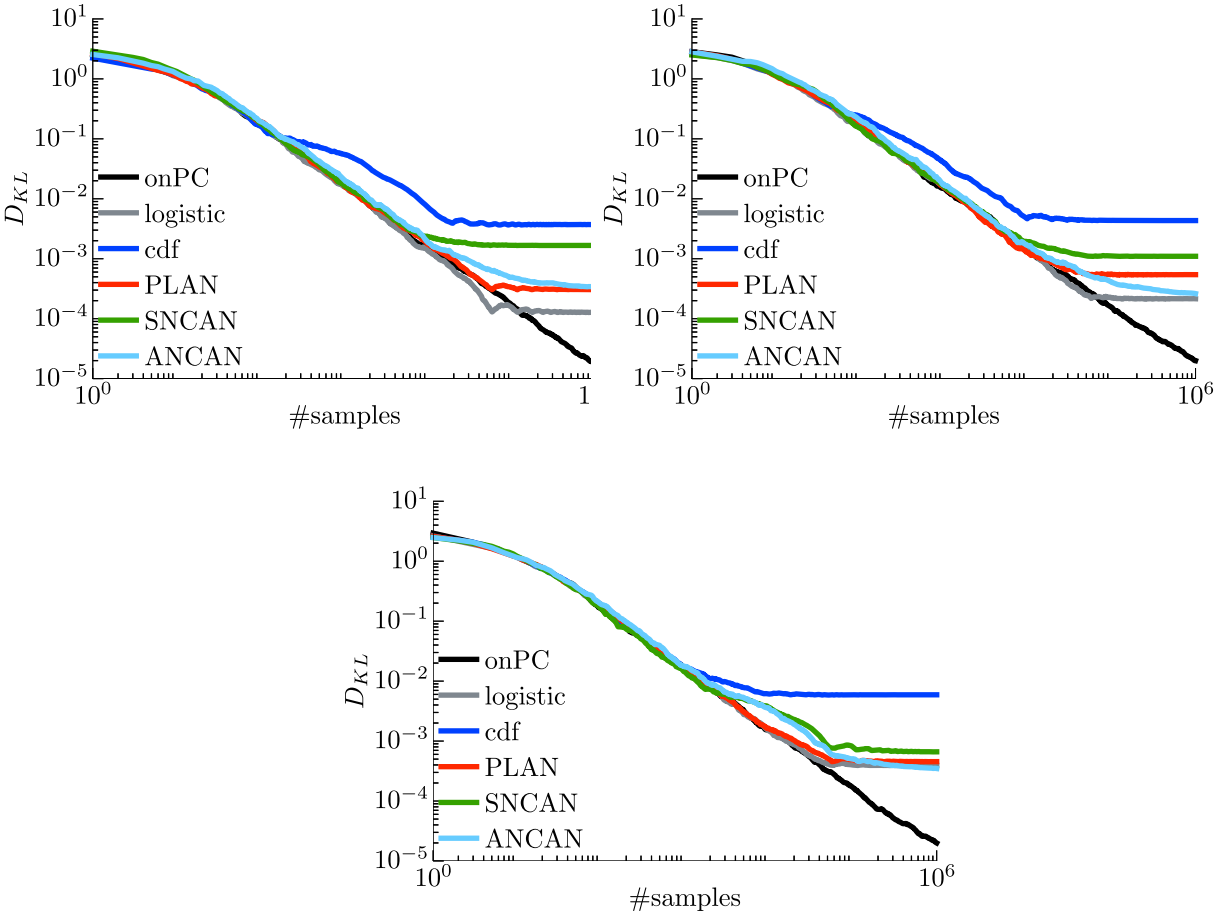


Fig. 5. Kullback-Leibler divergence D_{KL}. "on PC" implies implementation on a general purpose computer with 64-bit floating point numbers, the logistic function, and the Mersenne Twister pseudo-random number generator. The other results were obtained from implementation with fixed-point numbers with a scaling factor of 2^8, the corresponding activation function, and the 16-bit M-sequence random number generator (ab. M-seq. RNG).

generated a sample $\hat{\xi}$ from the binomial distribution $\mathcal{B}i(12, 0.5)$ and the noise term ξ was calculated as $\hat{\xi} + \frac{2}{3} - 6$. For the cdf-based approximation, the M-seq. RNG generated a sample $\tilde{\xi}$ from the binomial distribution $\mathcal{B}i(n_B, 0.5)$. We found $\xi = \frac{1}{2}(\tilde{\xi} - \frac{n_B}{2})$ for $n_B = 47$ has the minimum squared error from the logistic function. Figure 4 shows the activation functions and the differences from the logistic functions after implementation. Although the differences obtained from the CAN and the cdf-based approximation became larger, they remain comparatively small.

3.3 Kullback-leibler divergence

We evaluated approximation accuracy of the aforementioned approaches. A Bernoulli-Bernoulli restricted Boltzmann machine was prepared, where the number of neurons was set to $n_v = n_h = n$, and the synaptic weights $G_{k,l}$ 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 first implemented on a general purpose computer (PC) with 64-bit floating point numbers, the logistic function, and the Mersenne Twister pseudo-random number generator. This Boltzmann machine us hereafter referred to the original one. The Boltzmann machine was implemented also with the fixed-point numbers with scaling factor 2^8 by using the logistic function, the PLAN, the cdf-based approximation, and the proposed CAN. After the Boltzmann machines were initialized and the neurons were updated repeatedly, the Boltzmann machines reached stationary distributions. In this paper, we focus on the outputs of the first five visible neurons and their distribution. The Kullback-Leibler divergence D_{KL} was used to measure the similarity between the stationary distributions of the original Boltzmann machine and the imple-

Table I. Comparison of Implementation.

Model	Bit length	RNG
logistic (on PC)	64-bit float.	Mersenne Twister
logistic	8-bit fixed	M-seq. RNG of 16bit
PLAN	8-bit fixed	M-seq. RNG of 16bit
cdf	8-bit fixed	M-seq. RNG of 16bit
SNCAN	8-bit fixed	M-seq. RNG of 16bit
ANCAN	8-bit fixed	M-seq. RNG of 16bit

Table II. Comparison in Approximation of Boltzmann Machines.

Model	$\log_{10} D_{KL}$			#{occupied slice}	
	$n = 5$	$n = 20$	$n = 100$	$n = 5$	$n = 20$
logistic (on PC)	−4.77	−4.70	−4.70	–	–
logistic	−3.89	−3.67	−3.41	–	–
PLAN	−3.51	−3.27	−3.34	1,033	8,470
cdf	−2.43	−2.36	−2.23	2,612	13,758
SNCAN	−2.78	−2.96	−3.18	438	5,683
ANCAN	−3.47	−3.58	−3.46	435	5,974

mented Boltzmann machines. A smaller Kullback-Leibler divergence D_{KL} implies a better accuracy in terms of the approximated implementation. The number of neurons was set to $n = 5$, 20, and 100. The stationary distribution of the original Boltzmann machine was theoretically obtained for $n = 5$ and $n = 20$. Otherwise, the empirical distribution of 10^7 samples from the original Boltzmann machine was used. The average Kullback-Leibler divergences D_{KL} over 10 trials is shown in Fig. 5 and is summarized in Table I. When $n = 5$, the logistic function demonstrated the best performance. The PLAN and the ANCAN were comparable, while the SNCAN and the cdf-based approximation resulted in worse performances. When $n = 20$ and $n = 100$, the ACAN performance was comparable to the performance of the logistic function and excelled over that of the PLAN. Remarkably, the Kullback-Leibler divergence D_{KL} of the ANCAN continued to decrease after 10^6 samples, while that of the PLAN converged prior reaching 10^5 samples. The cdf-based approximation always converged after approximately 10^4 samples and achieved the worst results.

3.4 Implementation on FPGA device

The Boltzmann machines were also implemented on a field programmable gate array (FPGA) device. We used a Xilinx FPGA Kintex-7 XC7K325T-2FFG900C mounted on the Kintex-7 FPGA KC705 Evaluation Kit [27]. A bitstream file for the FPGA configuration was generated by the Xilinx design software environment ISE 14.7. Table I shows the corresponding implementation cost, i.e., the number of the occupied slices on the FPGA devices. Straight-forward implementation of the logistic function is troublesome and thus it was omitted. When $n = 5$, the ANCAN and the SNCAN require computational resources less than 50 % of that required by the PLAN. When $n = 20$, the ANCAN and the SNCAN reduced the implementation cost by 30 %. The cdf-based approximation required massive implementation costs. The normal distributions in the cdf-based approximation were approximated by the binomial distributions, which does not require nonlinear functions but the summation of samples from the M-seq. RNG, i.e., sequential addition operations. This requires significant computational cost.

3.5 Classification and generation of the MNIST database

In this section, the Boltzmann machines were evaluated for the MNIST handwritten digit database [24] (see Fig. 6 for examples). The MNIST database is a dataset of 28×28 grayscale images of 70,000 handwritten digits 0–9; 60,000 images for training and 10,000 images for testing. We constructed a

Table III. Comparison in Classification Task of the MNIST Dataset.

Model	Accuracy (%)
logistic (on PC)	95.27
logistic	95.22
PLAN	95.20
cdf	95.32
SNCAN	95.16
ANCAN	94.73

Table IV. Comparison in Generation Task of the MNIST Dataset.

Model	Log-likelihood		
Parameters learned by	each model	each model	PC
Images generated by	each model	PC	each model
training subset	70	–	–
logistic (on PC)	−179	−179	−179
logistic	−184	−181	−182
PLAN	−170	−164	−185
cdf	−163	−163	−178
SNCAN	−105	−167	−174
ANCAN	−53	−140	−59

Boltzmann machine with a hidden layer of 500 units and a visible layer of 794 units; 784 units for the pixels of a single image and 10 units for the one-hot coded class labels (see Fig. 2(b)). Pixel intensities were normalized to the range of $[0, 1]$, which represented the probability that the corresponding visible unit had a value of 1. We divided the 60,000 training images into 50,000 training images and 10,000 validation images. The Boltzmann machines were trained with the persistent constructive divergence algorithm of $k = 10$ [28] and a learning rate of 2^{-8} owing to the implementation with fixed-point numbers with a scaling factor of 2^8. The number of the learning iterations was chosen according to the classification accuracy for the validation subset.

First, we evaluated the classification accuracy of the Boltzmann machines. The average classification accuracies of five trials for the test subset are summarized in Table III. The ANCAN achieved the worst result but all the classification accuracies were almost comparable with each other. Second, we also evaluated the generation ability of the Boltzmann machines. Each Boltzmann machine generated 10,000 binarized images from the randomly initialized visible units, summarized in Fig. 7. They demonstrate that the Boltzmann machine implemented using the ANCAN generated a wide variety of images compared to the other Boltzmann machines. Following previous works on generative models [29–31], we evaluated the generation abilities of the Boltzmann machines by the log-likelihood of the test subset. The probability of the test subset was estimated by fitting a Gaussian Parzen window to the samples from the Boltzmann machines. The σ parameter of the Gaussian Parzen window was determined by cross-validation on the validation set. A larger log-likelihood implies that the Boltzmann machine builds a better model of the distribution of a given dataset. The second column of Table IV summarizes the test log-likelihood of each Boltzmann machine and the binarized samples of the training subset. Interestingly, the ACAN achieved a better log-likelihood than not only all the Boltzmann machines implemented with fixed point numbers but also the Boltzmann machines implemented on a PC.

For a more detailed comparison, we evaluated the Boltzmann machines implemented on a PC where the weight parameters were transferred from the Boltzmann machines trained with the other approximations of activation functions, summarized in the second rightmost column of Table IV. Even when the weight parameters were transferred from the SNCAN and the ANCAN, the Boltzmann machines on a PC did not achieve the log-likelihoods at similar levels to the original SNCAN and ANCAN.

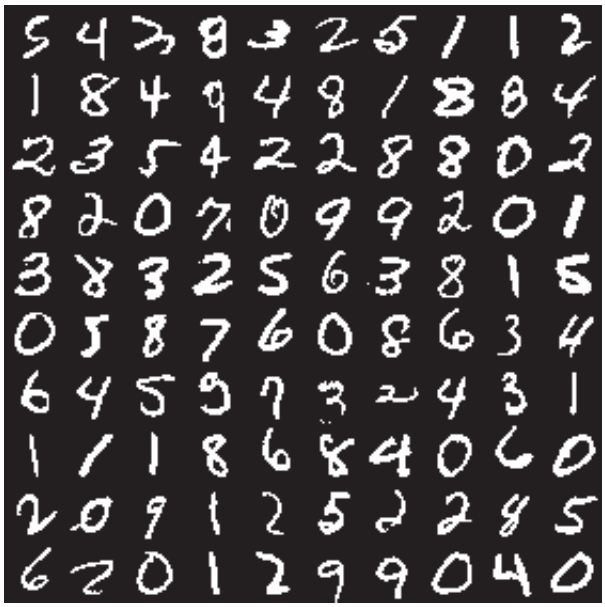


Fig. 6. Example images of MNIST database.

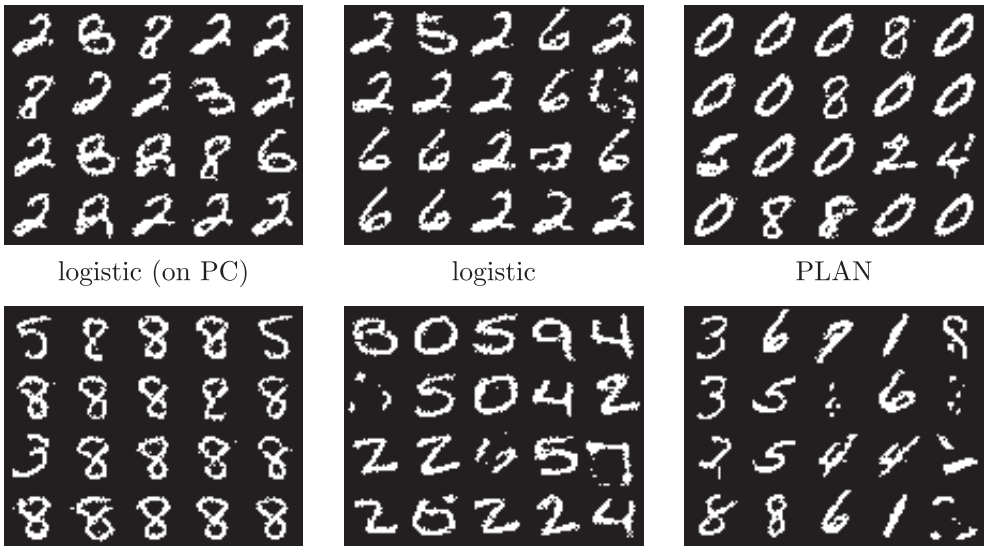


Fig. 7. Samples of binarized images of the Boltzmann machines.

However, the Boltzmann machines with the weight parameters transferred from the ANCAN achieved a relatively better log-likelihood compared to those with the weight parameters transferred from the other approaches. In addition, the parameters learned on a PC were transferred to the Boltzmann machines with the other approximations of activation functions, summarized in the rightmost column of Table IV. In this case, the SNCAN did not achieve a log-likelihood at a similar level to the original SNCAN despite that the ANCAN achieved a lower log-likelihood.

4. Discussion

Previous studies confirm that a Boltzmann machine has a risk of generating an insufficiently wide variety of images even when the Boltzmann machine classifies the MNIST database with an accuracy of better than 95% (e.g., [28]). For classification tasks, Boltzmann machines have to model at least limited features contributing to the classification. Thus, good classification accuracy does not always imply good approximation of the data distribution. The data distribution of the MNIST database has local minima, and the Boltzmann machines easily get trapped in them in both the learning phase and the generation phase. This would explain why the Boltzmann machines generated a limited variety

of images.

As shown in Table II, under the condition that all the approaches share common weight parameters, the ANCAN achieves better Kullback-Leibler divergences, i.e., the ANCAN represents the probability distribution similar to the original Boltzmann machine. However, the SNCAN does not achieve good Kullback-Leibler divergences. These results are consistent with the results on the generation of the MNIST database summarized in the rightmost column of Table IV, where the weight parameters were learned on a PC. Due to the asynchrony, the ANCAN has a larger potential to escape from the local minima by perturbing the states of the units and generate a wide variety of images. However, simultaneously, this asynchrony negatively influences the classification accuracy. The visible units corresponding to the class labels sometimes suffer from perturbation and converge to incorrect labels.

The second rightmost column in Table IV demonstrates that the weight parameters learned by the ANCAN contribute to the generation of a wider variety of images by the Boltzmann machine implemented on a PC. In this study, we use the persistent constructive divergence algorithm to train the Boltzmann machines [28]. With this algorithm, the Boltzmann machines continue to generate images for approximating gradients of the Kullback-Leibler divergence from the data distribution to the model distribution. Hence, the asynchrony of the ANCAN contributes not only to the generation but also to learning weight parameters suited for generation.

Conversely, the SNCAN achieved a better likelihood than the other Boltzmann machines, except for the ANCAN, when the weight parameters were learned by the SNCAN itself (see the second column of Table IV). The dynamics of the SNCAN has no asynchrony but has internal state transitions. Actually, the outputs of each unit of the SNCAN and the ANCAN are not totally random but are related to previous outputs. Immediately after a neuron elicits a spike, it has a less chance of eliciting another spike. This dynamics also perturbs the states of the units and would contribute to learning and generation. However, the weight parameters learned by the Boltzmann machine on a PC did not enhance the performance of the SNCAN and vice versa (see the two rightmost columns of Table IV). Even worse, the SNCAN does not represent a probability distribution similar to the original Boltzmann machine as shown in Table II. These results imply that the dynamics of the SNCAN is not always helpful for the original Boltzmann machine depending on the data distribution.

5. Conclusion

This study proposed a new type of asynchronous networks of cellular automaton-based neurons for hardware-oriented Boltzmann machines. It is a special type of cellular automata and is implemented as small asynchronous sequential logic circuits. The experimental results demonstrate that the proposed approach achieves remarkable results in tasks involving the approximation of a Boltzmann machine. Due to its asynchrony, it learns weight parameters better suited for approximating the distribution of the given dataset and generates a wide variety of artificial data. Although the proposed approach does not produce notable results for classification tasks, its performance is almost comparable to the competitive approaches. The main benefit of the proposed approach is that it requires much less computational resources than traditional implementation approaches. These results suggest that the ANCAN is a better approximation and requires much less computational resources compared with the Boltzmann machines implemented by using other approximations.

Acknowledgments

This study was partially supported by the KAKENHI (16K12487), 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," *Neural Networks*, vol. 61, pp. 85–117, 2015.

[3] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
[4] Y. LeCun *et al.*, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2323, 1998.
[5] R. Salakhutdinov and G.E. Hinton, "Deep Boltzmann machines," *International Conference on Artificial Intelligence and Statics*, no. 3, pp. 448–455, 2009.
[6] R. Salakhutdinov and G. Hinton, "An efficient learning procedure for deep Boltzmann machines," *Neural Computation*, vol. 24, no. 8, pp. 1967–2006, 2012.
[7] D.P. Kingma and M. Welling, "Auto-encoding variational bayes," *International Conference on Learning Representations*, no. Ml, pp. 1–14, 2014.
[8] D.P. Kingma, D.J. Rezende, and M. Welling, "Semi-supervised learning with deep generative models," *Advances in Neural Information Processing Systems*, pp. 1–9, 2014.
[9] D. Ackley, G.E. Hinton, and T. Sejnowski, "A learning algorithm for boltzmann machines," *Cognitive science*, vol. 9, no. 1, pp. 147–169, 1985.
[10] K. He *et al.*, "Deep residual learning for image recognition," *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.
[11] K. Ovtcharov *et al.*, "Accelerating deep convolutional neural networks using specialized hardware," *Microsoft Research*, pp. 3–6, 2015.
[12] 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.
[13] Y. Cao, Y. Chen, and D. Khosla, "Spiking deep convolutional neural networks for energy-efficient object recognition," *International Journal of Computer Vision*, vol. 113, no. 1, pp. 54–66, 2015.
[14] 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, pp. 836–852, 2016.
[15] E. Tu, N. Kasabov, and J. Yang, "Mapping temporal variables into the NeuCube for improved pattern recognition, predictive modeling, and understanding of stream data," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–13, 2016.
[16] 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.
[17] P. O'Connor *et al.*, "Real-time classification and sensor fusion with a spiking deep belief network," *Frontiers in Neuroscience*, vol. 7, no. 178, pp. 1–13, 2013.
[18] E. Neftci *et al.*, "Event-driven contrastive divergence for spiking neuromorphic systems," *Frontiers in Neuroscience*, vol. 7, no. 272, pp. 1–14, 2014.
[19] T. Hishiki and H. Torikai, "A novel rotate-and-fire digital spiking neuron and its neuron-like bifurcations and responses," *IEEE Transactions on Neural Networks*, vol. 22, no. 5, pp. 752–767, 2011.
[20] 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.
[21] 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.
[22] T. Matsubara and H. Torikai, "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.
[23] 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.
[24] "THE MNIST DATABASE of handwritten digits." url: http://yann.lecun.com/exdb/mnist/
[25] T. Matsubara and K. Uehara, "Efficient implementation of Boltzmann machine using asyn-

chronous network of cellular automaton-based neurons," *Proc. 2016 International Symposium on Nonlinear Theory and its Applications*, pp. 634–637, 2016.

[26] S. Sato, "On the moments of the firing interval of the diffusion approximated model neuron," *Mathematical Biosciences*, vol. 39, no. 1-2, pp. 53–70, 1978.

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

[28] T. Tieleman, "Training restricted Boltzmann machines using approximations to the likelihood gradient," *Proceedings of the 25th International Conference on Machine Learning*, vol. 307, p. 7, 2008.

[29] O. Breuleux, Y. Bengio, and P. Vincent, "Quickly generating representative samples from an RBM-derived process," *Neural Computation*, vol. 23, no. 8, pp. 2058–2073, 2011.

[30] Y. Bengio and É. Thibodeau-Laufer, "Deep generative stochastic networks trainable by backprop," *arXiv*, pp. 1–24, 2013.

[31] I.J. Goodfellow *et al.*, "Generative adversarial nets," *Advances in Neural Information Processing Systems*, pp. 2672–2680, 2014.