Spatially Arranged Sparse Recurrent Neural Networks for Energy Efficient Associative Memory

Gouhei Tanaka, *Member, IEEE*, Ryosho Nakane, *Member, IEEE*, Tomoya Takeuchi, *Member, IEEE*, Toshiyuki Yamane, *Member, IEEE*, Daiju Nakano, Yasunao Katayama, *Senior Member, IEEE*, and Akira Hirose, *Fellow, IEEE*

***Abstract*— The development of hardware neural networks, including neuromorphic hardware, has been accelerated over the past few years. However, it is challenging to operate very large-scale neural networks with low-power hardware devices, partly due to signal transmissions through a massive number of interconnections. Our aim is to deal with the issue of communication cost from an algorithmic viewpoint and study learning algorithms for energy-efficient information processing. Here, we consider two approaches to finding spatially arranged sparse recurrent neural networks with the high cost-performance ratio for associative memory. In the first approach following classical methods, we focus on sparse modular network structures inspired by biological brain networks and examine their storage capacity under an iterative learning rule. We show that incorporating long-range intermodule connections into purely modular networks can enhance the cost-performance ratio. In the second approach, we formulate for the first time an optimization problem where the network sparsity is maximized under the constraints imposed by a pattern embedding condition. We show that there is a tradeoff between the interconnection cost and the computational performance in the optimized networks. We demonstrate that the optimized networks can achieve a better cost-performance ratio compared with those considered in the first approach. We show the effectiveness of the optimization approach mainly using binary patterns and apply it also to gray-scale image restoration. Our results suggest that the presented approaches are useful in seeking more sparse and less costly connectivity of neural networks for the enhancement of energy efficiency in hardware neural networks.**

***Index Terms*— Associative memory, energy efficiency, Hopfield network, interconnection cost, iterative learning rule, sparse neural networks, sparse optimization.**

Manuscript received May 22, 2017; revised January 15, 2018, July 20, 2018, and December 23, 2018; accepted February 10, 2019. Date of publication March 15, 2019; date of current version January 3, 2020. The work of G. Tanaka was supported in part by Japan Society for the Promotion of Science KAKENHI under Grant JP16K00326. *(Corresponding author: Gouhei Tanaka.)*

G. Tanaka is with the Department of Electrical Engineering and Information Systems, Institute for Innovation in International Engineering Education, The University of Tokyo, Tokyo 113-8656, Japan, and also with the Institute of Industrial Science, The University of Tokyo, Tokyo 153-8505, Japan (e-mail: gouhei@sat.t.u-tokyo.ac.jp).

R. Nakane and A. Hirose are with the Department of Electrical Engineering and Information Systems, Institute for Innovation in International Engineering Education, The University of Tokyo, Tokyo 113-8656, Japan (e-mail: nakane@cryst.t.u-tokyo.ac.jp; ahirose@ee.t.u-tokyo.ac.jp).

T. Takeuchi is with the Graduate School of Mathematical Sciences, The University of Tokyo, Tokyo 153-8914, Japan (e-mail: take@ms.u-tokyo.ac.jp).

T. Yamane and D. Nakano are with IBM Research—Tokyo, Kawasaki 212-0032, Japan (e-mail: tyamane@jp.ibm.com; dnakano@jp.ibm.com).

Y. Katayama is with IBM Research—Tokyo, Tokyo 103-8510, Japan (e-mail: yasunaok@jp.ibm.com).

This article has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the author.

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2019.2899344

I. Introduction

UNLIKE the traditional rule-based computing, the parallel distributed computing based on neural information processing is able to perform intelligent tasks through learning from sample data, such as pattern recognition, classification, estimation, and prediction. Hardware implementations of artificial neural networks have been widely studied for laboratory experiments and commercial purposes over the two decades [1], [2]. In the past few years, a rapid progress has been made in the development of neuromorphic chips with low power consumption [3], [4]. For enhancing scalability, however, it is necessary to more efficiently implement a massive number of interconnections corresponding to biological synapses. In many current configurations, local interconnections are implemented with a crossbar array architecture and global interconnections are achieved indirectly with a routing system [5]. Therefore, the communication cost for signal transmissions between neuronal units increases as they become distant in the integrated circuit. Since the number of global interconnections tends to dramatically increase with the system size in many existing neural network models, the interconnection problem should be a bottleneck in realizing scalability of hardware neural networks and neuromorphic devices [6], [7].

For reducing the communication cost in hardware neural networks, it is significant to decrease the number of global interconnections as well as lessen the frequency of signal transmissions through each interconnection. These requirements also correspond to the important constraints in the brain and give a clue to understand an efficient synaptic plasticity [8]. Our study particularly focuses on the former point, i.e., how to save the interconnection cost and increase the cost-performance ratio. For this purpose, we develop learning methods which work with more sparse network structures and less costly connection topologies. This approach from the algorithm side would be complementary to the effort for efficient implementation techniques from the hardware side.

The original Hopfield network [9] with full connectivity is a typical example of densely connected recurrent neural

networks. Thanks to the fully connected structure, the Hopfield network has an energy function that monotonically decreases with each update of the neuronal state. Such a property is favorable for many applications including associative memory [10], image restoration [11], and combinatorial optimization problems [12]. In an autoassociative memory task, the Hopfield network embeds multiple patterns into the interconnection weights by a Hebbian learning rule. When a noisy version of one of the stored patterns is given as the initial network state, the network iteratively updates its state and retrieves the correct stored pattern in a successful case. The maximum number of successfully stored patterns is often represented as αN, where N is the number of neurons and α is the storage capacity used for evaluation of the computational performance. It was theoretically shown that the storage capacity of the Hopfield network is $\alpha \sim 0.14$ [10]. This performance is achieved with approximately N^2 interconnections which should cause the interconnection problem for large N.

Motivated by biological and physical interests, the Hopfield network with nonfully connected structures has been widely studied. Sompolinsky [13] investigated diluted Hopfield networks where the fraction c of the interconnections is randomly removed from the original fully connected network. It was found that the storage capacity of the diluted network is approximately given by $\alpha \sim 0.14(1-c)$. This means that the computational performance is degraded linearly with the dilution factor. This early study was followed by the analyses of the associative memory ability of randomly diluted Hopfield networks [14]–[19]. In addition, much attention has been paid to the Hopfield networks with complex topologies including small-world networks [20]–[22], scale-free networks [23], and other architectures [24], [25]. Some studies suggest that homogeneously connected random networks are better than heterogeneously connected ones in the storage capacity when the interconnection density is the same [20], [26], [27]. Most of the above-mentioned studies use the classical one-shot Hebbian learning rule [28] for the storage of patterns, but there is an option to use another learning rule, e.g., the iterative Hebbian learning rule (Perceptron-like learning) [29]–[31], for improving the storage capacity. So far, few studies have explored effective network structures under the iterative learning algorithm. Moreover, although a variety of sparse network structures have been considered, their efficiency in hardware computation has not yet been quantitatively evaluated.

The purpose of this paper is to examine spatially arranged sparse Hopfield-type networks for energy-efficient associative memory in terms of cost-performance relationship. We take two approaches for obtaining sparse recurrent network structures with low interconnection cost: one is to compare different network structures with a fixed sparsity; the other is to optimize the network structure for maximizing its sparsity during the learning process. In the first approach, we focus on sparse modular network structures which are biologically plausible [32]–[34]. We clarify how the storage capacity scales with the number of modules and demonstrate a modification of the interconnection topology that enables a compatibility between large storage capacity and low interconnection cost. In the second approach, we optimize the network structure such that the number of interconnections is minimized under the constraints imposed by a pattern embedding condition. We find a tradeoff between the interconnection cost and the storage capacity in the optimized networks. Our results show that the second approach can give a more cost-effective network structure than the first approach at the expense of computation time for optimization. Finally, we discuss remaining issues and possible efforts for enhancing energy efficiency in the associative memory networks.

The main novelty of this paper is to consider for the first time how to enhance energy efficiency of hardware neural networks for associative memories in a theoretical framework as well as to formulate the optimization problem for finding the optimal interconnection weights that maximize the sparsity of recurrent neural networks for associative memory. Our optimization method is regarded as a new learning method for associative memory, which combines the iterative learning algorithm [29]–[31] and the concept of sparse optimization [35]. The relevance of our method is validated by numerically showing that the optimized network connectivity is very sparse and costless while achieving reasonable computational performance. Our optimization method can be applied to a neural network of any size in principle, although more computation time and computer memory are needed for larger neural networks. If the size of patterns used for associative memory is very large, they can be divided into a set of smaller patterns so that they can be handled with smaller neural networks in parallel and the computation time is saved.

This paper is organized as follows. In Section II, we describe the two approaches for finding sparse recurrent neural networks with the high cost-performance ratio. In Section III, we show the results of associative memory tests with the sparse neural networks obtained by the two approaches. In Section IV, we summarize the results and give a discussion.

II. Methods

The associative memory task is divided into the *learning* phase and the *retrieval* phase. In this section, we introduce two approaches for conducting the associative memory task using sparse recurrent neural networks with low interconnection cost. The first approach is to assume a sparse modular network structure as described in Section II-A. The second approach is to optimize the network connectivity so as to maximize the network sparsity as described in Section II-B. In the retrieval phase, the network state is repeatedly updated to retrieve a correct pattern from a noisy one as explained in Section II-C. The measure for evaluating the interconnection cost of the sparse neural networks is introduced in Section II-D.

We consider a recurrent neural network consisting of N neurons which are interconnected with each other [9]. The neuronal states are represented by an N-dimensional vector $\mathbf{x} = (x_1, \ldots, x_N)^\top$ where $x_i \in [-1, 1]$ denotes the state of neuron i for $i = 1, \ldots, N$. The connection weights are represented by an $N \times N$ weight matrix $W = (w_{ij})$, where w_{ij} denotes the weight of the connection between neuron i and neuron j. We assume that the weight matrix is symmetric, i.e., $w_{ij} = w_{ji}$, and the self-loops are not

allowed, i.e., $w_{ii} = 0$ for all i. The interconnection density d is defined as the number of nonzero components in W divided by $N(N-1)$. The P sample patterns to be stored are represented as N-dimensional binary vectors, $\mathbf{s}^{(k)} = (s_1^{(k)}, s_2^{(k)}, \ldots, s_N^{(k)})^\top$ for $k = 1, \ldots, P$, where $s_i^{(k)} \in \{-1, 1\}$ for $i = 1, \ldots, N$. The set of stored patterns is simply represented as an $N \times P$ matrix $S = (\mathbf{s}^{(1)}\ \mathbf{s}^{(2)}\ \cdots\ \mathbf{s}^{(P)})$.

A. Learning Methods for Fixed Sparse Networks

In the original Hopfield model with full connectivity [9], the weight matrix W is determined with the *one-shot* Hebbian learning rule [28] described as follows:

$$W = \frac{1}{N} S S^\top. \tag{1}$$

The diagonal components are deleted by subtracting $diag(W)$ from W to prohibit self-loops. The symmetric weight matrix W determined in this way enables to well define an energy function that monotonically decreases with each update of the neuronal state [9]. Since the stored patterns are embedded into point attractors corresponding to the local minima of the energy landscape, each stored pattern is retrieved from a perturbed one after iterative updates of the network state, resulting in a successful memory association. However, in many Hopfield-type networks with sparse or diluted connectivity mentioned in Section I, the weight matrix is determined by the one-shot Hebbian rule in (1) and the weights of the absent interconnections are not used in the retrieval phase. This means that some information of the stored patterns assigned to these weights are lost compared with the full connectivity case and that it leads to a low storage capacity compared with the original Hopfield model. A possible way to mitigate the information loss is to distribute the information of the stored patterns only to the weights of the existing interconnections.

Such efficient information coding is possible using the *iterative* Hebbian learning rule [29], [30], which is often called the Perceptron-like learning rule. In fact, some studies have shown that the iterative learning rule yields a better storage capacity than the one-shot learning rule in sparse networks [36], [37]. Therefore, we mainly focus on the iterative learning rule as described in the following.

Each neuron receives the sum of weighted inputs from the other neurons. The net input is often called the local field h_i which is given by

$$h_i(\mathbf{x}) = \sum_{j \in V_i} w_{ij} x_j \quad \text{for } i = 1, \ldots, N \tag{2}$$

where $V_i \subseteq \{1, 2, \ldots, N\} \setminus \{i\}$ stands for the set of indices of neurons that are connected to neuron i. Then, the state of neuron i is updated using the sign function as follows:

$$x_i' = \mathrm{sgn}(h_i(\mathbf{x}) - \theta_i) \quad \text{for } i = 1, \ldots, N \tag{3}$$

where x_i' represents the updated state and θ_i represents the threshold. A necessary condition for associative memory is that the patterns $\mathbf{s}^{(k)}$ for $k = 1, \ldots, P$ correspond to the fixed points of (3). Therefore, the pattern embedding condition can be given by the set of NP simultaneous inequalities as follows [29]:

$$s_i^{(k)}(h_i(\mathbf{s}^{(k)}) - \theta_i) \geq \delta > 0 \quad \text{for } i = 1, \ldots, N \text{ and } k = 1, \ldots, P \tag{4}$$

where δ is a margin parameter introduced to widen the stability regions (i.e., the basins of attraction) of the fixed point attractors corresponding to the stored patterns [29]. Setting $\theta_i = 0$ for all i for simplicity, the above condition is rewritten as follows:

$$s_i^{(k)} \left(\sum_{j \in V_i} w_{ij} s_j^{(k)} \right) \geq \delta \quad \text{for } i = 1, \ldots, N \text{ and } k = 1, \ldots, P. \tag{5}$$

Once the network structure is given, the weights w_{ij} for existing interconnections can be determined to satisfy the condition (5) using an iterative Hebbian algorithm (see Algorithm 1). We initially set $w_{ij} = 0$ for all the weights. Until the condition (5) is satisfied, the weight is repeatedly updated as follows [29]:

$$w_{ij} \leftarrow w_{ij} + \frac{1}{N} s_i^{(k)} s_j^{(k)} \quad \text{for all } j \in V_i. \tag{6}$$

This process will give a suitable weight matrix satisfying the pattern embedding condition (5) after a finite number of iterations if it exists [29]. The same weight matrix can be obtained by setting $\delta = 1$ in the condition (5) and instead replacing (6) with $w_{ij} \leftarrow w_{ij} + s_i^{(k)} s_j^{(k)}/(\delta N)$. It is possible that there is no solution satisfying the condition (5), e.g., for a very sparse network. Therefore, in practice, we terminate the iterative algorithm after a preset number of maximum iterations, $l_{\max}$.

Algorithm 1: Iterative Hebbian Learning Algorithm [29]

```
Initialize the weight variables, w_ij = 0 for 1 ≤ i, j ≤ N ;
Initialize the flag for (5), flag = 0 ;
Initialize the iteration step, l = 0 ;
while flag = 0 or l < l_max do
    flag = 1 ;
    for k := 0 to P do
        for i := 0 to N do
            h_i = Σ_{j∈V_i} w_ij s_j^(k) ;
            if s_i^(k) h_i < δ then
                w_ij ← w_ij + s_i^(k) s_j^(k) / N ;
                flag = 0 ;
            end
        end
    end
    l ← l + 1 ;
end
```

B. Learning Methods With Network Sparsification

To seek a more sparse network structure, we formulate an optimization problem that minimizes the interconnection

cost under the pattern embedding conditions (5). We can use the techniques of sparse optimization, which have been widely applied in compressed sensing, signal processing, image processing, and machine learning [35]. Although a minimization of the total number of interconnections can be directly formulated as an L_0-norm optimization problem, it is well known that this problem is NP-hard in terms of computational complexity [38], [39]. A typical way for reducing the computational cost is to consider a relaxation problem using L_1-norm, which is a linear programming problem and solvable in polynomial time. Thus, we consider an approximated problem using the L_1-norm optimization where the objective function is given by the sum of the absolute values of the weight variables [40], [41]. Recently, the similar approach based on the L_1-norm optimization has been used for understanding energy and resource efficiency in the brain [8], [42], although recurrent connections are not considered. Some other studies have also used optimization methods to examine how biologically inspired constraints influence the structure and function of neural circuits [43]–[45].

In our formulation for recurrent neural networks, the constraints are given by the set of inequalities (5). Since the weight matrix is symmetric and the diagonal elements are null, we have $\sum_{i=1}^{N}\sum_{j=1}^{N}|w_{ij}| = 2\sum_{i=2}^{N}\sum_{j<i}|w_{ij}|$ and we can take the lower triangular elements of W as the independent variables to be optimized. For a given set of pattern vectors $\mathbf{s}^{(k)}$ $(k = 1, \ldots, P)$, the optimization problem is formulated as follows:

$$\min_{W} \sum_{i=2}^{N}\sum_{j<i}|w_{ij}| \tag{7}$$

$$\text{s.t. } s_i^{(k)}\left(\sum_j w_{ij}s_j^{(k)}\right) \geq \delta \quad \text{for } i = 1, \ldots, N \text{ and } k = 1, \ldots, P \tag{8}$$

$$w_{ji} = w_{ij} \quad \text{for all } i, j = 1, \ldots, N \tag{9}$$

$$w_{ii} = 0 \quad \text{for all } i = 1, \ldots, N. \tag{10}$$

This optimization problem can be transformed into a linear programming problem as described in the following. The lower triangular elements are represented as the following vector:

$$\mathbf{v} = (v_1, v_2, \ldots, v_K)^\top \tag{11}$$

where $K \equiv N(N-1)/2$ and

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ v_1 & 0 & 0 & 0 & 0 \\ v_2 & v_N & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & 0 & 0 \\ v_{N-1} & v_{2N-3} & \cdots & v_K & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ w_{21} & 0 & 0 & 0 & 0 \\ w_{31} & w_{32} & 0 & 0 & 0 \\ \vdots & \vdots & \ddots & 0 & 0 \\ w_{N1} & w_{N2} & \cdots & w_{N,N-1} & 0 \end{pmatrix}.$$

First, we rewrite the objective function (7) using the elements of $\mathbf{v}$ as follows:

$$\sum_{i=2}^{N}\sum_{j<i}|w_{ij}| = \sum_{n=1}^{K}|v_n|. \tag{12}$$

A minimization of $|v_n|$ can be reformulated by replacing v_n with the difference between two nonnegative variables as follows [46]:

$$v_n = v_n^+ - v_n^- \quad \text{for } n = 1, \ldots, K \tag{13}$$

where $v_n^+ \geq 0$ and $v_n^- \geq 0$. Assuming $v_n^+ v_n^- = 0$, we obtain $|v_n| = v_n^+ + v_n^-$ because $(v_n^+, v_n^-) = (v_n, 0)$ if $v_n \geq 0$ and $(v_n^+, v_n^-) = (0, -v_n)$ otherwise. We define the two K-dimensional vectors as follows:

$$\mathbf{v}^+ = \left(v_1^+, \ldots, v_K^+\right)^\top \tag{14}$$

$$\mathbf{v}^- = \left(v_1^-, \ldots, v_K^-\right)^\top \tag{15}$$

where $\mathbf{v} = \mathbf{v}^+ - \mathbf{v}^-$. Using $\tilde{\mathbf{v}} \equiv \begin{pmatrix}\mathbf{v}^+ \\ \mathbf{v}^-\end{pmatrix}$, the objective function (7) can be rewritten as $\mathbf{c}_{2K}^\top \tilde{\mathbf{v}}$, where $\mathbf{c}_j = (1, \ldots, 1)^\top$ is the j-dimensional column vector with all elements equal to 1. When all the elements of $\mathbf{c}_{2K}$ are nonnegative as in this case, we can drop the condition $v_n^+ v_n^- = 0$ [46].

Next, we rewrite the constraints (8) using the vector $\tilde{\mathbf{v}}$. We define the N^2-dimensional column vector containing all the elements of the weight matrix W as follows:

$$\mathbf{w} = \begin{pmatrix} \mathbf{w}_1 \\ \mathbf{w}_2 \\ \vdots \\ \mathbf{w}_N \end{pmatrix} \tag{16}$$

where

$$\mathbf{w}_i = (w_{1i}, w_{2i}, \ldots, w_{Ni})^\top \quad \text{for } i = 1, \ldots, N. \tag{17}$$

We also define the $P \times P$ diagonal matrix Q_i consisting of the pattern vector elements as follows:

$$Q_i = \begin{pmatrix} s_i^{(1)} & 0 & \ldots & 0 \\ 0 & s_i^{(2)} & \ldots & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & \ldots & s_i^{(P)} \end{pmatrix} \quad \text{for } i = 1, \ldots, N.$$

Using the componentwise inequality, the set of the NP inequalities (8) can be written as follows:

$$QR\mathbf{w} - \boldsymbol{\delta} \geq \mathbf{0} \tag{18}$$

where

$$Q = \operatorname{diag}(Q_1, Q_2, \ldots, Q_N) \tag{19}$$

$$R = \operatorname{diag}(S^\top, S^\top, \ldots, S^\top) \tag{20}$$

$$\boldsymbol{\delta} = (\delta, \ldots, \delta)^\top \tag{21}$$

$$\mathbf{0} = (0, \ldots, 0)^\top. \tag{22}$$

Since the diagonal elements of W are null and the upper triangular elements can be represented using the lower triangular elements from symmetry, we can find the $N^2 \times K$ matrix E that satisfies the following equation:

$$E\mathbf{v} = \mathbf{w} \tag{23}$$

where $E = (e_{ij})$ with $e_{ij} \in \{0, 1\}$. Substituting (23) into (18) and using $\mathbf{v} = \mathbf{v}^+ - \mathbf{v}^-$, we obtain

$$QRE(\mathbf{v}^+ - \mathbf{v}^-) - \boldsymbol{\delta} \geq \mathbf{0}. \tag{24}$$

This is equivalent to

$$QR\tilde{E}\tilde{\mathbf{v}} - \boldsymbol{\delta} \geq \mathbf{0} \tag{25}$$

where $\tilde{E} \equiv (E, -E)$ is the $N^2 \times 2K$ matrix.

As a result, the optimization problem (7)–(10) is converted to the following linear programming problem with respect to variables $\tilde{\mathbf{v}}$:

$$\min_{\tilde{\mathbf{v}}} \ \mathbf{c}_{2K}^\top \tilde{\mathbf{v}} \tag{26}$$

$$\text{s.t. } QR\tilde{E}\tilde{\mathbf{v}} - \boldsymbol{\delta} \geq \mathbf{0} \tag{27}$$

$$\tilde{\mathbf{v}} \geq \mathbf{0}. \tag{28}$$

Once $\tilde{\mathbf{v}}$ is found by solving the above-mentioned problem, the weight vector $\mathbf{w}$ is obtained from (23). We set $w_{ij} = 0$ if $|w_{ij}| < \epsilon$ where ϵ is the cutoff parameter.

We can add structural constraints in advance by forcing some elements of $\mathbf{v}$ to be zero. In this case, the variables which can be nonzero are represented by $\mathbf{u} = (u_1, \ldots, u_L)$ with $L < K$, where $u_1, \ldots, u_L \in \{v_1, \ldots, v_K\}$. We can find the $K \times L$ matrix F that satisfies the following equation:

$$F\mathbf{u} = \mathbf{v} \tag{29}$$

where $F = (f_{ij})$ with $f_{ij} \in \{0, 1\}$. Introducing new vectors $\mathbf{u}^+$ and $\mathbf{u}^-$, satisfying $\mathbf{u} = \mathbf{u}^+ - \mathbf{u}^-$, and defining $\tilde{\mathbf{u}} = \binom{\mathbf{u}^+}{\mathbf{u}^-}$, we can incorporate the structural constraint into the linear programming problem as follows:

$$\min_{\tilde{\mathbf{u}}} \ \mathbf{c}_{2L}^\top \tilde{\mathbf{u}} \tag{30}$$

$$\text{s.t. } QR\tilde{E}\tilde{F}\tilde{\mathbf{u}} - \boldsymbol{\delta} \geq \mathbf{0} \tag{31}$$

$$\tilde{\mathbf{u}} \geq \mathbf{0} \tag{32}$$

where $\tilde{F} = (F, F)^\top$. Once $\tilde{\mathbf{u}}$ is found by solving the above problem, the weight vector $\mathbf{w}$ is obtained from (23) and (29). We set $w_{ij} = 0$ if $|w_{ij}| < \epsilon$.

In numerical experiments, the linear programming problems are solved using the interior point method solver in MOSEK optimization toolbox [47] operating on the software package MATLAB [48].

C. Retrieval Phase

After the weight matrix W is determined in the learning phase, the associative memory ability is tested in the retrieval phase. We generate a noisy input pattern $\bar{\mathbf{s}}^{(k)}$ by flipping randomly chosen σN components in the pattern $\mathbf{s}^{(k)}$ for $k = 1, \ldots, P$, where $\sigma \in [0, 1]$ represents the noise level. The network state at time step t is represented as $\mathbf{x}(t)$. For pattern k, the initial state is set as $\mathbf{x}(0) = \bar{\mathbf{s}}^{(k)}$. Then, the neuronal states are repeatedly updated as follows:

$$x_i(t+1) = f\left(\sum_{j \in V_i} w_{ij} x_j(t)\right) \quad \text{for } i = 1, \ldots, N \tag{33}$$

where $f(x) = 2/(1 + e^{-x/a}) - 1$ is the sigmoidal activation function and its nonlinearity is adjusted by the slope parameter a. The real vector $\mathbf{x}(t)$ is transformed into a binary vector $\mathbf{y}(t)$ by thresholding as follows: $\mathbf{y}(t) = (y_1(t), \ldots, y_N(t))^\top$ where $y_i(t) = 1$ if $x_i(t) \geq 0$ and $y_i(t) = -1$ otherwise. The state update is performed asynchronously and continued until the network state converges to a steady state or the time step reaches the predefined maximum value $t_{\max}$. We denote the binary vector corresponding to the final network state by $\mathbf{y}_{\mathrm{f}}^{(k)}$.

Now, we can see how the margin parameter δ influences the retrieval process. We denote the optimal weights determined from the optimization problem (7)–(10) by $w_{ij}^*(\delta)$ for $i, j = 1 \ldots, N$. Then, it is obvious that $w_{ij}^*(\delta) = \delta w_{ij}^*(1)$, and therefore, $f(\sum_{j \in V_i} w_{ij}^*(\delta) x_j) = f(\delta \sum_{j \in V_i} w_{ij}^*(1) x_j)$. This means that, instead of using $w_{ij}^*(\delta)$ in the retrieval phase, we can use the optimal weights $w_{ij}^*(1)$ for $\delta = 1$ and the modified activation function $f(x) = 2/(1 + e^{-\delta x/a}) - 1$. Therefore, we only need the results of the optimization calculation for $\delta = 1$. An increase in the δ value is equivalent to a decrease in the slope parameter of the activation function, resulting in a steeper function.

The associative memory ability is evaluated by how the retrieved pattern is close to the correct pattern vector $\mathbf{s}^{(k)}$. For pattern k ($k = 1, \ldots, P$), the overlap between the retrieved pattern and the correct pattern is given by

$$m^{(k)} = \frac{1}{N} \mathbf{s}^{(k)} \cdot \mathbf{y}_{\mathrm{f}}^{(k)} \tag{34}$$

which ranges between -1 and 1. If the memory retrieval is perfect, then $m^{(k)} = 1$. For evaluation of the computational performance, we use the average overlap m defined as follows:

$$m = \frac{1}{P} \sum_{k=1}^{P} m^{(k)}. \tag{35}$$

We denote by $P_{\max}$ the maximum number of stored patterns such that $m \geq 1 - \xi$ where ξ ($0 \leq \xi \ll 1$) represents the acceptable error.

D. Cost Evaluation

In hardware neural networks, short-range communications are less costly than long-range ones [5]. Therefore, we simply assume that the communication cost is approximately evaluated by the interconnection cost for the existing interconnections in a 2-D space. We suppose that N neurons are arranged like grid points in the 2-D square space as illustrated in Fig. 1. The size of the square is $N^{(1/2)} \times N^{(1/2)}$. The length of the interconnection between neuron i at (x_i, y_i) and neuron j at (x_j, y_j) is defined as the L_1 (Manhattan) distance $D(i, j) = |x_i - x_j| + |y_i - y_j|$, which corresponds to the number of hops along the grid lines for signal transmissions. The total length of the interconnections, normalized by the space size, is given by

$$D = \frac{1}{N} \left(\sum_{i=1}^{N} \sum_{j \in V_i} D(i, j) \right). \tag{36}$$

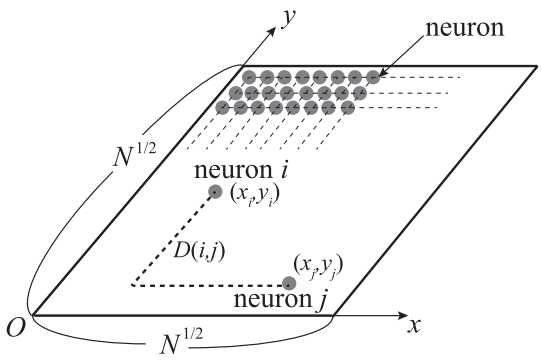

Fig. 1. Schematic of the arrangement of neuronal units in the square space. The size of the space is $N^{(1/2)} \times N^{(1/2)}$ for a network consisting of N neurons. The interconnection cost for connecting neuron i and neuron j in this space is evaluated by the L_1-norm (Manhattan) distance $D(i, j)$ as indicated by the dashed line.

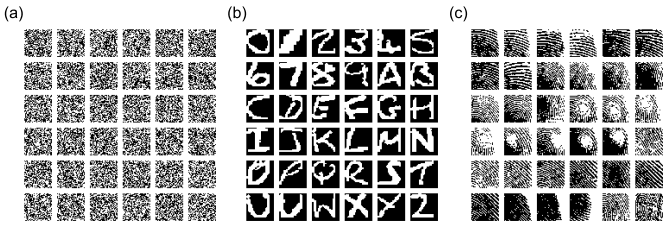

Fig. 2. Examples of the binary image patterns with size 32 × 32 used for the associative memory test. (a) Random patterns. (b) Handwritten character images [49]. (c) Fingerprint images [50].

We define the normalized interconnection cost as follows:

$$C = D/D_0 \tag{37}$$

where D_0 is the total cost for the original fully connected Hopfield network, i.e., $D_0 \equiv (1/N) \sum_{i=1}^{N} \sum_{j=1, j\neq i}^{N} D(i, j)$.

III. Results

A. Setup of Numerical Experiments

We mainly used the three kinds of image patterns shown in Fig. 2 for associative memory tests. The random patterns shown in Fig. 2(a) were generated by randomly assigning black or white color to each pixel with probability 0.5. Fig. 2(b) shows the handwritten character images from the U.S. postal database [49], including 26 alphabets and 10 digits. Fig. 2(c) shows the fingerprint images from the database used for Fingerprint Verification Competition [50]. The size of all of these images was scaled to 32 × 32. To handle these images, we fixed the number of neurons at $N = 1024$. We set the parameters at $\delta = 1$, $a = 0.1$, $\epsilon = 0.001$, $\xi = 0.05$, $l_{\max} = 50\,000$, unless otherwise noted. Furthermore, in order to demonstrate that our network sparsification is effective for more practical images, we employed gray-scale images in Section III-E.

B. Fixed Network Structures

Once a network structure is given, we can determine the weight matrix by the learning rule as described in Section II-A. The interconnection density d, or the network sparsity, is fixed for fair comparison between different network topologies. We seek a network topology which yields better storage capacity and enables efficient computing with less energy. Sparsity and modularity are the major characteristics of biological brain networks [32]–[34]. These factors can be a key for the brain function to operate with very low power consumption [51]. Based on this notion, we focus on sparse modular network structures in this section. This type of structure is advantageous in terms of interconnection cost as well. We assume that the whole network consists of M modules with the same size. Each module contains N/M locally interconnected neurons. As M increases, the interconnections become localized.

Fig. 3 shows the performance comparison based on the average overlap m between the one-shot rule (1) and the iterative rule (6) in the associative memory tests with random patterns. Fig. 3(a)–(c) correspond to the three types of sparse modular network structures with $M = 1$, $M = 4$, and $M = 16$, respectively. In all the networks, the interconnection density is fixed at $d = 0.05$. The first and second columns show the spatial structures of the modules and the connectivity matrices, respectively. The connectivity in each module is random and there are no intermodule connections. The third and fourth columns show the final overlap m for the one-shot rule and the iterative rule, respectively. In each panel, the results for three different noise levels σ are plotted. The results show that the iterative rule yields much better performance than the one-shot rule. The computational performance for the iterative learning rule is degraded with an increase in the noise level σ, but it is still much higher than that of the one-shot rule. Under the iterative learning rule, the associative memory ability gradually decreases with an increase in the number of modules M. Hereafter, we focus on the iterative learning rule.

Fig. 4 shows how the associative memory ability of the sparse modular networks depends on the number of stored patterns P and the noise level σ for the three kinds of image patterns. Fig. 4(a)–(c) correspond to the network structures shown in Fig. 3(a)–(c), respectively. The gray-scale color in each panel indicates the final overlap m. The white region surrounded by the dashed lines corresponds to the parameter conditions such that $m \geq 1 - \xi$. Therefore, the computational performance can be approximately compared based on the size of the white region. For all the image patterns, the single module network [Fig. 4(a)] yields the best performance. As the number of modules increases, both the memory capacity and the noise tolerance are deteriorated. Although local interconnections are desirable for reducing the communication cost, the increase in the number of modules causes the decline of the computational performance as seen from the comparison shown in Fig. 4(a)–(c). This result indicates that the networks only with very localized interconnections result in low computational performance.

Inspired by the fact that the biological brain networks are not perfectly modularized but have shortcut pathways between the modules [32]–[34], we introduce intermodule connections into the purely modular structure with $M = 16$ shown in Fig. 3(c). We randomly chose 30% of the intramodule connections in each module and rewired them to be intermodule connections. The interconnection density is not changed by this rewiring and kept at $d = 0.05$. By this structural

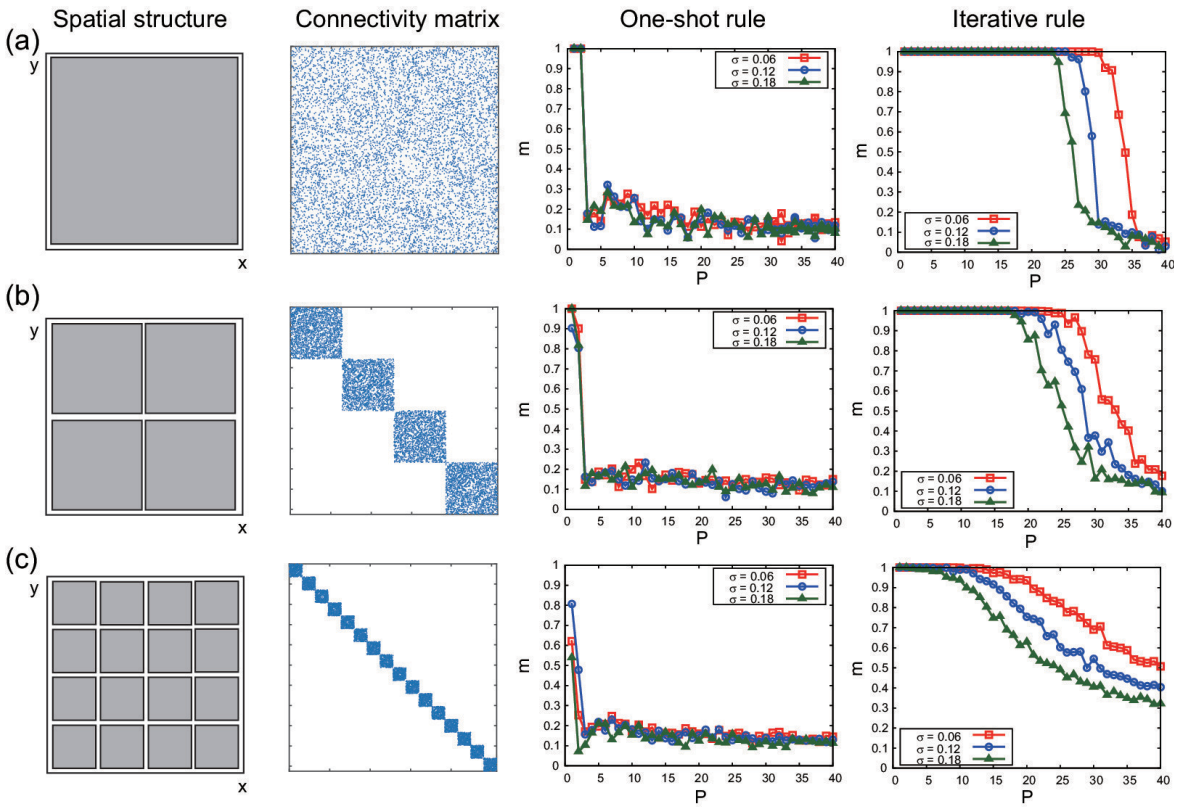

Fig. 3. Performance comparison between the one-shot and iterative Hebbian learning rules in sparse modular networks. The interconnection density is fixed at $d = 0.05$. The first and second columns show the spatial modular structures and the connectivity matrices, respectively. The third and fourth columns show the final overlap m for the random patterns shown in Fig. 2(a) under the one-shot and iterative rules, respectively. (a) $M = 1$. (b) $M = 4$. (c) $M = 16$.

modification, the storage capacity is expected to increase without largely increasing the interconnection cost. If the connections are randomly rewired, the network modification process is similar to that for generating a Watts–Strogatz small-world network [52] from a locally connected network. In the previous studies on small-world networks that are characterized by a small average shortest path length and a large clustering coefficient [20]–[22], it is reported that the storage capacity increases with the rewiring probability. The same property was confirmed in our previous study for different values of the rewiring fraction [51]. We have tested four topologies depending on whether the intramodule connections are regular or random and whether the intermodule connections are regular or random as in [51]. In all the four cases, the enhancement of computational performance was observed. The best performance among these four cases was obtained for the network structure with random intramodule and regular intermodule connections as shown in Fig. 5 (left). This network can be regarded as a hierarchical network including overlapped module structures [53]. Compared with the result for 16-module network shown in Fig. 4(c), the computational performance is considerably improved as shown in Fig. 5, suggesting the benefit of long-range intermodule connections in exchange for the increased interconnection cost. Such a positive effect is brought about by the small-world property which is necessary for embedding pattern correlations between distant neurons belonging to different modules.

The relationship between the interconnection cost C and the maximum number of stored patterns $P_{\max}$ for the sparse

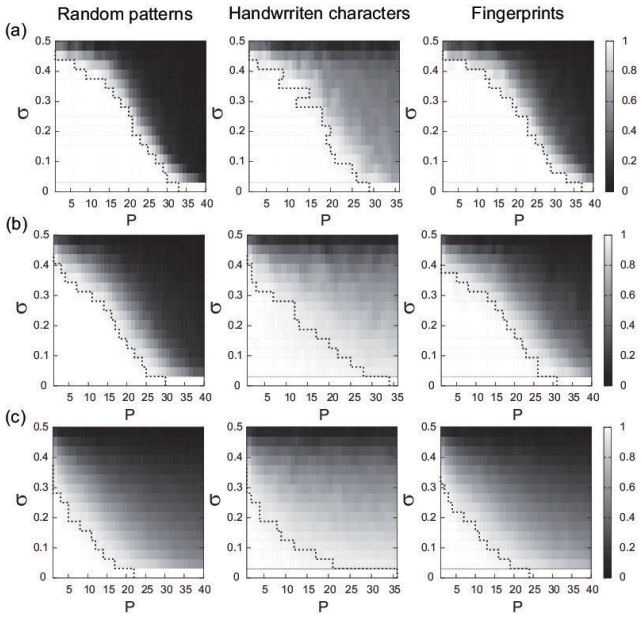

Fig. 4. Effect of the number of modules on the computational performance under the iterative Hebbian rule. The overlap m is indicated by the gray-scale color in the plane of the number of stored patterns P and the noise level σ. (a) $M = 1$ [Fig. 3(a)]. (b) $M = 4$ [Fig. 3(b)]. (c) $M = 16$ [Fig. 3(c)].

modular networks is summarized as shown in Fig. 6. The blue circles indicate the results for the purely modular networks with $M = 1, 2, 4, 8, 16$ where intermodule connections are absent. For these modular networks, the decreasing interconnection cost is represented as $C \sim d/M^{(1/2)}$ in the case of

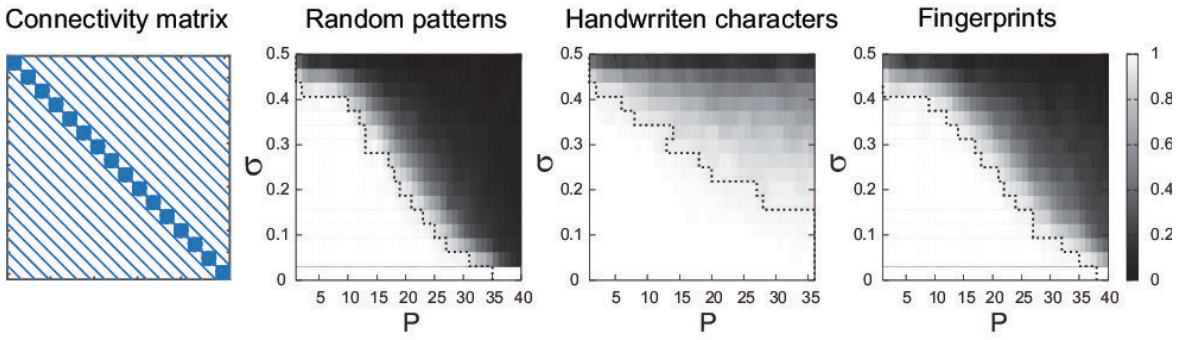

Fig. 5. Effect of intermodule connections on the associative memory performance. The connection matrix representing the network structure where 36% of the intramodule connections of the modular network shown in Fig. 3(c) were rewired to be regular intermodule connections (left). The overlap m indicated by the gray-scale color in the plane of the number of stored patterns P and the noise level σ for the random patterns, the handwritten character images, and the fingerprint images, respectively (right).

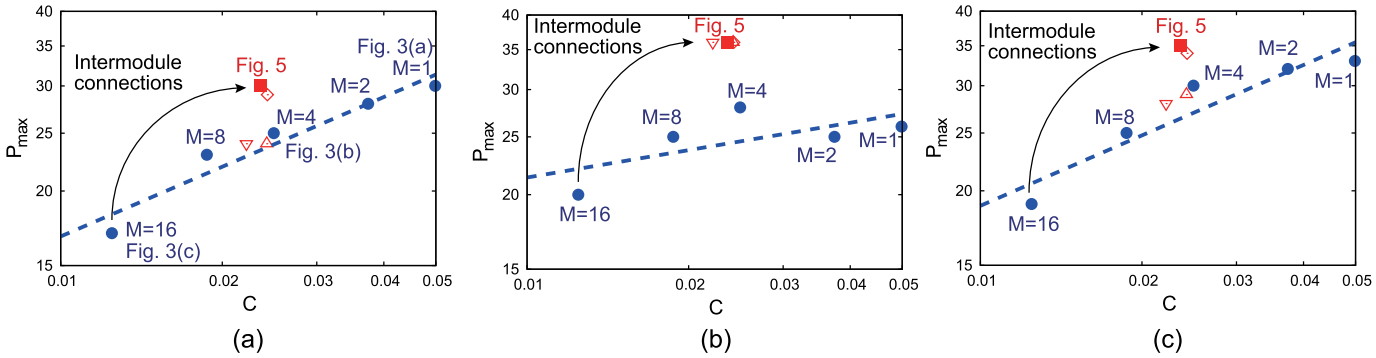

Fig. 6. Cost-performance relationship in the sparse modular networks. The maximum number of stored patterns $P_{\max}$ is plotted against the interconnection cost C. The interconnection density is fixed at $d = 0.05$ and the noise level is at $\sigma = 0.06$. Blue circles: purely modular networks without intermodule connections. Dashed line: fitting line for these five plots. Red marks: modular networks with intermodule connections, including the networks with regular intramodule and regular intermodule connections (upward triangles), regular intramodule and random intermodule connections (downward triangles), random intramodule and regular intermodule connections (filled squares), and random intramodule and random intermodule connections (diamonds). (a) Random patterns. (b) Handwritten character images. (c) Fingerprint images.

uncorrelated random patterns as shown in Fig. 6(a). The fitting (dashed) line indicates that the computational performance degrades with the decreasing interconnection cost in a power law fashion. Namely, there is a tradeoff between the cost effectiveness and the computational capability in the sparse modular networks. The red marks in Fig. 6 correspond to the modular networks with intermodule connections, including the networks with regular intramodule and regular intermodule connections (upper triangles), regular intramodule and random intermodule connections (downward triangles), random intramodule and regular intermodule connections (filled squares), and random intramodule and random intermodule connections (diamonds). As we mentioned, the best results are obtained in the case of random intramodule and regular intermodule connections (filled squares) shown in Fig. 6(a)–(c). These plots show the effectiveness of introducing the long-range interconnections into the 16-module network. In fact, the interconnection cost is similar to that of the 4-module network shown in Fig. 3(b) and the performance is better than or comparable to that of the 1-module network shown in Fig. 3(a). It is suggested that a network structure resembling biological brain networks enable compatibility between good computational performance and low communication cost.

C. Optimization of Network Structures

We perform the learning algorithm with network sparsification by solving the optimization problem formulated in Section II-B. In the optimization problem (26)–(28) where no structural constraint is imposed, the number of variables to be optimized is given by $K = N(N-1)/2$, which requires large computer memory and long computation time for large N. To save the computational cost for learning, we impose structural constraints on the network by setting the matrix F in (29) and solve the optimization problem (30)–(32). We demonstrate that the learning algorithm with network sparsification contributes to finding cost-effective network structures for associative memory.

Following the results in Section III-B, we first restrict the network structure to be similar to that shown in Fig. 5 (left). We assume that the intramodule connections are full (instead of the random ones shown in Fig. 5) in each module and the intermodule connections are regular (as shown in Fig. 5). The motivation for considering this module constraint is the fact that the combination of the local dense connections and the global sparse connections are suitable for the current configuration of the neuromorphic chips as mentioned in the beginning of the Introduction. Before the optimization and

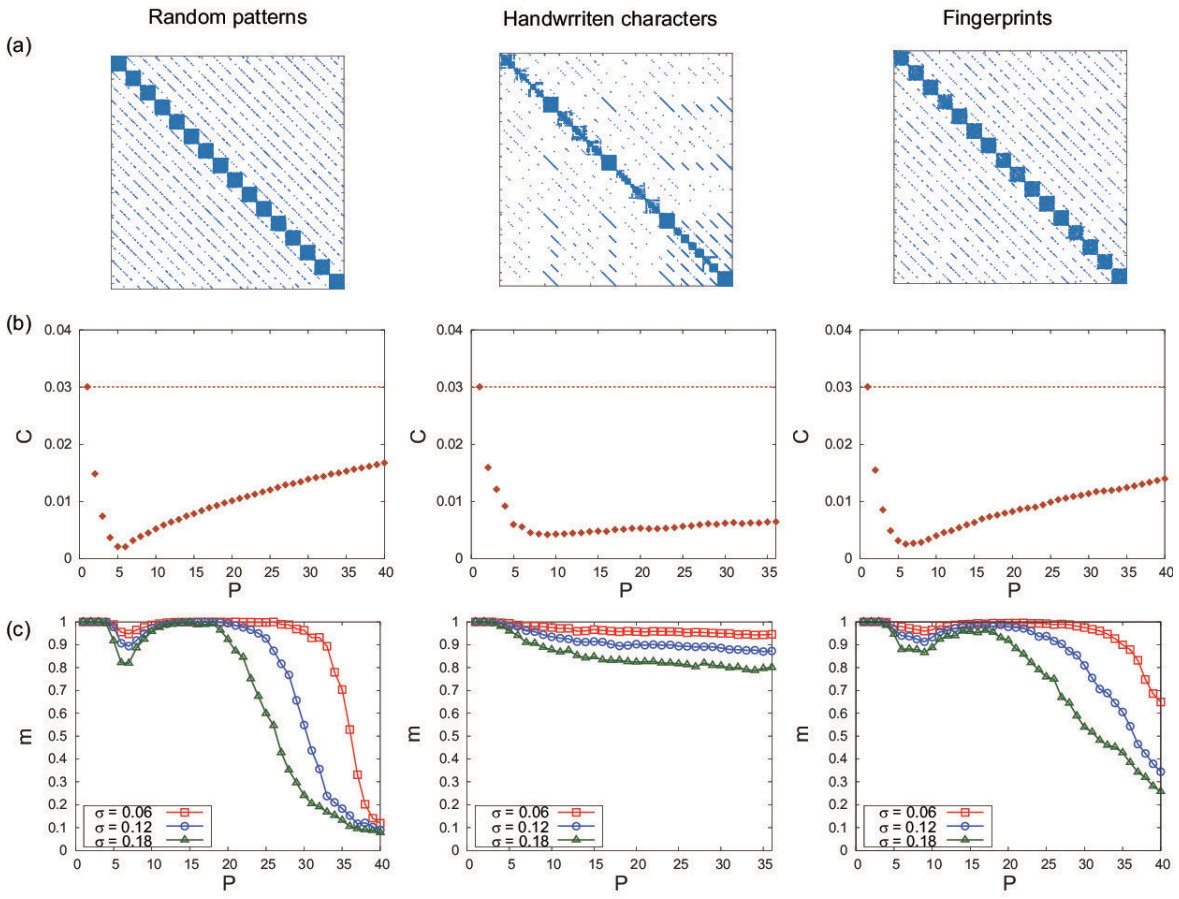

Fig. 7. Results of the network optimization under the constraint of modular structure. The size of the images is 32×32. The interconnection density is fixed at $d \sim 0.05$ before the connection cutoff for all the patterns. (a) Connectivity matrices of the optimized networks for $P = 10$. The size of the matrices is 1024×1024. (b) Interconnection cost C for a variation of the number of stored patterns P. (c) Overlap m plotted against P.

the connection cutoff, the interconnection density is given by $d \sim 0.076$ and the interconnection cost by $C \sim 0.03$. Fig. 7(a) shows the connectivity matrices of the optimized network structures for $P = 10$, which are different depending on the type of the stored patterns. Fig. 7(b) shows the interconnection cost C plotted against the number of stored patterns P. When $P = 1$, the interconnection cost is the same as that before the optimization and the connection cutoff (indicated by the horizontal dashed line) because all the adjusted weights have absolute values larger than the cutoff parameter value ϵ. As P is increased, the number of weights with $|w_{ij}| < \epsilon$ increases and accordingly the interconnection cost dramatically decreases. After the cost touches the bottom at around $P \sim 6$, it gradually increases. Fig. 7(c) shows the final overlap m in the associative memory test using the optimized networks. For the random and fingerprint patterns, the performance is slightly degraded at around $P \sim 6$. This degradation occurs even if the connections with small weights are not cut off from the optimized networks (i.e., even when $\epsilon = 0$). In other words, the fixed point attractors corresponding to the stored patterns have very small stability regions in the optimized networks for these P values. With a further increase in P, the overlap recovers to 1 and then falls again considerably. In contrast, for the handwritten character patterns, the overlap keeps a high level but gradually decreases as P is increased. When the noise level is $\sigma = 0.06$, the maximum number of successfully stored random patterns is $P_{\max} = 30$, which is achieved by the optimized network with interconnection cost $C \sim 0.014$. This optimized network yields the computational performance comparable to that for the network shown in Fig. 5 inspite of its lower interconnection cost.

Next, we demonstrate the result under the structural constraint that prohibits long-range connections. If the L_1 distance between neuron i and neuron j is larger than the radius $N^{(1/2)}/2$ in the 2-D square space, we remove the weights w_{ij} from the variables to be optimized. This radius constraint is motivated by the notion that shorter communications between neuronal units are more power efficient in the neuromorphic hardware [6]. Before the optimization and the connection cutoff, the interconnection density is given by $d \sim 0.036$ and the interconnection cost by $C \sim 0.18$. Fig. 8(a) shows the connectivity matrices of the optimized networks for $P = 10$, which are different depending on the type of the stored patterns. The interconnections outside the diagonal belt are absent due to the structural constraint. The interconnection cost and the final overlap are shown in Fig. 8(b) and (c), respectively, which are qualitatively similar to those shown in Fig. 7(b) and (c). In this case, however, a perfect memory association is achieved even at $P = 40$ for the random and fingerprint patterns as shown in Fig. 8(c). This is reasonable

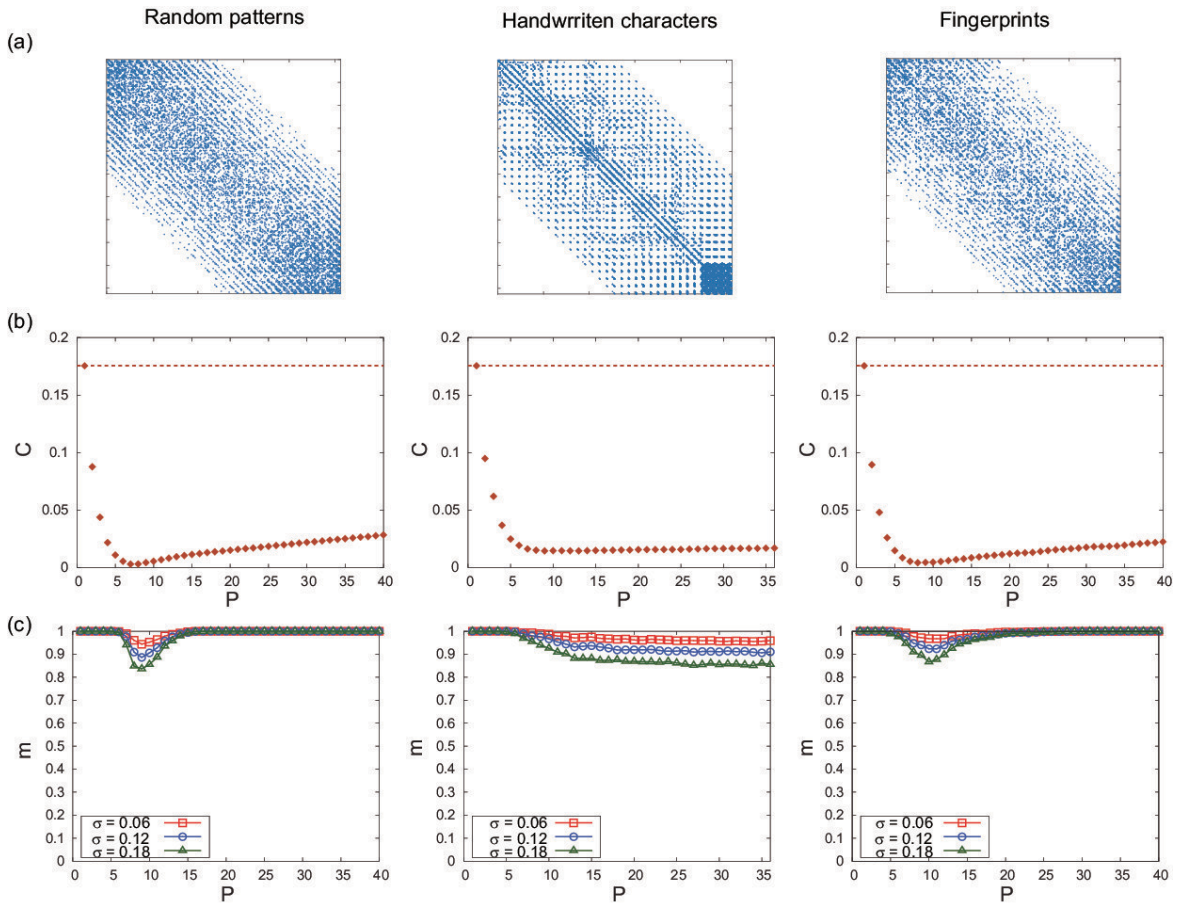

Fig. 8. Results of the network optimization under the radius constraint. The size of the images is 32×32. The interconnection density is fixed at $d \sim 0.05$ before the connection cutoff for all the patterns. (a) Connectivity matrices of the optimized networks for $P = 10$. The size of the matrices is 1024×1024. (b) Interconnection cost C for a variation of the number of stored patterns P. (c) Overlap m plotted against P.

because the optimized networks under the radius constraint are less sparse than those under the module constraint.

The cost-performance relationship of the optimized networks for the random patterns is summarized as shown in Fig. 9. The green downward triangles and orange upward ones correspond to the results of the optimized networks under the module constraint shown in Fig. 7 and the radius constraint shown in Fig. 8, respectively. Only the successful cases where $m \geq 1 - \xi$ are plotted. Under the module constraint, the memory association fails for $P > 30$, whereas under the radius constraint, the successful memory association is achieved up to $P = 65$. The positive slope of the sequential plots for the optimized networks indicates the tradeoff between the interconnection cost and the computational performance. The slope for the optimized sparse networks is steeper than that for the fixed sparse modular networks trained by the Perceptron-like learning (the dashed line), indicating that the optimization approach can yield more cost-effective networks than the other one with the fixed structures. For example, the interconnection cost C required for storing 30 random patterns in the optimized modular networks (the rightmost green downward triangle, $C \sim 0.13$) is approximately half of that in the fixed modular networks with long-range connections (the red square, $C \sim 0.24$). The results demonstrate that the learning method with network sparsification is highly useful for finding cost-effective sparse recurrent network structures for energy efficient hardware neural networks.

We numerically examine how the computation time for the optimization of the weight matrix depends on the system size N and the number of patterns P. We use random patterns with three different sizes, including $N = 8 \times 8$, $N = 16 \times 16$, and $N = 32 \times 32$. Fig. 10 shows the computation time plotted against P for different network sizes and different types of structural constraints (the radius and module constraints). If the plots are well fitted with a linear function in this figure, then it would indicate that the computation time grows exponentially with P. However, the gap between the neighboring plots for a fixed N seems to decrease as P increases. This suggests that the computation time is a polynomial function of P for a fixed number of N.

A practical option for handling large-scale images is to divide each of them into a set of smaller scale subimages in the same way and process them using smaller scale neural networks. By performing the associative memory task using the subimages in corresponding positions, we can obtain smaller scale retrieved patterns and combine them into a large-scale retrieved pattern. The learning and retrieval processes in the different positions are executable in parallel, meaning that the computation time is determined not by the size of the original large-scale images but by the size of the subimages. Fig. 11

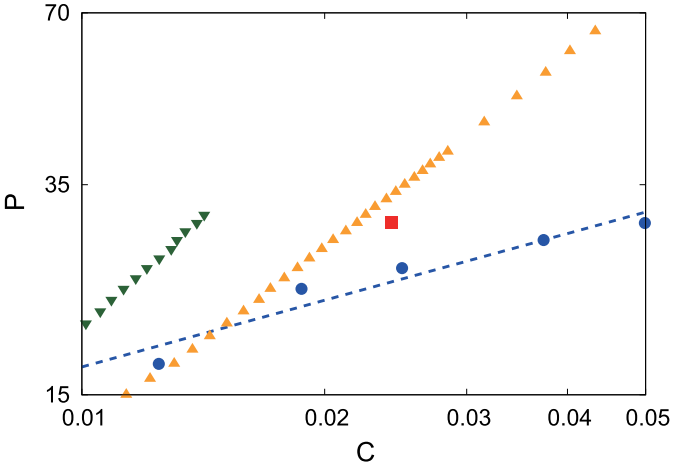

Fig. 9. Cost-performance relationship in the optimized networks for the random patterns of size 32×32. The number of stored patterns P is plotted against the interconnection cost C for the cases where $m \geq 1 - \zeta$. The noise level is fixed at $\sigma = 0.06$. The green downward triangles and the orange upward ones correspond to the optimized networks in Fig. 7 under the module constraint and Fig. 8 under the radius constraint, respectively. The blue circles and red square are the same as those in Fig. 6.

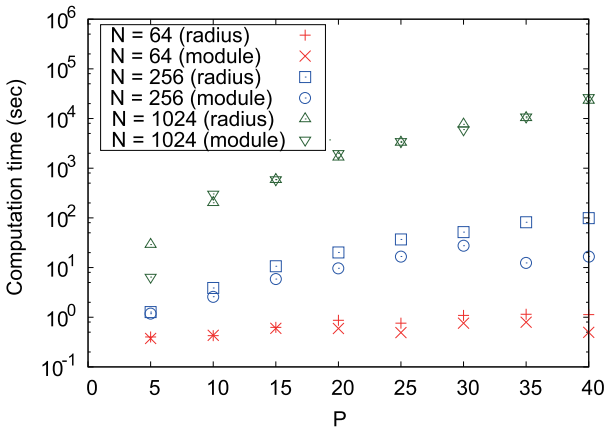

Fig. 10. Computation time for the optimization of the weight matrix, plotted against the number of stored patterns. The different marks correspond to different network sizes and different structural constraints: $N = 64$ and the radius constraint (red pluses); $N = 64$ and the module constraint (red crosses); $N = 256$ and the radius constraint (blue squares); $N = 256$ and the module constraint (blue circles); $N = 1024$ and the radius constraint (green upward triangles); $N = 1024$ and the module constraint (green downward triangles).

shows the result of the network optimization for handwritten character and fingerprint images with size 64×64, which are obtained by rescaling the images of size 32×32 shown in Fig. 2(b) and (c). We divide these images into four subimages of size 32×32 and perform the associative memory task under the radius constraint. The optimized weight matrices for the four parts are represented as the four diagonal blocks in the unified matrices as shown in Fig. 11(a). For handwritten character images, the division of the images is successful for $P = 6$ but not for $P = 9$, because the top left diagonal block is present in the former case but missing in the latter case. The null diagonal block is caused by a high correlation among the subimages in the corresponding position. For fingerprint images, such a null diagonal block is not found for $P \leq 40$. Fig. 11(b) shows the interconnection costs for the optimized networks, which are quite small compared with that ($C = 1$)

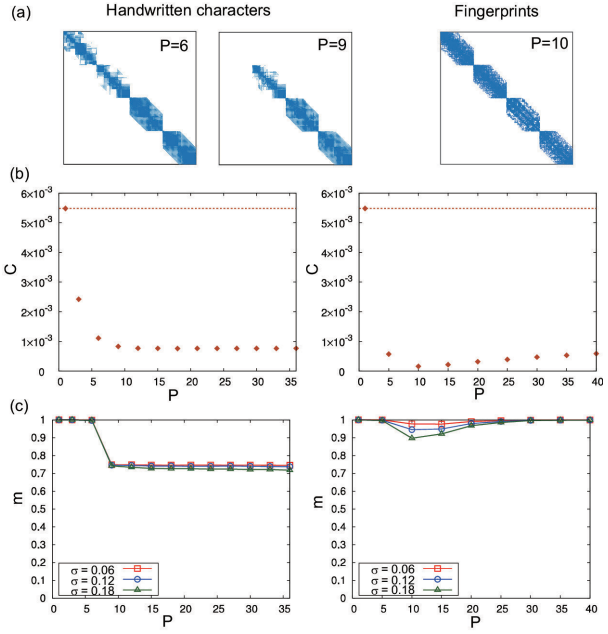

Fig. 11. Results of the network optimization under the radius constraint. The size of the images is 64×64. Each image is divided into four 32×32 subimages and the learning method is applied to each subimage. The interconnection density is fixed at $d \sim 0.05$ before the optimization and the connection cutoff in each neural network. (a) Connectivity matrices of the optimized networks. The size of the matrices is 4096×4096. (b) Interconnection cost C for a variation of the number of stored patterns P. (c) Overlap m plotted against P.

for the fully connected networks of size 4096×4096. Fig. 11(c) (left) shows that the overlap for the handwritten character images falls from 1 to around 3/4 at $P \sim 9$, due to the failure of obtaining an appropriate upper left diagonal block. For fingerprint images, the associative memory performance is good as shown in Fig. 11(c) (right) and comparable to that shown in Fig. 8(c) (right). The above-mebtioned results demonstrate that a division of images into subimages is a valid and feasible way for reducing the time for learning large-scale images if the subimages in corresponding positions are not highly correlated with each other. We omit the results for random patterns because the size of the random subimages is equal to 32×32 and the performance for those subimages is already shown in Fig. 8(c) (left). The total performance for random images of size 64×64 is approximately given by the average of the overlaps for the four random subimages, which is almost the same as those shown in Fig. 8(c) (left). In this case, the normalized cost C is much lower than that in Fig. 8(b) (left) because D_0 [the denominator in (37)] for the full connectivity case is much larger for $N = 4096$ than for $N = 1024$.

D. Modified Objective Function

The objective function (7) in the optimization problem can be modified to incorporate the information of the distance of interconnections. As an example, we consider the following objective function:

$$\min_{W} \sum_{i=2}^{N} \sum_{j<i} D(i, j)|w_{ij}| \tag{38}$$

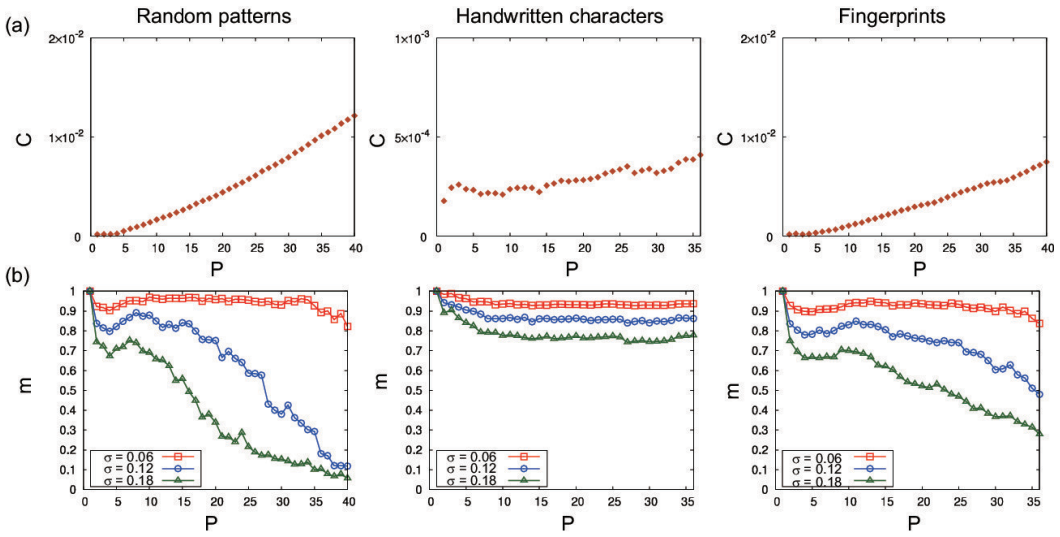

Fig. 12. Results of the network optimization under the radius constraint. The interconnection density is fixed at $d \sim 0.05$. (a) Connectivity matrix of the optimized networks. (b) Interconnection cost C for a variation of the number of stored patterns P. (c) Overlap m plotted against P.

where $D(i, j)$ is the distance between neurons i and j as defined in Section II-D. We can expect that the weights corresponding to long connections are more likely to be reduced (and removed) than those corresponding to short connections. The optimization problem composed of the objective function (38) and the constraints (8)–(10) can be transformed into a linear programming problem in a similar way to that in Section II-B. Fig. 12(a) shows that the interconnection costs of the optimized networks under the modified objective function are much smaller than those for the original objective function shown in Fig. 8(b). However, the overlaps for the modified objective function shown in Fig. 12(b) are not better than those for the original objective function shown in Fig. 8(c). As demonstrated earlier, the objective function in our learning method significantly affects the optimized weight matrix which determines the cost and performance of the associative memory model.

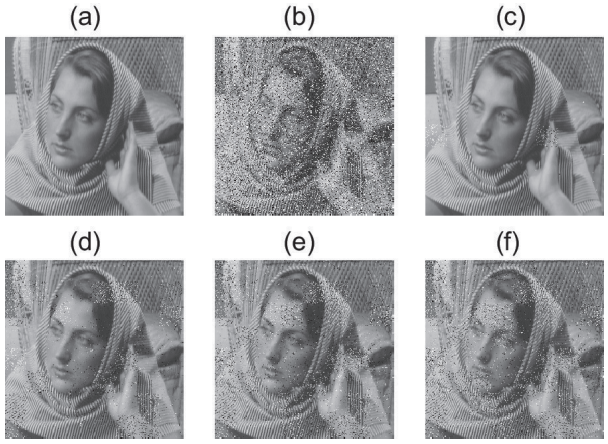

Fig. 13. (a) Original "Barbara" image. (b) Noisy image with 20% salt-and-pepper noise. (c)–(f) Recovered images for (c) $P = 4$, (d) $P = 8$, (e) $P = 12$, and (f) $P = 16$.

E. Application to Gray-Scale Image Restoration

So far, we have tested our method using binary patterns (Fig. 2). Here, we demonstrate that our method is applicable to restoration of gray-scale standard images (Supplementary Fig. S1). For associative memory with P images, the images with indices $p = 1, \ldots, P$ were used. Each image consists of 256×256 pixels and each pixel takes 256 gray levels ([0, 255]). To handle these images with binary neural networks, each image was divided into 64 subimages of size 32×32 (Supplementary Fig. S2), and then, each subimage was converted to 8 binary patterns of size 32×32 by representing a pixel value as an 8-bit binary sequence (e.g., 76="01001100") and collecting binary values in the corresponding bit. From the above-mentioned procedure, 512 ($= 64 \times 8$) binary patterns of size 32×32 were obtained from each gray-scale image. In an image restoration test with P standard images, we performed the learning under the radius constraint and the retrieval for 512 sets of P binary patterns in corresponding parts. A noisy version of a standard image was divided into binary patterns in the same way and used in the retrieval phase. After the retrieval, we unify 512 retrieved binary patterns into a gray-scale image.

The results of the experiments are shown in Fig. 13 and Table I. In the retrieval phase, by adding 20% salt-and-pepper noise to the original "Barbara" image [Fig. 13(a)], we generated a noisy image [Fig. 13(b)]. The images retrieved with the optimized sparse recurrent neural networks are shown in Fig. 13(c)–(f) for $P = 4, 8, 12, 16$, respectively. Image retrieval performance is evaluated using peak signal-to-noise ratio (PSNR), defined as $\mathrm{PSNR} = 20 \log_{10}(255/\mathrm{rms})$ where rms is the root mean square difference between a retrieved image and the original image. A higher value of PSNR indicates a higher similarity between them, or a more successful image restoration. Table I shows the average sparsity of the optimized connection matrices obtained by our method, the interconnection cost C defined by (37), and the PSNR value. The results indicate that our method achieves gray-scale image restoration with highly sparse and costless recurrent neural networks as in the case of binary pattern retrieval. The degradation of the PSNR values with an increase in the

TABLE I
RESULTS FOR GRAY-SCALE IMAGE RESTORATION

Pattern	Average sparsity	Cost C	PSNR (dB)
Noisy image	——	——	12.43
Retrieved ($P = 4$)	6.41×10^{-2}	3.01×10^{-2}	32.43
Retrieved ($P = 8$)	1.51×10^{-2}	6.59×10^{-3}	20.32
Retrieved ($P = 12$)	1.67×10^{-2}	7.40×10^{-3}	17.99
Retrieved ($P = 16$)	2.18×10^{-2}	9.83×10^{-3}	17.45

number of stored images is a significant issue to be overcome in a future study, which also occurs for the normal Hopfield network.

IV. CONCLUSION AND DISCUSSION

We have explored sparse recurrent neural networks for associative memory to realize energy efficient information processing in hardware neural networks. We have presented the two effective learning algorithms: one is the iterative Hebbian learning rule for given network structures; the other is the sparse optimization which maximizes the network sparsity under the constraints of pattern embedding conditions. We have shown that, for fixed sparse modular network structures, the iterative rule is much more effective than the one-shot rule. Moreover, we have demonstrated that adding intermodule connections to the purely modular networks is a good option to improve the balance between the interconnection cost and the computational performance. We have clarified that the network topology has a substantial impact on the associative memory ability. In the learning method with network optimization, we have demonstrated the tradeoff between the cost effectiveness and the storage capacity in the optimized network structures. We have shown that the proposed learning method is quite useful for finding a good network structure with high cost-performance ratio. In practice, the optimized weight matrix will be obtained by off-chip learning and used for low-power on-chip computation for pattern retrieval. The optimization-based learning is suited for batch processing, whereas the Perceptron-like learning is available for online processing.

The presented learning methods rely on the pattern embedding condition (5) represented by the set of simultaneous inequalities. The margin parameter δ controls the size of the stability regions (the basins of attraction) of the point attractors corresponding to the stored patterns. In this paper, we have fixed the value of δ, but the effect of δ on the computational performance should be clarified in a future work. It is possible to set different values of δ for each of the inequalities. Our problem formulation suggests that the sparsification of other recurrent neural networks is possible in a similar way if there is a conditional equation or inequality, under which good computational performance is expected, such as the pattern embedding condition for autoassociative memory.

In the optimization approach, we have reduced the number of interconnections by setting $w_{ij} = 0$ if $|w_{ij}| < \epsilon$. If ϵ is too large, then much information would be lost and the computational performance could be worse. If ϵ is too small, then the network sparsity would not be enhanced and a low interconnection cost would not be achieved. Therefore, it is significant to develop a method to appropriately set the value of ϵ for realizing both cost effectiveness and high performance.

The optimization approach is extremely useful for finding cost-effective network structures, but an excellent computational environment would be necessary for handling larger scale networks. In our formulation, we use the $N^2 \times 2K$ matrix E which contains around N^4 elements if structural constraints are not imposed. For large N, the operation of this matrix would require a large-capacity computer memory. The structural constraints are useful for reducing the number of variables in the optimization problem, saving the computer memory, and speeding up the optimization calculation. As demonstrated in Sections. III-C and III-E, a practical strategy for handling large-scale image data is to divide them into small subimages and process them using smaller scale neural networks. Another formulation of the optimization problem using more efficient information representation and coding could be worth investigation. It is also a future work to increase the memory capacity by improving the proposed method, e.g., by changing the spatial constraint (i.e., the initial condition of the optimization calculation) and the optimization solver.

In this paper, we have considered binary neural networks and used binary and gray-scale image patterns for the associative memory test. For more practical applications of the proposed methods, the performance evaluation should be carried out using color images. In such a case, the noise type could be an additional factor to influence the computational performance. Our method of learning with network sparsification can be applied to other variants of recurrent neural networks including complex-valued neural networks [54], [55] and other variants of associative memories including bidirectional associative memories [56]. The formulation of optimization problems based on the similar idea for other computational tasks, such as time series prediction and combinatorial optimization, is also a significant issue for expanding the applications of sparse recurrent neural networks and realizing energy-efficient neural information processing.

REFERENCES

[1] F. M. Dias, A. Antunes, and A. M. Mota, "Artificial neural networks: A review of commercial hardware," *Eng. Appl. Artif. Intell.*, vol. 17, no. 8, pp. 945–952, Dec. 2004.
[2] J. Misra and I. Saha, "Artificial neural networks in hardware: A survey of two decades of progress," *Neurocomputing*, vol. 74, nos. 1–3, pp. 239–255, 2010.
[3] P. A. Merolla *et al.*, "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345, no. 6197, pp. 668–673, Aug. 2014.
[4] B. V. Benjamin *et al.*, "Neurogrid: A mixed-analog-digital multichip system for large-scale neural simulations," *Proc. IEEE*, vol. 102, no. 5, pp. 699–716, May 2014.
[5] D. Kuzum, S. Yu, and H.-S. P. Wong, "Synaptic electronics: Materials, devices and applications," *Nanotechnology*, vol. 24, no. 38, 2013, Art. no. 382001.
[6] J. Hasler and B. Marr, "Finding a roadmap to achieve large neuromorphic hardware systems," *Frontiers Neurosci.*, vol. 7, no. 118, Sep. 2013.
[7] Y. Katayama, T. Yamane, D. Nakano, R. Nakane, and G. Tanaka, "Wave-based neuromorphic computing framework for brain-like energy efficiency and integration," *IEEE Trans. Nanotechnol.*, vol. 15, no. 5, pp. 762–769, Sep. 2016.
[8] J. Sacramento, A. Wichert, and M. C. van Rossum, "Energy efficient sparse connectivity from imbalanced synaptic plasticity rules," *PLoS Comput. Biol.*, vol. 11, no. 6, 2015, Art. no. e1004265.

[9] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proc. Nat. Acad. Sci. USA*, vol. 79, no. 8, pp. 2554–2558, 1982.
[10] R. McEliece, E. Posner, E. Rodemich, and S. Venkatesh, "The capacity of the Hopfield associative memory," *IEEE Trans. Inf. Theory*, vol. IT-33, no. 4, pp. 461–482, Jul. 1987.
[11] J. K. Paik and A. K. Katsaggelos, "Image restoration using a modified Hopfield network," *IEEE Trans. Image Process.*, vol. 1, no. 1, pp. 49–63, Jan. 1992.
[12] J. J. Hopfield and D. W. Tank, "Computing with neural circuits: A model," *Science*, vol. 233, no. 4764, pp. 625–633, 1986.
[13] H. Sompolinsky, "Neural networks with nonlinear synapses and a static noise," *Phys. Rev. A, Gen. Phys.*, vol. 34, no. 3, pp. 2571–2574, 1986.
[14] B. Derrida, E. Gardner, and A. Zippelius, "An exactly solvable asymmetric neural network model," *Europhys. Lett.*, vol. 4, no. 2, pp. 167–173, 1987.
[15] A. Treves and D. J. Amit, "Metastable states in asymmetrically diluted Hopfield networks," *J. Phys. A, Math. Gen.*, vol. 21, no. 14, pp. 3155–3169, 1988.
[16] E. Gardner, "Optimal basins of attraction in randomly sparse neural network models," *J. Phys. A, Math. Gen.*, vol. 22, no. 12, pp. 1969–1974, 1989.
[17] A. Bovier and V. Gayrard, "Rigorous bounds on the storage capacity of the dilute Hopfield model," *J. Stat. Phys.*, vol. 69, nos. 3–4, pp. 597–627, 1992.
[18] J. J. Arenzon and N. Lemke, "Simulating highly diluted neural networks," *J. Phys. A, Math. Gen.*, vol. 27, no. 15, pp. 5161–5165, 1994.
[19] M. Löwe and F. Vermet, "The Hopfield model on a sparse Erdös–Renyi graph," *J. Stat. Phys.*, vol. 143, no. 1, pp. 205–214, 2011.
[20] J. W. Bohland and A. A. Minai, "Efficient associative memory using small-world architecture," *Neurocomputing*, vols. 38–40, pp. 489–496, Jun. 2001.
[21] H. Oshima and T. Odagaki, "Storage capacity and retrieval time of small-world neural networks," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 76, no. 3, 2007, Art. no. 036114.
[22] P. Zheng, W. Tang, and J. Zhang, "A simple method for designing efficient small-world neural networks," *Neural Netw.*, vol. 23, no. 2, pp. 155–159, 2010.
[23] D. Stauffer, A. Aharony, L. da Fontoura Costa, and J. Adler, "Efficient Hopfield pattern recognition on a scale-free neural network," *Eur. Phys. J. B-Condens. Matter Complex Syst.*, vol. 32, no. 3, pp. 395–399, 2003.
[24] B. J. Kim, "Performance of networks of artificial neurons: The role of clustering," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 69, no. 4, 2004, Art. no. 045101.
[25] M. Löwe and F. Vermet, "Capacity of an associative memory model on random graph architectures," *Bernoulli*, vol. 21, no. 3, pp. 1884–1910, 2015.
[26] P. N. McGraw and M. Menzinger, "Topology and computational performance of attractor neural networks," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 68, no. 4, 2003, Art. no. 047102.
[27] J. Lu, J. He, J. Cao, and Z. Gao, "Topology influences performance in the associative memory neural networks," *Phys. Lett. A*, vol. 354, nos. 5–6, pp. 335–343, Jun. 2006.
[28] D. O. Hebb, *The Organization of Behavior: A Neuropsychological Theory*. Hoboken, NJ, USA: Wiley, 1949.
[29] S. Diederich and M. Opper, "Learning of correlated patterns in spin-glass networks by local learning rules," *Phys. Rev. Lett.*, vol. 58, no. 9, pp. 949–952, 1987.
[30] E. Gardner, "The space of interactions in neural network models," *J. Phys. A, Math. Gen.*, vol. 21, no. 1, p. 257, 1988.
[31] N. Davey, S. P. Hunt, and R. G. Adams, "High capacity recurrent associative memories," *Neurocomputing*, vol. 62, pp. 459–491, Dec. 2004.
[32] M. Rubinov and O. Sporns, "Complex network measures of brain connectivity: Uses and interpretations," *NeuroImage*, vol. 52, no. 3, pp. 1059–1069, 2010.
[33] L. R. Varshney, B. L. Chen, E. Paniagua, D. H. Hall, and D. B. Chklovskii, "Structural properties of the *caenorhabditis elegans* neuronal network," *PLoS Comput. Biol.*, vol. 7, no. 2, 2011, Art. no. e1001066.
[34] E. Bullmore and O. Sporns, "The economy of brain network organization," *Nature Rev. Neurosci.*, vol. 13, no. 5, pp. 336–349, May 2012.
[35] A. M. Bruckstein, D. L. Donoho, and M. Elad, "From sparse solutions of systems of equations to sparse modeling of signals and images," *SIAM Rev.*, vol. 51, no. 1, pp. 34–81, 2009.
[36] N. Davey and R. Adams, "High capacity associative memories and connection constraints," *Connection Sci.*, vol. 16, no. 1, pp. 47–65, 2004.
[37] G. Tanaka, T. Yamane, D. Nakano, R. Nakane, and Y. Katayama, "Hopfield-type associative memory with sparse modular networks," in *Proc. Int. Conf. Neural Inf. Process. (ICONIP)*. Kuching, Malaysia: Springer, Nov. 2014, pp. 255–262.
[38] B. K. Natarajan, "Sparse approximate solutions to linear systems," *SIAM J. Comput.*, vol. 24, no. 2, pp. 227–234, 1995.
[39] D. Ge, X. Jiang, and Y. Ye, "A note on the complexity of L_p minimization," *Math. Program.*, vol. 129, no. 2, pp. 285–299, 2011.
[40] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM Rev.*, vol. 43, no. 1, pp. 129–159, 2001.
[41] R. Tibshirani, "Regression shrinkage and selection via the lasso," *J. Roy. Stat. Soc., B (Methodol.)*, vol. 58, no. 1, pp. 267–288, 1996.
[42] C. Pehlevan and A. Sengupta, "Resource-efficient perceptron has sparse synaptic weight distribution," in *Proc. 25th Signal Process. Commun. Appl. Conf. (SIU)*, May 2017, pp. 1–4.
[43] D. B. Chklovskii, T. Schikorski, and C. F. Stevens, "Wiring optimization in cortical circuits," *Neuron*, vol. 34, no. 3, pp. 341–347, Apr. 2002.
[44] B. L. Chen, D. H. Hall, and D. B. Chklovskii, "Wiring optimization can relate neuronal structure and function," *Proc. Nat. Acad. Sci. USA*, vol. 103, no. 12, pp. 4723–4728, 2006.
[45] J. Chapeton, R. Gala, and A. Stepanyants, "Effects of homeostatic constraints on associative memory storage and synaptic connectivity of cortical circuits," *Frontiers Comput. Neurosci.*, vol. 9, p. 74, Jun. 2015.
[46] G. B. Dantzig, *Linear Programming and Extensions*. Princeton, NJ, USA: Princeton Univ. Press, 2016.
[47] MOSEK ApS. (2017). *The MOSEK Optimization Toolbox for MATLAB Manual. Version 8.0.0.52.*, [Online]. Available: http://docs.mosek.com/8.0/toolbox/index.html
[48] *MATLAB, Version 9.0 (R2016a)*, The MathWorks Inc., Natick, MA, USA, 2016.
[49] J. J. Hull, "A database for handwritten text recognition research," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 16, no. 5, pp. 550–554, May 1994.
[50] D. Maltoni, D. Maio, A. Jain, and S. Prabhakar, *Handbook of Fingerprint Recognition*. London, U.K.: Springer, 2009.
[51] G. Tanaka, T. Yamane, D. Nakano, R. Nakane, and Y. Katayama, "Regularity and randomness in modular network structures for neural associative memories," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2015, pp. 1–7.
[52] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, vol. 393, no. 6684, pp. 440–442, Jun. 1998.
[53] G. Zamora-López, C. Zhou, and J. Kurths, "Cortical hubs form a module for multisensory integration on top of the hierarchy of cortical networks," *Frontiers Neuroinform.*, vol. 4, p. 1, Mar. 2010.
[54] A. Hirose, *Complex-Valued Neural Networks*. Berlin, Germany: Springer, 2012.
[55] G. Tanaka and K. Aihara, "Complex-valued multistate associative memory with nonlinear multilevel functions for gray-level image reconstruction," *IEEE Trans. Neural Netw.*, vol. 20, no. 9, pp. 1463–1473, Sep. 2009.
[56] B. Kosko, "Bidirectional associative memories," *IEEE Trans. Syst., Man, Cybern.*, vol. 18, no. 1, pp. 49–60, Jan./Feb. 1988.

Gouhei Tanaka (M'15) received the B.E. degree in mathematical engineering and the M.S. and Ph.D. degrees in complexity science from The University of Tokyo, Tokyo, Japan, in 2000, 2002, and 2005, respectively.

From 2011 to 2013, he was a Project Associate Professor with the Institute of Industrial Science, The University of Tokyo. Since 2013, he has been a Project Associate Professor with the Graduate School of Engineering, The University of Tokyo. His current research interests include mathematical engineering, nonlinear science, network science, and their applications to real-world phenomena from complex systems viewpoint.

Dr. Tanaka is a member of the IEEE Computational Intelligence Society.

Ryosho Nakane (M'13) received the B.S. and M.S. degrees in electronic engineering from Hokkaido University, Sapporo, Japan, in 2000 and 2002, respectively, and the Ph.D. degree in electronics engineering from The University of Tokyo, Tokyo, Japan, in 2005.

Since 2013, he has been a Project Research Associate Professor with The University of Tokyo. His current research interests include semiconductor-based spintronic devices, semiconductor-based electronic devices for next-generation integrated circuits, and neuromorphic electronic devices for energy-efficient systems.

Dr. Nakane is a member of the IEEE Electron Device Society and the Japan Society of Applied Physics.

Tomoya Takeuchi (M'17) received the B.S., M.S., and Ph.D. degrees in mathematical science from The University of Tokyo, Tokyo, Japan, in 2001, 2003, and 2007, respectively.

He has been involved in a variety of projects related to industrial and applied mathematics. His current research interests include inverse problems in industry, numerical partial differential equations, and mathematical modeling and optimization, especially time-series-based forecasting of wind power generation.

Toshiyuki Yamane (M'13) received the B.S., M.S., and Ph.D. degrees in mathematical engineering and information physics from The University of Tokyo, Tokyo, Japan, in 1995, 1997, and 2000, respectively.

In 2000, he joined IBM Research—Tokyo, Kawasaki, Japan. He has been involved in a variety of projects related to mathematical engineering based on statistics and signal processing. His current research interests include error corrections, wireless communications, and design methodology and performance analysis of the conventional and future bioinspired computing systems.

Dr. Yamane is a member of the IEEE Communications Society.

Daiju Nakano received the B.S. and M.S. degrees in physics from The University of Tokyo, Tokyo, Japan, in 1994 and 1996, respectively.

In 1996, he joined IBM Research—Tokyo, Tokyo, where he was involved in liquid crystal display technology, thin-film transistor-array tester development, and wireless communication research. He is currently the Manager of Research at the Tokyo Research Laboratory, IBM Research—Tokyo, Tokyo. His major field is optical physics related to laser spectroscopy. His current research interests include neuromorphic science and engineering, especially designing hardware architecture for neuromorphic computing.

Yasunao Katayama (M'95–SM'11) received the B.S. and M.S. degrees in physics from Tokyo University, Tokyo, Japan, in 1984 and 1986, respectively, and the Ph.D. degree in electrical engineering from Princeton University, Princeton, NJ, USA, in 1994.

He is currently with IBM Research—Tokyo, Tokyo, where he has been involved with a variety of academic disciplines covering physics, information theory, and semiconductor/optical communication research, especially positron physics, fractional quantum Hall effect, quantum devices, numerical analysis, memory technology and systems, logic/dynamic random access memory integration, information theory, wireless and optical communication systems, and more recently on a new computing paradigm including neuromorphic computing.

Akira Hirose (F'13) received the Ph.D. degree in electronic engineering from The University of Tokyo, Tokyo, Japan, in 1991.

He is currently a Professor with the Department of Electrical Engineering and Information Systems, The University of Tokyo. His current research interests include wireless electronics and neural networks.

Dr. Hirose is a Fellow of The Institute of Electronics, Information and Communication Engineers (IEICE) and a member of Japanese Neural Network Society (JNNS). He served as the Founding President for Asia Pacific Neural Network Society in 2016, the President for JNNS from 2013 to 2015, the Vice President for IEICE Electronics Society from 2013 to 2015. He was the Chair of IEICE Neurocomputing Technical Group, the General Chair of the 2013 Asia-Pacific Conference on Synthetic Aperture Radar, Tsukuba, the 2016 International Conference on Neural Information Processing, Kyoto, and the IEEE International Geoscience and Remote Sensing Symposium, Yokohama, to be held in 2019. He serves as the Chair for Complex-Valued Neural Network Task Force in IEEE Computational Intelligence Society Neural Network Technical Committee and IEICE Electromagnetic Theory Technical Group. He was the Editor-in-Chief of IEICE Transactions on Electronics from 2011 to 2012 and an Associate Editor of journals such as the IEEE TRANSACTIONS ON NEURAL NETWORKS from 2009 to 2011 and the IEEE GEOSCIENCE AND REMOTE SENSING NEWSLETTER from 2009 to 2012.