SCIENCE CHINA
Information Sciences

• **Supplementary File** •

RGA-CNNs: Convolutional Neural Networks based on Reduced Geometric Algebra

Rui WANG[1], Miaomiao SHEN[1], Xiangyang WANG[1] & Wenming CAO[2*]

[1]*School of Communication and Information Engineering, Shanghai University, Shanghai, 200444, China;*
[2]*College of Information Engineering, Shenzhen University, Shenzhen, 518060, China*

Appendix A Preliminaries

Appendix A.1 Convolutional Neural Network (CNN)

Convolutional neural network (CNN) is an efficient recognition algorithm which shows a wide range of applications in image processing, pattern recognition and other fields in recent years [1, 2]. It has a simple structure, few training parameters and good adaptability and other advantages. The basic structure of CNN includes input, convolution, pooling, hidden, and output layers as shown in Figure A1.

Evidently, fully connect hidden layers can usually be added before the output layer, and correspondingly after pooling layer. The convolution layer is used for extracting local features in images and then the pooling layer is applied to reduce the location sensitivity of the features which are extracted from convolution layer. Usually, the convolution layer and the pooling layer may repeat for several times in the structure of CNNs. In a convolution layer with input o, the j-th channel is given by

$$o \bigotimes J^{(j)} + b^{(j)} \tag{A1}$$

where $\bigotimes$ is the convolution operation, and $\left\{J^{(j)}, b^{(j)}\right\}$ are the convolution kernels and bias terms, respectively.

Pooling is used to induce invariance to small translations, which is a characteristic of natural images. A pooling layer does so by splitting each input channel into patches, and replacing each patch with a single representative value in the output layer. Typical choices the maximal or average value, in max and average pooling, respectively. There exists instinct superiority that is, for large scale data, more layers and larger quantities of hidden elements are significantly necessary to be utilized. Then, by constructing such a structure, large-scale inputs can be reduced in size step by step and the characteristics becomes more obvious during learning process.

Recently, the research of CNNs theory has been extended from real number to hyper-complex number, typically presented as quaternion for three-dimensional data, such as color image. Since CNNs have proven to be very powerful in handling images, and quaternion numbers can produce meaningful representations in this domain, then the extension of real-valued convolutional neural networks to quaternion domain, namely quaternion-valued convolutional neural networks (QCNNs), have shown great potential in color image processing. Obviously, real-valued CNNs had achieved state-of-the-art performance when dealing with scalar data, typically as gray-scale image, and leading to some unsatisfying results with respect to color image in which the color image to be processed was usually separated into three independent color channels leading to the loss of relationship among three color channels. Then, QCNNs regard each color pixel as a pure quaternion and process multichannel information in a parallel way, which mimic the human perception of a visual environment and the inherent color structure can be preserved completely.

However, a series of operations will produce a quaternion containing the real part, which will inevitably lead to the data redundancy. In addition, the quaternion multiplication is non-commutative, which is not conducive to design a simple network. Thus, QCNNs lead to a large number of data redundancy and a complicated network.

* Corresponding author (email: wmcao@szu.edu.cn)

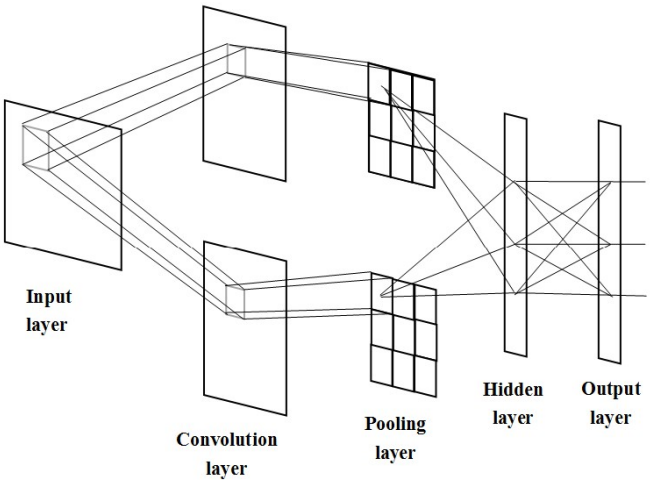

Figure A1 The typical basic structure of CNNs.

Appendix A.2 The basics of geometric algebra

This section briefly reviews the basics of Geometric Algebra (GA), which was introduced by William K. Clifford and also called Clifford Algebra [3, 4, 5, 6, 7, 8, 9, 10,11,12]. Suppose $\mathbb{G}_n$ is n-dimensional GA with an orthonormal basis of vectors , which leads to a basis

$$\left\{1, \{e_i\}, \{e_i e_j\}, \cdots, \{e_1 e_2 \cdots e_n\}\right\} \tag{A2}$$

Specifically, $\mathbb{G}_n$ can be represented by $\mathbb{G}_{p,q}$, where $n = p + q$ and p, q are the number of vectors with positive square and negative square in the basis of the space, respectively. That is

$$e_i^2 = \begin{cases} 1, & 1 \leqslant i \leqslant p \\ -1, & p+1 \leqslant i \leqslant n \end{cases} \tag{A3}$$

we focus on $\mathbb{G}_n$, in which $q = 0$. The geometric product of two basis is anti-commutative, and

$$e_i e_j = e_{ij} = -e_j e_i = -e_{ji}, i, j = 1, \cdots, n, i \neq j \tag{A4}$$

$$e_i^2 = 1, i = 1, \cdots, n \tag{A5}$$

$$e_i e_{ij} = e_i e_i e_j = e_j, i, j = 1, \cdots, n, i \neq j \tag{A6}$$

For vectors ω and z of $\mathbb{G}_n$, the geometric product is defined as:

$$wz = w \cdot z + w \wedge z \tag{A7}$$

where $w \cdot z$ denotes the inner product, $w \wedge z$ denotes the outer product. Since the e_i vectors are orthogonal, then $e_i e_j = e_i \cdot e_j + e_i \wedge e_j = e_i \wedge e_j$.

Take $\mathbb{G}_3$ for instance, the orthogonal bases are constructed by vectors with $2^3 = 8$ grades, which is given by: {1,{e_1,e_2,e_3}, {e_1e_2,e_1e_3,e_2e_3},{$e_1e_2e_3$}} and a simpler form: $\{1, e_1, e_2, e_3, e_{12}, e_{13}, e_{23}, e_{123}\}$. For a 2^n-dimensional signal, only n basis are needed. Generally, the outer product of k vectors is called a k-blade. The number k is called the grade of the blade.

The complete mathematical correlation between two vectors has been provided by GA. As the extension of vectors to higher dimensions, multivectors are the basic components in GA. Any multivectors $M \in \mathbb{G}_n$ can be described by

$$M = E_0 + \sum_{1 \leqslant i \leqslant n} E_i(M) e_i + \sum_{1 \leqslant i < j \leqslant n} E_{ij}(M) e_{ij} + \cdots + E_{1 \cdots n}(M) e_{1 \cdots n} \tag{A8}$$

where $E(M) \in \mathbb{R}$.

Appendix A.3 Reduced Geometric Algebra (RGA)

Appendix A.3.1 *The basics of RGA*

The definition of the reduced geometric algebra (RGA) [13] is given as follows:

$$\varepsilon_i = \frac{1}{2}\left(1 + e_i e_{n+i}\right) \in \mathbb{G}_n^R, i = 1, 2, \cdots, n \tag{A9}$$

According to (A4), the geometric product of and can be obtained by:

$$\varepsilon_i \varepsilon_j = \varepsilon_j \varepsilon_i, i \neq j \tag{A10}$$

Moreover, we define:

$$\varepsilon_i^2 = \varepsilon_i \varepsilon_j = \begin{cases} \varepsilon_{i+1}, & i = 1, 2, \cdots, n-1 \\ \varepsilon_1, & i = n \end{cases} \tag{A11}$$

Take $n = 3$ for instance, $\varepsilon_1^2 = \varepsilon_2, \varepsilon_2^2 = \varepsilon_3, \varepsilon_3^2 = \varepsilon_1$. Thinking together (A10) and (A11), it is clearly that the multiplication of ε_i is commutative. Specifically, RGA is denoted as $\mathbb{G}_n^R$ and can be seen as the space that is generated by the collection of $\left\{\{\varepsilon_1, \varepsilon_2, \cdots, \varepsilon_n\}, \{\varepsilon_{ij} = \varepsilon_i \varepsilon_j, 1 \leqslant i \neq j \leqslant n\}\right\}$. The element k in $\mathbb{G}_2^R$ has the following form:

$$k = a^1 \varepsilon_1 + a^2 \varepsilon_2 + a^3 \varepsilon_{12}, a^1, a^2, a^3 \in \mathbb{R} \tag{A12}$$

The addition and subtraction operations in $\mathbb{G}_2^R$ are almost the same as GA, here we present the multiplication operation only. $\forall k, l \in \mathbb{G}_2^R$, suppose $k = a^1 \varepsilon_1 + a^2 \varepsilon_2 + a^3 \varepsilon_{12}$ and $l = b^1 \varepsilon_1 + b^2 \varepsilon_2 + b^3 \varepsilon_{12}$, then the multiplication in $\mathbb{G}_2^R$ are given as

$$\begin{aligned} kl &= \left(a^1 \varepsilon_1 + a^2 \varepsilon_2 + a^3 \varepsilon_{12}\right)\left(b^1 \varepsilon_1 + b^2 \varepsilon_2 + b^3 \varepsilon_{12}\right) \\ &= \left(a^1 b^3 + a^2 b^2 + a^3 b^1\right) \varepsilon_1 + \left(a^1 b^1 + a^2 b^3 + a^3 b^2\right) \varepsilon_2 + \left(a^1 b^2 + a^2 b^1 + a^3 b^3\right) \varepsilon_{12} \end{aligned} \tag{A13}$$

where $\varepsilon_{12}\varepsilon_1 = \varepsilon_1\varepsilon_2\varepsilon_1 = \varepsilon_1\varepsilon_1\varepsilon_2 = \varepsilon_1^2\varepsilon_2 = \varepsilon_2\varepsilon_2 = \varepsilon_1$, similarly, $\varepsilon_{12}\varepsilon_2 = \varepsilon_2$. Moreover, $\varepsilon_{12}\varepsilon_{12} = \varepsilon_1\varepsilon_2\varepsilon_1\varepsilon_2 = \varepsilon_1\varepsilon_1\varepsilon_2\varepsilon_2 = \varepsilon_1^2\varepsilon_2^2 = \varepsilon_2\varepsilon_1 = \varepsilon_{12}$

It can be seen from (A13) that no superfluous components are produced in the results of the multiplication of k and l, which only contain ε_1, ε_2 and ε_{12} components. In this way, it successfully overcomes the data redundancy existed in the operations for color image based on quaternion.

We then define the norm of the elements in $\mathbb{G}_2^R$ as

$$\|k\| = a\varepsilon_1 + b\varepsilon_2 + c\varepsilon_{12} = |a + b + c| = \sqrt{a^2 + b^2 + c^2} \tag{A14}$$

The conjugate of k is defined as

$$k^* = a'\varepsilon_1 + b'\varepsilon_2 + c'\varepsilon_{12} \tag{A15}$$

Then

$$\begin{aligned} kk^* &= (a\varepsilon_1 + b\varepsilon_2 + c\varepsilon_{12})\left(a'\varepsilon_1 + b'\varepsilon_2 + c'\varepsilon_{12}\right) \\ &= \left(ca' + bb' + ac'\right)\varepsilon_1 + \left(aa' + cb' + bc'\right)\varepsilon_2 + \left(ba' + ab' + cc'\right)\varepsilon_{12} = \|k\|^2 \end{aligned} \tag{A16}$$

According to (A16), the following equations are obtained

$$\begin{cases} ca' + bb' + ac' = 0 \\ aa' + cb' + bc' = 0 \\ ba' + ab' + cc' = \|k\|^2 \end{cases} \tag{A17}$$

When solving the equations in (A17), the values of the individual components in (A15) are correspondingly yielded, but not all the elements of $\mathbb{G}_2^R$ are conjugate.

Thus, the inverse of element k in $\mathbb{G}_2^R$ can be defined as

$$k^{-1} = \frac{k^*}{\|k\|^2} \tag{A18}$$

In RGA, multivectors, which are the extension of vectors to higher dimensions, are the basic units. Each multivector $K \in \mathbb{G}_2^R$ is so described by

$$K = K^1 \varepsilon_1 + K^2 \varepsilon_2 + K^3 \varepsilon_{12} \tag{A19}$$

where $K^1, K^2, K^3 \in \mathbb{R}$.

Appendix A.3.2 *Convolution in RGA*

Refer to the definition of convolution for multivectors in GA space [8], we give the convolution of two multivectors in $\mathbb{G}_2^R$. Now, let $K = K^1\varepsilon_1 + K^2\varepsilon_2 + K^3\varepsilon_{12}$ and $L = L^1\varepsilon_1 + L^2\varepsilon_2 + L^3\varepsilon_{12}$ be a multivector field a multivector-valued filter, respectively. The RGA convolution is defined as

$$\mathbf{c}(x, y) = \iint_{\mathbb{G}_2^R} K(x, y) L(x - u, y - v) du dv \tag{A20}$$

where (x, y) indicates the coordinate. For discrete multivector space, the convolution has to be discretized and can be deduced as follows

$$\mathbf{c}(x, y) = \frac{1}{MN} \sum_{i=0}^{\infty} \sum_{j=0}^{\infty} K(x, y) L(x - i, y - j) \tag{A21}$$

Clearly, each color image pixel is represented as a RGA multivector, the convolution can be performed referring to (A21). Since the multiplication in $\mathbb{G}_2^R$ is commutative ($KL = LK$), the right-side and the left-side convolutions can be quite the same.

Suppose $f(x, y) \in \mathbb{G}_2^R$ and $g(x, y) \in \mathbb{G}_2^R$ are a color image and a filter in RGA space respectively. Consequently, we propose the definition of convolution in RGA for color image as follows

$$f(x, y) \bigotimes g(x, y) = \frac{1}{MN} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} f(x - i, y - j) g(i, j) \tag{A22}$$

where M and N denote the size of convolution window.

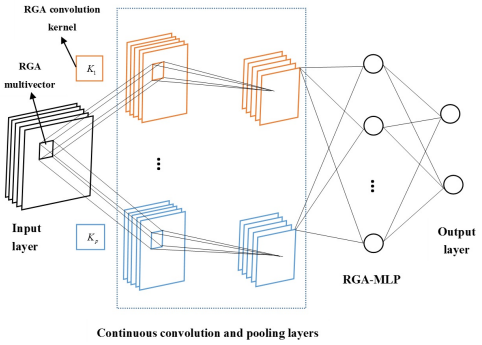

Figure B1 The structure of the proposed RGA-CNNs.

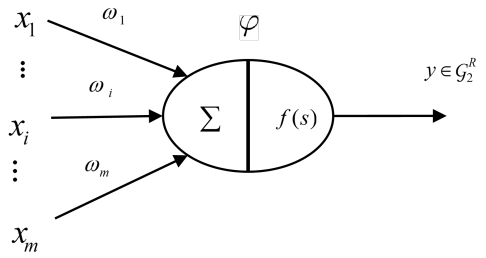

Figure B2 The structure of RGA neuron model.

Appendix B Convolutional Neural Networks based on Reduced Geometric Algebra

Appendix B.1 The structure of RGA-CNNs

Since existing CNNs treat each color image pixel as a scalar with scalar input and scalar convolution kernels which loses some color structures, or a pure quaternion with high data redundancy and a complicated network.

In this section, we propose a novel convolutional neural networks based on RGA, namely RGA-CNNs, which extend both the input color image pixels and convolution kernels as RGA multivectors, and all the operations are extended into the RGA domain, to capture the inherent color structures, simplify the network and remove data redundancy. The basic structure of the proposed RGA-CNNs, in which each representation and operation are based on RGA, is shown in Figure B1, including input layer, convolution layer, pooling layer, multilayer perceptron (MLP) and output layer.

Appendix B.2 RGA neuron model

Compared with conventional neuron model, RGA neuron extends all the operators, especially input, output and convolution kernels, as RGA multivectors. Figure B2 shows the structure of the proposed RGA neuron model. For the input RGA signal $x \in \mathbb{G}_2^R$, the output $y \in \mathbb{G}_2^R$ is represented as

$$y = f(s) \tag{B1}$$

and

$$s = \frac{\omega x}{\|\omega\|} - \varphi = \frac{\left(\omega^1\varepsilon_1 + \omega^2\varepsilon_2 + \omega^3\varepsilon_{12}\right)\left(x^1\varepsilon_1 + x^2\varepsilon_2 + x^3\varepsilon_{12}\right)}{\sqrt{(\omega^1)^2 + (\omega^2)^2 + (\omega^3)^2}} - \varphi = \frac{\begin{bmatrix}\left(\omega^1x^3 + \omega^2x^2 + \omega^3x^1\right)\varepsilon_1 \\ \left(\omega^1x^1 + \omega^2x^3 + \omega^3x^2\right)\varepsilon_2 \\ \left(\omega^1x^2 + \omega^2x^1 + \omega^3x^3\right)\varepsilon_{12}\end{bmatrix}}{\sqrt{(\omega^1)^2 + (\omega^2)^2 + (\omega^3)^2}} - \varphi \tag{B2}$$

where $\omega \in \mathbb{G}_2^R$ denotes the weight of the connections between neurons, φ is the internal state of the RGA neuron and also called bias.

The output of the neuron is determined by the output of the activation function $f(\cdot)$, which is defined by

$$f(s) = f(s^1)\varepsilon_1 + f(s^2)\varepsilon_2 + f(s^3)\varepsilon_{12} \tag{B3}$$

where $s = s^1\varepsilon_1 + s^2\varepsilon_2 + s^3\varepsilon_{12} \in \mathbb{G}_2^R$, $s^1, s^2, s^3 \in \mathbb{R}$ and $f(u) = \frac{1}{1+e^{-u}}, u \in \mathbb{R}$.

Obviously, a left-side (right-side) weight association neuron is generally inferior to a spinor one in terms of learning ability in some tasks. However, the computational complexity of the network with spinor neuron is more complicated than that of the single-side one due to the two-side weight association. Fortunately, RGA is commutative ($\omega x = x\omega$), that is to say the single-side weight association neuron can achieve almost the same learning ability as spinor neuron while removing the data redundancy and simplifying the network.

Appendix B.3 RGA multilayer perceptron (RGA-MLP) and its learning algorithm

In this subsection, we extend conventional multilayer perceptron (MLP) from real number to RGA, in which all the signals, weights and activation values are substituted for RGA numbers taken from a given RGA, to accommodate color image processing. Figure B3 shows the basic structure of RGA-MLP with 3 fully connect hidden layers.

In Figure B3, the input and output signals of each neuron, the weights between two neurons of adjacent two layers are all RGA multivectors, that is $y, \omega \in \mathbb{G}_2^R$. y_j^{l-1} denotes the output of the neuron j of the $(l-1)$-th layer and ω_{jr}^{l-1} means the connection weight between neuron j in the $(l-1)$-th layer and neuron r in the (l)-th layer, that is the output layer. Suppose $x_i^l = \left(x^1\varepsilon_1 + x^2\varepsilon_2 + x^3\varepsilon_{12}\right)_i^l \in \mathbb{G}_2^R$ is the input of neuron i of the (l)-th layer in RGA-MLP, $y_i^l \in \mathbb{G}_2^R$ and $\varphi_i^l \in \mathbb{G}_2^R$

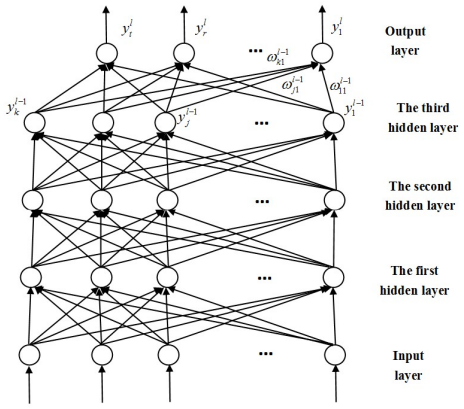

Figure B3 The structure of RGA-MLP containing 3 hidden layers.

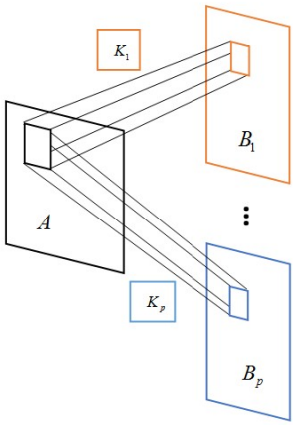

Figure B4 The convolution layer of RGA-CNNs.

are output and bias of this neuron. $\omega_{ji}^{l-1} = \left(\omega^1\varepsilon_1 + \omega^2\varepsilon_2 + \omega^3\varepsilon_{12}\right)_{ji}^{l-1} \in \mathbb{G}_2^R$ denotes the weight between neuron j in the $(l-1)$-th layer and neuron i in the (l)-th layer, then

$$x_i^l = \varphi_i^l + \sum_{j=1}^{k} \omega_{ji}^{l-1} y_j^{l-1} \tag{B4}$$

$$y_i^l = f(x_i^l) \tag{B5}$$

As a normally and effective updating algorithm, back propagation (BP) is utilized to minimize the output error, which can be calculated by outputs of neurons in the output layer and the desired output signals. In this subsection, BP algorithm has been extended to the RGA domain and then adopted to train the proposed RGA-CNNs scheme.

Suppose g_i is the desired output of the neuron r in the output layer, and it is represented as a RGA multivector. As for a series of input multi-dimensional data, the output data are determined firmly by the connection weights, we define the error E as a form of square error function.

$$\begin{aligned} E &= E(y_1^l, \cdots, y_t^l, \cdots, y_r^l) = \frac{1}{2}\sum_{t=1}^{r}\left(y_t^l - g_t\right)^2 = \frac{1}{2}\sum_{t=1}^{r}\left(\left(y^1\varepsilon_1 + y^2\varepsilon_2 + y^3\varepsilon_{12}\right)_t^l - \left(g^1\varepsilon_1 + g^2\varepsilon_2 + g^3\varepsilon_{12}\right)_t\right)^2 \\ &= \frac{1}{2}\sum_{t=1}^{r}\left(\left((y^1)_t^l - (g^1)_t\right)^2 + \left((y^2)_t^l - (g^2)_t\right)^2 + \left((y^3)_t^l - (g^3)_t\right)^2\right) \end{aligned} \tag{B6}$$

where the l-th layer is the output layer, and g_t is the desired response of neuron t in the output layer.

Specifically, the goal of training is to acquire minimal error, usually accompanied by the adjustment of weights, and the adjusting orientation of weight parameters can be denoted as

$$\Delta\omega_{ji}^l = -\eta\frac{\partial E}{\partial\omega_{ji}^l} = -\eta\frac{\partial E}{\partial(\omega^1\varepsilon_1 + \omega^2\varepsilon_2 + \omega^3\varepsilon_{12})_{ji}^l} = -\eta(\frac{\partial E}{\partial(\omega^1)_{ji}^l}\varepsilon_1 + \frac{\partial E}{\partial(\omega^2)_{ji}^l}\varepsilon_2 + \frac{\partial E}{\partial(\omega^3)_{ji}^l}\varepsilon_{12}) \tag{B7}$$

Similarly, that of bias can be obtained by

$$\Delta\varphi_i^l = -\eta\frac{\partial E}{\partial\varphi_i^l} = -\eta\frac{\partial E}{\partial(\varphi^1\varepsilon_1 + \varphi^2\varepsilon_2 + \varphi^3\varepsilon_{12})_i^l} = -\eta(\frac{\partial E}{\partial(\varphi^1)_i^l}\varepsilon_1 + \frac{\partial E}{\partial(\varphi^2)_i^l}\varepsilon_2 + \frac{\partial E}{\partial(\varphi^3)_i^l}\varepsilon_{12}) \tag{B8}$$

Then, we give

$$\frac{\partial E}{\partial\omega_{ji}^l} = \Gamma_{ji}^l(y^{l-1})^* \tag{B9}$$

$$\frac{\partial E}{\partial\varphi_i^l} = \Gamma_i^l \tag{B10}$$

where $*$ is set as the conjugation, η is a constant denoting the learning coefficient and Γ denotes the residual error, also called sensitivity. In the update of neuron i in the output layer, Γ is given by

$$\Gamma_i^l = \left(f'(s^1)\left((y^1)_i^l - (g^1)_i\right)\right)\varepsilon_1 + \left(f'(s^2)\left((y^2)_i^l - (g^2)_i\right)\right)\varepsilon_2 + \left(f'(s^3)\left((y^3)_i^l - (g^3)_i\right)\right)\varepsilon_{12} \tag{B11}$$

while Γ of neuron i in the l-th hidden layer is given by

$$\begin{aligned}\Gamma_i^l = &\left(f'(s^1)\left(\left(\sum_j(\omega^1)_{ji}^{l+1}\right)^*(\Gamma^1)_i^{l+1}\right)\right)\varepsilon_1 + \left(f'(s^2)\left(\left(\sum_j(\omega^2)_{ji}^{l+1}\right)^*(\Gamma^2)_i^{l+1}\right)\right)\varepsilon_2\\ &+\left(f'(s^3)\left(\left(\sum_j(\omega^3)_{ji}^{l+1}\right)^*(\Gamma^3)_i^{l+1}\right)\right)\varepsilon_{12}\end{aligned} \tag{B12}$$

where

$$f'(s) = f'(s^1)\varepsilon_1 + f'(s^2)\varepsilon_2 + f'(s^3)\varepsilon_{12} = \frac{\partial f(s^1)}{\partial s^1}\varepsilon_1 + \frac{\partial f(s^2)}{\partial s^2}\varepsilon_2 + \frac{\partial f(s^3)}{\partial s^3}\varepsilon_{12} \tag{B13}$$

Appendix B.4 RGA-CNNs

As mentioned above, the basic structure of our proposed RGA-CNNs in the RGA domain is generally composed of an input layer, a convolution layer, a pooling layer, a RGA-MLP, and an output layer. Specifically, the RGA-MLP can be added between the last pooling layer and the output layer, that is to say it exists before the output layer, and correspondingly after the pooling layer. The convolution layer is utilized to extracted local feature of images, and the pooling layer is used to reduce the location sensitivity of features extracted from the convolution layer. Usually, a CNN can contain several convolution layers and pooling layers.

The process of our proposed RGA-CNNs can be summarized as follows.

we extend conventional convolution kernel in real domain to RGA domain. As shown in Figure B4, the input matrix and the convolution kernel are represented as RGA multivectors $A \in \left(\mathbb{G}_2^R\right)^{M\times M}$ and $K \in \left(\mathbb{G}_2^R\right)^{N\times N}$, respectively.

In the convolution layer, the output $B \in \left(\mathbb{G}_2^R\right)^{(M-N+1)\times(M-N+1)}$ is generated by convolution of input with convolution kernel K as follows

$$b = f\left(\sum \frac{ka}{|k|} + \varphi\right) \tag{B14}$$

where k, a, b stand for each pixel of K, A, B respectively, and $b = f(\cdot)$ has been defined by (A3). Then a feature matrix B is obtained and so called feature map. Since the convolution layer is composed of several feature maps, that is to say, p feature maps $B_1, B_2, \cdots, B_p$ can be generated through convolution of input A with p RGA convolution kernels $K_1, K_2, \cdots, K_p$.

The output B is fed to the pooling layer, which aims to reduce the position sensitivity of the features extracted by convolution layer. Here, we extend the conventional max pooling method in the real domain to RGA domain, and specifically it is denoted as

$$c_{ij} = \max_{(k,l)\in C_{ij}} b_{kl} \tag{B15}$$

where C_{ij} is the local patch in the output of pooling layer C, c_{ij} is the pixel of C_{ij}, and b is the pixel of the output of convolution layer B.

Appendix C Simulation results

Appendix C.1 Data sets

Appendix C.1.1 *3D geometrical shapes data sets*

To form the 3D geometrical shapes data sets, we generate 4 classes of 3D geometrical shapes, in which 4000 patterns are contained in each of class. In terms of our 3D geometrical shapes data sets, different shapes belong to different classes and different patterns of the same class are generated by the rotation of the first pattern with different angles, around the origin and by the translation with different vectors. Figure C1 shows the example 3D geometrical shapes with 4 classes and 4 patterns for each of class, used in the classification experiments.

Appendix C.1.2 *Color image data sets*

we include the CIFAR-10 dataset for color image classification experiments. The CIFAR-10 dataset comprises totally 60000 natural color images with 10 different classes and 6000 images for each of class, in which 50000 images are acted as the training set and the rest 1000 images are applied as the testing set. All the images in the dataset consists of 32×32 (= 1024 pixels), each of which has its color values represented by 24 bits (Red, Green and Blue components are represented by 8 bits). More details of the CIFAR-10 dataset can be found at http://www.cs.toronto.edu/ kriz/cifar.html.

Appendix C.2 Experimental Setup

The architectures of real-valued CNNs, QCNNs and the proposed RGA-CNNs are mainly similar and composed of an input layer, a convolution layer, a pooling layer, a MLP and an output layer, respectively. The number of convolution kernels in the convolution layer is set to 12 with the size of 5×5 and pooling size is 2×2. Specifically, the input data of size 32×32 becomes 28×28 after convolution with a 5×5 convolution kernel. After max pooling of 4×4, output of pooling layer becomes 6×6. For k class problem, output layer has k neurons. Outputs of pooling layer are converted to quaternion or RGA numbers for QCNNs and RGA-CNNs, respectively. We initialize the learning rate η to 0.1 for the three networks. All

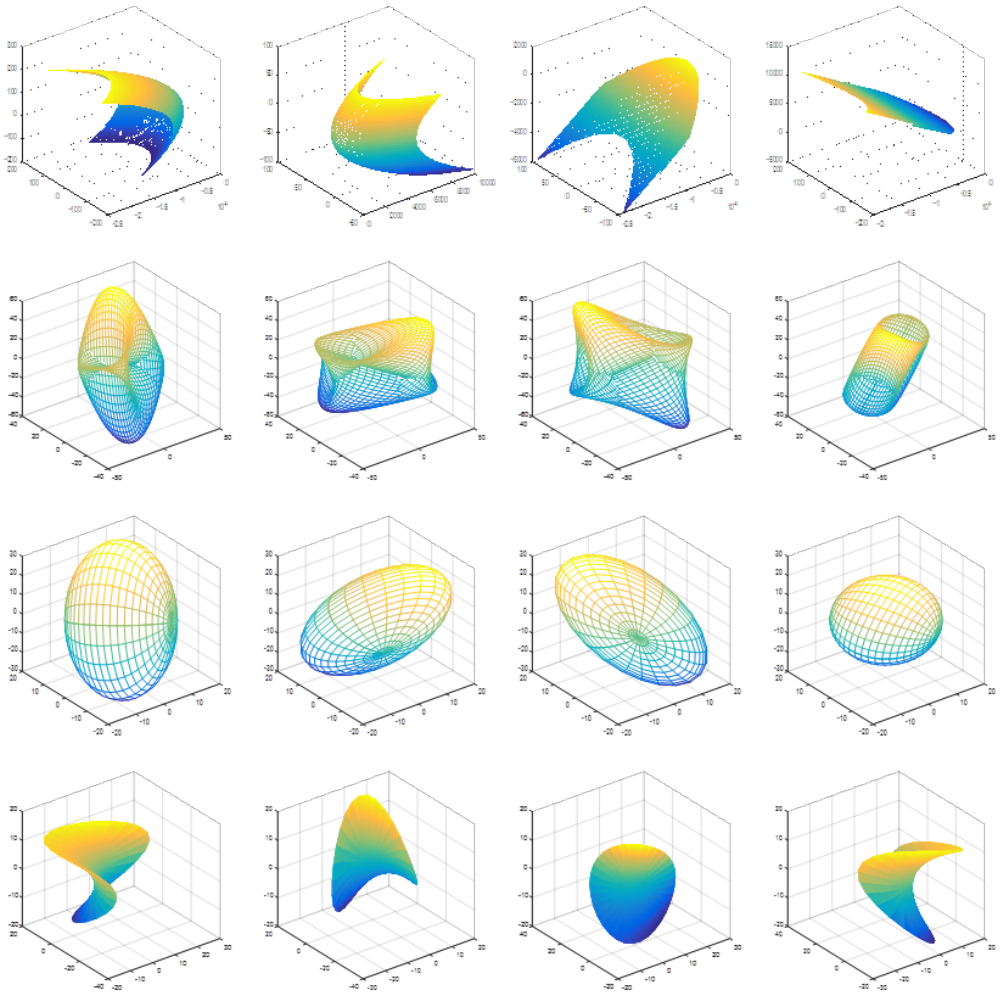

Figure C1 Four classes of plane curves.

the experiments in this paper are performed by using MATLAB on a computer with Intel(R) Core(TM) i5-3470 3.20GHz CPU and 4 GB memory. We choose sigmoid as the activation function, and max pooling by magnitude. The training error and the test accuracy are evaluated on the three networks for comparison experiments.

Appendix C.3 3D Geometrical Shapes Classification

Based on the 3D geometrical shapes database described above, we focus on analyzing the influence on the network performance, including the network depth, the convolution kernel size and the convolution kernel number. We choose 16000 3D geometrical shapes from 4 classes (where 3000 shapes and 1000 of each class are selected to construct the training dataset and the test dataset, respectively).

Appendix C.3.1 *The size of convolution kernels*

In order to explore the relationship between the performance of the proposed RGA-CNNs and the size of convolution kernels in the convolution layer, we conduct 3D geometrical shapes classification experiments in which the size of convolution kernels are set to 3×3 , 5×5, 7×7, 9×9 and 11×11. The training loss achieved by the proposed RGA-CNNs with different size of convolution kernels is depicted in Figure C2. And Table C1 shows the average training time of QCNNs and the proposed RGA-CNNs with different size of kernels.

However, much more pixel data are correspondingly read while processing the whole feature image with the increasing size of convolution kernels, which inevitably leads to higher training time of the network, and the lager the size of convolution kernel is, the higher training time it takes. And it is obvious that the average training time of the proposed RGA-CNNs is much lower than QCNNs, for the reason that RGA is commutative while quaternion is noncommutative. Thus, combine the effects of training errors and training times, we choose 5×5 as the most reasonable size of convolution kernels.

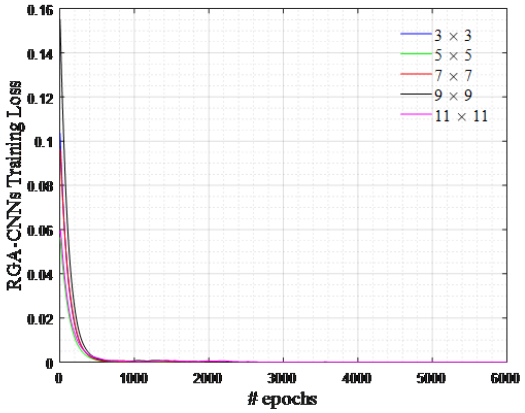

Figure C2 The training loss curves achieved by the proposed RGA-CNNs with different size of convolution kernels.

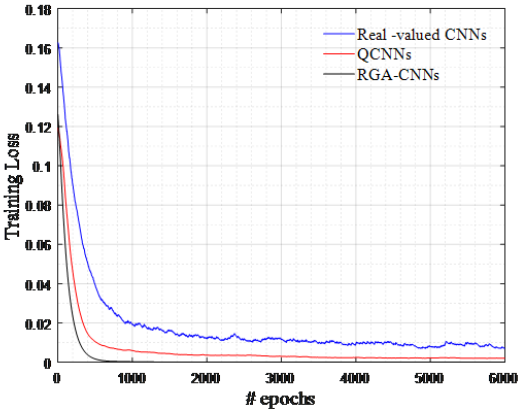

Figure C3 The training loss curves achieved by the real-valued CNNs, QCNNs and the proposed RGA-CNNs.

Table C1 Average training time (s) of QCNNs and the proposed RGA-CNNs with different size of convolution kernels.

The size of convolution kernels	3×3	5×5	7×7	9×9	11×11
QCNNs[14]	2509.9803	2773.0203	3110.7038	3519.2984	3921.6121
RGA-CNNs	1527.4333	1677.9276	1828.0485	1999.1379	2176.4174

Appendix C.3.2 *Comparisons of real-valued CNNs, QCNNs and the proposed RGA-CNNs*

The training loss curves achieved by real-valued CNNs and the proposed RGA-CNNs based on 3D geometrical shapes database are depicted in Figure C3 for comparison, in which the convolution kernels are all set to 5×5. We trained both the networks for 6000 iterations with a batch size of 10. As shown in Figure C3, it is clearly that the proposed RGA-CNNs achieve faster and more stable converge rate compared to real-valued CNNs and QCNNs. Furthermore, under the same iterations, the traditional real-valued CNNs shows a little bit higher training errors in contrast to QCNNs, and then the training errors achieved by the proposed RGA-CNNs achieve are lower than QCNNs.

To give a more precise comparison between the proposed RGA-CNNs, QCNNs and the traditional real-valued CNNs, the training errors with respect to the number of training iterations, and the results are listed in Table C2. As can be indicated in Table C2, when the number of training iterations go up from 1 to 400, the corresponding training errors of the traditional real-valued CNNs go down from 0.1626 to 0.0507 with reduction of nearly 0.1119, and the training errors of QCNNs decrease from 0.1192 to 0.0146 with reduction of nearly 0.1046, while that of the proposed RGA-CNNs decrease from 0.1261 to 0.0038 with reduction of nearly 0.1223. That is to say, with the first 400 iterations of training, the training errors of the proposed RGA-CNNs exhibit an extremely steeper downward trend compared with real-valued CNNs and QCNNs, commonly shown as a faster convergence rate. And as the iterations range from 1000 to 6000, the real-valued CNNs and QCNNs show approximately 0.0124 and 0.0040 decrease of training errors while the proposed RGA-CNNs reveal a nearly 0.0002 decrease, usually presented to be more stable.

Appendix C.4 Color Image Classification

In this experiment, 4 classes of color images in CIFAR-10 are used, 6000 color images of each class in training database are selected to form the training set and the test set is composed of 2000 color images of each class in test database. The

Table C2 The training errors of the proposed RGA-CNNs, QCNNs and real-valued CNNs with respect to training iterations.

Iterations	1	400	1000	2000	4000	6000
Real-valued CNNs[10]	0.1626	0.0507	0.0193	0.0124	0.0093	0.0069
QCNNs[14]	0.1192	0.0146	0.0060	0.0036	0.0024	0.0020
RGA-CNNs	0.1261	0.0038	2.4937×10^{-4}	6.1066×10^{-5}	2.0599×10^{-5}	1.3382×10^{-6}

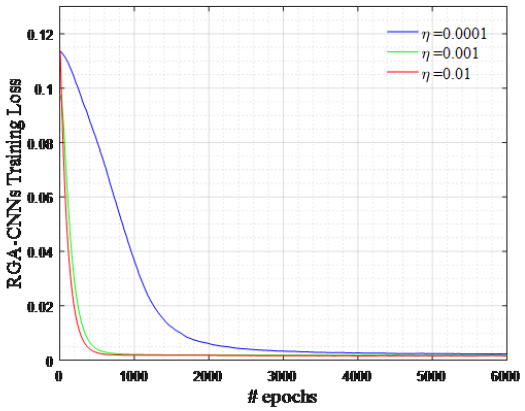

Figure C4 The training loss curves achieved by the proposed RGA-CNNs with different learning rate η.

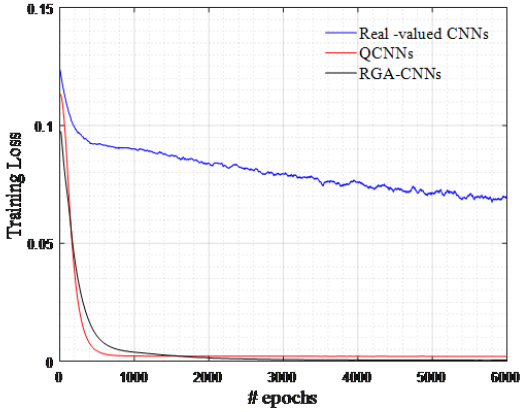

Figure C5 The training loss curves achieved by the real-valued CNNs, QCNNs and the proposed RGA-CNNs.

Table C3 The test accuracy of the real-valued CNNs, QCNNs and the proposed RGA-CNNs (%).

Methods	The first class	The second class	The third class	The fourth class	Total
Real-valued CNNs[10]	75.9	76.4	80.6	77.3	77.51
QCNNs[14]	85.8	87.0	88.3	88.4	87.4
RGA-CNNs	85.7	88.4	90.5	89.0	88.9

networks are trained so that the original input should be reconstructed at the output layer.

Appendix C.4.1 *Learning rate*

We conducted an experiment to discuss the effect of the learning rate η on the performance of the proposed RGA-CNNs. We correspondingly set the learning rate η to 0.0001, 0.001, and 0.01. As indicated in Figure C4, the proposed RGA-CNNs can achieves good convergence and the lowest training loss with different learning rates. When the learning rate η is set to 0.01, the proposed RGA-CNNs can achieve the best convergence. It is easily indicated that by adjusting the learning rate, the approximately optimal solution with respect to the optimization problem of training loss function can be obtained.

Appendix C.4.2 *Comparisons of real-valued CNNs, QCNNs and the proposed RGA-CNNs*

We evaluate the training error and the test accuracy on real-valued CNNs and the proposed RGA-CNNs for comparison experiments. For real-valued CNNs the learning rate is fixed at 0.1, for QCNNs and RGA-CNNs, the learning rate is 0.01. We trained both the networks for 6000 iterations with a batch size of 10, and the curves of training loss are shown in Figure C5.

As seen in the training loss in Figure C5, we conclude that both QCNNs and the proposed RGA-CNNs converged faster to less training loss than real-valued CNNs. The training loss curves of QCNNs and the proposed RGA-CNNs show stable convergence while real-valued CNNs fluctuates greatly and is very unstable during the training process.

Moreover, less training errors are achieved by QCNNs and the proposed RGA-CNNs compared with real-valued CNNs under the same iterations because of the holistically processing ways provided by quaternion and RGA frameworks. The traning loss curves of QCNNs and the proposed RGA-CNNs exhibit that the convergence and stability are almostly similar. With the increasing number of training iterations from about 200 to 1200, the corresponding training errors of QCNNs are a bit lower than the proposed RGA-CNNs. However, after 1200 iterations, the proposed RGA-CNNs achieve much lower training errors coampared to QCNNs.

For a more objective comparison, we list the test accuracy for CIFAR-10 color image datasets with real-valued CNNs, QCNNs and the proposed RGA-CNNs, in which the test accuracies of 4 classes are presented in Table C3. As indicated in Table C3, the most high test accuracy of real-valued CNNs reaches 80.6% with respect to the third class while that of the proposed RGA-CNNs goes to 90.5% also with regard to the third class. More generally, the total test accuracy obtained by RGA-CNNs is approximately 88.9 % while that of real-valued CNNs is only 74.3%. Moreover, it should be noted that the proposed RGA-CNNs shows a bit higher test accuracy than QCNNs, though the test accuracy of the first class is 0.1% lower. That is to say, the experimental results show that the proposed method achieves nearly the highest test accuracy, which verify the advantages of the proposed RGA-based CNNs.

Appendix C.5 Complexity Analysis

In this subsection, we will evaluate the efficiency of QCNNs and the proposed RGA-CNNs with respect to the computation time. The approximate computation time is calculated by using MATLAB on a computer with Intel(R) Core(TM) i5-3470 3.20GHz CPU and 4 GB memory. The results are shown in Table C4 with the computation time of QCNNs and the proposed RGA-CNNs based on 3D geometrical shapes and color images databsets. As shown in Table C4, the proposed RGA-CNNs efficiently achieves less computation time, which is nearly the half of QCNNs, for the reason that the RGA framework provides a powerful way to represent a color image as a multivector and process it in a holistic manner. In this way, the relationship of different channels in color image can be completely preserved and the network has been simplified due to the communicative multiplication of RGA.

Table C4 Computation time(s) of QCNNs and the proposed RGA-CNNs.

Datasets	QCNNs	RGA-CNNs
3D Geometrical Shapes	2772.5113	1678.2767
Color Images	3144.7837	1681.9981

References

1 Li Q, Peng Q, Yan C. Multiple VLAD Encoding of CNNs for Image Classification. Computing in Science Engineering, 2018, (99): 1–1
2 Zhang X, Zou J, He K, et al. Accelerating Very Deep Convolutional Networks for Classification and Detection. IEEE Transactions on Pattern Analysis Machine Intelligence, 2015, 38(10): 1943–1955
3 Gunn C G. A new approach to euclidean plane geometry based on projective geometric algebra. Mathematics, 2015, 704(4): 41–58
4 Wareham R J, Cameron J, Lasenby. Applications of conformal geometric algebra in computer vision and graphics. LNCS 3519, 2004: 329–349
5 Wang R , Shen M , Wang T , et al. L1-Norm Minimization for Multi-dimensional Signals Based on Geometric Algebra. Advances in Applied Clifford Algebras, 2019, 29(2).
6 Ebling J, Scheuermann G. Clifford Fourier Transform on Vector Fields. IEEE Transactions on Visualization Computer Graphics, 2005
7 Wang R, Zhou Y X, Jin Y L, et al. Sparse fast Clifford Fourier transform. Frontiers of Information Technology Electronic Engineering, 2017, 18(8): 1131–1141
8 Li Y, Liu W, Li X, et al. GA-SIFT: A new scale invariant feature transform for multispectral image using geometric algebra. Information Sciences, 2014: 559–572
9 Li H B, Cao Y. On Geometric Theorem Proving with Null Geometric Algebra. Guide to Geometric Algebra in Practice, 2011: 195–215
10 Li H B. Symbolic Geometric Reasonong with Advanced Invariant Algebra. International Conference on Mathematical Aspects of Computer and Information Sciences, 2015: 35–49
11 Wang R, Shen M M, Cao W M, Multivector Sparse Representation for Multispectral Images Using Geometric Algebra. IEEE Access. 2019, 7: 12755 - 12767
12 Wang R, Shi Y J, Cao W M. GA-SURF: A new Speeded-Up robust feature extraction algorithm for multispectral images based on geometric algebra. Pattern Recognition Letters, 2018.
13 Shen M M, Wang R, Cao W M, Joint sparse representation model for multi-channel image based on reduced geometric algebra. IEEE Access. Image Process, 2018, 6: 24213–24223
14 Zhang F, Cai N, Wu J, et al. Image denoising method based on a deep convolution neural network. Iet Image Processing, 2018, 12(4): 485–493
15 Kominami Y, Ogawa H, and Murase K. Convolutional neural networks with multi-valued neurons. International Joint Conference on Neural Networks IEEE, 2017: 2673–2678