Multi-Valued Autoencoders and Classification of Large-Scale Multi-Class Problem

Ryusuke Hata
Graduate School of Engineering
University of Fukui
Fukui 910-8507, Japan

M. A. H. Akhand
Dept. of Computer Science and Engineering
Khulna University of Engineering & Technology

Kazuyuki Murase
Graduate School of Engineering
University of Fukui
Fukui 910-8507, Japan

***Abstract*—Two-layered neural networks are well known as autoencoders (AEs) in order to reduce the dimensionality of data. AEs are successfully employed as pre-trained layers of neural networks for classification tasks. Most of the existing studies conceived real-valued AEs in real-valued neural networks. This study investigated complex- and quaternion-valued AEs for complex- and quaternion-valued neural networks. Inputs, weights, biases, and outputs in complex-valued AE (CAE) are complex variables, whereas those in quaternion-valued AE (QAE) are quaternions. In both methods, a split-type activation function is used in the hidden and output units. To deal with the images using the proposed methods, pairs of pixels are allotted to complex-valued inputs in the CAE and quartets of pixels are allotted to quaternion-valued inputs in the QAE. Proposed autoencoders are tested and performance compared with conventional AE for several tasks which are encoding/decoding, handwritten numeral recognition and large-scale multi-class classification. Proposed CAE and QAE revealed as good recognition methods for the tasks and outperformed conventional AE with significance performance in case of large-scale multi-class images recognition.**

***Keywords*—Autoencoder; classification; complex-valued autoencoder; quaternion-valued autoencoder; recognition**

I. Introduction

Autoencoding refers the automatic learning of encoding and decoding functions from examples without engineered by an expert or a human. A two-layered neural network is well known as an autoencoder (AE) in order to reduce the dimensionality of data. Recent studies proposed many types of AEs [1]-[6] which are composed of input, hidden, and output units, and are based on the gradient descent method. AEs generally deal with image data. If a network is trained with image data, some features of the input image appear in the learned weights. These parameters can be used as the initial parameters to train neural networks for classification tasks. Most of the existing studies conceived real-valued AEs in real-valued neural networks [1]-[6].

Artificial neural networks involve in a large number of applications with significant varieties and recent multi-valued version is found efficient for higher-dimension data. Nowadays, real-world data contain higher-dimensional information; examples include image, medical, and web data. In conventional real-valued neural networks (RNNs), a multi-dimension values are often treated by using multiple real-valued neurons. The use of these multi-valued quantities is now spreading to artificial neural networks in the form of complex-valued neural networks (CVNNs) and quaternion neural networks (QNNs).

Complex and quaternion numbers are widely used in various areas of engineering. Complex numbers are used to deal with two-dimensional vectors and wave information, whereas quaternions are used for three-dimensional graphics and computer vision. The gradient descent method to tune complex-valued weights in CVNNs [7] and quaternion-valued weights in QNNs [8] made efficient to tackle such high dimensional problems efficiently. With the advent of CVNNs and QNNs, multi-valued data can now be used as complex and quaternion signals. The convergence of CVNNs and QNNs is found better than that of RVNNs to solve such higher dimension problems. The study of CVNNs has been developing widely in various areas [9]-[20]. Applications of CVNNs include those in radar image processing [17], real-time image recognition [19], and traffic and power systems [20]. There have also been active studies of QNNs [21]-[24] in, for example, color image compression [21] and color night vision [22].

This study proposed two multi-valued autoencoders extending conventional AE which are complex-valued AE (CAE) and quaternion-valued AE (QAE). The CAE is a complex-valued neural network with input, hidden, and output units; its learning is based on the complex gradient descent method. The QAE is a quaternion neural network with input, hidden, and output units; its learning is based on the quaternion gradient descent method. The signal flows in the networks are almost the same as those of the AE. In order to simplify the network calculations, easy-to-use split-type activation functions are considered in the hidden and output units of the CAE and QAE. Proficiency of the proposed AEs are identified comparing with the conventional AE for encoding/decoding and classification of image objects.

Although CAE and QAE have been outlined in our previous study [25], the present study is extended and complete presentation in both theoretical analysis and experimental results. In this study, proposed methods are tested for two different activation functions (sigmoid and

This work was supported by the Grants-in-Aid from JSPS; Nos. 15K00333 for KM and 16J11219 for RH. The funding source had no role in study design; in the collection, analysis and interpretation of data; in the writing of the report; and in the decision to submit the article for publication.

rectified linear unit). Recognition of handwritten numerals and large-scale multi-class objects is the main significance of the present study. In another study, complex-valued are investigated for linear autoencoders [26]; the algorithm is different from those of our methods. The autoencoders have considered in this study are nonlinear in category based on neural network with nonlinear activation function and have focused on the classification task.

The remainder of this paper is structured as follows. Conventional AE and proposed multi-valued autoencoders (i.e., CAE and QAE) are explained in Section II. This section also demonstrates autoencoder based classification. In Section III, performance of proposed autoencoders are investigated for several tasks which are encoding/decoding, handwritten numeral recognition and a large-scale multi-class classification. Finally, the study is concluded in Section IV with future research directions.

II. Multi-Valued Autoencoders and Classification with Those

This section first explains conventional autoencoder (AE) with a sample architecture for better understanding of proposed multi-valued autoencoders. It then presents proposed complex-valued autoencoder (CAE) and quaternion-valued autoencoder (QAE) extending conventional AE. Finally, classification based on autoencoders is demonstrated.

A. Conventional Autoencoder(Ae)

An AE is a two-layered neural network that is based on the gradient descent method. In an AE, the number of outputs is the same as the number of inputs, and common weights are used in the first and second layers (weight sharing). To describe the network architecture clearly, consider a network with four-input, three-hidden, four-output units as shown in Fig. 1.

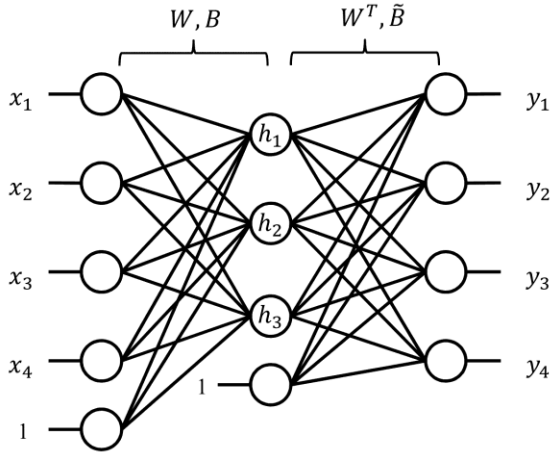


Fig. 1. Network structure of an autoencoder with four-input, three-hidden and four-output units. In conventional case the inputs and outputs are real number.

Here, the input vector is $X = [x_1 \cdots x_4]$, and the bias vectors in the first and second layers are $B = [b_1 \cdots b_3]$ and $\tilde{B} = [\tilde{b}_1 \cdots \tilde{b}_4]$, respectively. The weight matrix W is represented as:

$$W = \begin{pmatrix} w_{11} & \cdots & w_{14} \\ \vdots & \ddots & \vdots \\ w_{31} & \cdots & w_{34} \end{pmatrix}. \quad (1)$$

The hidden-unit output vector $H = [h_1 \cdots h_3]$ is obtained as

$$H = f(WX + B), \quad (2)$$

where $f(x)$ is an activation function such as the sigmoid or rectified linear unit (ReLU) function. The output vector $Y = [y_1 \cdots y_4]$ is computed as

$$Y = f\left(W^T H + \tilde{B}\right). \quad (3)$$

Here, W^T is the transpose of W. When training data are given to this network, the weights and biases are tuned by back propagation to minimize the error between inputs and outputs. The squared error given by (4) is applied as the error function:

$$E = \|X - Y\|^2. \quad (4)$$

The tuning equations of the network parameters b_q, $\tilde{b}_p$, and w_{qp} are as follows:

$$\Delta\tilde{b}_p = -\alpha \frac{\partial E}{\partial \tilde{b}_p} = 2\alpha(x_p - y_p) f'(z_p) \quad (5)$$

$$\Delta b_q = -\alpha \frac{\partial E}{\partial b_q} = \sum_{p=1}^{4} \{\Delta\tilde{b}_q w_{qp}\} f'(c_q) \quad (6)$$

$$\Delta w_{qp} = -\alpha \frac{\partial E}{\partial w_{qp}} = \Delta\tilde{b}_p h_q, \quad (7)$$

Where, α is the learning rate and $z_p = \sum_{q=1}^{3} w_{qp} h_q + \tilde{b}_p$ and $c_q = \sum_{p=1}^{4} w_{qp} x_p + b_q$ are the net inputs to the p^{th} output and q^{th} hidden unit, respectively. The learning process is performed by giving initial values to the parameters and iterating (5)–(7).

Autoencoders can generate some features in learned parameters by training with data. For example, if an AE is trained on a dataset of cat images, features such as silhouettes, eyes, and ears appear in the learned weights. Furthermore, AEs can be employed for pre-training weights of different layers of deep neural networks and hence perform classification tasks (e.g., image classification). By stacking AEs, deep neural networks are shown better convergence than in the case without pre-training of AEs [2].

B. Complex-Valued Autoencoder(CAE)

Proposed CAE is an extension of conventional AE to the complex domain with complex-valued neurons. To consider network structure of Fig. 1 for CAE, inputs, weights, biases, and outputs are all complex valued. CAE operation steps are similar to AE but perform in complex domain. Input signals are given to the network through the input units; then, the weighted sum of the inputs is given to some activation function in each of the hidden units. Finally, in the output units, the weighted sums of the hidden outputs are passed through some activation function.

A complex value contains a real and an imaginary parts and CAE learning algorithm is based on the complex-valued gradient descent method. For network structure with Fig. 1,

the input vector is $X = [x_1 \cdots x_4]$, the bias vector in the first layer $B = [b_1 \cdots b_3]$, the bias vector in the second layers $\tilde{B} = [\tilde{b}_1 \cdots \tilde{b}_4]$ and the weight matrix W (represented by the same form as (1)) are all complex-valued numbers in CAE. To describe the real and imaginary parts of the parameters, $x_p = x_p^R + \mathrm{i}x_p^I$, $w_{qp} = w_{qp}^R + \mathrm{i}w_{qp}^I$, $b_q = b_q^R + \mathrm{i}b_q^I$, and $\tilde{b}_p = \tilde{b}_p^R + \mathrm{i}\tilde{b}_p^I$ ($\mathrm{i}^2 = -1$). The net input $c_q = c_q^R + \mathrm{i}c_q^I$ and output $h_q = h_q^R + \mathrm{i}h_q^I$ of the q^{th} hidden unit are calculated as

$$c_q = \sum_{p=1}^{4} w_{qp}x_p + b_q$$

$$= \left\{\sum_{p=1}^{4}\left(w_{qp}^R x_p^R - w_{qp}^I x_p^I\right) + b_q^R\right\}$$

$$+\mathrm{i}\left\{\sum_{p=1}^{4}\left(w_{qp}^R x_p^I + w_{qp}^I x_p^R\right) + b_q^I\right\} \quad (8)$$

$$h_q = f\left(c_q^R\right) + \mathrm{i}f\left(c_q^I\right) \quad (9)$$

Here, the hidden output is generated by a split-type activation function [27]. The net input $z_p = z_p^R + \mathrm{i}z_p^I$ and output $y_p = y_p^R + \mathrm{i}y_p^I$ of the p^{th} output unit are calculated similarly as

$$z_p = \sum_{q=1}^{3} w_{qp}h_q + \tilde{b}_p \quad (10)$$

$$y_p = f\left(z_p^R\right) + \mathrm{i}f\left(z_p^I\right) \quad (11)$$

The error function to be minimized is the same formula as (4). For the training, we update the weights and biases by using (12)–(14):

$$\Delta\tilde{b}_p = -\beta\frac{\partial E}{\partial \tilde{b}_p^R} - \mathrm{i}\beta\frac{\partial E}{\partial \tilde{b}_p^I} = \Delta\tilde{b}_p^R + \mathrm{i}\Delta\tilde{b}_p^I \quad (12)$$

$$\Delta b_q = -\beta\frac{\partial E}{\partial b_q^R} - \mathrm{i}\beta\frac{\partial E}{\partial b_q^I} = \Delta b_q^R + \mathrm{i}\Delta b_q^I \quad (13)$$

$$\Delta w_{qp} = -\beta\frac{\partial E}{\partial w_{qp}^R} - \mathrm{i}\beta\frac{\partial E}{\partial w_{qp}^I} = \Delta w_{qp}^R + \mathrm{i}\Delta w_{qp}^I$$

$$= \Delta b_q \bar{h}_q, \quad (14)$$

Where, β is the learning rate and $\bar{h}_q$ is the complex conjugate of h_q. The following equations are for the partial derivatives within (12)–(14):

$$\frac{\partial E}{\partial \tilde{b}_p^R} = -2\left(x_p^R - y_p^R\right)f'\left(z_p^R\right) \quad (15)$$

$$\frac{\partial E}{\partial \tilde{b}_p^I} = -2\left(x_p^I - y_p^I\right)f'\left(z_p^I\right) \quad (16)$$

$$\frac{\partial E}{\partial b_q^R} = \left(\sum_{p=1}^{4}\Delta\tilde{b}_p^R w_{qp}^R + \sum_{p=1}^{4}\Delta\tilde{b}_p^I w_{qp}^I\right)f'\left(c_q^R\right) \quad (17)$$

$$\frac{\partial E}{\partial b_q^I} = \left(\sum_{p=1}^{4}\Delta\tilde{b}_p^I w_{qp}^R - \sum_{p=1}^{4}\Delta\tilde{b}_p^R w_{qp}^I\right)f'\left(c_q^I\right) \quad (18)$$

The learning process is performed by giving initial values to the parameters and iterating (12)–(14).

C. Quaternion-Valued Autoencoder(QAE)

Proposed QAE is an extension of conventional AE to the quaternion domain with quaternion-valued neurons. To consider network structure of Fig. 1 for QAE, inputs, weights, biases, and outputs are all quaternion valued. The signal flow in a QAE network is the same as that in an AE or CAE but perform in quaternion domain.

A quaternion value contains one real and three imaginary parts and QAE learning algorithm is based quaternion-valued gradient descent method [21]. For network structure with Fig. 1, the input vector is $X = [x_1 \cdots x_4]$, the bias vector in the first layer $B = [b_1 \cdots b_3]$, the bias vector in the second layers $\tilde{B} = [\tilde{b}_1 \cdots \tilde{b}_4]$ and the weight matrix W (represented by the same form as (1)) are all quaternion-valued numbers in QAE. To describe the real and imaginary parts of the parameters $x_p = x_p^R + \mathrm{i}x_p^I + \mathrm{j}x_p^J + \mathrm{k}x_p^K$, $w_{qp} = w_{qp}^R + \mathrm{i}w_{qp}^I + \mathrm{j}w_{qp}^J + \mathrm{k}w_{qp}^K$, $b_q = b_q^R + \mathrm{i}b_q^I + \mathrm{j}b_q^J + \mathrm{k}b_q^K$, and $\tilde{b}_p = \tilde{b}_p^R + \mathrm{i}\tilde{b}_p^I + \mathrm{j}\tilde{b}_p^J + \mathrm{k}\tilde{b}_p^K$ ($\mathrm{i}^2 = \mathrm{j}^2 = \mathrm{k}^2 = \mathrm{ijk} = -1$). The net input $c_q = c_q^R + \mathrm{i}c_q^I + \mathrm{j}c_q^J + \mathrm{k}c_q^K$ and output $h_q = h_q^R + \mathrm{i}h_q^I + \mathrm{j}h_q^J + \mathrm{k}h_q^K$ of the q^{th} hidden unit are calculated as:

$$c_q = \sum_{p=1}^{4} w_{qp}x_p + b_q$$

$$= \left\{\sum_{p=1}^{4}\left(w_{qp}^R x_p^R - w_{qp}^I x_p^I - w_{qp}^J x_p^J - w_{qp}^K x_p^K\right) + b_q^R\right\}$$

$$+\mathrm{i}\left\{\sum_{p=1}^{4}\left(w_{qp}^R x_p^I + w_{qp}^I x_p^R + w_{qp}^J x_p^K - w_{qp}^K x_p^J\right) + b_q^I\right\}$$

$$+\mathrm{j}\left\{\sum_{p=1}^{4}\left(w_{qp}^R x_p^J + w_{qp}^I x_p^K + w_{qp}^J x_p^R - w_{qp}^K x_p^I\right) + b_q^J\right\}$$

$$+\mathrm{k}\left\{\sum_{p=1}^{4}\left(w_{qp}^R x_p^K + w_{qp}^I x_p^J + w_{qp}^J x_p^I - w_{qp}^K x_p^R\right) + b_q^K\right\} \quad (19)$$

$$h_q = f\left(c_q^R\right) + \mathrm{i}f\left(c_q^I\right) + \mathrm{j}f\left(c_q^J\right) + \mathrm{k}f\left(c_q^K\right). \quad (20)$$

Here, a split-type activation function is adopted to generate the hidden output. The net input $z_p = z_p^R + \mathrm{i}z_p^I + \mathrm{j}z_p^J + \mathrm{k}z_p^K$ and output $y_p = y_p^R + \mathrm{i}y_p^I + \mathrm{j}y_p^J + \mathrm{k}y_p^K$ of the p^{th} output unit are also calculated as

$$z_p = \sum_{q=1}^{3} h_q w_{qp} + \tilde{b}_p \quad (21)$$

$$y_p = f\left(z_p^R\right) + \mathrm{i}f\left(z_p^I\right) + \mathrm{j}f\left(z_p^J\right) + \mathrm{k}f\left(z_p^K\right). \quad (22)$$

The same formula as (4) is used as the error function. For the training, the weights and biases are updated using the following equations:

$$\Delta\tilde{b}_p = -\gamma\frac{\partial E}{\partial \tilde{b}_p^R} - \mathrm{i}\gamma\frac{\partial E}{\partial \tilde{b}_p^I} - \mathrm{j}\gamma\frac{\partial E}{\partial \tilde{b}_p^J} - \mathrm{k}\gamma\frac{\partial E}{\partial \tilde{b}_p^K}$$

$$= \Delta\tilde{b}_p^R + \mathrm{i}\Delta\tilde{b}_p^I + \mathrm{j}\Delta\tilde{b}_p^J + \mathrm{k}\Delta\tilde{b}_p^K \quad (23)$$

$$\Delta b_q = -\gamma\frac{\partial E}{\partial b_q^R} - \mathrm{i}\gamma\frac{\partial E}{\partial b_q^I} - \mathrm{j}\gamma\frac{\partial E}{\partial b_q^J} - \mathrm{k}\gamma\frac{\partial E}{\partial b_q^K}$$

$$= \Delta b_q^R + \mathrm{i}\Delta b_q^I + \mathrm{j}\Delta b_q^J + \mathrm{k}\Delta b_q^K \quad (24)$$

$$\Delta w_{qp} = -\gamma\frac{\partial E}{\partial w_{qp}^R} - \mathrm{i}\gamma\frac{\partial E}{\partial w_{qp}^I} - \mathrm{j}\gamma\frac{\partial E}{\partial w_{qp}^J} - \mathrm{k}\gamma\frac{\partial E}{\partial w_{qp}^K}$$

$$= \Delta w_{qp}^R + \mathrm{i}\Delta w_{qp}^I + \mathrm{j}\Delta w_{qp}^J + \mathrm{k}\Delta w_{qp}^K$$

$$= \bar{h}_q\Delta b_q\,, \quad (25)$$

Where, γ is the learning rate and $\bar{h}_q$ is the quaternion conjugate of h_q. The following equations are for the partial derivatives within (23)–(25):

$$\frac{\partial E}{\partial \tilde{b}_p^R} = -2\left(x_p^R - y_p^R\right) f'\left(z_p^R\right) \quad (26)$$

$$\frac{\partial E}{\partial \tilde{b}_p^I} = -2\left(x_p^I - y_p^I\right) f'\left(z_p^I\right) \quad (27)$$

$$\frac{\partial E}{\partial \tilde{b}_p^J} = -2\left(x_p^J - y_p^J\right) f'\left(z_p^J\right) \quad (28)$$

$$\frac{\partial E}{\partial \tilde{b}_p^K} = -2\left(x_p^K - y_p^K\right) f'\left(z_p^K\right) \quad (29)$$

$$\frac{\partial E}{\partial b_q^R} = \begin{pmatrix} \sum_{p=1}^{4} \Delta\tilde{b}_p^R w_{qp}^R + \sum_{p=1}^{4} \Delta\tilde{b}_p^I w_{qp}^I \\ + \sum_{p=1}^{4} \Delta\tilde{b}_p^J w_{qp}^J + \sum_{p=1}^{4} \Delta\tilde{b}_p^K w_{qp}^K \end{pmatrix} f'\left(c_q^R\right) \quad (30)$$

$$\frac{\partial E}{\partial b_q^I} = \begin{pmatrix} \sum_{p=1}^{4} \Delta\tilde{b}_p^I w_{qp}^R - \sum_{p=1}^{4} \Delta\tilde{b}_p^R w_{qp}^I \\ + \sum_{p=1}^{4} \Delta\tilde{b}_p^K w_{qp}^J - \sum_{p=1}^{4} \Delta\tilde{b}_p^J w_{qp}^K \end{pmatrix} f'\left(c_q^I\right) \quad (31)$$

$$\frac{\partial E}{\partial b_q^J} = \begin{pmatrix} \sum_{p=1}^{4} \Delta\tilde{b}_p^J w_{qp}^R - \sum_{p=1}^{4} \Delta\tilde{b}_p^K w_{qp}^I \\ - \sum_{p=1}^{4} \Delta\tilde{b}_p^R w_{qp}^J + \sum_{p=1}^{4} \Delta\tilde{b}_p^I w_{qp}^K \end{pmatrix} f'\left(c_q^J\right) \quad (32)$$

$$\frac{\partial E}{\partial b_q^K} = \begin{pmatrix} \sum_{p=1}^{4} \Delta\tilde{b}_p^K w_{qp}^R + \sum_{p=1}^{4} \Delta\tilde{b}_p^J w_{qp}^I \\ - \sum_{p=1}^{4} \Delta\tilde{b}_p^I w_{qp}^J - \sum_{p=1}^{4} \Delta\tilde{b}_p^R w_{qp}^K \end{pmatrix} f'\left(c_q^K\right). \quad (33)$$

The learning process is performed by giving initial values to the parameters and iterating (23)–(25).

D. Classification using Autoencoder

Conventional AEs are found effective to build networks for classification task. Proposed CAE and QAE based networks also might perform well in classification; which is the main intuition of this study. Size of output layer (i.e., nodes in the layer) of the network depends on number of classes to be identified; and size of input layer depends on data and how it process. Hidden layer numbers and sizes are user defined parameters. Autoencoder(s) is used to pre-train hidden layer(s). Output layer is trained only in fine tuning with backpropagation in supervised mode.

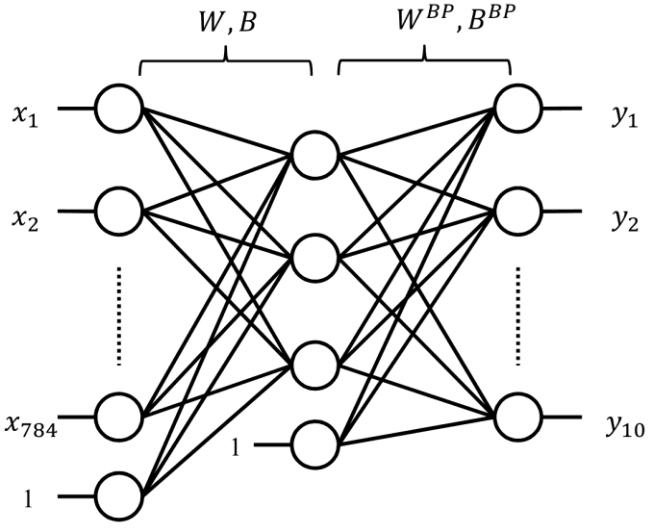


Fig. 2. Network structure using conventional autoencoder for handwritten numeral recognition from 28×28 (=784) size images.

For better understanding of classification using autoencoder, Fig. 2 is the network structure based on conventional AE for MNIST handwritten numeral recognition [28]. MNIST contains handwritten numeral images of 28×28 pixels. Therefore, total input nodes I_{AE} is 784 (=28×28) considering an individual node for an individual pixel value. To classify the digit images, 10 outputs (corresponding to the class labels from 0 to 9) are considered in the output layer. If number of nodes in the hidden layer is defined as $H_{AE,}$ hidden layer weights (W) and biases (B) are size of $H_{AE} \times I_{AE}$ and $H_{AE,}$ respectively. W and B are pre-trained through conventional AE. W^{BP} and B^{BP} are the weight and bias vectors of output layer, respectively, that have to be tuned by back propagation. In the output units, signals from the hidden units are processed by the sigmoid function.

The networks for classification using CAE and QAE are complex and quaternion-valued, respectively. In CAE based network, the complex-valued input neuron manipulate two conjugative pixel values in real and imaginary parts of it. Therefore, number of input in CAE based network (I_{CAE}) will be half of conventional AE based network of Fig. 2. Similarly, number of input in QAE based network (I_{QAE}) will be one fourth of conventional AE based network. At a glance, $I_{AE} = 2I_{CAE} = 4I_{QAE.}$ Due to higher dimension operation, relatively less number of hidden neurons in CAE and QAE might be sufficient. However, real valued output neuron is necessary to classify in both the cases. In order to generate real-valued outputs from the CAE output units, activation functions for CAE and QAE are shown in (34) and (35), respectively.

$$f_{C\to R}(x) = \left(f(x^R) - f(x^I)\right)^2 \quad (34)$$

$$f_{Q\to R}(x) = \left(f(x^R) - f(x^I)\right)^2 - \left(f(x^J) - f(x^K)\right)^2 \quad (35)$$

In both the equations, $f(x) = 1/(1 + e^{-x})$, i.e., sigmoid function. Equation (34) has been investigated for complex-valued neural networks to solve real-valued classification problems [27]. Equation (35) is the proposed activation function for QAE to convert quaternions to real numbers. The methods used back propagation to tune the weights and biases between the hidden and output units.

III. Experimental Studies

This section investigates effectiveness of proposed multivalued AEs (i.e., CAE and QAE) in encoding/decoding and classification of image objects. The outcome of the proposed methods compared with conventional AE. Encoding/decoding ability is observed on handwritten numeral images. Recognition of handwritten numeral is also considered to observe classification ability. Finally, recognition on large-scale objects with many classes is performed. Following individual sections explain experimental setup and compare outcomes of the encoding/decoding and both classification tasks. The algorithms are implemented in PGI® Accelerator C Workstation. Experiments of this study have been conducted on HP Z440 Workstation having CPU Intel(R) Xeon (R) CPU

E5-1603 @ 2.80GHz and RAM 32.0GB in Windows 10 Pro (64 bit) environment.

A. Encoding/Decoding

Performance of encoding/decoding is observed on MNIST database [28]. MNIST database comprises 28×28-pixel gray-scale images of handwritten digits from 0 to 9. From the available samples, 2500 samples are considered as training set and different 2500 samples are used as test set. In each set (training/test) 250 of each digit from 0 to 9 are considered. Same training and test set are used for all three networks (conventional AE, proposed CAE and QAE).

In training/testing, a pattern is represented in different forms in AE, CAE and QAE. Each individual pixel value is an input in AE; therefore, AE required total 784 input neurons. On the other hand, number of input is less in CAE and QAE due to multi-valued neurons. Fig. 3 shows pattern construction for CAE and QAE from a sample numeral image. CAE treated each pair of pixels as one complex number, and QAE considered each quartet of pixels as one quaternion number. Therefore, CAE and QAE required input neurons 392 (=784/2) and 196 (=784/4), respectively.

Two popular activation functions sigmoid and ReLU were considered to the hidden units of each method. On the other hand, only the sigmoid function was applied to the output units of each method. The number of hidden nodes were considered less than input of a method as of many previous studies. Experiments conducted for two different number of hidden units. Table 1 shows the parameters for all the three methods. Here, the notation $AE_{784\text{-}272}$ signifies the method name "AE" as conventional method; and the number of input and hidden units are 784 and 272, respectively. Due to less number of inputs as well as much less number of hidden units considered in proposed multi-valued autoencoders, total parameters were much less than conventional AE.

To assess the learning abilities of the proposed methods, mean squared error (MSE) for the test set and the execution time required for the training were compared. The test error was calculated by

$$\text{MSE} = \frac{1}{2N}\sum_{n=1}^{N}\|X_n - Y_n\|^2 \qquad (36)$$

Where, n is the pattern number and N is the total number of test samples. Furthermore, we discuss the features appeared in the learned weights of each method.

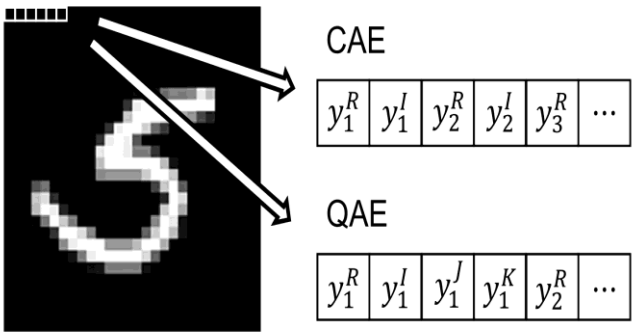


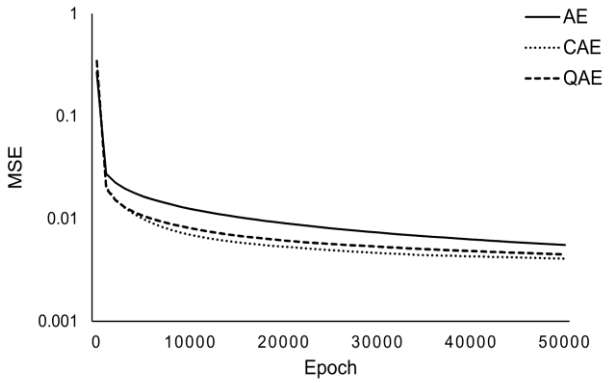
Fig. 3. Pattern construction from a sample MNIST image for CAE and QAE.

TABLE I. NUMBER OF PARAMETERS FOR ENCODER/DECODER TEST ON MNIST IMAGES

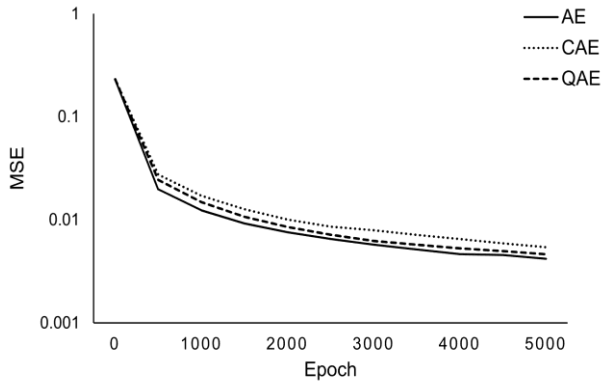
Method	Number of parameters			
	Input unit	***Hidden unit***	***Output unit***	***Total*[a]**
$AE_{784\text{-}272}$	784	272	784	214,304
$AE_{784\text{-}392}$	784	392	784	308,504
$CAE_{392\text{-}136}$	392	136	392	107,680
$CAE_{392\text{-}196}$	392	196	392	154,840
$QAE_{196\text{-}68}$	196	68	196	54,368
$QAE_{196\text{-}98}$	196	98	196	78,008

[a] Total number of parameters includes real and imaginary parts of weights and biases in the CAE and QAE.

Fig. 4 depicts MSE for three methods ($AE_{784\text{-}392}$, $CAE_{392\text{-}196}$, and $QAE_{196\text{-}98}$) for both sigmoid and ReLU functions in hidden units on a sample run. It is observed from the figure that the nature of the MSE curve almost same for all the three methods for a particular activation function. The significance observation from the figure is that all the methods with the ReLU function (Fig. 4(b)) converged much faster than the methods with the sigmoid function (Fig. 4(a)). For ReLU function, MSE reached steady state position for 5000 epochs; whereas, steady state position with similar MSE value for sigmoid was shown to reach for 50000 epochs. Therefore, in further expereiments, 5000 and 50000 epochs are considered for ReLU and sigmoid functions, respectively.



(a) Sigmoid function



(b) ReLU function

Fig. 4. Training process of $AE_{784\text{-}392}$, $CAE_{392\text{-}196}$, and $QAE_{196\text{-}98}$ methods with sigmoid and ReLU activation functions in the hidden units.

TABLE II. AVERAGE TEST ERROR ON AE, CAE AND QAE WITH DIFFERENT ARCHITECTURE AND ACTIVATION FUNCTIONS

Method	Hidden unit act. function	Epoch	MSE ($\times 10^{-3}$)	Time (s)
$AE_{784-272}$	Sigmoid	50000	10.62 ± 0.14	1356
	ReLU	5000	6.41 ± 0.16	154
$AE_{784-392}$	Sigmoid	50000	7.39 ± 0.08	2048
	ReLU	5000	6.08 ± 0.12	228
$CAE_{392-136}$	Sigmoid	50000	9.17 ± 0.07	376
	ReLU	5000	10.53 ± 0.35	43
$CAE_{392-196}$	Sigmoid	50000	6.23 ± 0.08	579
	ReLU	5000	8.60 ± 0.35	63
QAE_{196-68}	Sigmoid	50000	8.88 ± 0.19	206
	ReLU	5000	7.39 ± 0.18	24
QAE_{196-98}	Sigmoid	50000	6.33 ± 0.09	306
	ReLU	5000	6.10 ± 0.15	35

Averages are taken over 5 independent runs.

Table 2 shows test error and required time for each method after fixed number epoch. In relation to the test error, the AE and the QAE with the ReLU function showed better results than those of the methods with the sigmoid function even training epoch is much less in case of ReLU. However, the error of the CAE with the sigmoid function was better than that with the ReLU function. Furthermore, each of the methods showed better convergence as the number of parameters increased. As an example, MSE value of QAE_{196-98} (heaving 98 hidden units) was less than QAE_{196-68} (heaving 68 hidden units) for both sigmoid and ReLU. In terms of the execution time, the period for QAE was much shorter than period for the other two methods as seen from Table 2. This is because the execution time is related to the number of parameters and the computational complexity of the methods. For better understanding, Fig. 5 shows sample output images of $AE_{784-392}$, $CAE_{392-196}$, and QAE_{196-98} with the ReLU function in the hidden units. Comparing with the output images, the proposed methods showed almost the same qualities as that of the conventional AE.

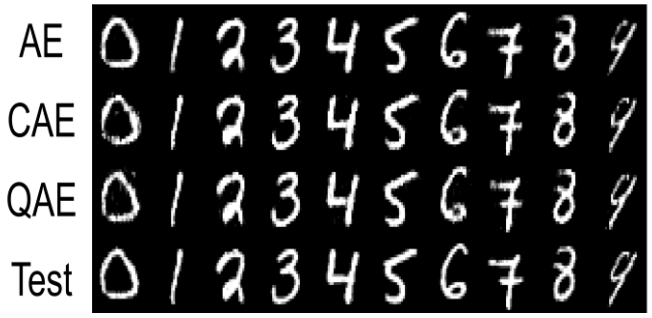


Fig. 5. Sample output images of the conventional and proposed methods. The images were obtained using $AE_{784-392}$, $CAE_{392-196}$, and QAE_{196-98} with ReLU function in the hidden units. The bottom images are the desired outputs of the test set.

B. Handwritten Numeral Recognition (HNR)

HNR is a complex classification task and MNIST database is well studied for this purpose. Classification performance using proposed CAE and QAE is observed and compared with conventional AE on MNIST database [28]. Autoencoder based network construction for classification task is already explained in Section II-D. The learned parameters that were obtained from the encoding/decoding problem in previous section are used as pre-trained hidden layer. To classify the digit images, output layer heaving 10 output nodes (corresponding to the class labels from 0 to 9) is added. In the output units, signals from the hidden units are processed by the sigmoid function. Finally, back propagation is used to tune the weights and biases between the hidden and output units. Fine tuning performed on fixed iteration to compare execution time among the methods. The methods with the ReLU function converged faster than the methods with the sigmoid function; therefore the epochs for fine tuning were 5000 and 10000 for ReLU and sigmoid, respectively.

Table 3 compares the methods in relation to the test set accuracy and the execution time for both sigmoid and ReLU as activation function in hidden units. The method $AE_{784-392-10}$ indicates $AE_{784-392}$ autoencoder from previous section is used and output layer weight (W^{BP}) size is 10×392 which are trained in fine tuning. In relation to the accuracy rate, ReLU achieved better results for AE and QAE; but sigmoid showed better for CAE. However, both proposed methods is found better than conventional AE regardless the activation function. As an example, accuracy for $AE_{784-392-10}$ with ReLU was 81.0%; one the other hand, $CAE_{392-196-10}$ and $QAE_{196-98-10}$ achieved 85.5% and 85.4 %, respectively, for same activation function. Although both CAE and QAE showed competitive accuracy; in relation to the execution time, the QAE was faster than CAE and much faster than AE. For 5000 epochs with ReLU, $QAE_{196-98-10}$ took 18 seconds; whereas, $CAE_{392-196-10}$ and $AE_{784-392-10}$ took 31 and 139 seconds, respectively.

TABLE III. AVERAGE TEST SET ACCURACY ON MNIST HANDWRITTEN NUMERAL RECOGNITION

Method	Hidden unit act. function	Epoch	Accuracy rate (%)	Time (s)
$AE_{784-392-10}$	Sigmoid	10000	76.1	280
	ReLU	5000	81.0	139
$CAE_{392-196-10}$	Sigmoid	10000	85.6	64
	ReLU	5000	85.5	31
$QAE_{196-98-10}$	Sigmoid	10000	85.2	38
	ReLU	5000	85.4	18

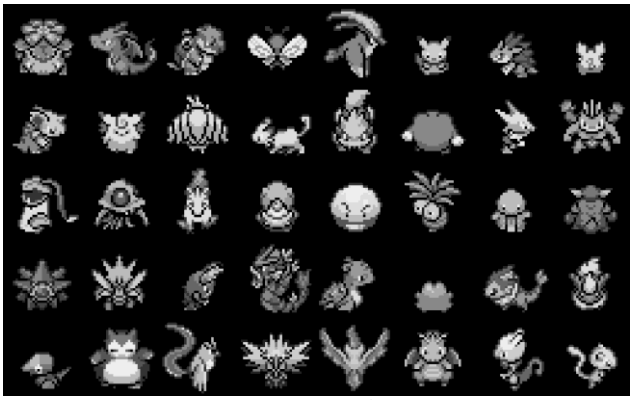
Averages are taken over 5 independent runs.

C. Pokémon Character Recognition (PCR)

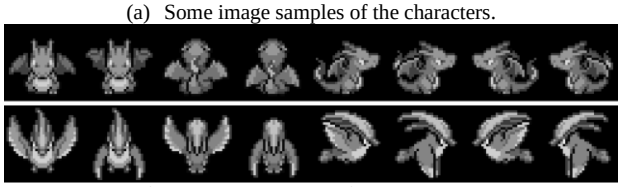
In this section, proposed CAE and QAE are evaluated and compared with AE on Pokémon dataset[1] which is relatively much complex problem. Pokémon is the registered trademark of Nintendo /Creatures Inc. /GAME FREAK Inc. The dataset is a collection of RGB images of 151 Pokémons where each character image is 32×32 pixels. A gray-scaled dataset is considered in this study to perform experiments. Fig. 6 shows few samples from of the dataset. A single character has eight patterns: two for each of the front, back, right, and left sides as shown in Fig. 6(b). Therefore, the dataset is a collection of 1208 (=151×8) images; and the task is to recognize the images into 151 character classes. Depending on the characters, the images have quite different patterns. Due to large number of classes, PCR is much complex than MNIST recognition task.

[1] http://hikochans.com/pixelart/

Network structure and total parameters for AE, CAE and QAE are shown in Table 4. An image with 32×32 pixels is feed to AE network as 1024 (=32×32) inputs. Pattern construction for CAE and QAE is similar to pattern construction from MNIST image data: one CAE neuron processes a pair of pixels as a complex number and one QAE processes four conjugative pixels values as one quaternion number. Therefore, in the experiments, inputs of CAE and QAE networks were 512 (=1024/2) and 256 (1024/4), respectively. Hidden nodes of the networks were also selected in similar fashion. Total parameter of a method depends on nodes in the input and hidden layers. Due to quaternion presentation and less number of hidden neurons, total parameters in QAE is less than CAE and AE. The activation function in hidden units was the ReLU function for each method. 75% of available objects were considered as training set and rest 25% were used for test purpose. In two different selections, two different data sets (training and test sets) were prepared.



(a) Some image samples of the characters.



(b) Each of eight pattern images of two characters (classes).

Fig. 6. Image samples of the Pokémon characters.

TABLE IV. PARAMETERS FOR POKÉMON CHARACTER RECOGNITION

Method	Number of parameters			
	Input unit	***Hidden unit***	***Output unit***	***Total[a]***
$AE_{1024\text{-}400\text{-}151}$	1024	400	151	470,551
$CAE_{512\text{-}350\text{-}151}$	512	350	151	465,102
$QAE_{256\text{-}200\text{-}151}$	256	200	151	327,004

[a] Total number of parameters includes real and imaginary parts of weights and biases in the CAE and QAE.

Similar to other autoencoder based classification, training performs in two different phases: autoencoder based pre-training of hidden layer and fine tuning of output layer. More specifically, in $AE_{1024\text{-}400\text{-}151,}$ the hidden layer is conventional AE with size 400×1024 and output layer weight (W^{BP}) with size 151×400 are trained in fine tuning through back propagation. Training epochs in first phase (autoencoder) were 10000 for all three methods. On the other hand, training epochs of a method in second phase (fine tuning) were 5000 with mini batch of 151.

Fig. 7 shows sample output images of the methods for training set of data set 1 in the first phase. It is noticeable from the figure that all the methods were able to learn the training images. Table 5 compares test set recognition accuracy as well as required times in both phases for the three methods. It is noticeable that conventional AE method showed the worst recognition accuracy for both the data sets which were only 11.4% and 11.9% for data set 1 and 2, respectively. Besides better encoding in first phase (as seen in Fig. 7), the worst recognition performance of AE revealed the limitation of real valued network for a problem to classify objects in such a large number of classes. Number of hidden node enlargement might improve performance but not significant level. In such a case number of parameters and hence computation complexity will increase much. With similar number of parameters, CAE showed very good recognition accuracy which were 94.1% and 96.2% for data set 1 and 2, respectively. On the other hand, with less number of parameters, QAE showed competitive performance to CAE and which were 92.1% for both the data sets. In relation to training time, QAE took less time in both the phases with resect CAE and AE. Finally, proposed CAE and QAE revealed as good recognition methods for such large-scale multi-class images.

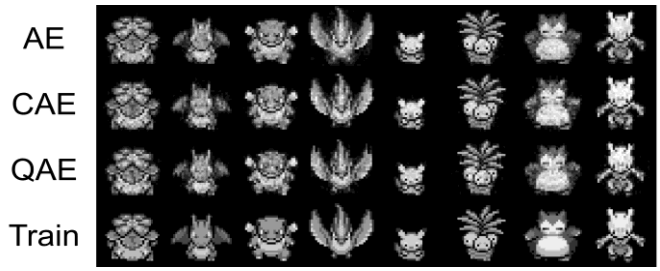


Fig. 7. Sample output images of the conventional and proposed methods for training set 1 in the first phase. The bottom images are the desired outputs.

TABLE V. TEST SET ACCURACY ON POKÉMON CHARACTER RECOGNITION

Method	Data Set	Accuracy Rate (%)	Time (s)	
			First Phase	***Second Phase***
$AE_{1024\text{-}400\text{-}151}$	1	11.4	349	1211
	2	11.9	350	1183
$CAE_{512\text{-}350\text{-}151}$	1	94.1	196	685
	2	96.2	196	637
$QAE_{256\text{-}200\text{-}151}$	1	92.1	115	595
	2	92.1	115	560

Averages are taken over 5 independent runs.

IV. CONCLUSIONS

This paper investigated two multi-valued autoencoders by extending the conventional AE to complex and quaternion domains which are complex-valued autoencoder (CAE) and quaternion-valued autoencoder (QAE). Proposed CAE is a two-layered neural network with inputs, outputs, weights, and biases in complex domain. The tuning equations of the weights and biases are based on the complex gradient descent method. We adopted an easy-to-use split-type activation function in the hidden and output units. On the other hand,

proposed QAE is also a two-layered neural network but with inputs, outputs, weights, and biases in quaternion domain. The tuning equations of the parameters are based on the quaternion gradient descent method. The split-type activation function is also applied to the QAE. Although computational complexities become higher in the proposed multi-valued encoders relatively small sized architecture is found worthy to handle a given task.

Proposed multi-valued autoencoders outperformed conventional AE while tested for encoding/decoding and classification tasks. In encoding/decoding task, proposed CAE and QAE showed better convergence than AE for fixed number of epochs. In terms of the execution time, QAE took the shortest time and CAE also took less time than AE. In case of MNIST handwritten numeral recognition based on the individual autoencoders, proposed CAE and QAE was better than conventional AE. The most significant outcomes of the proposed methods are observed on Pokémon Character Recognition (PCR) which is a large-scaled multi-class problem having 151 classes. In PCR, CAE and QAE achieved more than 90% accuracy; whereas accuracy for AE was below 20%. Moreover, proposed methods took less time than convention AE. Experimental studies with different settings identified the proficiency of the proposed multi-valued autoencoders.

A number of future researches are opened from this study. In the present study, split-type activation functions were considered for the proposed autoencoders. Complex-valued and quaternion neural networks with fully complex- and quaternion-valued activation functions have been studied recently [29], [30]. Thus, such activation functions into CAE and QAE might improve their performance and remained as future work. Furthermore, only gray-scale image data is used in the experiments. In a previous study, a QNN was used to treat color image data [22]. It dealt with RGB color values as quaternion numbers and showed good performance. Applications of the QAE to such color image data would be desired. Moreover, deep neural networks based on the proposed autoencoders might perform well and remain as future study.

REFERENCES

[1] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," Science, vol. 313, pp.504–507, 2006.

[2] Y. Bengio, P. Lamblin, D. Popovici and H. Larochelle, "Greedy layer-wise training of deep networks," Adv. Neural Inf. Process. Syst., vol. 19, pp. 153–160, 2007.

[3] P. Vincent, H. Larochelle, Y. Bengio and P. A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in Proc. 25th Int. Conf. Machine Learning, ACM, pp. 1096–1103, 2008.

[4] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio and P. A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," J. Machine Learning Research, vol. 11, pp. 3371–3408, 2010.

[5] R. Socher, E. H. Huang, J. Pennin, C. D. Manning and A. Y. Ng, "Dynamic pooling and unfolding recursive autoencoders for paraphrase detection," in Advances in Neural Information Processing Systems, pp. 801–809, 2011.

[6] M. Längkvist and A. Loutfi, "Learning feature representations with a cost-relevant sparse autoencoder," Int. J. Neural Syst., vol. 25(01), 1450034, 2015.

[7] T. Nitta, "An extension of the back-propagation algorithm to complex numbers," Neural Networks, vol. 10(8), pp. 1391–1415, 1997.

[8] T. Nitta, "A quaternary version of the back-propagation algorithm," in Proc. IEEE Int. Conf. Neural Networks, pp. 2753–2756, 1995.

[9] A. Hirose, "Complex-Valued Neural Networks: theories and applications," World Scientific, vol. 5, 2003.

[10] A. Hirose, "Complex-Valued Neural Networks," Springer Science & Business Media, vol. 400, 2006.

[11] A. Hirose, "Complex-Valued Neural Networks: Advances and Applications," IEEE Press, 2013.

[12] I. Aizenberg, "Complex-Valued Neural Networks with Multi-Valued Neurons," Heidelberg: Springer 353, 2011.

[13] T. Nitta, "Complex-Valued Neural Networks: Utilizing High-Dimensional Parameters," IGI Global, 2009.

[14] X. Chen, Z. Tang, C. Variappan, S. Li and T. Okada, "A modified error backpropagation algorithm for complex-value neural networks," Int. J. Neural Syst., vol. 15(06), pp. 435–443, 2005.

[15] T. Nitta, "The uniqueness theorem for complex-valued neural networks with threshold parameters and the redundancy of the parameters," Int. J. Neural Syst., vol. 18(02), pp. 123–134, 2008.

[16] H. Aoki, "Applications of Complex-Valued Neural Networks for Image Processing," in Complex-Valued Neural Networks: Theories and Applications, World Scientific Publishing, Singapore, vol. 5, pp. 181–204, 2003.

[17] A. B. Suksmono and A. Hirose, "Adaptive interferometric radar image processing by complex-valued neural networks," in Complex-Valued Neural Networks: Theories and Applications, World Scientific Publishing, Singapore, vol. 5, pp. 277–301, 2003.

[18] S. Buchholz and N. Le Bihan, "Polarized signal classification by complex and quaternionic multi-layer perceptrons," Int. J. Neural Syst., vol. 18(02), pp. 75–85, 2008.

[19] A. R. Hafiz, M. F. Amin and K. Murase, "Real-time hand gesture recognition using complex-valued neural network (CVNN)," in Int. Conf. Neural Information Processing, Springer Berlin Heidelberg, pp. 541–549. 2011.

[20] I. Nishikawa, T. Iritani, K. Sakakibara and Y. Kuroe, "Phase dynamics of complex-valued neural networks and its application to traffic signal control," Int. J. Neural Systems, vol. 15(01n02), pp. 111–120, 2005.

[21] N. Matsui and T. Isokawa, "Quaternion neural network with geometrical operators," Intelligent and Fuzzy Systems, vol. 15(3), pp. 149–164, 2004.

[22] H. Kusamichi, T. Isokawa, N. Matsui, Y. Ogawa and K. Maeda, "A new scheme for color night vision by quaternion neural network," in 2nd Int. Conf. Autonomous Robots and Agents, pp. 101–106, 2004.

[23] M. Yoshida, Y. Kuroe and T. Mori, "Models of Hopfield-type quaternion neural networks and their energy functions," Int. J. Neural Systems, vol. 15(01n02), pp. 129–135, 2005.

[24] T. Isokawa, H. Nishimura, N. Kamiura and N. Matsui, "Associative memory in quaternionic hopfield neural network," Int. J. Neural Syst., vol. 18(02), pp. 135–145, 2008.

[25] R. Hata and K. Murase, "Multi-valued autoencoders for multi-valued neural networks," in IEEE Int. Joint Conf. Neural Networks (IEEE World Congress on Computational Intelligence), pp. 4412–4417, 2016.

[26] P. Baldi and Z. Lu, "Complex-valued autoencoders," Neural Networks, vol. 33, pp. 136–147, 2012.

[27] M. F. Amin and K. Murase, "Single-layered complex-valued neural network for real-valued classification problems," Neurocomputing, vol. 72(4), pp. 945–955, 2009.

[28] Y. LeCun, C. Cortes and C. J. Burges, The MNIST database of handwritten digits, 1989.

[29] T. Kim and T. Adali, "Approximation by fully complex multilayer perceptrons," Neural Computation, vol. 15(7), pp. 1641–1666, 2003.

[30] T. Isokawa and H. Nishimura, "Quaternionic multilayer perceptron with local analyticity," Information, vol. 3(4), pp. 756–770, 2003.