Utilization of SAM-based Network for Developing Function Approximation

Minoru Motoki, Hirohito Shintani
Deparment of Information
Communication and Electronic Engineering
National Institute of Technology
Kumamoto College, Kumamoto, Japan
{motoki@kumamoto-nct.ac.jp} {hsintani@kumamoto-nct.ac.jp}

Kazunori Matsuo
Dept. of Control and Information
Systems Engineering National Institute of Technology
Kumamoto College, Kumamoto, Japan
{matsuo@kumamoto-nct.ac.jp}

Thomas Martin McGinnity
College of Science and Technology, Nottingham Trent University
Nottingham, UK
{martin.mcginnity@ntu.ac.uk}

JDIM
Journal of Digital Information Management

ABSTRACT: *We have previously reported progress in developing a multilayer SAM spiking neural network and a training algorithm, suitable for implementation on an FPGA with "On- Chip Learning". Here we report on utilization of a SAM -based network for continuous function approximation, which to date has proved difficult to achieve on a LIF type spiking neural network, by using a spike coding approach called 'NFR-coding'. We demonstrate "interpolated XOR" and 3-polynominal function approximation of this SAM network in computational experiments. It is demonstrated that the SAM network has the capability to perform these function approximations to high accuracy.*

Subject Categories and Descriptors: [C.2 COMPUTER-COMMUNICATION NETWORKS]: Neural nets; [G.1.2 Approximation]

General Terms: Neural Networks, Approximation, Training algorithm, SAM neurons

Keywords: Spiking neural network(SNN), SAM Neuron Model, FPGA implementation, On-chip learning, Function Approximation

Received: 18 June 2022, Revised 10 August 2022, Accepted 24 August 2022

Review Metrics: 0/6, Review Score: 5.05, Inter-reviewer consistency 81.5%

DOI: 10.6025/jdim/2022/20/4/148-155

1. Introduction

The effectiveness of neural networks in deep-learning research and development is well known. Models in these algorithms often use the sigmoidal activation function, or the ReLU activation function, which is derived from the sigmoidal activation function. Alternatively, spiking neural network (SNN) models have been proposed in the biological, brainscience, and engineering areas, and have been applied to a number of industrial applications. In particular, a spiking neuron model can treat time series data with only one neuron element, and has advantages for realisation on digital hardware such as FPGAs.

In this paper, we focus attention on a spiking neuron model called the SAM neuron model proposed by Shigematsu et al..[1] The SAM neuron model was proposed for a brain-type computer, and is a type of Leaky Integrated and Fire (LIF) spiking neuron model. The SAM model is an adaptation of the LIF model, incorporating one additional parameter, and is able to express bursting and other spiking behaviours. It is reported that the model has the advantage of treating time information within a single neu-

ron body, avoiding a reduction in the frequency of output spikes even if the sampling interval is wide.

We achieved to develop a supervised training algorithm for the layered SAM neural network which is constructed by connecting the SAM neuron model in a hierarchical manner. The algorithm aims to implement the SAM network into digital circuits such as FPGAs, while avoiding the use of on-chip circuit multipliers. We have already succeeded to achieve “on-chip supervised learning” on FPGAs[2].

However, application of the SAM network has only been demonstrated with classification problems such as the Iris and WBC datasets. It has not been applied in function approximation or more general problems so far. This paper reports on the application of networks based on the SAM model to two function approximation problem, namely the Interpolated XOR (as in Bohte et al.’s SpikeProp[3]) and a 3rd degree polynomial function approximation (as Iannella’s SNN[4]).

2. SAMNEURAL Network and Trainnig Algorithm

2.1. SAM Neuron Model

The Spike Accumulation and Modulation (SAM) neuron model is a type of spiking neuron model, which reflects biological neuronal functions (Figure 1) The behavior of the SAM neuron model is as follows. At a discrete time *t*, if the *j* -th neuron receives a spike $X_i(t) \in \boldsymbol{B} = \{0,1\}$ = $\{0,1\}$ from the i-th neuron, then the inner potential $U_j(t) \in \boldsymbol{R}$ of the j -th neuron is calculated by the sum of product between the link weights $W_{ji} \in \boldsymbol{R}$ and input spikes $X_i(t)$, plus the addition of the weighted previous inner potential $aV_j(t-1)$ (where α is a decay parameter). Thus,

$$U_j(t) = \sum_{i=1}^{n_1} W_{ji}\, X_i(t) + aV_j(t-1) \qquad (1)$$

The j-th neuron output $X_j(t) \in \boldsymbol{B}$ is obtained by applying the activation function $g()$ to $U_j(t)$, where $g()$ is a step function such that if $U_j(t)$ is less than θ the output is 0,

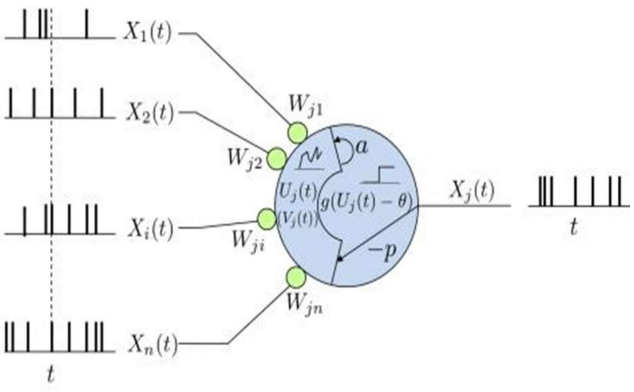


Figure 1. SAM neuron model

corresponding to no spike; if $U_j(t)$ is greater than θ the output is 1 and a spike is generated. θ is the threshold, i.e.,

$$X_j(t) = g\big(U_j(t) - \theta\big), \qquad (2)$$

$$u = U_j(t) - \theta, \qquad (3)$$

$$g(u) = \begin{cases} 0 & (u < 0) \\ 1 & (u \geq 0). \end{cases} \qquad (4)$$

Moreover, the inner potential is decreased by an amount p upon activation, such that

$$V_j(t) = U_j(t) - pX_j(t). \qquad (5)$$

The SAM model is essentially similar to the LIF model except that it treats time discretely, incorporates the term p on activation; and does not set a refractory period.

2.2. SAM Neural Network

We define the membrane potentials and outputs of neurons in the hidden(2nd layer) as

$$U_j^{\langle 2\rangle}(t) = \sum_{i=1}^{n_1} W_{ji}^{\langle 2\rangle}\, X_i^{\langle 1\rangle}(t) + aV_j^{\langle 2\rangle}(t-1), \qquad (6)$$

$$X_j^{\langle 2\rangle}(t) = g\big(U_j^{\langle 2\rangle}(t) - \theta_j{}^{\langle 2\rangle}\big), \qquad (7)$$

$$V_j^{\langle 2\rangle}(t) = U_j^{\langle 2\rangle}(t) - pX_j^{\langle 2\rangle}(t) \qquad (8)$$

Where $V_j^{\langle 2\rangle} = 0$.

For neurons in the 3rd(output) layer, we define

$$U_k^{\langle 3\rangle}(t) = \sum_{j=1}^{n_2} W_{kj}^{\langle 3\rangle}\, X_j^{\langle 2\rangle}(t) + aV_k^{\langle 3\rangle}(t-1), \qquad (9)$$

$$X_k^{\langle 3\rangle}(t) = g\big(U_k^{\langle 3\rangle}(t) - \theta_k{}^{\langle 3\rangle}\big), \qquad (10)$$

$$V_k^{\langle 3\rangle}(t) = U_k^{\langle 3\rangle}(t) - pX_k^{\langle 3\rangle}(t) \qquad (11)$$

Where $V_k^{\langle 3\rangle}(0) = 0$. Here, superscript $\langle l\rangle$ indicates the layer number.

The thresholds of the SAM neuron models are set for each neuron, namely $\theta_j^{\langle 2\rangle}$, $\theta_k^{\langle 3\rangle}$; but are varied during training, in a similar manner to link weights.

Moreover, we define an inner potential u which contains the threshold parameter for deriving the training algorithm, as follows

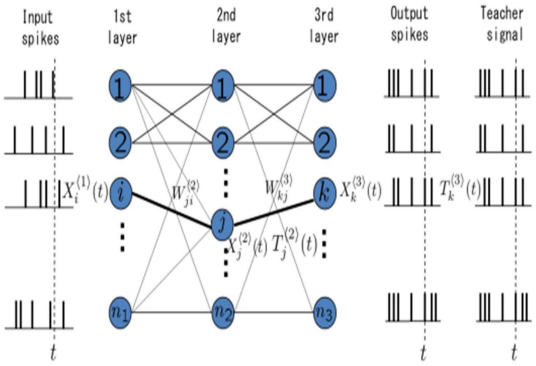


Figure 2. Multilayer SAM neural network

$$u_j^{\langle 2\rangle}(t) \stackrel{\text{def}}{\equiv} U_j^{\langle 2\rangle}(t) - \theta_j^{\langle 2\rangle}, \tag{12}$$

$$u_j^{\langle 2\rangle}(t) \stackrel{\text{def}}{\equiv} U_j^{\langle 2\rangle}(t) - \theta_j^{\langle 2\rangle}, \tag{13}$$

2.3. Training Algorithm for SAM neural network

The training approach for the multilayer SAM neural network is based on an extension of the approach by Motoki et al. [5] We define the objective function E as

$$E \stackrel{\text{def}}{\equiv} \frac{1}{N}\sum_{P=1}^{N}\sum_{t=1}^{t_N}\frac{1}{2}\sum_{k=1}^{n_3}\left(X_{P,k}^{\langle 3\rangle}(t) - T_{P,K}^{\langle 3\rangle}(t)\right)^2, \tag{14}$$

where $T_{P,K}^{\langle 3\rangle}(t) \in \boldsymbol{B}$ is the teacher signal of *k*-th output neuron for a pattern P at a time t. We derived the training algorithm by calculating the gradient for link weights W of E and updating the training parameters (link weights and thresholds). To simplify we considered the number of pattern $P = 1$ in the followings.

For the neurons of the output layer, we can state

$$\frac{\partial E}{\partial W_{kj}^{\langle 3\rangle}} = \sum_{t=1}^{t_N}\left(X_k^{\langle 3\rangle}(t) - T_k^{\langle 3\rangle}(t)\right)\frac{\partial X_k^{\langle 3\rangle}(t)}{\partial W_{kj}^{\langle 3\rangle}}$$

$$= \sum_{t=1}^{t_N}\left(X_k^{\langle 3\rangle}(t) - T_k^{\langle 3\rangle}(t)\right)\frac{\partial X_k^{\langle 3\rangle}(t)}{\partial u_k^{\langle 3\rangle}(t)}\frac{\partial u_k^{\langle 3\rangle}(t)}{\partial W_{kj}^{\langle 3\rangle}}$$

$$= \sum_{t=1}^{t_N}\left(X_k^{\langle 3\rangle}(t) - T_k^{\langle 3\rangle}(t)\right)g'\left(u_k^{\langle 3\rangle}(t)\right)\frac{\partial u_k^{\langle 3\rangle}(t)}{\partial W_{kj}^{\langle 3\rangle}}. \tag{15}$$

By expressing a general equation of progressing time t

$$\frac{\partial u_k^{\langle 3\rangle}(t)}{\partial W_{kj}^{\langle 3\rangle}} = X_j^{\langle 2\rangle}(t) + aX_j^{\langle 2\rangle}(t-1) + a^2X_j^{\langle 2\rangle}(t-2) + \cdots$$

$$+\ a^{t-1}X_j^{\langle 2\rangle}(1)$$

$$-\ p\{a^{t-1}g'\left(u_k^{\langle 3\rangle}(1)\right)\frac{\partial u_k^{\langle 3\rangle}(1)}{\partial W_{kj}^{\langle 3\rangle}}$$

$$+\ a^{t-2}g'(u_k^{\langle 3\rangle}(2))\frac{\partial u_k^{\langle 3\rangle}(2)}{\partial W_{kj}^{\langle 3\rangle}} + \cdots$$

$$+\ ag'(u_k^{\langle 3\rangle}(t-1))\frac{\partial u_k^{\langle 3\rangle}(t-1)}{\partial W_{kj}^{\langle 3\rangle}}\}$$

$$=\ H_{kj}^{\langle 3\rangle}(t) -\ apJ_{kj}^{\langle 3\rangle}(t-1) \tag{16}$$

Here, $H_{kj}^{\langle 3\rangle}(t)$ represents input historical values, defined by

$$H_{kj}^{\langle 3\rangle}(t) \stackrel{\text{def}}{\equiv} aH_{kj}^{\langle 3\rangle}(t-1) + X_j^{\langle 2\rangle}(t). \tag{17}$$

In addition,

$$J_{kj}^{\langle 3\rangle}(t) \stackrel{\text{def}}{\equiv} aJ_{kj}^{\langle 3\rangle}(t-1) + g'\left(u_k^{\langle 3\rangle}(t)\right)\frac{\partial u_k^{\langle 3\rangle}}{\partial W_{kj}^{\langle 3\rangle}} \tag{18}$$

By approximating we used

$$\frac{\partial u_k^{\langle 3\rangle}(t)}{\partial W_{kj}^{\langle 3\rangle}} = H_{kj}^{\langle 3\rangle}(t). \tag{19}$$

As a result,

$$\frac{\partial E(t)}{\partial W_{kj}^{\langle 3\rangle}(t)} = \left(X_k^{\langle 3\rangle}(t) - T_k^{\langle 3\rangle}(t)\right)g'\left(u_k^{\langle 3\rangle}(t)\right)H_{kj}^{\langle 3\rangle}(t). \tag{20}$$

For the hidden layer,

$$\frac{\partial E(t)}{\partial W_{ji}^{\langle 2\rangle}(t)} = \left(X_j^{\langle 2\rangle}(t) - T_j^{\langle 2\rangle}(t)\right)g'\left(u_j^{\langle 2\rangle}(t)\right)H_{ji}^{\langle 2\rangle}(t) \tag{21}$$

where, $T_j^{\langle 2\rangle}(t) \in \boldsymbol{B}$ is Teacher signal of j-th hidden neuron,

$$T_j^{\langle 2\rangle}(t) = \begin{cases} 1 & \left(X_j^{\langle 2\rangle}(t) - \dfrac{\partial E(t)}{\partial X_j^{\langle 2\rangle}(t)}\right) \geq 0.5 \\ 0 & \left(X_j^{\langle 2\rangle}(t) - \dfrac{\partial E(t)}{\partial X_j^{\langle 2\rangle}(t)}\right) < 0.5. \end{cases} \tag{22}$$

Here,

$$\frac{\partial E(t)}{\partial X_j^{\langle 2\rangle}(t)} = \sum_{k=1}^{n_3}\frac{\partial E(t)}{\partial X_k^{\langle 3\rangle}(t)}\frac{\partial X_k^{\langle 3\rangle}(t)}{\partial u_k^{\langle 3\rangle}(t)}\frac{\partial u_k^{\langle 3\rangle}(t)}{\partial X_j^{\langle 2\rangle}(t)}$$

$$= \sum_{k=1}^{n_3}\left(X_k^{\langle 3\rangle}(t) - T_k^{\langle 3\rangle}(t)\right)g'\left(u_k^{\langle 3\rangle}(t)\right)\partial W_{kj}^{\langle 3\rangle}(t). \tag{23}$$

Generally, for the *l*-th layer,

$$\frac{\partial E(t)}{\partial W_{nm}^{\langle l\rangle}(t)} = \left(X_n^{\langle l\rangle}(t) - T_n^{\langle l\rangle}(t)\right)g'\left(u_n^{\langle l\rangle}(t)\right)H_{nm}^{\langle l\rangle}(t) \tag{24}$$

$$W_{nm}^{\langle l \rangle}(t+1) \leftarrow W_{nm}^{\langle l \rangle}(t) - \eta \frac{\partial E(t)}{\partial W_{nm}^{\langle l \rangle}(t)} \quad (25)$$

And,

$$\frac{\partial E(t)}{\partial \theta_n^{\langle l \rangle}(t)} = \left(X_n^{\langle l \rangle}(t) - T_n^{\langle l \rangle}(t)\right) g'\left(u_n^{\langle l \rangle}(t)\right) \quad (26)$$

$$\theta_n^{\langle l \rangle}(t+1) \leftarrow \theta_n^{\langle l \rangle}(t) - \iota \frac{\partial E(t)}{\partial \theta_n^{\langle l \rangle}(t)}. \quad (27)$$

3. NFR-CODING

The manner in which information is encoded into spikes is crucially important both for learning and implementation efficiency. A number of coding systems have been proposed by SNN researchers e.g.[3]. In this paper we state a coding system called NFR-coding which is appropriate for the characteristics of the SAM neuron model. NFR-coding is based on rate coding.

The NFR-coding system is explained as follows. The values of [0.0, 1.0] ∈ $\boldsymbol{R}$ are expressed by n_C neurons and t_C discrete times. Resolution r_C is 1.0/(n_C × t_C). Figure shows an example when $r_C = 0.01, n_C = 10, t_C = 10$. A single neuron can express 0.1 which is the maximum expression value of 1.0 divided by the number of neuron n_C. Similarly a single neuron expresses [0.0, 0.1] within time interval t_C (we refer to this as Neuron Focused). The spikes expressed by a single neuron represent rate coding (Frequency), and therefore the spike rate is the expression value. Moreover, the set of spike intervals is cyclic and the spikes are evenly distributed (a property known as Cyclicity). For example, when expressing the value $x = 0.02$, there are 2 spikes every 5 time units; when expressing a value $x = 0.03$, there are 3 spikes every 3.3 time units on average.

Furthermore, the spikes of n_C neurons within the time [1, t_C] are repeated in the window $t_C + 1$ to $2t_C$. This repeat is for the 'decode' to enhance decode robustness (we refer to this as 'Distinguishable').

A definition of NFR-coding can be formulated as follows. Codes for $x(\in \boldsymbol{R} : [0.0, 1.0]$, rounded in resolution $r_C = 1/(t_C \times n_C))$ are expressed in a matrix $\boldsymbol{A}_x$ whose element is $a_{t,n} \in \boldsymbol{B} = \{0,1\}$ $(t \in N: [1, t_C], n \in N: [1, n_C])$,

$$a_{t,n} \stackrel{\text{def}}{=} \begin{cases} 1 & (n < x \cdot n_C) \\ c(t,n) & \left(x \cdot n_C \le n < (x \cdot n_C + 1)\right) \\ 0 & \left((x \cdot n_C + 1) \le n\right), \end{cases} \quad (28)$$

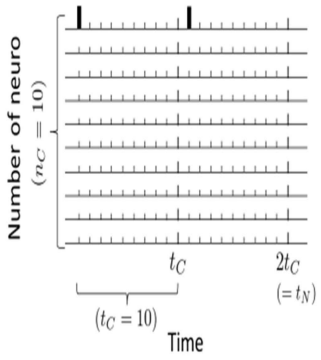
$$c(t,n) = \begin{cases} 1 & (t = 1, \left\lfloor 1 + \frac{1}{f_r \cdot n_C} \right\rfloor, \left\lfloor 1 + \frac{1}{f_r \cdot n_C} + \frac{1}{f_r \cdot n_C} \right\rfloor, \cdots \\ & \mid t \le t_C) \text{ and} \\ & (t = t_C + 1, \left\lfloor t_C + 1 + \frac{1}{f_r \cdot n_C} \right\rfloor \\ & \left\lfloor t_C + 1 + \frac{1}{f_r \cdot n_C} + \frac{1}{f_r \cdot n_C} \right\rfloor, \cdots t_C < t < 2t_C) \\ 0 & otherwise. \end{cases} \quad (29)$$

Here $f_r = x - (n-1)\frac{1}{n_C}$, and means a fragment of the value x.

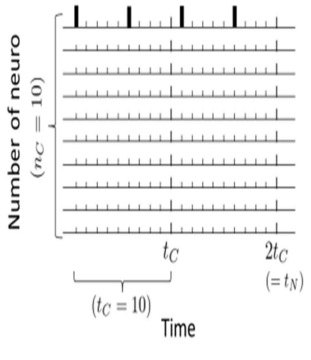
The term $\frac{1}{f_r \cdot n_C}$ in Eq.(29) is the spike interval. An example of NFR-coding A of value x = 0.13 is shown in Fig.4.

Secondly, we explain the decoding method. The identifying (decode) from spike sequences is calculated not from the period $[1, t_C]$ but $[t_C + 1, 2t_C]$. Namely, the decoded value x_d is

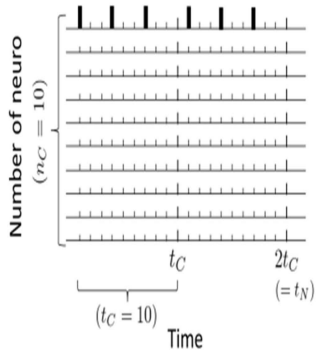
$$x_d = \frac{1}{n_C t_C} \sum_{t=t_C+1}^{2t_C} \sum_{n=1}^{n_C} a_{t,n}. \quad (30)$$



(a) $x = 0.01$



(b) $x = 0.02$



(c) $x = 0.03$

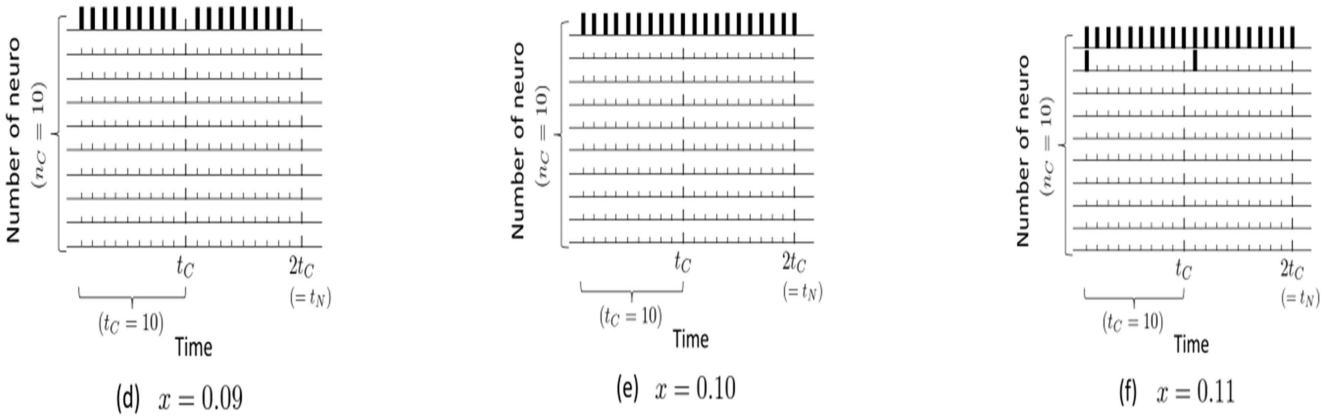


Figure 3. NFR-coding

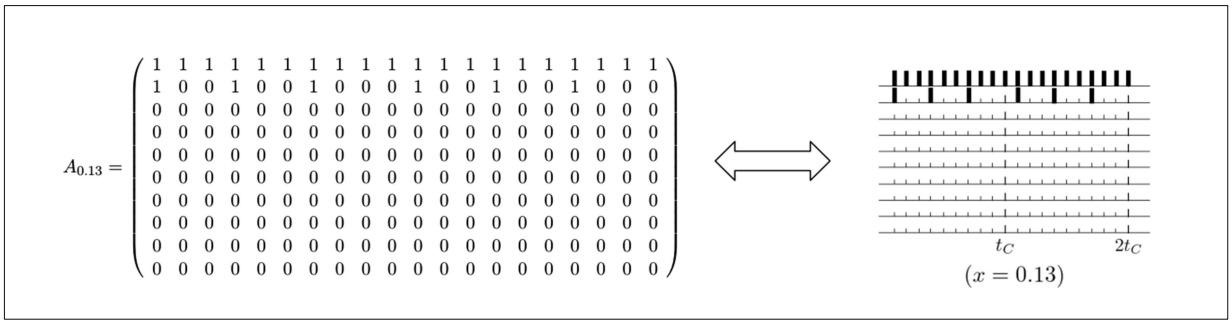


Figure 4. Matrix expresstion of NFR-coding $(x = 0.13, t_C = 10, n_C = 10)$

The NFR-coding approach for the SAM neural network is used because of its properties of Continuity, Frequency, Cyclicity, Distinguishable, Dispersability. NFR-coding satisfies all of these features. We consider that these 5 features are generally required to achieve good performance for neural network learning algorithms. Existing coding methods do not satisfy all such features at the same time, and there is no coding method exploiting a repeat of the spikes.

4. Function Approximation By Computational Experiment

This section describes the application of the SAM network and NFR coding to the two function approximations cases previously mentioned. First, we show the trainability of the model with a 2 term function approximation called Interpolated XOR. Second, we examined the generalization ability with a 3 degree polynomial function.

4.1. Interporated

Bohte et al. applied a training algorithm called SpikeProp to an SRM type SNN, and showed results for the Interpolated XOR task[4]. We examine the SAM neural network trainability for the same task. The function equation is:

$$y = f(x_1, x_2) = 10 + \frac{6}{1 + e^{-2(|x_1 - x_2| - 3)}}. \quad (31)$$

We normalized linearly the values x_1, x_2, y of this function to [0.1, 0.9] in order to be trainable easily for the SAM network. The SAM network was trained using total 961points of training data, 31 points for x_1, x_2 respectively.

The parameters used for the NFR-coding are $n_C = 10, t_C = 10$ and the resolution of the values was $r_C = 0.01$.

The results are shown in Figure 5. The SAM network can express the smooth curved surface. The network configuration is 10:130:10 (input: hidden: output), while the training epoch is 1400.

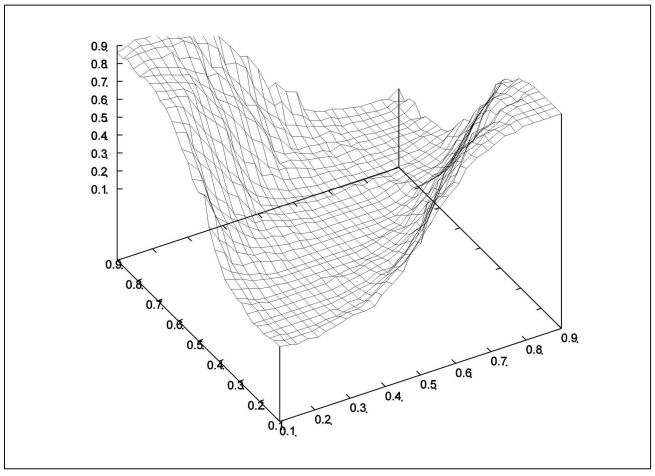


Figure 5. Interpolated XOR

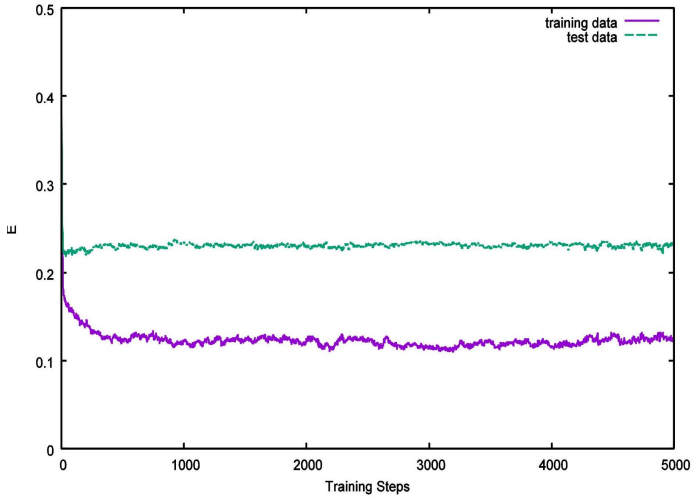


Figure 6. Tranining curve of E

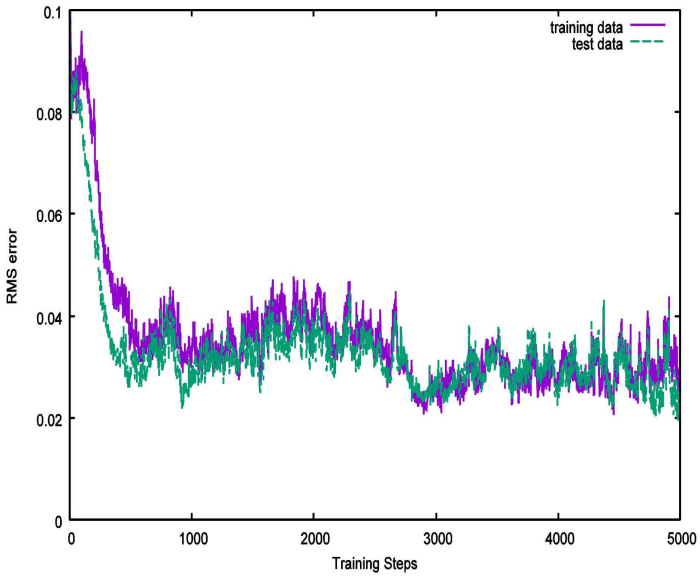


Figure 7. Training curve of RMSE

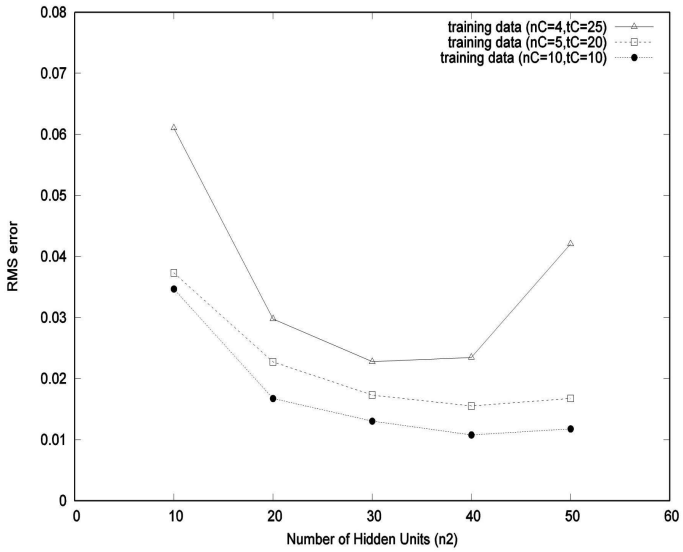


Figure 8. Training ability

4.2. 3rd degree polynomial function approximation

We examined the generalization abilities of the SAM network with the 3rd degree polynomial function approximation. The function used (Eq 32) is the same as the function of Iannella et al.'s paper [4],

$$y = f(x) = (2x - 1.6)^3 - 2x + 4 \qquad (32)$$

This function is neither monotonically increasing nor monotonically decreasing, but demonstrates both upward convex and downward convex behaviours.

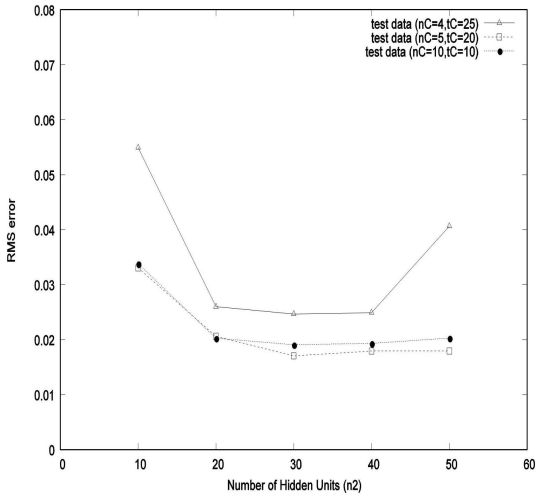


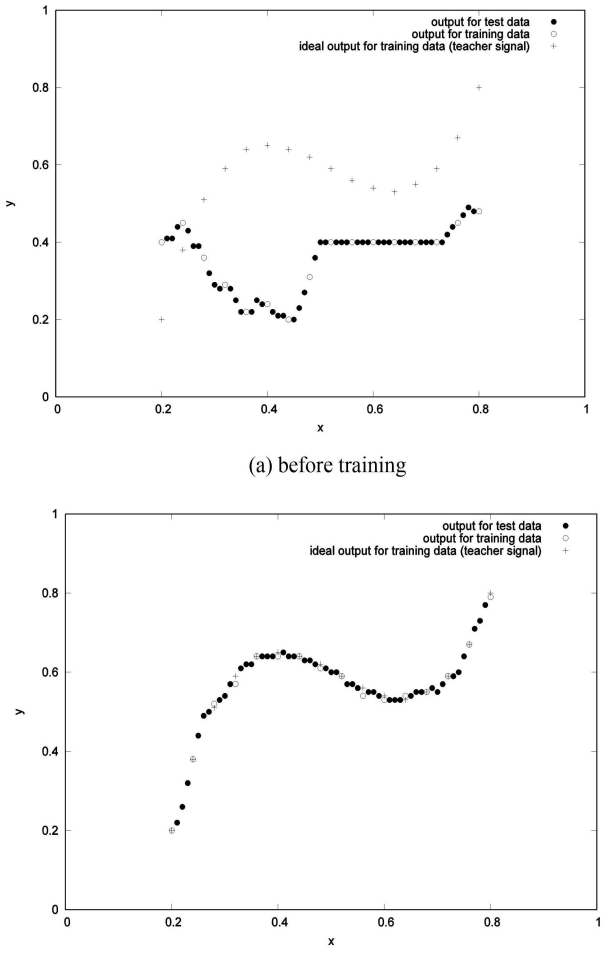
Figure 9. Generalization ability

We generated experimental data by normalizing linearly [0.0, 1.5] to [0.2, 0.8] of x of Eq.(32) and [min max] to [0.2, 0.8] of y. The resolution r_c is 0.01 and the values are rounded to this resolution. Of the 61 data points, we used 16 points for training and 45 points for testing.

The SAM neuron parameters were fixed to $a = 0.9,\ p = 0.5$. For the initial values of link weights and thresholds, we randomly set the values within [-1.0, 1.0) . We tried the experiments with various combinations of (n_C, t_C) including (4, 25), (5,20), (10,10), and the number of hidden units $n_2 = 10,20,30,40,50$. The learning coefficient was set to $\eta = 0.001, 0.0001,$ or 0.00001 $(\iota = 0.2\eta)$. We experimented with 10 conditions of initial seed values of the link weights and thresholds.

As an example of the training, we show the curve of the objective function E (Fig.6) and RMS error of the function approximation (Fig.7), where the conditions are (n_C, t_C) = (5, 20), n_2 = 40, η = 0.0001 $(\iota = 0.2\eta)$. Here E is equivalent to the averaging error per 1 spike, and the RMS error means the averaging error per 1 value of function approximation, as given by Eq.(33).

$$RMSE = \sqrt{\frac{1}{N}\sum_{P=1}^{N}(\hat{y} - y_d)^2} \qquad (33)$$



(a) before training

(b) after training

Figure 10. Function approximation results

where y_d is the decoded value from the network output (function approximation value), $\hat{y}$ is value of function (desired valued), N is number of data (= 61). As shown in this figure, it is clear that both E and the RMS error are decreasing as the training proceeds. Generally, the training curve of the sigmoidal MLP is smooth, whereas the changes in this training curve are more rapid. This phenomena is considered to come from the characteristic SNN feature of outputs = 1 or 0.

In the RMS error graph, we can observe that the RMSE of test data is higher than RMSE of training data in some cases. In other words, it is possible that the approximation is good for the test data, even though the output spikes are slightly different from the teacher spikes by NFR-decoding. This phenomenon cannot be observed in ordinary machine learning. We consider that this phenomenon comes from the fact that the number of training data samples is relatively large (1:3; training data : test data), the resolution of values r_C is not so fine, and the effect of 'Continuity' in NFR-coding. In future work, we plan to confirm this hypothesis by further changing the experimental conditions.

Next, we show the training error and the generalization

error in Figure 8 and Figure 9 respectively. These values are averaged values of the minimum values for the RMS error for training data (training error) and the RMS error for test data (generalization error) during 10000 training epoch each for the 10 sets of initial values of the link weights and thresholds. These figures show that there are cases which the generalization ability is better than the training ability; the best generalization performance are found when $(n_C, t_C) = (5,20\ )$. The improvement in performance ceases if the number of hidden units n_2 is increased, i.e. a saturation is reached.

Finally, we show an example of function approximation results before training and after training in Fig.10, with the parameters set at $n_2 = 40, n_C = 5, t_C = 20$. Before training, the approximated values are random (Fig.10(a)), whereas after training, the approximated values are close to the teacher data at the training epoch 4000 (Fig.10(b)). Moreover, from Fig.10(a) it is observed that the test output values coincide with the training output values, even though the network parameters are random (before training). This result expresses the 'Continuity' aspect of the NFR characteristics as mentioned before, showing the effectiveness of NFR-decoding. Earlier work in [4], reported final learned values within an error of 0.01 in all training data points, whereas the corresponding SAM network's learned values (linearly transformed values) were within 0.128 (maximum error), while the average error was 0.04. We consider that this performance arises from the chosen resolution of the NFR-coding (r_c).

5. Conclusions

This paper describes function approximation experiments using neural networks composed of the SAM neuron model, while exploiting an innovative NFR-coding scheme. In the Interpolated XOR experiment, the SAM network can perform a smooth approximation. Better performance was observed in the 3rd degree polynomial function approximation experiment. The case when the approximation of test data is better than the training data is observed.

The observation that the SAM neuron model (one of most simplest of SNN models) can achieve good performance in function approximation (including a function with both upward convex and downward convex aspects), is important, since approximation using LIF-type SNN is very difficult. Future work will explore other characteristics of this model in detail.

References

[1] Shigematsu, Y. & Matsumoto, G. (2000) Temporal learning rule and dynamicneural network model. *Applied Mathematics and Computation*, 111, 209–218.

[2] Motoki, M., Soh, S.J.H., Matsuo, K., Shintani, H. & McGinnity, T.M. (Under review) A supervised training algorithm for multilayer SAM spiking neural network. *IEEE Transactions on Neural Networks and Learning Systems*.

[3] Bohte, S.M., Kok, J.N. & La Poutré, H.L. (2002) Error-backpropagation in temporally encoded networks of spiking neurons. *Neurocomputing*, 48, 17–37.

[4] Iannella, N. & Back, A.D. (2001) [Special Issue] A spiking neural network architecture for nonlinear function approximation. Neural Networks. *Neural Network,* 14, 933–939.

[5] Motoki, M. & Matsuo, K. (2014) (in Japanese) *An error-correction learning algorithm for SAM neuron model and its HDL design Workshop on Circuits and Systems,* Vol. 27, pp. 244–249.

[6] Maass, W. & Bishop, C.M. (2001). Pulsed Neural Networks. MIT Press: Cambridge, USA.