Journal of Computers Vol. 36 No. 1, February 2025, pp. 75-88
https://doi.org/10.63367/19911599202502360100 6

Color Image Recognition Models based on Quaternion Residual Networks

Ya-Mei Xu*, Zhi-Yao Wang, and Chan-Fei Wang

School of Computer and Communication, Lanzhou University of Technology,
Lanzhou 730050, Gansu Province, China
yameixu@126.com, wangzhy_djf@163.com, wangchanfei@163.com

Received 7 May 2024; Revised 12 September 2024; Accepted 1 October 2024

Abstract. Color image recognition based on real-number neural networks can encounter challenges such as redundant self-information and insufficient mutual information in ternary color feature extraction. These issues can lead to inaccurate perception of color information by the model. Furthermore, deep neural networks with over 100 layers may experience a decline in model performance due to the large number of parameters. To overcome these challenges, this study extends real-number neural networks into the quaternion domain to construct a series of color image recognition models based on quaternion residual networks. The model designs include a lightweight structure, quaternion residual blocks, quaternion bilateral convolution, and quaternion L_1 regularization. The quaternion residual network has half the number of parameters as the corresponding real-number network, and the lightweight design for parameters helps solve the computational complexity problem of deep networks. Moreover, the quaternion bilateral convolution can reasonably preserve the image color structure, and the quaternion L_1 regularization can effectively sparse the training parameters, thereby mitigating network overfitting. Experimental results confirmed that the proposed models can improve image recognition rates while reducing algorithmic complexity compared to the corresponding real-number networks.

Keywords: color image recognition, quaternion, residual network, lightweight

1 Introduction

In the mid-19th century, Hamilton created quaternion algebra [1], an extension of complex numbers. Quaternions are hyper-complex numbers, containing a real part and three imaginary parts. These numbers represent four-dimensional vectors with one real part and three imaginary parts, unlike complex numbers, which represent two-dimensional vectors using one real and one imaginary part. Quaternions are extensively utilized in various fields, such as robotic space exploration [2], three-dimensional human motion modeling [3], and computer graphics [4]. Their unique structure makes them highly effective for computations and modeling of image graphics in multi-dimensional spaces.

Image recognition is a crucial technology utilized in computer vision, pattern recognition, machine learning, and other technical fields. In recent years, significant progress in image recognition has been driven by convolutional neural networks (CNNs), leading to the development of several prominent models such as ShuffleNet [5], Visual Geometry Group (VGG) [6], AlexNet [7], GoogLeNet [8], residual network (ResNet) [9], and the recently proposed PathNet [10]. Among these networks, ResNet can effectively mitigate performance degradation caused by gradient disappearance and gradient explosion [11] during the training of deep neural networks, thus attracting the attention of researchers.

Despite advancements in CNN architectures, challenges remain in accurately representing color information in color images. Real-number CNNs often suffer from redundant self-information and insufficient mutual information during ternary color feature extraction. Furthermore, deep neural networks with over 100 layers, such as ResNet101 and ResNet152, may experience reduced performance compared to their corresponding shallower networks due to the excessive number of parameters.

To address the aforementioned issues, we enhance the utilization of color information in color images by describing image color structure in the quaternion domain. This approach allows for a more accurate representation and perception of color information during image processing tasks. The bilateral multiplication of a general qua-

* Corresponding Author

ternion and a pure imaginary quaternion produces another pure imaginary quaternion. Leveraging this principle, we design a quaternion residual network structure that preserves the three-dimensional nature of image features. In which the ternary colors in color images are represented as three imaginary axes of pure imaginary quaternions. Moreover, a quaternion bilateral convolution is introduced to eliminate redundant ternary color information while preserving mutual color information, utilizing the properties of quaternion bilateral multiplication. We also design a quaternion L_1 regularization method to address the challenges of increased algorithmic complexity. This method computes the amplitude of each quaternion element and applies L_1 norms to these amplitudes. In this framework, sparsity is encouraged in the trained model parameters, effectively reducing overfitting in quaternion deep residual networks.

In summary, we propose a novel quaternion residual network structure, termed QResNet, for color image recognition. We develop a series of models based on QResNet, including QResNet-18, QResNet-34, QResNet-50, QResNet-101, and QResNet-152. These models feature a lightweight architecture with quaternion residual blocks, quaternion bilateral convolution, quaternion pooling, quaternion batch normalization, and quaternion L1 regularization. Compared to traditional real-number networks, QResNet models offer a more efficient representation of ternary color properties and significantly reduce the number of parameters, facilitating the effective deployment of deep residual networks. This approach enhances recognition accuracy and stability for color images while reducing algorithmic complexity.

The rest of this paper is organized as follows: Section 2 reviews related works, while Section 3 introduces the quaternion algebra underlying the proposed model. Section 4 details the proposed QResNet models, and Section 5 presents the experimental results. Finally, Section 6 concludes the paper.

2 Related Works

As ResNet continues to deliver excellent performance and gains popularity, it has become a foundational framework in many research fields. Numerous modified versions have emerged in the image recognition domain. In 2017, G. Huang et al. [12] proposed the Dense Convolutional Network (DenseNet), which enhances ResNet by connecting each layer to every other layer in a feed-forward manner, thereby improving the network's anti-overfitting capability. In 2019, C.-Y. Yu et al. [13] introduced S-DenseNet, a compact variant of DenseNet that enhances feature extraction by replacing standard convolution with group convolution. In 2021, D. O'Neill et al. [14] presented Sparse DenseNet, which utilizes a genetic algorithm to explore the space between a standard feed-forward network and DenseNet, identifying and removing redundant skip connections to simplify the network's structure.

In recent years, researchers have been exploring the integration of quaternion feature extraction with CNN networks, leveraging the rapid advancements in neural network algorithms [15]. In 2018, X.-Y. Zhu et al. [16] introduced quaternion convolutional layers and quaternion fully connected layers, constructing a quaternion convolutional neural network (QCNN) based on the VGG-S architecture [17]. This QCNN achieved an image recognition accuracy of approximately 76.95% on the Oxford 102 Flowers dataset [18]. However, the study did not extend the quaternion domain to components such as batch normalization and dropout layers. In 2019, Q.-L. Yin et al. [19] improved the QCNN model by incorporating an attention mechanism, which enabled the model to focus on critical regions within the images. This enhancement led to an image recognition accuracy of about 85.37% on the CIFAR-10 dataset [20].

The applications discussed above extend on real-number network in the quaternion domain. They have proven that the quaternion networks can reduce the number of parameters and improve classification accuracy, compared to real-number networks. However, there is limited literature on quaternion networks for residual networks, primarily because the three primary colors in an image do not align with the four dimensions required by quaternions. In 2018, C. J. Gaudet et al. [21] introduced a quaternion convolutional kernel for residual networks using unilateral quaternion multiplication, achieving accuracies of approximately 94.56% and 73.99% on the CIFAR-10 and CIFAR-100 datasets [20], respectively. To meet the four-dimensions requirement for quaternion multiplication, this algorithm incorporates grayscale information to create the fourth dimension. This strategy increases the number of convolution kernel parameters, which in turn raises algorithm complexity and exacerbates overfitting

Based on the above discussion, we investigate the effective integration of quaternions with deep residual networks and propose a series of residual network models enhanced by quaternions. The main contributions of this paper are as follows:

(1) We introduce a quaternion bilateral convolution kernel with 4 parameters specifically for color images. This kernel maintains the three-dimensional nature of image features, removes redundancy in ternary color information, and preserves color mutual information. Based on this quaternion convolution kernel, we propose a novel quaternion residual network structure termed QResNet.

(2) We propose a novel quaternion L1 regularization method for the QResNet structure. This regularization technique promotes parameter sparsity in the network, effectively reducing overfitting during model training.

(3) We develop a series of color image recognition models based on QResNet, including QResNet-18, QResNet-34, QResNet-50, QResNet-101, and QResNet-152. These models feature lightweight architectures that reduce algorithmic complexity compared to their real-number counterparts. Additionally, we introduce other model designs such as quaternion pooling and quaternion batch normalization.

3 Quaternion Algebra

A quaternion q in the quaternion space $\mathbb{Q}$ comprises a real part and three imaginary parts, represented as follows:

$$q = q_{\mathrm{r}} + q_{\mathrm{i}} i + q_{\mathrm{j}} j + q_{\mathrm{k}} k, \quad q \in \mathbb{Q}, \quad q_{\mathrm{r}}, q_{\mathrm{i}}, q_{\mathrm{j}}, q_{\mathrm{k}} \in \mathbb{R} \ , \tag{1}$$

where q_{r} is the real part, q_{i}, q_{j}, q_{k} denote the three imaginary parts, and $\mathbb{R}$ is the real-number space. Additionally, i, j, and k denote the three imaginary axes, linearly independent, which must obey the Hamiltonian rule [1]:

$$i^2 = j^2 = k^2 = ijk = -1, \quad ij = -ji = k, \quad jk = -kj = i, \quad ki = -ik = j \ . \tag{2}$$

A quaternion with a real part of 0 is called a pure imaginary quaternion, and its spatial representation is denoted as $\hat{\mathbb{Q}}$:

$$\hat{q} = 0 + q_{\mathrm{i}} i + q_{\mathrm{j}} j + q_{\mathrm{k}} k, \quad q_{\mathrm{i}}, q_{\mathrm{j}}, q_{\mathrm{k}} \in \mathbb{R}, \quad q \in \hat{\mathbb{Q}} \ . \tag{3}$$

We consider two quaternions, $p = p_{\mathrm{r}} + p_{\mathrm{r}} i + p_{\mathrm{j}} j + p_{\mathrm{k}} k$ and $q = q_{\mathrm{r}} + q_{\mathrm{r}} i + p_{\mathrm{j}} j + q_{\mathrm{k}} k$. The standard operation rules for quaternions are as follows.

Add and subtract:

$$p \pm q = (p_{\mathrm{r}} \pm q_{\mathrm{r}}) + (p_{\mathrm{i}} \pm q_{\mathrm{i}}) i + (p_{\mathrm{j}} \pm q_{\mathrm{j}}) j + (p_{\mathrm{k}} \pm q_{\mathrm{k}}) k \ . \tag{4}$$

Multiply with a real-number:

$$\lambda q = \lambda q_{\mathrm{r}} + \lambda q_{\mathrm{i}} i + \lambda q_{\mathrm{j}} j + \lambda q_{\mathrm{k}} k \ . \tag{5}$$

Multiplication of two quaternions (Hamiltonian product):

$$\begin{aligned} pq &= (p_{\mathrm{r}} + p_{\mathrm{i}} i + p_{\mathrm{j}} j + p_{\mathrm{k}} k)(q_{\mathrm{r}} + q_{\mathrm{i}} i + q_{\mathrm{j}} j + q_{\mathrm{k}} k) \\ &= (p_{\mathrm{r}} q_{\mathrm{r}} - p_{\mathrm{i}} q_{\mathrm{i}} - p_{\mathrm{j}} q_{\mathrm{j}} - p_{\mathrm{k}} q_{\mathrm{k}}) + (p_{\mathrm{r}} q_{\mathrm{i}} + p_{\mathrm{i}} q_{\mathrm{r}} + p_{\mathrm{j}} q_{\mathrm{k}} - p_{\mathrm{k}} q_{\mathrm{j}}) i \\ &\quad + (p_{\mathrm{r}} q_{\mathrm{j}} - p_{\mathrm{i}} q_{\mathrm{k}} + p_{\mathrm{j}} q_{\mathrm{r}} + p_{\mathrm{k}} q_{\mathrm{i}}) j + (p_{\mathrm{r}} q_{\mathrm{k}} + p_{\mathrm{i}} q_{\mathrm{j}} - p_{\mathrm{j}} q_{\mathrm{i}} + p_{\mathrm{k}} q_{\mathrm{r}}) k \end{aligned} \ . \tag{6}$$

Conjugate:

$$q^* = q_{\mathrm{r}} - q_{\mathrm{i}} i - q_{\mathrm{j}} j - q_{\mathrm{k}} k \ . \tag{7}$$

Amplitude:

$$|q| = \sqrt{qq^*} = \sqrt{q_{\mathrm{r}}^2 + q_{\mathrm{i}}^2 + q_{\mathrm{j}}^2 + q_{\mathrm{k}}^2} \ . \tag{8}$$

4 QResNet Models Design

4.1 Quaternionic Bilateral Convolution

The bilateral multiplication of a pure imaginary quaternion, denoted as $\hat{q}$, and a general quaternion, denoted as w, is expressed as follows:

$$\begin{aligned}\hat{p} = wqw^* &= (w_{\mathrm{r}} + w_{\mathrm{i}}i + w_{\mathrm{j}}j + w_{\mathrm{k}}k)(0 + q_{\mathrm{i}}i + q_{\mathrm{j}}j + q_{\mathrm{k}}k)(w_{\mathrm{r}} - w_{\mathrm{i}}i - w_{\mathrm{j}}j - w_{\mathrm{k}}k) \\ &= [(w_{\mathrm{r}}^2 + w_{\mathrm{i}}^2 - w_{\mathrm{j}}^2 - w_{\mathrm{k}}^2)q_{\mathrm{i}} + 2(w_{\mathrm{i}}w_{\mathrm{j}} - w_{\mathrm{r}}w_{\mathrm{k}})q_{\mathrm{j}} + 2(w_{\mathrm{r}}w_{\mathrm{j}} + w_{\mathrm{i}}w_{\mathrm{k}})q_{\mathrm{k}}]i \\ &+ [2(w_{\mathrm{r}}w_{\mathrm{k}} + w_{\mathrm{i}}w_{\mathrm{j}})q_{\mathrm{i}} + (w_{\mathrm{r}}^2 - w_{\mathrm{i}}^2 + w_{\mathrm{j}}^2 - w_{\mathrm{k}}^2)q_{\mathrm{j}} + 2(w_{\mathrm{j}}w_{\mathrm{k}} - w_{\mathrm{r}}w_{\mathrm{i}})q_{\mathrm{k}}]j \\ &+ [2(w_{\mathrm{i}}w_{\mathrm{k}} - w_{\mathrm{r}}w_{\mathrm{j}})q_{\mathrm{i}} + 2(w_{\mathrm{r}}w_{\mathrm{i}} + w_{\mathrm{j}}w_{\mathrm{k}})q_{\mathrm{j}} + (w_{\mathrm{r}}^2 - w_{\mathrm{i}}^2 - w_{\mathrm{j}}^2 + w_{\mathrm{k}}^2)q_{\mathrm{k}}]k\end{aligned} \ , \tag{9}$$

where w^* is the conjugate of w, as described in (8). The computation result $\hat{p}$ in (9) shows that it is a pure imaginary quaternion because its real part is 0.

Therefore, leveraging the above property of pure imaginary quaternions, we consider a color image of size $N \times N'$, denoted as $\hat{\boldsymbol{H}}$, as a pure imaginary quaternion matrix, represented as follows:

$$\hat{\boldsymbol{H}} = \left\{\hat{h}_{nn'}\right\} \in \hat{\mathbb{Q}}^{N \times N'}, \quad n = 1,...,N, \quad n' = 1,...,N' \ , \tag{10}$$

where the element $\hat{h}_{nn'}$ of the matrix $\hat{\boldsymbol{H}}$ is a pure imaginary quaternion, with its three imaginary parts representing the R, G, and B color data of the pixel.

By utilizing (9), we define the quaternion bilateral convolution for the QResNet model. First, we define a quaternion bilateral convolution kernel of size $L \times L'$:

$$\boldsymbol{w} = \left\{w_{ll'}\right\} \in \mathbb{Q}^{L \times L'}, \quad l = 1,...,L, \quad l' = 1,...,L' \ , \tag{11}$$

where the element $w_{ll'}$ of the quaternion convolution kernel $\boldsymbol{w}$ is a general quaternion. Then, we assume the row and column strides for the image convolution are denoted as t and t', respectively. In this case, the quaternion bilateral convolution operation is defined as follows:

$$\boldsymbol{w} \otimes \hat{\boldsymbol{H}} \otimes \boldsymbol{w}^* = \left\{\hat{f}_{mm'}\right\} = \boldsymbol{F} \in \hat{\mathbb{Q}}^{\left(\left\lfloor \frac{N-L}{t} \right\rfloor + 1\right) \times \left(\left\lfloor \frac{N'-L'}{t'} \right\rfloor + 1\right)} \ , \tag{12}$$

$$\hat{f}_{mm'} = \sum_{l=0}^{L-1} \sum_{l'=0}^{L'-1} \frac{1}{|w_{ll'}|} w_{ll'} \ \hat{h}_{(mt+l)(m't'+l')} w_{ll'}^*, \quad m = 1,...,\left\lfloor \frac{N-L}{t} \right\rfloor + 1, \quad m' = 1,...,\left\lfloor \frac{N'-L'}{t'} \right\rfloor + 1 \ , \tag{13}$$

where, $|w_{ll'}|$ is the magnitude of $w_{ll'}$, computed according to (8), $\lfloor \cdot \rfloor$ is a downward rounding operation.

As shown in the above equations, multiple layers of quaternion bilateral convolution operations are stacked sequentially, as each convolutional layer outputs a pure imaginary quaternion matrix.

For complex numbers, the rotation of a vector is described as the multiplication of complex numbers in a two-dimensional plane. Similarly, for quaternions, the bilateral multiplication of a pure imaginary quaternion and a general quaternion describes the scaling transformation and rotational transformation of a three-dimensional vector [4]. To illustrate the impact of bilateral convolution on the scaling and rotational transformations of an image, we express an element of the quaternion convolution kernel $w = w_{\mathrm{r}} + w_{\mathrm{i}}i + w_{\mathrm{j}}j + w_{\mathrm{k}}k$ as a quaternion characterized by its magnitude s and three angles θ, ϕ, φ, as follows:

$$w = s\{\cos\theta + \sin\theta[i\cos\phi + \sin\phi(j\cos\varphi + k\sin\varphi)]\} \ , \tag{14}$$

where $s \in \mathbb{R}^+$ and θ, ϕ, $\varphi \in [-\pi/2, -\pi/2]$. Each convolution kernel element w has four trainable parameters: s, θ, ϕ and φ. If we set $\phi = \arg\cos(\sqrt{3}/3)$ and $\varphi = \pi / 4$, the three imaginary axes values of the quaternion w become equal. In this case, (14) simplifies to a form with two trainable parameters, s and θ, as follows:

$$w = s\left[\cos\theta + \frac{\sqrt{3}}{3}\sin\theta(i + j + k)\right] . \tag{15}$$

If we set $\theta = \arg\cos(1/2)$, the four axes values of this quaternion become equal, and (15) further simplifies to a form with a single trainable parameter, s, as follows:

$$w = \frac{s}{2}(1 + i + j + k) \ . \tag{16}$$

The three convolution kernel elements described in (14)-(16) provide three degrees of freedom for training the QResNet model. Ablation experiments revealed the following insights: First, the model achieves the highest test recognition accuracy when the quaternion convolution kernel elements use the four trainable parameters s, θ, ϕ and φ described in (14). However, this configuration also results in the highest network complexity. Second, when the quaternion convolution kernel elements employ the two trainable parameters s, θ, as described in (15), the model requires approximately half the number of parameters compared to the corresponding real-number network. This approach significantly reduces the algorithm complexity with only a slight performance decrease. Lastly, the model's recognition accuracy noticeably declines when using kernel elements with a single trainable parameter s, as described in (16). Therefore, considering efficiency and complexity based on theoretical analysis and experimental results, we selected the two trainable parameters s and θ described in (15) as the quaternion convolution kernel elements for the QResNet models.

4.2 Quaternionic Residual Network Structure

Table 1. Network architecture of QResNet models (Image size: 32 × 32)

Network layer	Output size	Stride	Convolutional kernel				
			QResNet18	QResNet34	QResNet50	QResNet101	QResNet152
conv1	3×32×32, 16	1	3×3, 16				
		1	3×3, max pooling				
conv2_x	3×32×32, 16	1	$\begin{bmatrix}3\times3,\ 16\\3\times3,\ 16\end{bmatrix}\times2$	$\begin{bmatrix}3\times3,\ 16\\3\times3,\ 16\end{bmatrix}\times3$	$\begin{bmatrix}1\times1,\ 16\\3\times3,\ 16\\1\times1,\ 64\end{bmatrix}\times3$	$\begin{bmatrix}1\times1,\ 16\\3\times3,\ 16\\1\times1,\ 64\end{bmatrix}\times3$	$\begin{bmatrix}1\times1,\ 16\\3\times3,\ 16\\1\times1,\ 64\end{bmatrix}\times3$
conv3_x	3×16×16, 32	2	$\begin{bmatrix}3\times3,\ 32\\3\times3,\ 32\end{bmatrix}\times2$	$\begin{bmatrix}3\times3,\ 32\\3\times3,\ 32\end{bmatrix}\times4$	$\begin{bmatrix}1\times1,\ 32\\3\times3,\ 32\\1\times1,\ 128\end{bmatrix}\times4$	$\begin{bmatrix}1\times1,\ 32\\3\times3,\ 32\\1\times1,\ 128\end{bmatrix}\times4$	$\begin{bmatrix}1\times1,\ 32\\3\times3,\ 32\\1\times1,\ 128\end{bmatrix}\times8$
conv4_x	3×8×8, 64	2	$\begin{bmatrix}3\times3,\ 64\\3\times3,\ 64\end{bmatrix}\times2$	$\begin{bmatrix}3\times3,\ 64\\3\times3,\ 64\end{bmatrix}\times6$	$\begin{bmatrix}1\times1,\ 64\\3\times3,\ 64\\1\times1,\ 256\end{bmatrix}\times6$	$\begin{bmatrix}1\times1,\ 64\\3\times3,\ 64\\1\times1,\ 256\end{bmatrix}\times23$	$\begin{bmatrix}1\times1,\ 64\\3\times3,\ 64\\1\times1,\ 256\end{bmatrix}\times36$
conv5_x	3×4×4, 128	2	$\begin{bmatrix}3\times3,\ 128\\3\times3,\ 128\end{bmatrix}\times2$	$\begin{bmatrix}3\times3,\ 128\\3\times3,\ 128\end{bmatrix}\times3$	$\begin{bmatrix}1\times1,\ 128\\3\times3,\ 128\\1\times1,\ 512\end{bmatrix}\times3$	$\begin{bmatrix}1\times1,\ 128\\3\times3,\ 128\\1\times1,\ 512\end{bmatrix}\times3$	$\begin{bmatrix}1\times1,\ 128\\3\times3,\ 128\\1\times1,\ 512\end{bmatrix}\times3$
Pooling	3×1×1, 128	global average pooling, dropout					
Output	number of categories						

We constructed five QResNet models: QResNet-18, QResNet-34, QResNet-50, QResNet-101, and QResNet-152, using quaternion bilateral convolution. Table 1 presents the specific network architecture parameters for each model. Square brackets in the table are used to denote residual blocks.

The specific steps for setting the network parameters of these models are as follows.

(1) The Conv1 layer adopts a 3×3 kernel size with 64 filters, based on the structural parameters of the real-number ResNet-18 network. The total number of parameters for this layer is calculated as 3×3×64 = 576.

(2) We initialize the quaternion network with the same number of parameters as the real-number network. For the QResNet-18 model, using quaternion convolution kernel elements with two trainable parameters s and θ, as described in (15), the number of filters in the Conv1 layer can be set to 32. This configuration yields 3×3×2×32 = 576 parameters, maintaining equivalence with the real-number network.

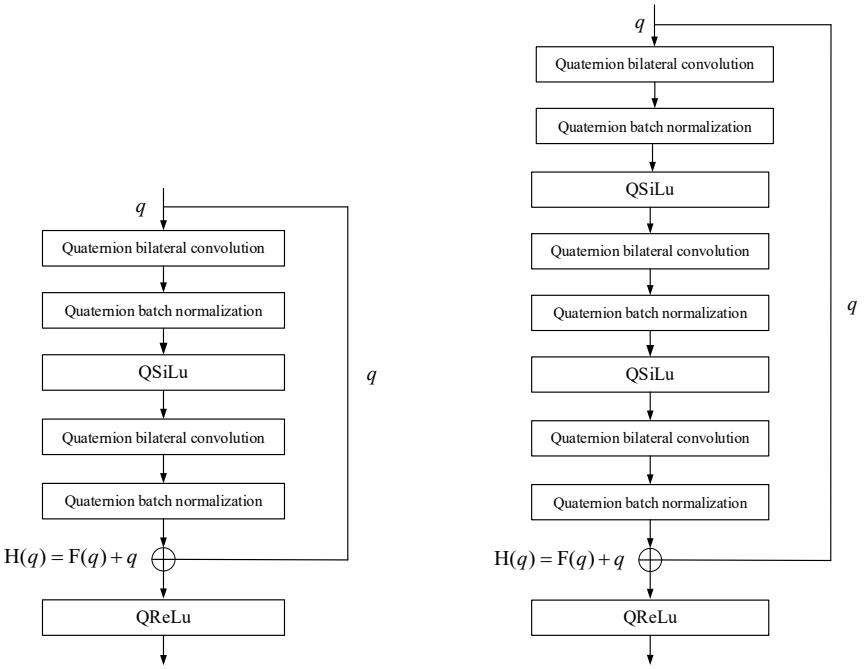
(3) To exploit the enhanced feature extraction capability of quaternion bilateral convolution, we further reduce the number of quaternion convolution filters. Setting the Conv1 layer to 16 filters results in 3×3×2×16=288 parameters, effectively halving the parameter count compared to the corresponding real-number network layer.

(4) This parameter reduction approach is systematically applied to the remaining convolutional layers and other QResNet models, leading to the final network structures outlined in Table 1.

4.3 QResNet Residual Blocks

The residual blocks in the QResNet models are designed as an improvement over the real-number residual blocks [9], tailored specifically for quaternion deep residual networks. Unlike the Rectified Linear Unit (ReLU) activation function, the Sigmoid Linear Unit (SiLU) activation function offers smoother and more continuous characteristics, which help mitigate prevent neuron death. As a result, the SiLU activation function is used in the residual blocks of our QResNet models to improve model performance and training efficiency. Additionally, the ReLU activation function is retained at the output of the entire residual block to prevent gradient vanishing.

Fig. 1 shows the residual blocks in the QResNet models. The QResNet-18 and QResNet-34 networks utilize an improved BasicBlock structure as shown in Fig. 1(a), where each stage consists of two BasicBlocks. In contrast, the QResNet-50, QResNet-101, and QResNet-152 networks adopt an improved BottleNeck structure as shown in Fig. 1(b), where each stage consists of three BottleNecks. At the end of the network, the final feature maps undergo a global average pooling to reduce spatial dimensions to a single value. Subsequently, these feature maps pass through a dropout layer followed by the output layer, as detailed in Table 1.



(a) Improved BasicBlock structure (b) Improved BottleNeck structure

Fig. 1. QResNet residual blocks

The QResNet model outputs a purely imaginary quaternion matrix after the bilateral convolution, allowing a single activation function to process the three imaginary channels simultaneously. The QReLu and QSiLu activation functions are defined as follows:

$$\mathrm{QReLu}(\hat{\boldsymbol{q}}) = \mathrm{ReLu}(\boldsymbol{q}_{\mathrm{i}})i + \mathrm{ReLu}(\boldsymbol{q}_{\mathrm{j}})j + \mathrm{ReLu}(\boldsymbol{q}_{\mathrm{k}})k \ , \tag{17}$$

$$\mathrm{QSiLu}(\hat{\boldsymbol{q}}) = \mathrm{SiLu}(\boldsymbol{q}_{\mathrm{i}})i + \mathrm{SiLu}(\boldsymbol{q}_{\mathrm{j}})j + \mathrm{SiLu}(\boldsymbol{q}_{\mathrm{k}})k \ . \tag{18}$$

4.4 Quaternion L1 Regularization

Real-number neural networks typically employ the L_2 norm [17] as a regularizer. However, for quaternion weights, direct norm calculation is infeasible due to the presence of imaginary components. To address this, existing literature treats the four components of quaternion weights as separate real numbers and applies regularization methods like L_1, L_2, or $L_{1/2}$ norms to these real-number components [9, 22-24]. However, these methods quadruple the parameter count and introduce redundancy among the components. To overcome these issues, we propose a novel quaternion L_1 regularization method. This approach computes the magnitude of each quaternion parameter and then combines all the magnitudes to calculate the L_1 norm, which is given by,

$$\mathrm{L}_1(\boldsymbol{\sigma}) = \sum_{w \in \boldsymbol{\sigma}} \left| l(w) \right| \ . \tag{19}$$

In this method, the magnitude $|l(w)|$ of the quaternion weight w is calculated using (8). This approach allows the network to learn fewer quaternion weights, thereby reducing the number of parameters and lowering the model complexity.

The parameter optimization process incorporates quaternion L_1 regularization into the network training, and it is expressed as follows:

$$\boldsymbol{\sigma}^* = \arg\min_{\boldsymbol{\sigma}} \frac{1}{T} \sum_{t=1}^{T} \xi(\boldsymbol{y}^{(t)}, H(\boldsymbol{x}^{(t)}; \boldsymbol{\sigma})) + \lambda \, \mathrm{L}_1(\boldsymbol{\sigma}) \ , \tag{20}$$

where $\xi(\cdot)$ denotes the loss function, T is the number of training samples in a batch, $H(\cdot)$ represents the quaternion residual network to be learned, and $\boldsymbol{\sigma}$ denotes its network parameters.

4.5 Quaternion Pooling

In quaternion residual networks, performing max pooling on individual channels, as done in real-number networks, is not meaningful because it would mix the values of different color layers, leading to a loss of color context information. Instead, we apply pooling by maximizing the magnitudes of pixels within a given submatrix. This approach preserves color context information and achieves good recognition results. For a pooling block $\hat{\boldsymbol{q}}^{K \times K'}$ in the quaternion output matrix with a size of $K \times K'$, the result of magnitude-max pooling is given by,

$$\hat{q}_{\max} = \arg\max_{\hat{q} \in \hat{\boldsymbol{q}}^{K \times K'}} \left| \hat{q} \right| \ , \tag{21}$$

where $|\hat{q}|$ represents the magnitude of the quaternion $\hat{q}$ as described in (8).

4.6 Quaternion Batch Normalization

Quaternion batch normalization (QBN) differs from real-valued batch normalization in how it calculates the mean and variance. Given T as the number of training samples in a batch, the mean (QE) and variance (QV) of

all the purely imaginary quaternion matrices $\hat{\boldsymbol{M}}_{\mathrm{q}} = \{\hat{\boldsymbol{q}}_1, ...\boldsymbol{q}_T\}$ are calculated as follows:

$$QE(\hat{\boldsymbol{M}}_{\mathrm{q}}) = \frac{1}{T}\sum_{t=1}^{T}\hat{\boldsymbol{q}}_t \quad , \tag{22}$$

$$QV(\hat{\boldsymbol{M}}_{\mathrm{q}}) = \frac{1}{T}\sum_{t=1}^{T}(\hat{\boldsymbol{q}}_t - QE(\hat{\boldsymbol{M}}_{\mathrm{q}}))\ (\hat{\boldsymbol{q}}_t - QE(\hat{\boldsymbol{M}}_{\mathrm{q}}))^{*} \cdot \tag{23}$$

Therefore, the QBN of the quaternion matrix $\hat{\boldsymbol{q}}_t \in \hat{\boldsymbol{M}}_{\mathrm{q}}$ is defined as follows:

$$QBN(\hat{\boldsymbol{q}}_t) = \gamma\left(\frac{\hat{\boldsymbol{q}}_t - QE(\hat{\boldsymbol{M}}_{\mathrm{q}})}{\sqrt{QV(\hat{\boldsymbol{M}}_{\mathrm{q}}) + \varepsilon}}\right) + \hat{\boldsymbol{\beta}}, \quad t = 1, ..., T \quad , \tag{24}$$

where γ represents the scaling factor as a scalar, $\hat{\boldsymbol{\beta}}$ represents the shifting scale as a purely imaginary quaternion matrix, and a small non-zero value ε is added for numerical stability. Both γ and $\hat{\boldsymbol{\beta}}$ are trainable parameters that are updated during the training process, adapting to changes in the network weights.

5 Experiments

5.1 Experimental Setup

We conducted experiments on the Cifar-10, Cifar-100 [18], and Oxford 102 Flowers [20] datasets to evaluate the effectiveness of the QResNet models. The Cifar-10 and Cifar-100 datasets consist of color images with 10 and 100 categories, respectively, each containing 50,000 training samples and 10,000 test samples. Model hyperparameter tuning was performed on the training set. The Oxford 102 flowers dataset contains color images of flowers across 102 categories, comprising a total of 8189 images. From the dataset, 6849 images were selected for training, 910 images for testing, and the remaining 430 images for validation and hyperparameter tuning.

The experimental were conducted on a computer equipped with an Intel Xeon(R) Gold 6271C 2.60Hz CPU, NVIDIA GeForce RTX 4090(24GB) GPU × 2, and 128GB RAM. We used Python 3.10 for the software experiment. To enhance model accuracy, various data augmentation techniques were applied during training, including horizontal flipping, color jittering, random erasing, and rotation. All models were trained using the cross-entropy loss function and the SGD optimization algorithm. The initial learning rate was set to 0.001, with a scheduling strategy employed to gradually decrease the learning rate during training. The models were trained for 300 epochs with a batch size of 128, and additional hyperparameters were fine-tuned separately for each network.

5.2 Ablation Experiments

The influence of quaternion convolution kernels. We conducted experiments to evaluate the impact of the number of the trainable parameters in the quaternion convolution kernel elements of our models. The QResNet models utilized two parameters s and φ as trainable parameters for each element of the quaternion convolution kernel, as described in (15). When all four parameters, s, θ, ϕ, and φ, were employed, as described in (14), these networks were referred to as QResNet(s, θ, ϕ, φ) models. We compared the classification performance of the real-number ResNet, QResNet(s, θ, ϕ, φ), and QResNet models using the Cifar-100 dataset.

Table 2 presents the experimental results, where the number of parameters denotes the count of non-zero parameters in the trained model. The test loss and the test accuracy represent the average loss value and recognition accuracy on the test set, respectively. As discussed in Section 3.1, the number of training parameters set for the QResNet(s, θ, ϕ, φ) models is approximately similar to their corresponding real-number ResNet models. However, due to the lightweight architecture of QResNet, the parameter count in QResNet models is roughly reduced to half of that in the real-number ResNet models. As shown in Table 2, the non-zero parameters in

the QResNet models decrease to about 1/4 to 1/2 of those in their corresponding real-number ResNet models after training. Despite this reduction, the QResNet models maintain comparable test recognition accuracy to QResNet(s, θ, ϕ, φ) models, demonstrating that QResNet effectively enhances image recognition rates while minimizing algorithm complexity.

Table 2. Comparison of model performance with different convolutional kernels (Cifar-100 dataset)

Model	Cifar-100		
	Test loss	Test accuracy (%)	Number of parameters
ResNet-18	0.8426	75.93%	11,173,962
QResNet-18(s, θ, ϕ, φ)	0.4922	83.11%	5,731,266
QResNet-18	**0.5026**	**82.89%**	**2,865,600**
ResNet-34	0.8210	76.30%	21,282,122
QResNet-34(s, θ, ϕ, φ)	0.4831	83.47%	10,796,480
QResNet-34	**0.4981**	**83.30%**	**5,398208**
ResNet-50	0.7161	77.71%	25,041,482
QResNet-50(s, θ, ϕ, φ)	0.4547	84.31%	24,926,712
QResNet-50	**0.4939**	**83.92%**	**13,963,456**
ResNet-101	0.6901	79.97%	44,033,610
QResNet-101(s, θ, ϕ, φ)	0.4074	86.11%.	43,905,344
QResNet-101	**0.4202**	**85.62%**	**26,452,672**
ResNet-152	0.7103	78.23%	59,677,258
QResNet-152(s, θ, ϕ, φ)	0.3509	88.32%	59,182,592
QResNet-152	**0.3538**	**88.06%**	**36,591,296**

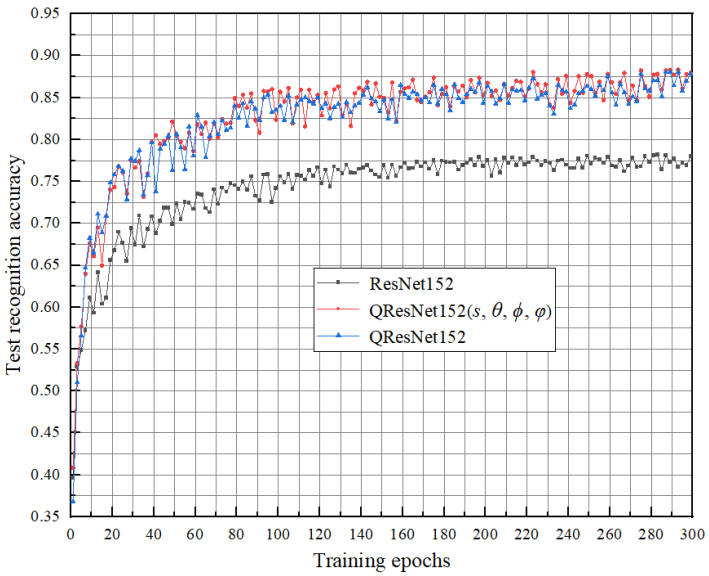


Fig. 2. Test recognition accuracy of models corresponding to different convolutional kernels (Cifar-100 dataset)

Fig. 2 depicts the recognition accuracy curves on the test set of the real-number ResNet-152, QResNet-152(s, θ, ϕ, φ), and QResNet-152 models on the Cifar-100 dataset. It is evident that both the QResNet-152(s, θ, ϕ, φ) and QResNet-152 models achieve significantly higher recognition accuracy on the test set compared to the real-number ResNet-152 network, further confirming the effectiveness of the QResNet models. On the other hand, the QResNet(s, θ, ϕ, φ) models utilize quaternion convolution kernels with four parameters, which yields slightly higher recognition accuracy. However, this comes at the cost of increased algorithm complexity, making it a viable choice for specific application scenarios.

The Influence of Quaternion Regularization Terms. In Section 4.4, the current regularization strategy for training quaternion neural networks treats each of the four components of quaternion weights as individual real-number weights. It then calculates the L_1 or L_2 norm of all the weights to obtain a regularization term. For comparison, if the QResNet model's regularization term adopts the L_1 or L_2 norm of individual real-number components, these can be denoted as QResNet(realL_1) and QResNet(realL_2) models, respectively. We performed experiments on the Cifar-100 dataset to compare the classification performance of the QResNet(realL_1), QResNet(realL_2), and QResNet models, thereby validating the effectiveness of the proposed quaternion L_1 regularization strategy.

Table 3. Comparison of model performance with different regularization terms (Cifar-100 dataset)

Model	Cifar-100				
	Training loss	Training accuracy (%)	Test loss	Test accuracy (%)	Time (ms/frame)
QResNet-18(realL_2)	0.1012	97.58	0.6417	78.34	4.6
QResNet-18(realL_1)	0.2122	92.63	0.5389	81.42	2.9
QResNet-18	**0.3493**	**90.48**	**0.5026**	**82.89**	**2.6**
QResNet-34(realL_2)	0.0746	98.16	0.6212	78.65	5.5
QResNet-34(realL_1)	0.1479	94.88	0.5298	82.16	3.7
QResNet-34	**0.2775**	**92.48**	**0.4981**	**83.30**	**3.3**
QResNet-50(realL_2)	0.0506	98.83	0.6194	79.49	6.3
QResNet-50(realL_1)	0.1418	95.24	0.4957	82.83	5.4
QResNet-50	**0.2389**	**93.67**	**0.4939**	**83.92**	**4.9**
QResNet-101(realL_2)	0.0926	97.63	0.5808	80.64	8.9
QResNet-101(realL_1)	0.1376	95.39	0.4756	84.04	7.4
QResNet-101	**0.1695**	**94.26**	**0.4202**	**85.62**	**7.1**
QResNet-152(realL_2)	0.1073	97.16	0.5750	80.68	10.7
QResNet-152(realL_1)	0.1179	96.10	0.4203	85.73	9.3
QResNet-152	**0.2178**	**94.86**	**0.3538**	**88.06**	**9.1**

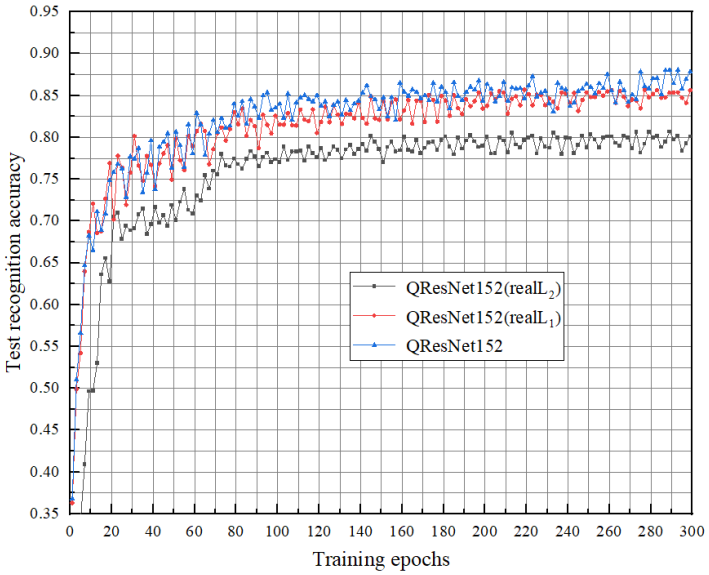


Fig. 3. Test recognition accuracy of models corresponding to different regularization terms (Cifar-100 dataset)

Table 3 shows that the proposed quaternion L_1 regularization outperforms the realL_1 and realL_2 regularization in terms of recognition accuracy on both the training and test sets. The theoretical analysis suggests that this is mainly because realL_2 regularization tends to smooth parameter solutions but does not promote parameter sparsity. In contrast, while realL_1 regularization achieves sparse solutions, it disrupts the balance among color com-

ponents by adjusting each quaternion component individually. The proposed quaternion L_1 regularization calculates the L_2 norm for quaternion components first and then applies the L_1 norm to these L_2 norms. This approach preserves the color information balance for each pixel and reduces the overall feature complexity of the image, improving generalization and effectively suppressing overfitting. Additionally, Table 3 shows that QResNet models with quaternion L1 regularization have the lowest time overhead per test image, further demonstrating the effectiveness of this strategy.

Fig. 3 illustrates the curves of the test recognition accuracy of the models with different regularization methods on the Cifar-100 dataset. It can be observed that the QResNet-152 model consistently outperforms the QResNet-152(realL_2) model. As training epochs increase, the QResNet-152(realL_2) model shows signs of early saturation due to overfitting. In contrast, both the QResNet-152 and QResNet-152(realL_1) models effectively mitigate overfitting, leading to continued performance improvement beyond 150 training epochs. Notably, the QResNet-152 model achieves the highest performance. This underscores the superior effectiveness of the proposed quaternion L1 regularization strategy employed in QResNet models.

5.3 Performance Comparison between QResNet Models and Real-number ResNet Networks

After 300 training epochs, the average loss values and image recognition accuracies for each model on the training and testing datasets are shown in Table 4 and Table 5 respectively.

Table 4. Performance comparison between QResNets and real-number ResNets on the training set

Model	Cifar-10		Cifar-100		Oxford 102 flowers	
	Loss	Accuracy (%)	Loss	Accuracy (%)	Loss	Accuracy (%)
ResNet-18	0.0461	98.87	0.4506	87.73.	0.0583	98.49
QResNet-18	**0.0421**	**99.01**	**0.3493**	**90.48**	**0.0447**	**98.92**
ResNet-34	0.0461	98.88	0.4465	88.03	0.0525	98.63
QResNet-34	**0.0346**	**99.23**	**0.2775**	**92.48**	**0.0421**	**99.02**
ResNet-50	0.0466	98.85	0.4218	88.56	0.0464	98.86
QResNet-50	**0.0306**	**99.39**	**0.2389**	**93.67**	**0.0395**	**99.08**
ResNet-101	0.0479	98.79	0.3253	89.83	0.0447	98.92
QResNet-101	**0.0299**	**99.42**	**0.1695**	**94.26**	**0.0361**	**99.19**
ResNet-152	0.0612	98.42	0.3590	90.25	0.0461	98.88
QResNet-152	**0.0292**	**99.46**	**0.2178**	**94.86**	**0.0339**	**99.26**

Table 5. Performance comparison between QResNets and real-number ResNets on the test set

Model	Cifar-10		Cifar-100		Oxford 102 flowers	
	Loss	Accuracy (%)	Loss	Accuracy (%)	Loss	Accuracy (%)
ResNet-18	0.2331	92.87	0.8426	75.93	0.1267	96.53
QResNet-18	**0.1728**	**94.30**	**0.5026**	**82.89**	**0.1023**	**97.14**
ResNet-34	0.2327	92.96	0.8210	76.30	0.1006	97.18
QResNet-34	**0.1767**	**94.59**	**0.4981**	**83.30**	**0.0980**	**97.23**
ResNet-50	0.2252	93.12	0.7161	77.71	0.0804	97.48
QResNet-50	**0.1545**	**94.74**	**0.4939**	**83.92**	**0.0773**	**97.89**
ResNet-101	0.2386	93.25	0.6901	79.97	0.0848	97.87
QResNet-101	**0.1280**	**95.73**	**0.4202**	**85.62**	**0.0544**	**98.57**
ResNet-152	0.2521	93.11	0.7103	78.23	0.0993	97.25
QResNet-152	**0.1186**	**96.12**	**0.3538**	**88.06**	0.0362	99.18

As illustrated in Table 4 and Table 5, the QResNet-152 model achieves impressive training recognition accuracies of 99.46% on the Cifar-10, 94.86% on the Cifar-100, and 99.26% on the Oxford 102 flowers dataset. Correspondingly, the test recognition accuracies are 96.12%, 88.06%, and 99.18%, respectively. These results suggest that the QResNet models outperform their real-number ResNet counterparts with the same depth in both training and test performance, demonstrating the effectiveness of quaternion-based enhancements in deep residual networks.

Furthermore, the QResNet-152 model outperforms the QResNet-18 model in testing recognition accuracy across the Cifar-10, Cifar-100, and Oxford 102 flowers datasets, with improvements of 1.82%, 5.17%, and

2.04%, respectively. These results, shown in Table 5, indicate that the classification performance of QResNet models enhances with increased model depth, further demonstrating the scalability and effectiveness of the proposed quaternion-based residual networks.

Interestingly, the recognition accuracy of the real-number ResNet model on the Oxford 102 flowers dataset decreases when the network depth reaches 152 layers. This decline is mainly due to the extensive parameter scale of the real-number residual network. In contrast, the QResNet models exhibit a different trend, with the QResNet-152 model maintaining high recognition accuracy even at greater depths. This stability is attributed to the QResNet framework, where each convolutional layer contains roughly half the number of parameters compared to real-number residual networks, significantly reducing the total number of trainable parameters. Furthermore, the quaternion L_1 regularization enhances parameter sparsity after training, effectively minimizing overfitting and contributing to improved model performance.

5.4 Comparative Performance Analysis of QResNet Models and Other Neural Networks

The QResNet models proposed in this paper leverage quaternion algebra to enhance residual networks for image recognition tasks. To assess the performance of our models, we conducted comparative experiments with various advanced network models, including improved ResNet models, existing quaternion-based networks, and the relatively new CNN-based model PathNet [10]. The selected improved ResNet models include DenseNet [12], S-DenseNet [13], and Sparse DenseNet [14]. Existing quaternion neural networks are relatively scarce, but notable ones include QCNN_v1 (6 layers) and QCNN_v2 (8 layers) proposed by X.-Y. Zhu [16], a quaternion convolutional network with an attention mechanism by Q.-L. Yin [19], and two quaternion residual networks with 13 and 110 layers introduced by C. J. Gaudet [21].

Table 6 presents the performance comparison between the QResNet models and these existing neural networks on the experimental datasets. A dash "-" in the table indicates that the corresponding algorithm did not provide experiments or data on the specified database. This comprehensive evaluation highlights the comparative strengths of the QResNet models in enhancing image recognition performance.

Table 6. Performance comparison of QResNet models with existing algorithms

Model	Network layer	Dataset		
		Cifar-10	Cifar-100	Oxford 102 flowers
PathNet	16	**95.16**	76.02	54.06
DenseNet-40	40	93.00	72.45	-
DenseNet-100	100	94.23	76.21	-
S-DenseNet	86	**94.83**	**85.38**	-
Sparse DenseNet	40	93.65	72.89	-
QCNN_v1	6	77.78	-	-
QCNN_v2	8	-	-	76.95
Ref. [19]	6	85.37	-	-
Ref. [21]_v1	13	93.23	69.41	-
Ref. [21]_v2	110	**94.56**	73.99	-
QResNet-18	**18**	**94.30**	**82.89**	**97.14**
QResNet-34	**34**	**94.59**	**83.30**	**97.23**
QResNet-50	**50**	**94.74**	**83.92**	**97.89**
QResNet-101	**101**	**95.73**	**85.62**	**98.57**
QResNet-152	**152**	**96.12**	**88.06**	**99.18**

As shown in Table 6, while the CNN-based improved model PathNet achieves high recognition accuracy on the Cifar-10 test set, its performance on the Cifar-100 and Oxford 102 flowers datasets is comparatively poor. This suggests that PathNet is more suited for small-scale image classification tasks. A similar issue is observed with the DenseNet and Sparse DenseNet models. Among the improved models based on ResNet, the S-ResNet model demonstrates the best performance.

Among quaternion-based image recognition algorithms, QCNN, the first quaternion convolutional neural network model, shows poor recognition performance on the Cifar-10 and Oxford 102 flowers datasets. The model presented in Ref. [19] enhances QCNN by incorporating an attention mechanism, resulting in improved performance on the Cifar-10 dataset compared to QCNN. The quaternion residual neural network proposed in Ref. [21]

further enhances recognition accuracy on the Cifar-10 dataset, surpassing the model in Ref. [19]. However, its performance on the Cifar-100 dataset remains suboptimal.

Table 6 demonstrates that the QResNet models proposed in this paper achieve superior image recognition performance, surpassing the existing neural networks reported in the literature. Although both the QResNet models and the model presented in Ref. [21] are residual quaternion neural networks, they differ significantly in their approaches. The model in Ref. [21] performs quaternion convolution using the quaternion unilateral multiplication described in equation (6) and incorporates a fourth-dimensional feature by adding grayscale information to the RGB channels. This additional dimension introduces considerable redundancy, negatively affecting the algorithm's performance. In contrast, the QResNet models use quaternion bilateral multiplication, as described in equation (9), which preserves the pure imaginary quaternion nature of the input without the need for a fourth-dimensional feature. Furthermore, the integration of quaternion L_1 regularization, quaternion batch normalization, and quaternion pooling in the proposed QResNet models further enhances their performance, making them more efficient and effective.

The comparison highlights that the incorporation of fourth-dimensional information in existing quaternion residual networks introduces information redundancy, negatively impacting model performance. In contrast, QResNet models utilize quaternion bilateral convolution and purely imaginary quaternions to represent the RGB colors of images, ensuring that the final image features remain strictly imaginary quaternions. This representation offers a more effective and natural depiction of color image attributes. Furthermore, QResNet models restrict the quaternion convolution kernels to two trainable parameters, significantly reducing computational complexity. This approach minimizes the risk of overfitting while maintaining sufficient degrees of freedom for learning convolution kernels. Additionally, the quaternion L_1 regularization exploits the algebraic properties of quaternions, treating each quaternion as a single entity. This method enhances sparsity by reducing convolutional kernel parameters while maintaining the balance of color information for individual pixels. Combined with carefully designed model structures, residual blocks, pooling, and normalization, the QResNet models achieve superior recognition accuracy while lowering algorithmic complexity.

6 Conclusion

This study presents five QResNet models: QResNet-18, QResNet-34, QResNet-50, QResNet-101, and QResNet-152. The model design incorporates various advanced features, including lightweight structures, quaternion residual blocks, quaternion bilateral convolution, quaternion pooling, quaternion batch normalization, and quaternion L_1 regularization. The QResNet models represent the RGB color images using purely imaginary quaternions and utilize quaternion bilateral multiplication for convolution, effectively preserving color structural information. This approach reduces the number of convolutional parameters to approximately half compared to real-number networks, thereby decreasing computational complexity. Furthermore, the quaternion L_1 regularization enhances sparsity, mitigating overfitting. The QResNet-152 model demonstrates significant improvements in test recognition accuracy over real-number residual networks of the same depth, achieving increases of 3.01%, 9.83%, and 1.93% on the Cifar-10, Cifar-100, and Oxford 102 flowers datasets, respectively. A current limitation of the algorithm is its relatively large number of network parameters. Future work will focus on simplifying model parameters, extending quaternion algebra to additional network architectures, and applying these advancements to broader image processing tasks, such as color image object detection and scene text localization.

7 Acknowledgement

This work is supported by the National Natural Science Foundation of China (No. 62061024, No. 62176110, No. 61562058).

References

[1] W.R. Hamilton, Lectures on quaternions: containing a systematic statement of a new mathematical method, 1st ed., Hodges and Smith, Whittaker & Co., MacMillan & Co., Dublin, 1853.

[2] B. Xian, M.S. de Queiroz, D. Dawson, I. Walker, Task-space tracking control of robot manipulators via quaternion feedback, IEEE Transactions on Robotics and Automation 20(1)(2004) 160-167.
[3] D. Pavllo, C. Feichtenhofer, M. Auli, D. Grangier, Modeling human motion with quaternion-based neural networks, International Journal of Computer Vision 128(10)(2020) 855-872.
[4] J. Vince. Quaternions for computer graphics, 1st ed., Springer, London, 2011.
[5] X.-Y. Zhang, X.-Y. Zhou, M.-X. Lin, J. Sun, Shufflenet: an extremely efficient convolutional neural network for mobile devices, in: Proc. 2018 the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018.
[6] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition. <https://arxiv.org/abs/1409.1556>, 2014 (accessed 22.12.2023).
[7] A. Krizhevsky, I. Sutskever, G.E. Hinton, ImageNet classification with deep convolutional neural networks, in: Proc. 2012 the 25th Advances in Neural Information Processing Systems (NIPS), 2012.
[8] C. Szegedy, S. Ioffe, V. Vanhoucke, A. Alemi, Inception-v4, inception-resnet and the impact of residual connections on learning, in: Proc. 2017 the 31st AAAI Conference on Artificial Intelligence, 2017.
[9] K.-M. He, X.-Y. Zhang, S.-Q. Ren, J. Sun, Deep residual learning for image recognition, in: Proc. 2016 the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.
[10] Z.-H. Fan, D.-B. Sun, H.-Y. Yu, W.-D. Zhang, PathNet: a novel multi-pathway convolutional neural network for few-shot image classification from scratch, Multimedia Systems 30(2)(2024) 1-13.
[11] R. Pascanu, T. Mikolov, Y. Bengio, On the difficulty of training recurrent neural networks, in: Proc. 2013 International Conference on Machine Learning, 2013.
[12] G. Huang, Z. Liu, L.V.D. Maaten, K.Q. Weinberger, Densely connected convolutional networks, in: Proc. 2017 the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.
[13] C.-Y. Yu, X. He, H.-T. Ma, X. Qi, J.-R. Lu, Y.-H. Zhao, S-DenseNet: a DenseNet compression model based on convolution grouping strategy using Skyline method, IEEE Access 7(2019) 183604-183613.
[14] D. O'Neill, B. Xue, M.-J. Zhang, Evolutionary neural architecture search for high-dimensional skip-connection structures on DenseNet style networks, IEEE Transactions on Evolutionary Computation 25(6)(2021) 1118-1132.
[15] G. Altamirano-gomez, C. Gershenson, Quaternion convolutional neural networks: current advances and future directions, Advances in Applied Clifford Algebras 34(3)(2024) 1-63.
[16] X.-Y. Zhu, Y. Xu, H.-T. Xu, C.-J. Chen, Quaternion convolutional neural networks, in: Proc. 2018 the European Conference on Computer Vision (ECCV), 2018.
[17] K. Chatfield, K. Simonyan, A. Vedaldi, A. Zisserman, Return of the devil in the details: delving deep into convolutional nets. <https://arxiv.org/abs/1405.3531>, 2014 (accessed 15.01.2024).
[18] M.E. Nilsback, A. Zisserman, Automated flower classification over a large number of classes, in: Proc. 2008 the Sixth Indian Conference on Computer Vision, Graphics & Image Processing, 2008.
[19] Q.-L. Yin, J.-W. Wang, X.-Y. Luo, J.-T. Zhai, S.K. Jha, Y.Q. Shi, Quaternion convolutional neural network for color image classification and forensics, IEEE Access 7(2019) 20293-20301.
[20] A. Krizhevsky, G. Hinton, Learning multiple layers of features from tiny images, Handbook of Systemic Autoimmune Diseases 1(4)(2009) 1-58.
[21] C.J. Gaudet, A.S. Maida, Deep quaternion networks, in: Proc. 2018 International Joint Conference on Neural Networks, 2018.
[22] J.-W. Wang, T. Li, X.-Y. Luo, Y.-Q. Shi, S.K. Jha, Identifying computer generated images based on quaternion central moments in color quaternion wavelet domain, IEEE Transactions on Circuits and Systems for Video Technology 29(9)(2018) 2775-2785.
[23] Y. Xu, L.-C. Yu, H.-T. Xu, H. Zhang, T. Nguyen, Vector sparse representation of color image using quaternion matrix analysis, IEEE Transactions on Image Processing 24(4)(2015) 1315-1329.
[24] Z.-G. Xu, F.-X. Yuan, H.-L. Zhu, Y.-M. Xu, Color image super-resolution reconstruction based on quaternion sparse regularization, Journal of Huazhong University of Science & Technology (Natural Science Edition) 46(1)(2018) 75-80.