Artif Life Robotics (2016) 21:106–111
DOI 10.1007/s10015-015-0247-4

ORIGINAL ARTICLE

Quaternionic multistate Hopfield neural network with extended projection rule

Toshifumi Minemoto[1] · Teijiro Isokawa[1] · Haruhiko Nishimura[2] · Nobuyuki Matsui[1]

Received: 8 April 2015 / Accepted: 13 November 2015 / Published online: 15 December 2015

Abstract The aim of this paper is to investigate storing and recalling performances of embedded patterns on associative memory. The associative memory is composed of quaternionic multistate Hopfield neural network. The state of a neuron in the network is described by three kinds of discretized phase with fixed amplitude. These phases are set to discrete values with arbitrary divide size. Hebbian rule and projection rule are used for storing patterns to the network. Recalling performance is evaluated through storing random patterns with changing the divide size of the phases in a neuron. Color images are also embedded and their noise tolerance is explored.

Keywords Hopfield neural network · Multistate · Projection rule · Quaternion

1 Introduction

Research on complex-valued and hypercomplex-valued extensions for neural networks has been gaining much interest [1, 2]. Natural treatment for multi-dimensional data on neural networks, such as phase and amplitude in a signal, can be achieved by this kind of extensions. Complex-valued multistate neural network (CVMNN) is a complex-valued Hopfield-type neural network, and the state of a neuron is encoded by a distinct point on a unit circle in the complex plane, i.e., the discretized phase [3]. Multi-level values, such as pixel values of images, can be easily represented using this type of encoding, and thus associative memory that is capable of storing gray-scaled images can be constructed. For CVMNNs there are several studies for the learning schemes that embed patterns to the networks [4, 5].

The use of quaternions, which are four-dimensional hypercomplex numbers, is expected to extend CVMNN so that multi-dimensional vectors with discretized values can be accepted. In this respect, quaternionic multistate neural network (QMNN) has been proposed [6], and several learning schemes have also been formulated [5, 7]. QMNNs enable three-dimensional patterns, such as color images or three-dimensional body images, to be embedded in the network using phasor representation of quaternions. Though these learning schemes for QMNNs are available, the applicability and performances using memory patterns have not yet been investigated.

In this paper, we investigate the performances of QMNNs through storing to and retrieving patterns from the networks. The Hebbian rule and the extended projection rule for QMNN are used for storing memory patterns to the networks. First experiment is to ensure the projection rule actually works, i.e., random patterns can be stably embedded. In the second experiment, the noise robustness of QMNN with the projection rule is also investigated from the viewpoints of discretized level, the number of memory patterns in the network, and noise-affected patterns for the initial configuration of the network. Third experiment is to

This work was presented in part at the 20th International Symposium on Artificial Life and Robotics, Beppu, Oita, January 21–23, 2015.

✉ Toshifumi Minemoto
eu14n001@steng.u-hyogo.ac.jp

[1] Graduate School of Engineering, University of Hyogo, 2167, Shosha, Himeji 671-2280, Japan

[2] Graduate School of Applied Informatics, University of Hyogo, Computational Science Center Building 5-7F, 7-1-28 Minatojima-minamimachi, Chuo-ku, Kobe, Hyogo 650-0047, Japan

demonstrate that color images can be stored to and retrieved from a QMNN using the projection rule. Three images with many pixels and with high resolution of the pixel level are used for the memory patterns, and the robustness of the noises for the input patterns is explored.

This paper is organized as follows. Section 2 gives preliminaries, such as the definitions and representations of quaternion and quaternionic multistate neural network. Then, the learning schemes are described in Sect. 3. This is followed in Sect. 4 by showing the experimental results. We finish with conclusions in Sect. 5.

2 Preliminaries

2.1 Definition of quaternions

Quaternions form a class of hypercomplex numbers that consist of a real number and three kinds of imaginary number, $\boldsymbol{i}$, $\boldsymbol{j}$, $\boldsymbol{k}$. Formally, a quaternion is defined as a vector in a four-dimensional vector space,

$$\boldsymbol{x} = x^{(e)} + x^{(i)}\boldsymbol{i} + x^{(j)}\boldsymbol{j} + x^{(k)}\boldsymbol{k} \tag{1}$$

where $x^{(e)}$, $x^{(i)}$, $x^{(j)}$ and $x^{(k)}$ are real numbers. The multiplication rules between the three imaginary numbers are:

$$\begin{aligned} &\boldsymbol{i}^2 = \boldsymbol{j}^2 = \boldsymbol{k}^2 = \boldsymbol{ijk} = -1, \\ &\boldsymbol{ij} = -\boldsymbol{ji} = \boldsymbol{k},\ \boldsymbol{jk} = -\boldsymbol{kj} = \boldsymbol{i},\ \boldsymbol{ki} = -\boldsymbol{ik} = \boldsymbol{j}. \end{aligned} \tag{2}$$

It is also written using 4-tuple or 2-tuple notations as follows:

$$\boldsymbol{x} = \left(x^{(e)}, x^{(i)}, x^{(j)}, x^{(k)}\right) = \left(x^{(e)}, \vec{x}\right), \tag{3}$$

where $\vec{x} = (x^{(i)},\ x^{(j)},\ x^{(k)})$. In this representation $x^{(e)}$ is the scalar part of x, and $\vec{x}$ forms the vector part.

Now, we define the operation between quaternions, $\boldsymbol{p}$ and $\boldsymbol{q}$. The addition and subtraction of quaternions are defined in the same manner as those of complex numbers or vectors by

$$\boldsymbol{p} \pm \boldsymbol{q} = \left(p^{(e)} \pm q^{(e)},\ \vec{p} \pm \vec{q}\right) \tag{4}$$

$$= \left(p^{(e)} \pm q^{(e)},\ p^{(i)} \pm q^{(i)},\ p^{(j)} \pm q^{(j)}, p^{(k)} \pm q^{(k)}\right). \tag{5}$$

The product of $\boldsymbol{p}$ and $\boldsymbol{q}$, denoted as $\boldsymbol{pq}$, is calculated by Eq. (2) as

$$\boldsymbol{pq} = \left(p^{(e)}q^{(e)} - \vec{p}\cdot\vec{q},\ p^{(e)}\vec{q} + q^{(e)}\vec{p} + \vec{p}\times\vec{q}\right). \tag{6}$$

where $\vec{p}\cdot\vec{q}$ and $\vec{p}\times\vec{q}$ denote the dot and cross products respectively between three-dimensional vectors $\vec{p}$ and $\vec{q}$.

The quaternion conjugate is defined as

$$\boldsymbol{x}^* = \left(x^{(e)}, -\vec{x}\right) = x^{(e)} - x^{(i)}\boldsymbol{i} - x^{(j)}\boldsymbol{j} - x^{(k)}\boldsymbol{k}. \tag{7}$$

The conjugate of product holds the relation of $(\boldsymbol{pq})^* = \boldsymbol{q}^*\boldsymbol{p}^*$.

The quaternion norm of $\boldsymbol{x}$, notation $|\boldsymbol{x}|$, is defined by

$$|\boldsymbol{x}| = \sqrt{\boldsymbol{xx}^*} = \sqrt{x^{(e)2} + x^{(i)2} + x^{(j)2} + x^{(k)2}}. \tag{8}$$

The multiplication between scalar $a = (a, \vec{0})$ and quaternion $\boldsymbol{x}$ is given by

$$a\boldsymbol{x} = \left(ax^{(e)}, a\vec{x}\right) = \left(ax^{(e)}, ax^{(i)}, ax^{(j)}, ax^{(k)}\right). \tag{9}$$

2.2 Phasor representation of quaternions

A complex number in the form $\boldsymbol{c} = c^{(e)} + c^{(i)}\boldsymbol{i}$ can be represented in phasor form by $\boldsymbol{c} = r\mathrm{e}^{\boldsymbol{i}\theta}$, where $r = \sqrt{c^{(e)2} + c^{(i)2}}$, and $\theta = \arctan\left(c^{(i)}/c^{(e)}\right)$. Similarly, every quaternion can be represented in the phasor form. The phasor form of quaternions [8] adopted in this paper is as follows:

$$\boldsymbol{q} = |\boldsymbol{q}|\mathrm{e}^{\boldsymbol{i}\varphi}\mathrm{e}^{\boldsymbol{k}\psi}\mathrm{e}^{\boldsymbol{j}\theta}, \quad \text{where} \begin{cases} \mathrm{e}^{\boldsymbol{i}\varphi} = \cos\varphi + \boldsymbol{i}\sin\varphi \\ \mathrm{e}^{\boldsymbol{j}\theta} = \cos\theta + \boldsymbol{j}\sin\theta \\ \mathrm{e}^{\boldsymbol{k}\psi} = \cos\psi + \boldsymbol{k}\sin\psi \end{cases}. \tag{10}$$

φ, θ, and ψ are the phases of $\boldsymbol{q}$, and they are defined in the ranges of $[-\pi, \pi), [-\pi/2, \pi/2)$, and $[-\pi/4, \pi/4]$, respectively.

3 Quaternionic multistate Hopfield neural network

We describe the quaternionic multistate Hopfield neural network model in this section [6]. All the variables in the model are described by quaternions, i.e., neuron states and connection weights. The state of a neuron p is represented using three phases φ_p, ψ_p, and θ_p as

$$\boldsymbol{u}_p = \mathrm{e}^{\boldsymbol{i}\varphi_p}\mathrm{e}^{\boldsymbol{k}\psi_p}\mathrm{e}^{\boldsymbol{j}\theta_p} \tag{11}$$

where $|\boldsymbol{u}_p| = 1$. The action potential of the neuron p at a discrete time t is defined as

$$\begin{aligned} \boldsymbol{h}_p(t) &= \sum_q \boldsymbol{w}_{pq}\boldsymbol{u}_q(t) \\ &= \sum_q \boldsymbol{w}_{pq}\mathrm{e}^{\boldsymbol{i}\varphi_q(t)}\mathrm{e}^{\boldsymbol{k}\psi_q(t)}\mathrm{e}^{\boldsymbol{j}\theta_q(t)} \end{aligned} \tag{12}$$

where $\boldsymbol{w}_{pq}$ is the connection weights between neuron q to neuron p. The output state of the neuron p at the time $(t+1)$ is determined by

$$\boldsymbol{u}_p(t+1) = \mathrm{qsign}\left(\boldsymbol{h}_p(t)\right), \tag{13}$$

where

$$\text{qsign}(\boldsymbol{h}) = \text{qsign}\left(e^{\boldsymbol{i}\varphi}e^{\boldsymbol{k}\psi}e^{\boldsymbol{j}\theta}\right) \tag{14}$$

$$= \text{csign}_A\left(e^{\boldsymbol{i}\varphi}\right)\text{csign}_B\left(e^{\boldsymbol{k}\psi}\right)\text{csign}_C\left(e^{\boldsymbol{j}\theta}\right). \tag{15}$$

The update is conducted for each of phases in $\boldsymbol{h}$, i.e., φ, ψ, θ. The function csign_A is used for updating φ, and it is defined as

$$\text{csign}_A\left(e^{\boldsymbol{i}\varphi}\right) = \begin{cases} \boldsymbol{q}_0^{(\varphi)} & \text{for } -\pi \le \arg\left(e^{\boldsymbol{i}\varphi}\right) < -\pi + \varphi_0 \\ \boldsymbol{q}_1^{(\varphi)} & \text{for } -\pi + \varphi_0 \le \arg\left(e^{\boldsymbol{i}\varphi}\right) < -\pi + 2\varphi_0 \\ \vdots & \\ \boldsymbol{q}_{A-1}^{(\varphi)} & \text{for } -\pi + (A-1)\varphi_0 \le \arg\left(e^{\boldsymbol{i}\varphi}\right) < -\pi + A\varphi_0 \end{cases}, \tag{16}$$

where A is the phase resolution for φ, and φ_0 defines a quantization unit which is represented by $\varphi_0 = 2\pi/A$. $\boldsymbol{q}_a^{(\varphi)}$ is a distinct point on a unit circle which is defined as $\boldsymbol{q}_a^{(\varphi)} = \exp(\boldsymbol{i}(-\pi + a\varphi_0 + \frac{\varphi_0}{2}))$. Therefore, the function csign_A outputs the closest quaternion in $\{\boldsymbol{q}_0^{(\varphi)}, \ldots, \boldsymbol{q}_{A-1}^{(\varphi)}\}$ corresponding to the input. Similarly, the function csign_B for updating ψ and the function csign_C for updating θ are defined as follows:

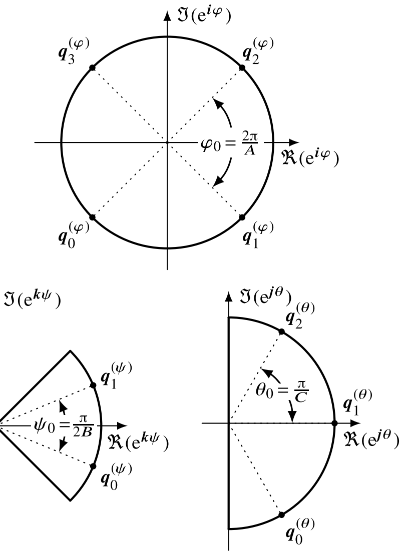
$$\text{csign}_B\left(e^{\boldsymbol{k}\psi}\right) = \begin{cases} \boldsymbol{q}_0^{(\psi)} & \text{for } -\frac{\pi}{4} \le \arg\left(e^{\boldsymbol{k}\psi}\right) < -\frac{\pi}{4} + \psi_0 \\ \boldsymbol{q}_1^{(\psi)} & \text{for } -\frac{\pi}{4} + \psi_0 \le \arg\left(e^{\boldsymbol{k}\psi}\right) < -\frac{\pi}{4} + 2\psi_0 \\ \vdots & \\ \boldsymbol{q}_{B-1}^{(\psi)} & \text{for } -\frac{\pi}{4} + (B-1)\psi_0 \le \arg\left(e^{\boldsymbol{k}\psi}\right) < -\frac{\pi}{4} + B\psi_0 \end{cases}, \tag{17}$$

$$\text{csign}_C\left(e^{\boldsymbol{j}\theta}\right) = \begin{cases} \boldsymbol{q}_0^{(\theta)} & \text{for } -\frac{\pi}{2} \le \arg\left(e^{\boldsymbol{j}\theta}\right) < -\frac{\pi}{2} + \theta_0 \\ \boldsymbol{q}_1^{(\theta)} & \text{for } -\frac{\pi}{2} + \theta_0 \le \arg\left(e^{\boldsymbol{j}\theta}\right) < -\frac{\pi}{2} + 2\theta_0 \\ \vdots & \\ \boldsymbol{q}_{C-1}^{(\theta)} & \text{for } -\frac{\pi}{2} + (C-1)\theta_0 \le \arg\left(e^{\boldsymbol{j}\theta}\right) < -\frac{\pi}{2} + C\theta_0 \end{cases}, \tag{18}$$

where B and C are the phase resolutions for φ and θ, respectively. The quantization units ψ_0 and θ_0 are defined by $\psi_0 = \pi/2B$ and $\theta_0 = \pi/C$, respectively. $\boldsymbol{q}_b^{(\psi)}$ and $\boldsymbol{q}_c^{(\theta)}$ are also defined as follows: $\boldsymbol{q}_b^{(\psi)} = \exp(\boldsymbol{j}(-\frac{\pi}{4} + b\psi_0 + \frac{\psi_0}{2}))$, $\boldsymbol{q}_c^{(\theta)} = \exp(\boldsymbol{k}(-\frac{\pi}{2} + c\theta_0 + \frac{\theta_0}{2}))$.

From the above equations, the quaternionic neuron takes a total of $A \times B \times C$ states. An example of the quantized output points of the quaternionic neuron is shown in Fig. 1, where $A = 4, B = 2$, and $C = 3$.

The energy function of the network with N quaternionic neurons is given as:



$\Im(e^{\boldsymbol{i}\varphi})$
$\Re(e^{\boldsymbol{i}\varphi})$
$\boldsymbol{q}_3^{(\varphi)}$
$\boldsymbol{q}_2^{(\varphi)}$
$\boldsymbol{q}_0^{(\varphi)}$
$\boldsymbol{q}_1^{(\varphi)}$
$\varphi_0 = \frac{2\pi}{A}$
$\Im(e^{\boldsymbol{k}\psi})$
$\Re(e^{\boldsymbol{k}\psi})$
$\boldsymbol{q}_1^{(\psi)}$
$\boldsymbol{q}_0^{(\psi)}$
$\psi_0 = \frac{\pi}{2B}$
$\Im(e^{\boldsymbol{j}\theta})$
$\Re(e^{\boldsymbol{j}\theta})$
$\boldsymbol{q}_2^{(\theta)}$
$\boldsymbol{q}_1^{(\theta)}$
$\boldsymbol{q}_0^{(\theta)}$
$\theta_0 = \frac{\pi}{C}$

Fig. 1 An example of quantized output points in the quaternionic multistate neuron ($A = 4, B = 2, C = 3$)

$$\boldsymbol{E}(t) = -\frac{1}{2}\sum_{p=1}^{N}\sum_{q=1}^{N} \boldsymbol{u}_p^*(t)\boldsymbol{w}_{pq}\boldsymbol{u}_q(t). \tag{19}$$

$\boldsymbol{E}$ takes real value ($\boldsymbol{E} = \boldsymbol{E}^*$) when the connection weights $\boldsymbol{w}_{pq}$ satisfy the following conditions:

$$\boldsymbol{w}_{pq} = \boldsymbol{w}_{qp}^*,\ \boldsymbol{w}_{pp} = \boldsymbol{w}_{pp}^* = (\boldsymbol{w}^{(e)}, \boldsymbol{0}),\ \boldsymbol{w}^{(e)} \ge 0 \tag{20}$$

The function monotonically decreases under the conditions $|\Delta\varphi| < \varphi_0$, $|\Delta\psi| < \psi_0$, and $|\Delta\theta| < \theta_0$. In these conditions, $\Delta\varphi$, $\Delta\psi$, and $\Delta\theta$ are a phase difference between the state at time $(t+1)$ and the action potential at time t for the neuron undergoing its update [6].

4 Learning schemes

4.1 Hebbian rule

Let $(\xi_{\varphi,1}^{\mu}, \ldots, \xi_{\varphi,N}^{\mu})$, $(\xi_{\psi,1}^{\mu}, \ldots, \xi_{\psi,N}^{\mu})$, and $(\xi_{\theta,1}^{\mu}, \ldots, \xi_{\theta,N}^{\mu})$ be multistate memory patterns. Here, $\xi_{\varphi}^{\mu} \in \{0, \ldots, A-1\}$, $\xi_{\psi}^{\mu} \in \{0, \ldots, B-1\}$, $\xi_{\theta}^{\mu} \in \{0, \ldots, C-1\}$, and μ denotes pattern index. These three patterns can be represented as a quaternionic pattern $(\boldsymbol{\epsilon}_1^{\mu}, \ldots, \boldsymbol{\epsilon}_N^{\mu})$, where $\boldsymbol{\epsilon}^{\mu} = e^{\boldsymbol{i}(\xi_{\varphi}^{\mu}\varphi_0 + \frac{\varphi_0}{2})} e^{\boldsymbol{k}(\xi_{\psi}^{\mu}\psi_0 + \frac{\psi_0}{2})} e^{\boldsymbol{j}(\xi_{\theta}^{\mu}\theta_0 + \frac{\theta_0}{2})}$.

A straightforward way to embed patterns into the associative memory is the use of Hebbian rule. The Hebbian rule is represented as

$$\boldsymbol{w}_{pq} = \sum_{\mu=1}^{P} \boldsymbol{\epsilon}_p^{\mu}\boldsymbol{\epsilon}_q^{\mu*}, \tag{21}$$

where P is the number of embedded patterns, and $\boldsymbol{\epsilon}_p^{\mu}$ denotes the state of the neuron p about μ-th memory pattern.

The connection weights calculated by Eq. (21) satisfies the condition Eq. (20), thus the network with these connections is stable. In this scheme, the memory patterns can be stable points in the network when the memory patterns are orthogonal to each other [9].

4.2 Projection rule

Projection rule [10–12] is a learning scheme that can embed non-orthogonal (correlated) memory patterns in a network. A key idea of the projection rule is that non-orthogonal patterns are first projected onto orthogonal ones, and then the Hebbian rule is applied to these projected patterns.

Quaternionic extended projection rule [7] is given as follows:

$$\tilde{\boldsymbol{w}}_{pq} = \sum_{\nu,\mu}^{P} \boldsymbol{\epsilon}_p^{\mu} \left(\boldsymbol{Q}^{-1}\right)_{\mu\nu} \boldsymbol{\epsilon}_q^{\nu *}, \quad \boldsymbol{Q}_{\mu\nu} = \sum_{p=1}^{N} \boldsymbol{\epsilon}_p^{\mu *} \boldsymbol{\epsilon}_p^{\nu}, \tag{22}$$

where $\boldsymbol{Q}^{-1}$ denotes the pseudo inverse matrix of $\boldsymbol{Q}$. By the use of orthogonalization of correlated patterns, the number of patterns that can be stored in the network equals to the number of neurons in the network.

5 Simulations

5.1 Stability of stored patterns

We first evaluate the stability of the embedded memory patterns which are stored into QMNNs using the Hebbian rule and the projection rule. In this experiment, the patterns with randomly generated values are used as memory patterns. The size of the patterns, which is the number of neurons in the network, is set to 100, and the phase resolutions are set as follows: $(A, B, C) = (4, 1, 2)$, (8, 2, 4), (16, 4, 8), (32, 8, 16), (64, 16, 32). In these conditions, the quantization units φ_0, ψ_0, and θ_0 are the same size. The number of the memory patterns, denoted by P, varies such that $P = 1, 2, \ldots, 100$.

The stability of the patterns is investigated by the following procedure. First, for given A, B, C, and P, memory patterns are generated and stored into the network. Next, each of the memory patterns is set to the network as its initial states, then the states for all neurons are updated. If the network state does not change, this stored pattern can be regarded as stable.

Figure 2 shows P dependency of the retrieval success rates against various phase resolutions. The retrieval success rates are calculated from 1000 trials. From Fig. 2, we find that the memory patterns are hardly embedded to the network using the Hebbian rule. The memory patterns tend to be more unstable with increases of the phase resolutions and the number of stored patterns. If the phase resolutions (A, B, C) are set to (32, 8, 16) or (64, 16, 32), only one pattern is stable in the network as shown in Fig. 2. In other words, two or more memory patterns cannot to be stable in the network using the Hebbian rule for large phase resolutions. On the other hand, the projection rule can store up to 99 patterns in the network regardless of the phase resolutions as shown in Fig. 3. In this figure, the retrieval success rate is 0 % in the case of $P = 100$. It is due to that the self-connection weights, $\tilde{\boldsymbol{w}}_{pp}$s, are set to 0. If these weights are set to positive values, the retrieval success rate becomes 100 %. From these results, all the stored patterns are local minima in the network using the projection rule. Therefore, the memory patterns stored using the projection rule have higher stability than those using the Hebbian rule in QMNN.

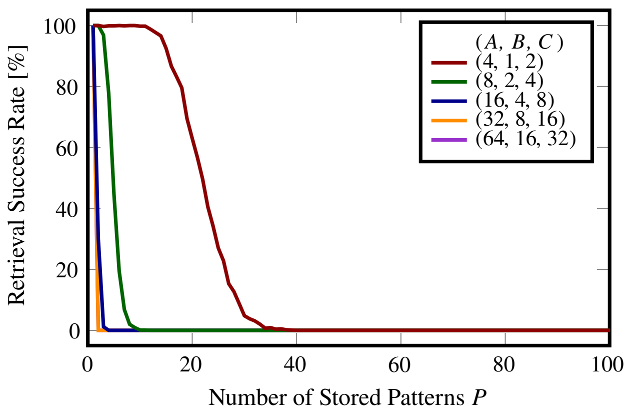


Fig. 2 Stability of the memory patterns stored by Hebbian rule

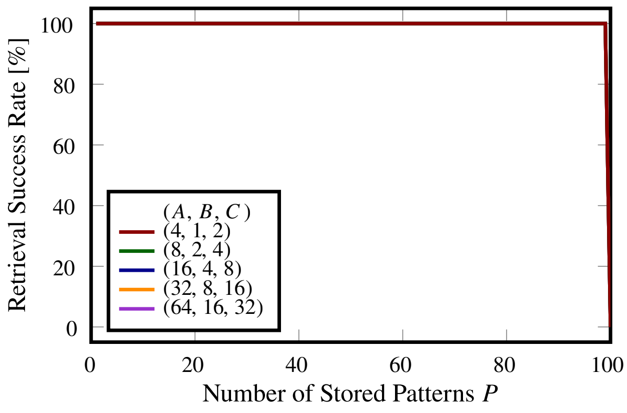


Fig. 3 Stability of the memory patterns stored by projection rule

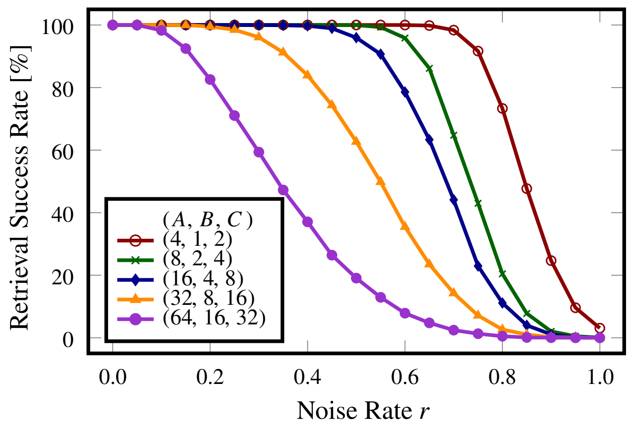


Fig. 4 Retrieval success rate of the stored patterns against the noise rate ($P = 10$)

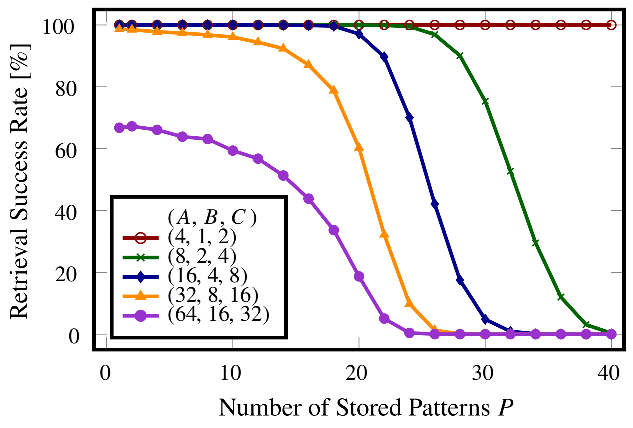


Fig. 5 Retrieval success rate of the stored patterns against the number of stored patterns ($r = 0.3$)

5.2 Noise robustness of projection rule

We have shown the stabilities of the stored patterns by retrieving stored patterns from the network. In this section, we show the retrieval performance by retrieving patterns from noisy patterns, i.e., the patterns modified from the stored patterns. This evaluates the basins of attractors (stored patterns) produced by the projection rule. We set the experimental conditions for the number of neurons and phase resolutions with the same ones in the previous section.

Figures 4 and 5 show the retrieval success rates when noisy patterns are used as initial configurations of the network. A noisy input pattern is generated from one of the stored patterns with each pixel value being changed with probability r (noise rate). In the case of Fig. 4, the number of stored patterns is set to $P = 10$ and the noise rate of the input pattern varies from 0.1 to 1.0. In the case of Fig. 5, the noise rate is fixed to 0.3 and the number of stored patterns is changed from 1 to 40. From these results, we find that the retrieval success rate is decreased with increasing the number of stored patterns. The success rate is also decreased with increasing the phase resolutions.

The deterioration of noise robustness is caused by spurious patterns, i.e., mixture patterns, inverted patterns and rotated patterns of memory patterns. These spurious patterns increase with increasing the number of stored patterns and the phase resolutions. Therefore, the noise robustness of QMNN depends on the number of stored patterns and phase resolutions.

5.3 Image retrieval task

We have explored the performances of QMNN with the projection rule and Hebbian rule in the previous sections, and found that the projection rule could successfully work for storing random patterns. In this section, we investigate the performances of QMNNs by storing and retrieving natural images that have more intensity resolutions in each pixel from the viewpoint of practical applications such as color image database.

Figure 6 shows three types of memory patterns to be embedded to the network. These images consist of $64 \times 64 = 4096$ pixels and each pixel value is represented by 8 bits (256 levels) of three channels. The three channels, which are red, green, and blue, are assigned to φ, θ, and ψ of a quaternion in phasor representation. Thus, the phase resolutions are set to $A = B = C = 256$. The number of neurons in the network is the same as the number of pixels of the images, i.e., $N = 4096$. These memory patterns are embedded to the networks with the Hebbian rule and the projection rule.

Figure 7 shows input images and their corresponding output images from the networks. The images on the top rows are the input images, and the pixel values on these images are affected by noises with probability r. The images on the second and third rows are the output images from the networks with patterns being embedded by the Hebbian rule and the projection rule, respectively. From the output images, the network with the Hebbian rule cannot retrieve the stored pattern, even if the original memory pattern is input to the network. Thus the three color images are not stable in the network using the Hebbian rule. On the other hand, the network with the projection rule successfully retrieves stored patterns from the noisy images. Therefore, for a large-scaled network with high resolution of phases, the patterns can be stored by the projection rule.

This experimental result shows that QMNN can store natural images each of which has some correlation to each other and it can retrieve the stored images from noisy input. In this case, loading rate, which is a ratio of the number of stored patterns (P) with respect to the number of neurons (N) in the network, is low ($P = 3/N = 4096$). When the loading rate for the network gets higher, the retrieval of

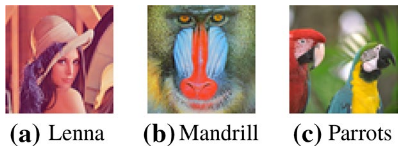


Fig. 6 Memory patterns for the image retrieval experiment (64 × 64 pixels, 24 bit color)

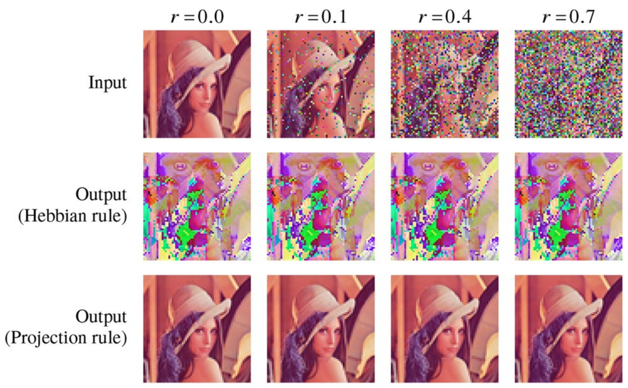


Fig. 7 Image retrieval results (Lenna)

images tends to be failed, as shown in Figs. 4 and 5. This is due to the spurious patterns in the QMNN caused by the memory patterns.

6 Conclusion

In this paper, the stabilities of embedded patterns are investigated in the quaternionic associative memories. The associative memory is based on quaternionic multistate Hopfield neural network. The Hebbian rule and the projection rule are used for embedding memory patterns.

The stability of stored memory patterns has been explored with randomly generated patterns with changing the phase resolutions. From the experimental results, the Hebbian rule hardly stores the memory patterns with increasing the phase resolutions. In contrast, the projection rule can stabilize all the memory patterns into the network regardless of phase resolutions. The noise robustness of the retrieval patterns is also investigated and it is found that the performance from noisy input depends on the phase resolutions of the quaternionic neuron states and the number of stored memory patterns. The practical experimental results show that the color images can be embedded to the network by utilizing the projection rule. Three types of color images are embedded (by the projection rule) and they are retrieved from noisy input images successfully.

To obtain better performances in embedding patterns, we will explore encoding schemes for representing pixel values of images for practical applications. The experimental results show that the QMNNs with the projection rule can store correlated patterns such as natural images successfully, but also show that the retrieval performances become lower when the number of patterns is higher or the resolutions in the neuron are higher. This is due to the spurious patterns by the memory patterns. This problem also lies in CVMNNs, but another type of CVMNN, which has a combination of real-valued and complex-valued neurons in a network, could overcome this problem [13]. It is possible to extend this CVMNN to QMNN for improving the noise robustness. This remains for our future work.

References

1. Hirose A (2003) Complex-valued neural networks: theories and applications. World Scientific, Singapore
2. Nitta T (2009) Complex-valued neural networks: utilizing high-dimensional parameters. Information Science Reference (IGI Global), New York
3. Jankowski S, Lozowski A, Zurada J (1996) Complex-valued multistate neural associative memory. Neural Netw IEEE Trans 7(6):1491–1496
4. Muezzinoglu M, Guzelis C, Zurada J (2003) A new design method for the complex-valued multistate hopfield associative memory. Neural Netw IEEE Trans 14(4):891–899
5. Isokawa T, Nishimura H, Matsui N (2009) An iterative learning scheme for multistate complex-valued and quaternionic hopfield neural networks. In: Proceedings of International Joint Conference on Neural Networks (IJCNN2009), pp 1365–1371
6. Isokawa T, Nishimura H, Saitoh A, Kamiura N, Matsui N (2008) On the scheme of quaternionic multistate hopfield neural network. In: Proceedings of SCIS and ISIS, Japan Society for Fuzzy Theory and Intelligent Informatics, pp 809–813
7. Isokawa T, Nishimura H, Matsui N (2013) Quaternionic neural networks for associative memories. In: Hirose A (ed) Complex-valued neural networks: advances and applications. Springer, New York, pp 103–131
8. Bülow T, Sommer G (2001) Hypercomplex signals-a novel extension of the analytic signal to the multidimensional case. Signal Process IEEE Trans 49(11):2844–2852
9. Isokawa T, Nishimura H, Kamiura N, Matsui N (2008) Associative memory in quaternionic hopfield neural network. Int J Neural Syst 18(2):135–145
10. Personnaz L, Guyon I, Dreyfus G (1986) Collective computational properties of neural networks: New learning mechanisms. Phys Rev A 34(5):4217
11. Kohonen T (1988) Self-organization and associative memory. Springer, Berlin
12. Lee DL (2006) Improvements of complex-valued hopfield associative memory by using generalized projection rules. Neural Netw IEEE Trans 17(5):1341–1347
13. Suzuki Y, Kobayashi M (2014) Complex-valued bipartite auto-associative memory. IEICE Trans Fundam Electron Commun Comput Sci 97(8):1680–1687