Deep octonion networks

J. Wu, L. Xu, F. Wu, Y. Kong, L. Senhadji, H. Shu

▶ **To cite this version:**

J. Wu, L. Xu, F. Wu, Y. Kong, L. Senhadji, et al.. Deep octonion networks. Neurocomputing, 2020, 397, pp.179-191. 10.1016/j.neucom.2020.02.053 . hal-02865295

HAL Id: hal-02865295

https://univ-rennes.hal.science/hal-02865295

Submitted on 12 Jun 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Journal Pre-proof

Deep Octonion Networks

Jiasong Wu[1, 2, 3, 4], Ling Xu[1, 4], Fuzhi Wu[1, 4], Youyong Kong[1, 4], Lotfi Senhadji[2, 3, 4], Huazhong Shu[1,4]

[1]*Laboratory of Image Science and Technology, Key Laboratory of Computer Network and Information Integration, Southeast University, Ministry of Education, Nanjing 210096, China*

[2]*INSERM, U1099, Rennes, F-35000, France*

[3]*Université de Rennes 1, LTSI, Rennes, F-35042, France*

[4]*Centre de Recherche en Information Médicale Sino-français (CRIBs), Université de Rennes, Inserm, Southeast University*

Abstract: Deep learning is a hot research topic in the field of machine learning methods and applications. Real-value neural networks (Real NNs), especially deep real networks (DRNs), have been widely used in many research fields. In recent years, the deep complex networks (DCNs) and the deep quaternion networks (DQNs) have attracted more and more attentions. The octonion algebra, which is an extension of complex algebra and quaternion algebra, can provide more efficient and compact expressions. This paper constructs a general framework of deep octonion networks (DONs) and provides the main building blocks of DONs such as octonion convolution, octonion batch normalization and octonion weight initialization; DONs are then used in image classification tasks for CIFAR-10 and CIFAR-100 data sets. Compared with the DRNs, the DCNs, and the DQNs, the proposed DONs have better convergence and higher classification accuracy. The success of DONs is also explained by multi-task learning.

Keywords: Convolutional neural network, complex, quaternion, octonion, image classification

Journal Pre-proof

Journal Pre-proof

1. Introduction

Real-value neural networks (Real NNs) [1-12] attracted the attention of many researchers and recently made major breakthroughs in many areas such as signal processing, image processing, natural language processing, etc. Many models of Real NNs have been constructed and reported in the literature. These models can generally be categorized into two kinds: non-deep models and deep models. The non-deep models are mainly constructed by multilayer perceptron module [13] and hard to train, if we only use the real-valued back propagation (BP) algorithm [14], when their layers are larger than 4. The deep models can be roughly constructed by the following two strategies: multilayer perceptron models assisted by the unsupervised pretrained methods (for example, deep belief nets [15], deep auto-encoder [16], etc.) and real-value convolutional neural networks (Real CNNs), including LeNet-5 [17], AlexNet [18], Inception [19-22], VGGNet [23], HighwayNet [24], ResNet [25], ResNeXt [26], DenseNet [27], FractalNet [28], PolyNet [29], SENet [30], CliqueNet [31], BinaryNet [32], SqueezeNet [33], MobileNet [34], etc.

Although Real CNNs have achieved great success in various applications, the correlations between convolution kernels are generally not taken into consideration, that is, there is no connection or special relationship established between convolution kernels. On the opposite of Real CNNs, real-value recurrent neural networks (Real RNNs) [35-38], obtains the correlations by establishing connections between convolution kernels and by learning their weights. This approach increases significantly the training difficulty and has poor convergence. *Thus, a first question raised: Can we consider correlations between convolution kernels by mean of some special relationships, which do not require learning, instead of adding the connections between convolution kernels?*

Many researchers showed that the performance can be improved when the relationships between convolution kernels are modeled by complex algebra, quaternion algebra [39-42], and also octonion algebra [43-45]. Therefore, they attached a lot of attention for extending neural

Journal Pre-proof

networks from real domain to complex, quaternion, and also octonion domains. These extension models, as in Real CNNs, can also be divided into two categories: non-deep models [46-60] and deep models [61-71]. There are many research work focusing on the non-deep models, for example, Widrow *et al*. [46] first introduced the complex-valued neural networks (Complex NNs), which have been widely used in recent years in radar imaging, image processing, communication signal processing, and many others [47]. Compared to the Real NNs, the Complex NNs have better generalization ability due to their time-varying delays and impulse effects [48]. Arena *et al*. [49] then extended the neural networks from complex to quaternion domain and proposed quaternion-valued neural networks (Quaternion NNs), which have been applied to color image compression [50], color night vision [51], and 3D wind forecasting [52]. Furthermore, the quaternion-valued BP algorithms achieve correct geometrical transformations in color space for an image compression problem, whereas real-valued BP algorithms fail [53, 54]. Popa [55] further extended the neural networks from quaternion to octonion domain and proposed octonion-valued neural networks (Octonion NNs), whose leakage delays, time-varying delays, and distributed delays were also introduced [56, 57]. Moreover, Clifford-valued neural networks [58-60] were also proposed for the extension of complex NNs and quaternion NNs. For the deep models, Reichert and Serre [61] proposed complex-valued deep networks, which interpreted half of the cell state as an imaginary part and used complex values to simulate the phase dependence of biologically sound neurons. Then, some researchers proposed complex-valued convolutional neural networks (Complex CNNs) [62-65]. Among them, Trabelsi *et al*. [65] proposed deep complex networks (DCNs) which provide the key atomic components (complex convolution, complex batch normalization, and complex weight initialization strategy, etc.) for the construction of DCNs and also obtain lower error rate than corresponding deep real networks (DRNs) [25] in CIFAR-10 and CIFAR-100 [66]. Then, Gaudet and Maida [67] extended the deep networks from complex to quaternion domain and proposed deep quaternion networks (DQNs), which achieve better image classification performance than DCNs [65] in CIFAR-10 and

Journal Pre-proof

CIFAR-100. Meanwhile, Titouan *et al*. also proposed deep quaternion neural network (QDNN) [68, 69], quaternion recurrent neural network (QRNN) [70], bidirectional quaternion long-short term memory (BQLSTM) [71], and quaternion convolutional autoencoder (QCAE) [72]. *Then, a second question raised: Can we extend the deep networks from quaternion to octonion domain to obtain a further benefit? How to explain the success of these deep networks on these various domains (complex, quaternion, octonion)*?

In an attempt to solving this second question, in this paper, we propose deep octonion networks which can be seen as an extension of deep networks from quaternion domain to octonion domain. The contributions of the paper are as follows:

1） The key atomic components of deep octonion networks, such as octonion convolution module, octonion batch normalization module and octonion weight initialization method.

2） When applying the proposed deep octonion networks on the classification tasks on CIFAR-10 and CIFAR-100, the classification results are better than deep real networks, deep complex networks and deep quaternion networks.

3） The explanation of deep complex networks, deep quaternion networks, and deep octonion networks behaviors from the perspective of multi-task learning [73-75].

The rest of the paper is organized as follows. In Section 2, octonion representation, its main properties and characteristics are briefly introduced. The architectural components needed to build deep octonion networks is described in Section 3. The classification performance of deep octonion networks is analyzed and also compared to the deep real networks, deep complex networks, and deep quaternion networks in Section 4. Then, Section 5 explains deep networks behaviors on these domains from the perspective of multi-task learning. The conclusions are formulated in Section 6.

Journal Pre-proof

Journal Pre-proof

2. Octonion Representation

An octonion number x is a hypercomplex number, which is an extension of complex number and quaternion number, consists of one real part and seven imaginary parts:

$$x=x_0e_0+x_1e_1+x_2e_2+x_3e_3+x_4e_4+x_5e_5+x_6e_6+x_7e_7\in O \tag{1}$$

where O denotes the octonion number field, $x_i\in R, i=1,2,...,7$ (R denotes the real number field), e_0=1, and e_i, i=1,2,…,7, are seven imaginary units obeying the following rules:

$$\begin{cases} e_i^2 = -1 \\ e_ie_j = -e_je_i \\ (e_ie_j)e_k = -e_i(e_je_k) \end{cases}, \qquad \forall i\neq j\neq k,\ 1\le i,j,k\le 7, \tag{2}$$

The above equation shows that the octonion multiplication is neither commutative nor associative. The multiplication tables of the imaginary units are also shown in Table 1.

The conjugate of this octonion $x\in O$ is given by

$$x^* = x_0e_0 - x_1e_1 - x_2e_2 - x_3e_3 - x_4e_4 - x_5e_5 - x_6e_6 . \tag{3}$$

The unit norm octonion of $x\in O$ is

$$x^{\triangleleft}=\frac{x}{\sqrt{x_0^2+x_1^2+x_2^2+x_3^2+x_4^2+x_5^2+x_6^2+x_7^2}} . \tag{4}$$

Journal Pre-proof

For a complete review of the properties of octonion, the reader can refer to [43].

3. Deep Octonion Networks

This section introduces the methods and the modules required to construct the deep octonion networks and to initialize them: octonion internal representation method (Section 3.1), octonion convolution module (Section 3.2), octonion batch normalization module (Section 3.3), and octonion weight initialization method (Section 3.4).

3.1 Octonion internal representation method

We represent the real part and seven imaginary parts of an octonion number as logically distinct real valued entities and simulate octonion arithmetic using real-valued arithmetic internally. If we assume that an octonion convolutional layer has N feature maps where N is divisible by 8, then, these feature maps can be split into 8 parts to form an octonion representation. Specifically, as shown in Fig. 1, we allocate the first $N/8$ feature maps to the real part and the remaining seven $N/8$ feature maps to the seven imaginary parts.

3.2 Octonion convolution module

Octonion convolution can be implemented by convolving an octonion vector $\mathbf{x}_o \in O^N$ by an octonion filter matrix $\mathbf{W}_o \in O^{N \times N}$ as follows:

Journal Pre-proof

$$
\begin{aligned}
\mathbf{W}_o *_o \mathbf{x}_o =& \left(\mathbf{W}_0+\mathbf{W}_1e_1+\mathbf{W}_2e_2+\mathbf{W}_3e_3+\mathbf{W}_4e_4+\mathbf{W}_5e_5+\mathbf{W}_6e_6+\mathbf{W}_7e_7\right)\\
&*_o\left(\mathbf{x}_0+\mathbf{x}_1e_1+\mathbf{x}_2e_2+\mathbf{x}_3e_3+\mathbf{x}_4e_4+\mathbf{x}_5e_5+\mathbf{x}_6e_6+\mathbf{x}_7e_7\right)\\
=&\left(\mathbf{W}_0*\mathbf{x}_0-\mathbf{W}_1*\mathbf{x}_1-\mathbf{W}_2*\mathbf{x}_2-\mathbf{W}_3*\mathbf{x}_3-\mathbf{W}_4*\mathbf{x}_4-\mathbf{W}_5*\mathbf{x}_5-\mathbf{W}_6*\mathbf{x}_6-\mathbf{W}_7*\mathbf{x}_7\right)\\
&+\left(\mathbf{W}_0*\mathbf{x}_1+\mathbf{W}_1*\mathbf{x}_0+\mathbf{W}_2*\mathbf{x}_3-\mathbf{W}_3*\mathbf{x}_2+\mathbf{W}_4*\mathbf{x}_5-\mathbf{W}_5*\mathbf{x}_4-\mathbf{W}_6*\mathbf{x}_7+\mathbf{W}_7*\mathbf{x}_6\right)\mathbf{e}_1\\
&+\left(\mathbf{W}_0*\mathbf{x}_2-\mathbf{W}_1*\mathbf{x}_3+\mathbf{W}_2*\mathbf{x}_0+\mathbf{W}_3*\mathbf{x}_1+\mathbf{W}_4*\mathbf{x}_6+\mathbf{W}_5*\mathbf{x}_7-\mathbf{W}_6*\mathbf{x}_4-\mathbf{W}_7*\mathbf{x}_5\right)\mathbf{e}_2\\
&+\left(\mathbf{W}_0*\mathbf{x}_3+\mathbf{W}_1*\mathbf{x}_2-\mathbf{W}_2*\mathbf{x}_1+\mathbf{W}_3*\mathbf{x}_0+\mathbf{W}_4*\mathbf{x}_7-\mathbf{W}_5*\mathbf{x}_6+\mathbf{W}_6*\mathbf{x}_5-\mathbf{W}_7*\mathbf{x}_4\right)\mathbf{e}_3\\
&+\left(\mathbf{W}_0*\mathbf{x}_4-\mathbf{W}_1*\mathbf{x}_5-\mathbf{W}_2*\mathbf{x}_6-\mathbf{W}_3*\mathbf{x}_7+\mathbf{W}_4*\mathbf{x}_0+\mathbf{W}_5*\mathbf{x}_1+\mathbf{W}_6*\mathbf{x}_2+\mathbf{W}_7*\mathbf{x}_3\right)\mathbf{e}_4\\
&+\left(\mathbf{W}_0*\mathbf{x}_5+\mathbf{W}_1*\mathbf{x}_4-\mathbf{W}_2*\mathbf{x}_7+\mathbf{W}_3*\mathbf{x}_6-\mathbf{W}_4*\mathbf{x}_1+\mathbf{W}_5*\mathbf{x}_0-\mathbf{W}_6*\mathbf{x}_3+\mathbf{W}_7*\mathbf{x}_2\right)\mathbf{e}_5\\
&+\left(\mathbf{W}_0*\mathbf{x}_6+\mathbf{W}_1*\mathbf{x}_7+\mathbf{W}_2*\mathbf{x}_4-\mathbf{W}_3*\mathbf{x}_5-\mathbf{W}_4*\mathbf{x}_2+\mathbf{W}_5*\mathbf{x}_3+\mathbf{W}_6*\mathbf{x}_0-\mathbf{W}_7*\mathbf{x}_1\right)\mathbf{e}_6\\
&+\left(\mathbf{W}_0*\mathbf{x}_7-\mathbf{W}_1*\mathbf{x}_6+\mathbf{W}_2*\mathbf{x}_5+\mathbf{W}_3*\mathbf{x}_4-\mathbf{W}_4*\mathbf{x}_3-\mathbf{W}_5*\mathbf{x}_2+\mathbf{W}_6*\mathbf{x}_1+\mathbf{W}_7*\mathbf{x}_0\right)\mathbf{e}_7,
\end{aligned}
\tag{5}
$$

which can be expressed as the real matrix form as follows:

$$
\begin{bmatrix}
\mathcal{R}(\mathbf{W}_o *_o \mathbf{x}_o)\\ \mathcal{I}(\mathbf{W}_o *_o \mathbf{x}_o)\\ \mathcal{J}(\mathbf{W}_o *_o \mathbf{x}_o)\\ \mathcal{K}(\mathbf{W}_o *_o \mathbf{x}_o)\\ \mathcal{E}(\mathbf{W}_o *_o \mathbf{x}_o)\\ \mathcal{L}(\mathbf{W}_o *_o \mathbf{x}_o)\\ \mathcal{M}(\mathbf{W}_o *_o \mathbf{x}_o)\\ \mathcal{N}(\mathbf{W}_o *_o \mathbf{x}_o)
\end{bmatrix}
=
\begin{bmatrix}
\mathbf{W}_0 & -\mathbf{W}_1 & -\mathbf{W}_2 & -\mathbf{W}_3 & -\mathbf{W}_4 & -\mathbf{W}_5 & -\mathbf{W}_6 & -\mathbf{W}_7\\
\mathbf{W}_1 & \mathbf{W}_0 & -\mathbf{W}_3 & \mathbf{W}_2 & -\mathbf{W}_5 & \mathbf{W}_4 & \mathbf{W}_7 & -\mathbf{W}_6\\
\mathbf{W}_2 & \mathbf{W}_3 & \mathbf{W}_0 & -\mathbf{W}_1 & -\mathbf{W}_6 & -\mathbf{W}_7 & \mathbf{W}_4 & \mathbf{W}_5\\
\mathbf{W}_3 & -\mathbf{W}_2 & \mathbf{W}_1 & \mathbf{W}_0 & -\mathbf{W}_7 & \mathbf{W}_6 & -\mathbf{W}_5 & \mathbf{W}_4\\
\mathbf{W}_4 & \mathbf{W}_5 & \mathbf{W}_6 & \mathbf{W}_7 & \mathbf{W}_0 & -\mathbf{W}_1 & -\mathbf{W}_2 & -\mathbf{W}_3\\
\mathbf{W}_5 & -\mathbf{W}_4 & \mathbf{W}_7 & -\mathbf{W}_6 & \mathbf{W}_1 & \mathbf{W}_0 & \mathbf{W}_3 & -\mathbf{W}_2\\
\mathbf{W}_6 & -\mathbf{W}_7 & -\mathbf{W}_4 & \mathbf{W}_5 & \mathbf{W}_2 & -\mathbf{W}_3 & \mathbf{W}_0 & \mathbf{W}_1\\
\mathbf{W}_7 & \mathbf{W}_6 & -\mathbf{W}_5 & -\mathbf{W}_4 & \mathbf{W}_3 & \mathbf{W}_2 & -\mathbf{W}_1 & \mathbf{W}_0
\end{bmatrix}
*
\begin{bmatrix}
\mathbf{x}_0\\ \mathbf{x}_1\\ \mathbf{x}_2\\ \mathbf{x}_3\\ \mathbf{x}_4\\ \mathbf{x}_5\\ \mathbf{x}_6\\ \mathbf{x}_7
\end{bmatrix},
\tag{6}
$$

where $*_o$ and $*$ denote octonion convolution and real convolution, respectively. $\mathbf{x}_i \in R^N$ and $\mathbf{W}_i \in R^{N\times N}$ with i=1,2,…,7. $\mathcal{R}(\bullet)$ denotes the real part of $\bullet$, $\mathcal{I}(\bullet)$, $\mathcal{J}(\bullet)$, $\mathcal{K}(\bullet)$, $\mathcal{E}(\bullet)$, $\mathcal{L}(\bullet)$, $\mathcal{M}(\bullet)$ and $\mathcal{N}(\bullet)$ denote the seven different imaginary parts of $\bullet$, respectively. The implementation of octonion convolutional operation is shown in Fig. 2, where M_r, M_i, M_j, M_k, M_e, M_l, M_m, M_n refer to eight parts of feature maps and K_r, K_i, K_j, K_k, K_e, K_l, K_m, K_n refer to eight parts of kernels, and $M_{p_1} * K_{p_2} (\mathbf{p}_1, \mathbf{p}_2 = \mathbf{r}, \mathbf{i}, \mathbf{j}, \mathbf{k}, \mathbf{e}, \mathbf{l}, \mathbf{m}, \mathbf{n})$ refer to the result of a real convolution between the feature maps and the kernels. The real representations of complex convolution, quaternion convolution, and octonion convolution are

Journal Pre-proof

Journal Pre-proof

shown in Appendix 1. From this latter, we can see that the octonion convolution is a kind of mixed convolution, similar to a mixture of standard convolution and depth separable convolution, with certain links to the original convolution [76]. Traditional real-valued convolution simply multiplies each channel of the kernel by the corresponding channel of the image. The goal of the octonion convolution is to generate a unique linear combination of each axis based on the results of a single axis, allowing each axis of the kernel to interact with each axis of the image, thereby allowing the linear depth of the channel to be mixed, depending on the structure of the octonion multiplication. For example, using 8 kernels ($m \times n \times 8$) to convolve an 8 channels of feature maps ($M \times N \times 8$), finally generates one feature map. Conventional convolution is one convolution kernel applied to one feature map, and then added to the result of the previous operation, regardless of the correlation between the feature maps. The octonion convolution, using the octonion arithmetic rule, applies eight convolution kernels to each feature map. Then applying a 1×1 convolution to the result of the previous operation allows to obtain the linear interaction of the feature map and thus to derive the new feature map space.

3.3 Octonion batch normalization module

Batch normalization [31] can accelerate deep network training by reducing internal covariate shift. It allows us to use much higher learning rates and be less careful about initialization. When applying the batch normalization to real numbers, it is sufficient to translate and scale these numbers such that their mean is zero and their variance is one. However, when applying the batch normalization to complex or quaternion numbers, this can't ensure equal variance in both the real and imaginary components. In order to overcome this problem, a whitening approach is used in [65, 67], which scales the data by the square root of their variances along each principle components. In this section, we use a similar approach, but treating this issue as a "whitening" of 8D vector problem.

Firstly, whitening is accomplished by multiplying the zero-centered data (**x**-E[**x**]) by the inverse square root of the covariance matrix **V**:

$$\tilde{\mathbf{x}} = \frac{(\mathbf{x} - \mathrm{E}[\mathbf{x}])}{\sqrt{\mathbf{V}}}, \tag{7}$$

and

$$\mathbf{V} = \begin{bmatrix} V_{rr} & V_{ri} & V_{rj} & V_{rk} & V_{re} & V_{rl} & V_{rm} & V_{rn} \\ V_{ir} & V_{ii} & V_{ij} & V_{ik} & V_{ie} & V_{il} & V_{im} & V_{in} \\ V_{jr} & V_{ji} & V_{jj} & V_{jk} & V_{je} & V_{jl} & V_{jm} & V_{jn} \\ V_{kr} & V_{ki} & V_{kj} & V_{kk} & V_{ke} & V_{kl} & V_{km} & V_{kn} \\ V_{er} & V_{ei} & V_{ej} & V_{ek} & V_{ee} & V_{el} & V_{em} & V_{en} \\ V_{lr} & V_{li} & V_{lj} & V_{lk} & V_{le} & V_{ll} & V_{lm} & V_{ln} \\ V_{mr} & V_{mi} & V_{mj} & V_{mk} & V_{me} & V_{ml} & V_{mm} & V_{mn} \\ V_{nr} & V_{ni} & V_{nj} & V_{nk} & V_{ne} & V_{nl} & V_{nm} & V_{nn} \end{bmatrix}, \tag{8}$$

where E[**x**] refers to the average value of each batch of training data $\mathbf{x} \in O^N$, and $\mathbf{V} \in O^{8\times 8}$ is the covariance matrix of each batch of data **x**.

In order to avoid calculating the $\left(\sqrt{\mathbf{V}}\right)^{-1}$, Eq. (7) can be computed as follows

$$\tilde{\mathbf{x}} = \mathbf{U}(\mathbf{x} - \mathrm{E}[\mathbf{x}]), \tag{9}$$

where **U** is one of the matrices from the Cholesky decomposition of $\mathbf{V}^{-1}$, and each item of the matrix **U** is shown in Appendix 2.

Secondly, the forward conduction formula of the octonion batch normalization layer is defined as

$$\mathrm{OctonionBN}(\mathbf{x}) = \boldsymbol{\gamma}\mathbf{x} + \boldsymbol{\beta}, \tag{10}$$

where $\boldsymbol{\beta} = E(\mathbf{x}) \in O^8$ is a learned parameter with eight real parameters (one real part and seven imaginary parts) and $\boldsymbol{\gamma} = \sqrt{\mathbf{V}} \in O^{8\times 8}$ is also a learned parameter with only 36 independent real parameters,

Journal Pre-proof

Journal Pre-proof

$$\boldsymbol{\gamma}=\begin{pmatrix} \gamma_{rr} & \gamma_{ri} & \gamma_{rj} & \gamma_{rk} & \gamma_{re} & \gamma_{rl} & \gamma_{rm} & \gamma_{rn} \\ \gamma_{ir} & \gamma_{ii} & \gamma_{ij} & \gamma_{ik} & \gamma_{ie} & \gamma_{il} & \gamma_{im} & \gamma_{in} \\ \gamma_{jr} & \gamma_{ji} & \gamma_{jj} & \gamma_{jk} & \gamma_{je} & \gamma_{jl} & \gamma_{jm} & \gamma_{jn} \\ \gamma_{kr} & \gamma_{ki} & \gamma_{kj} & \gamma_{kk} & \gamma_{ke} & \gamma_{kl} & \gamma_{km} & \gamma_{kn} \\ \gamma_{er} & \gamma_{ei} & \gamma_{ej} & \gamma_{ek} & \gamma_{ee} & \gamma_{el} & \gamma_{em} & \gamma_{en} \\ \gamma_{lr} & \gamma_{li} & \gamma_{lj} & \gamma_{lk} & \gamma_{le} & \gamma_{ll} & \gamma_{lm} & \gamma_{ln} \\ \gamma_{mr} & \gamma_{mi} & \gamma_{mj} & \gamma_{mk} & \gamma_{me} & \gamma_{ml} & \gamma_{mm} & \gamma_{mn} \\ \gamma_{nr} & \gamma_{ni} & \gamma_{nj} & \gamma_{nk} & \gamma_{ne} & \gamma_{nl} & \gamma_{nm} & \gamma_{nn} \end{pmatrix}. \tag{11}$$

Similar to [65] and [67], the diagonal of $\boldsymbol{\gamma}$ is initialized to $1/\sqrt{8}$, the off diagonal terms of $\boldsymbol{\gamma}$ and all components of $\boldsymbol{\beta}$ are initialized to 0.

3.4 Octonion weight initialization method

Before starting to train the network, we need to initialize its parameters. If the weights are initialized to the same value, the updated weights will be the same, which means that the network can't learn the features. For deep neural networks, such initialization will make deeper meaningless and will not match the effects of linear classifiers. Therefore, the initial weight values are all different and close but not equal to 0, which not only ensures the difference between the input and output, but also allows the model to converge stably and quickly. In view of this, we provide an initialization method for octonion weight. The 8 parts of every octonion weight $\mathbf{W}_o \in O^{N\times N}$ are assumed to be independent Gaussian random variables with zero-mean and the same variance σ^2. Then, the variance of $\mathbf{W}_o$ is provided by:

$$Var\left(\mathbf{W}_o\right)=\mathbb{E}\left[|\mathbf{W}_o|^2\right]-\left|\mathbb{E}\left[\mathbf{W}_o\right]\right|^2 \tag{12}$$

As the 8 parts are zero-mean then $\mathbb{E}\left[\mathbf{W}_o\right]=0$ and as they have the same variance then $Var\left(\mathbf{W}_o\right)=8\sigma^2$. The value of the standard deviation σ is set according to the following:

$$\sigma=\begin{cases}1/\left(2\sqrt{n_{in}+n_{out}}\right), & \text{if Glorot's initialization [77] is used}\\ 2/\sqrt{n_{in}}, & \text{if He's initialization [78] is used}\end{cases} \tag{13}$$

4. Implementation and Experimental Results

Similar to the 110-layer deep real networks [25], we designed an octonion convolutional neural network named deep octonion networks, whose schematic diagram are shown in Fig. 3. Fig. 3(a) shows the detailed convolution structure of the four stages, and Fig. 3(b) shows the entire structure including the input and output modules. Then we performed the image classification tasks of CIFAR-10 and CIFAR-100 [66] to verify the validity of the proposed deep octonion networks. The following experiment was implemented using Keras (Tensorflow as backend) on a PC machine, which sets up Ubuntu 16.04 operating system and has an Intel(R) Core(TM) i7-2600 CPU with speed of 3.40 GHz and 64 GB RAM, and has also two NVIDIA GeForce GTX1080-Ti GPUs.

4.1 Models configurations

4.1.1 Octonion input construction

Since the images in datasets of CIFAR-10 and CIFAR-100 are real-valued, however, the input of the proposed deep octonion networks needs to be an octonion matrix, which we have to derive first. The octonion has one real part and seven imaginary parts, we put the original N training real images into the real part, and similar to [65] and [67], the seven imaginary parts of the octonion matrix are obtained by performing a single real-valued residual block ($\mathrm{BN}\rightarrow\mathrm{ReLU}\rightarrow\mathrm{Conv}\rightarrow\mathrm{BN}\rightarrow\mathrm{ReLU}\rightarrow\mathrm{Conv}$) [25] 7 times at the same time. Then, the 8 vectors are connected according to a given axis to form a brand new octonion vector.

4.1.2 The structure of deep octonion networks

Journal Pre-proof

Journal Pre-proof

The $OctonionConv \rightarrow OctonionBN \rightarrow ReLU$ operation is performed on the obtained octonion input, where $OctonionConv$ denotes the octonion convolution module shown in section 3.2 and $OctonionBN$ denotes the octonion batch normalization module shown in section 3.3. Then the octonion output is sent to the next three stages. In each stage, there are several residual blocks with double convolution layers. The shape of the feature maps in three stages are the same, and the number of them are increased gradually to ensure the expressive ability of the output features. To speed up the training, the following layer is an $AveragePooling2D$ layer, which is then followed by a fully connected layer called $Dense$ to classify the input. The deep octonion network model sets the number of residual blocks in the three stages to 10, 9, and 9, respectively, and the number of convolution filters is set to 32, 64, and 128. The batch size is set to 64.

4.1.3 The training of deep octonion networks

Deep octonion networks are then compiled, the cross entropy loss function and the stochastic gradient descent method are chosen for training the model. The Nesterov Momentum is set to 0.9 in the back propagation of stochastic gradient descent in order to increase the stability and speed up the convergence. The learning rate is shown in Table 2. Using a custom learning rate schedule, different learning rates are used in different epochs in order to make the network more stable. Here, the learning rate from 0 to 80'th epoch is divided into three stages. The first 20 epochs are preheated at a learning rate of 0.01, and for the middle 40 epochs we increase the learning rate by a factor ten. For the latter 20 epochs, we restore it to 0.01. The deep octonion networks are trained on 120 epochs, which is less than 200 epochs in [65] and [67], because the convergence speed of the deep octonion networks is higher than the deep real networks [25], deep complex networks [65], and deep quaternion networks [67].

4.2 Experimental results and analysis

Journal Pre-proof

There are two methods for choosing the learning rate. One is to have the same learning rate during training which is called "smooth" learning rate (blue line in Fig. 4), and the other is to adjust the learning rate during training which is called "convex" learning rate (red line in Fig. 4). It is worth noting that two initialization methods of Glorot et al. [77] and He et al. [78] were used in the experiment, which are shown in the form of solid lines and dotted lines, respectively. Experiments show that the "convex" learning rate setting from epoch 0 to epoch 80 is better than the "smooth" learning rate. Although the accuracy will fluctuate during the 20 to 60 epochs, the accuracy will increase slightly in the next 20 epochs. Therefore, in the subsequent experiments, the "convex" learning rate was used for scheduling. The accuracy of the deep octonion networks, deep complex networks [65], and deep quaternion networks [67] in the first 20 epochs are compared in Fig. 5, from which we can see that the proposed deep octonion networks perform better compared to the other deep networks. Besides, as shown in Table 3, the deep octonion networks have less learning parameters than the other compared deep networks. In addition, as shown in the second and third column of Table 3, we also use floating point operations (FLOPs) and multiply-accumulate operations (MACCs) to statistically count the model's calculation volume to obtain the speed of the model. Regarding the calculation of FLOPs and MACCs, we followed the same steps as in [79, 80]. The calculation performances show that DONs use less computations and achieve the best results under the same conditions.

For the classification tasks of CIFAR-10 and CIFAR-100: Firstly, we use the 10-fold cross-validation method, that is, we divide the data set into ten parts, and take nine parts as training data and one part as testing data, and then the Top-1 error rate is obtained by averaging the 10 results. Secondly, the 10-fold cross-validation are performed 10 times, and the mean value is obtained as an estimation of the Top-1 error rate of the algorithm. Table 3 shows the Top-1 error rate of the deep octonion networks, deep real networks [25], deep complex networks [65], and deep quaternion networks [67] on CIFAR-10 and CIFAR-100. It can be seen from Table 3 that the deep octonion networks can achieve lower error rate compared to the other deep

Journal Pre-proof

networks and the advantage becomes more apparent when there are more classes to distinguish. From the fourth column of Table 3, we can see that, compared with deep quaternion networks [67], the improvement of the deep octonion networks is not significant as the relative improvement is 1.7%. However, as shown in the fifth column of Table 3, when the number of classes increases, the proposed deep octonion networks the relative improvement becomes 5.4% when compared to deep quaternion networks [67]. This also implies that the deep octonion networks have better generalization ability than the other compared deep networks. The phenomenon is also explained by the multi-task learning in the next section.

5. Explanation of deep octonion networks via multi-task learning

In Machine Learning, the standard algorithm is to learn one task at a time, we generally train a single model or an ensemble of models to perform our desired task and the output of the system is real. When facing the complex learning problems, traditional methods chose a similar approach. Firstly, decompose the complex learning problems into simple and independent sub-problems, and then study each sub-problem separately. Finally, establish the mathematical model of complex problems by combining the sub-problem learning results, and the model is refined through fine tuning until the performance no longer improves. These operations seem reasonable but not accurate, for the reason that many problems in the real world cannot be decomposed into independent sub-problems, the rich interrelated information between the sub-problems cannot be ignored. Even if it can be decomposed, the sub-problems are interrelated, and connected by sharing factors or share representations. In order to solve this problem, multi-task learning (MTL) was born [73]. Compared to single-task learning (STL—learning just one task at a time), MTL is a kind of joint learning method which learns multiple related tasks together based on shared representations. The purpose of the shared

Journal Pre-proof

representation is to improve the generalization. Multiple tasks are learned in parallel, and the results affect each other.

5.1 The relationship between DONs and MTL

MTL can be seen as a method that inspired by human learning. We often learn tasks to acquire the necessary skills in order to master more complex problems. There are many forms of MTL that can further improve CNN performance. Figures 6(a) and 6(b) show a single-task learning and multi-task feedforward neural network with one input layer, two hidden layers and one output layer, respectively. In single-task learning, learning between tasks is independent of each other. In MTL, parameters between multiple tasks are shared. MTL methods can be divided into two categories based on how parameters are shared between different task models. In the soft parameter sharing category, each task has its own model and its own parameters. The methods in this category focus on how to design weight-sharing approaches. The most common way is to share all convolutional layers and split on fully connected layers for heuristic decisions for loss of specific tasks, such as Cross-Stitch Networks [81], Sluice Network [82], etc. In hard parameter sharing category, all task models share exactly the same feature extractor, and each branch head executes its own task. In the context of deep learning, MTL is usually done by sharing hard or soft parameters of the hidden layer [74]. Fig. 6 (b) shows the MTL mode of hard parameter sharing. We refer to the structure between the input layer and the output layer as the shared layer.

However, currently it is difficult to determine the best shared feature position [83], the best sharing / splitting scheme [81] and the existing CNN structure only receives feature tensors with a fixed number of feature channels. If multitasking is used for the existing CNN structure, the number of channels will increase as the number of multitasking tasks increases. The tandem features will not be available to subsequent layers of the CNN. There are multiple solutions to this problem. One is the NDRR layer proposed in [84], which is plug-and-play extended to the existing CNN architecture and uses feature transformation to discriminate cascaded features and to reduce their dimensions. The DONs network proposed in this

Journal Pre-proof

paper solves the problem of channel number mismatch by improving the network structure, and constructs a new octonion convolutional neural network for eight-task learning.

Generally, when considering the optimization of more than one loss function, we are effectively dealing with a MTL problem. Next, we only focus on one task, that is, there is only one optimization goal, and the learning of auxiliary tasks may still help improving the learning performance of the main task. Auxiliary tasks can provide inductive bias, which makes the model more inclined to those solutions that can explain multiple tasks at the same time, and the generalization performance of the model is then better. The DONs shown in Fig. 6 (c) uses a network structure with hard parameter sharing similar to Fig. 6 (b) to learn eight related tasks together. The input of the last seven tasks is learned through the input of the first task, which plays a supporting role in the first task. Therefore, there are relevant and irrelevant parts in these eight tasks. The details of the DONs shared layer follow the rules in Fig. 2.

5.2 Effectiveness of DONs

In [75], it has been proven that the number of parameters in the multi-task model is less than the number of parameters for establishing multiple models, and the task is optimized to reduce the risk of over-fitting and the generalization ability is then stronger. The effectiveness of MTL is mainly reflected in the following five aspects: implicit data augmentation, attention focusing, eavesdropping, representation bias and regularization. Therefore, considering the DONs as a specific MTL is supported by the following:

- Neural networks can help the hidden layer to avoid local minima through the interaction between different tasks during learning. When learning the main task, the parts that are not related to the task will produce noise during the learning process. Since different tasks have different noise patterns, a model that learns eight tasks simultaneously is able to learn a more general representation. Eight tasks learned at the same time

Journal Pre-proof

can average the noise patterns, which can make the model better representative of the data. This is similar to implicit data augmentation for MTL.

- DONs takes the first of the eight tasks as the main one, and the latter seven tasks are learned through the input of the first task, which assists the first task. The gap between tasks is not particularly large, so the model can be focused on those features that do have an impact, as in the process of attention focusing for MTL. Studies have shown that if auxiliary tasks and main tasks use the same characteristics for decision-making, they will benefit more from MTL. Therefore, we need to find suitable auxiliary tasks to benefit from MTL. The choice of auxiliary tasks is diverse [74].

- DONs restrict the model by using octonion operation rules, so that models that are more in line with real rules can be selected from the hypothesis space. This kind of regularization is such that the risk of overfitting as well as the complexity of the mode are reduced.

In addition, as described in the previous section, hard parameter sharing (DONs can be considered as such models) is the most common method of MTL in neural networks, which can greatly reduce the risk of overfitting [74]. It has been demonstrated in [75] that the risk of overfitting the shared parameters is smaller than overfitting the specific parameters for each task. What's more, regards the number of tasks as N, a larger N means that the more tasks are learned simultaneously, the more the model can find a representation that captures all tasks, and the less likely the original task is to overfitting. This also explains that the performance can be improved when the relationships between convolution kernels are modeled by complex algebra, quaternion algebra, and also octonion algebra and why deep neural networks can achieve better results on octonion domains.

6. Conclusion

Journal Pre-proof

In this paper, we propose deep octonion networks (DONs) as an extension of DRNs, the DCNs, and the DQNs. The main building blocks of DONs are given, such as octonion convolution, octonion batch normalization, and octonion weight initialization. DONs were applied to the image classification tasks of CIFAR-10 and CIFAR-100 to verify their validity. Experiments showed that compared with the DRNs, the DCNs, and the DQNs, the proposed DONs have better convergence, less parameters, and higher classification accuracy. The success of DONs is also explained through multi-task learning approach.

ACKNOWLEDGEMENT

This work was supported in part by the National Natural Science Foundation of China under Grants 61876037, 31800825, 61871117, 61871124, 61773117, 31571001, 61572258, and in part by the National Key Research and Development Program of China under Grants 2017YFC0107903, 2017YFC0109202, 2018ZX10201002-003, and in part by the Short-Term Recruitment Program of Foreign Experts under Grant WQ20163200398.

Author Contributions:
Jiasong Wu: Conceptualization, Methodology, Writing- Reviewing and Editing.
Ling Xu: Writing- Original draft preparation, Software, Visualization.
Fuzhi Wu: Software, Validation, Data Curation.
Youyong Kong: Validation, Project administration.
Lotfi Senhadji: Formal analysis, Writing- Reviewing and Editing.
Huazhong Shu: Supervision, Resources.

Conflicts of Interest: The authors declare that they have no known conflicts of interest.

Journal Pre-proof

Journal Pre-proof

References

[1] W.S. McCulloch and W. Pitts, A Logical Calculus of Ideas Immanent in Nervous Activity, Bulletin of Mathematical Biophysics 5 (4) (1943) 115-133.
[2] S. Haykin, Neural Networks: A Comprehensive Foundation (2 ed.). Prentice Hall. ISBN 0-13-273350-1, 1998.
[3] Y. Bengio, A. Courville, P.Vincent, Representation learning: a review and new perspectives, IEEE Trans. Pattern Anal. Mach. Intell. 35 (8) (2013) 1798-1828.
[4] Y. LeCun, Y. Bengio, G.E. Hinton, Deep learning, Nature 521 (2015) 436-444.
[5] I. Goodfellow, Y. Bengio, A. Courville, Deep learning, MIT Press, 2016.
[6] L. Deng, D. Yu, Deep learning: methods and applications, Found Trends Signal Process, 7 (3-4) (2014) 197-387.
[7] J. Schmidhuber, Deep learning in neural networks: An overview, Neural Networks, 61 (2015) 85-117.
[8] H. Wang, et al., On the origin of deep learning, arXiv preprint arXiv:1702.07800v1, 2017.
[9] J. Gu, et al., Recent advances in convolutional neural networks, arXiv preprint arXiv:1512.07108v1, 2015.
[10] Y. Guo, et al., Deep learning for visual understanding: A review. Neurocomputing, 187 (2016) 27-48.
[11] A. Prieto, B. Prieto, E. M. Ortigosa, et al., Neural networks: An overview of early research, current frameworks and new challenges. Neurocomputing, 214 (2016) 242-268.
[12] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, F. E. Alsaadi, A survey of deep neural network architectures and their applications, Neurocomputing 741 234 (2017) 11–26.
[13] F. Rosenblatt. Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms. Spartan Books, Washington DC, 1961.
[14] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning representations by back-propagating errors, Cognitive modeling, 1988.
[15] G. Hinton, S. Osindero, Y.W. Teh, A fast learning algorithm for deep belief nets, Neural computation 18 (7) (2006) 1527-1554.
[16] G.E. Hinton, R.R. Salakhutdinov, Reducing the dimensionality of data with neural networks, Science 313 (5786) (2006) 504-507.
[17] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient based learning applied to document recognition, Proc IEEE, 86(11) (1998) 2278-2324.
[18] A. Krizhevsky, I. Sutskever, G.E. Hinton, ImageNet classification with deep convolutional neural networks, in: Proceedings of NIPS, 2012.
[19] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, Going deeper with convolutions, in: Proceedings of the CVPR, 2015.
[20] S. Ioffe, C. Szegedy, Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, arXiv preprint arXiv: 1502.03167, 2015.
[21] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna, Rethinking the Inception Architecture for Computer Vision, 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, 2818-2826.
[22] C. Szegedy, S. Ioffe, V. Vanhoucke, Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning, arXiv preprint arXiv: 1602.07261, 2016.
[23] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv preprint arXiv: 1409.1556, 2014.
[24] R. K. Srivastava, K. Greff, J. Schmidhuber, Training very deep networks. Computer Science, 2015.
[25] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of CVPR, 2016.
[26] S. Xie, et al., Aggregated residual transformations for deep neural networks, arXiv preprint arXiv: 1611.05431, 2016.
[27] G. Huang, Z. Liu, K.Q. Weinberger, Densely Connected Convolutional Networks, arXiv preprint arXiv: 1608.06993v3, 2016.
[28] G. Larsson, et al., FractalNet: Ultra-Deep Neural Networks without Residuals, arXiv preprint arXiv: 1605.07648, 2016.
[29] X. Zhang, Z. Li, C. C. Loy, D. Lin, Polynet: a pursuit of structural diversity in very deep networks, arXiv preprint arXiv: 1611.05725, 2016.

Journal Pre-proof

[30] J. Hu, L. Shen, G. Sun, Squeeze-and-Excitation Networks, arXiv preprint arXiv: 1709.01507,2017.
[31] Y. Yang, Z. Zhong, T. Shen, et al., Convolutional Neural Networks with Alternately Updated Clique, in: Proceedings of the IEEE CVPR, 2018.
[32] I. Hubara, et al., Binarized Neural Networks, in: Proceedings of NIPS, 2016.
[33] F. N. Iandola, et al., SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5MB model size, arXiv preprint arXiv: 1602.07360, 2016.
[34] A. G. Howard, et al., MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications, arXiv preprint arXiv: 1704.04861, 2017.
[35] J. L. Elman, Finding Structure in Time, Cognitive Science. 14 (2) (1990) 179-211.
[36] S. Hochreiter, J. Schmidhuber, Long short-term memory. Neural Computation, 9(8), 1735-1780., Long Short Term Memory, Neural Computation, 9(8) (1997) 1735-1780.
[37] A. Graves, Jürgen Schmidhuber, Framewise phoneme classification with bidirectional LSTM and other neural network architectures, Neural Networks, IJCNN, 18 (5) (2005) 602–610.
[38] K. Cho, B. V. Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, Learning phrase representations using rnn encoder-decoder for statistical machine translation, arXiv preprint arXiv: 1406.1078, 2014.
[39] W. R. Hamilton, Lectures on quaternions, Nature, 57(1462) (2010) 7.
[40] T. A. Ell, S. J. Sangwine, Hypercomplex Fourier transforms of color images, IEEE Trans Image Process, 16(1) (2007) 22-35.
[41] C. C. Took, D. P. Mandic, The quaternion LMS algorithm for adaptive filtering of hypercomplex processes, IEEE Trans Signal Process, 57 (2009) 1316-1327.
[42] R. Zeng, J.S. Wu, Z.H. Shao, Y. Chen, B.J. Chen, L. Senhadji and H.Z. Shu, Color image classification via quaternion principal component analysis network, Neurocomputing, vol. 216 (2016) 416-428.
[43] S. Okubo, Introduction to Octonion and Other Non-Associative Algebras in Physics, Cambridge University Press, 1995.
[44] H. Y. Gao, K. M. Lam, From quaternion to octonion: Feature-based image saliency detection, IEEE International Conference on Acoustics, speech and signal processing, 2014.
[45] Ł. Błaszczyk, K. M. Snopek, Octonion Fourier transform of real-valued functions of three variables - selected properties and examples, Signal Processing, 136 (2017) 29–37.
[46] B. Widrow, J. Mccool, M. Ball , The complex LMS algorithm, Proceedings of the IEEE, 63(4) (1975) 719-720.
[47] A. Hirose, Complex-Valued Neural Networks: Advances and Applications. 2013.
[48] Q. Song, H. Yan, Z. Zhao, Y. Liu, Global exponential stability of complex-valued neural networks with both time-varying delays and impulsive effects, Neural Networks, 79 (2016) 108-116.
[49] P. Arena, L. Fortuna, L. Occhipinti, M. G. Xinilia, Neural networks for quaternion-valued function approximation, In Internationa l symposium on circuits and systems, 6 (1994) 307–310.
[50] T. Isokawa, T. Kusakabe, N. Matsui, F. Peper, Quaternion neural network and its application, International Conference on Knowledge-Based and Intelligent Information and Engineering Systems, (2003) 318–324.
[51] H. Kusamichi, T. Isokawa, N. Matsui, Y. Ogawa, K. Maeda, A new scheme for color night vision by quaternion neural network, In: International conference on autonomous robots and agents (2004) 101–106.
[52] C. Jahanchahi, C. Took, D. Mandic, On HR calculus, quaternion valued stochastic gradient, and adaptive three dimensional wind forecasting, In International joint conference on neural networks, IEEE, (2010) 1–5.
[53] T. Isokawa, T. Kusakabe, N. Matsui, F. Peper, Quaternion Neural Network and Its Application, International Conference on Knowledge-Based and Intelligent Information and Engineering Systems, (2003) 318-324.

Journal Pre-proof

[54] N. Matsui, T. Isokawa, H. Kusamichi, et al., Quaternion neural network with geometrical operators, Journal of Intelligent & Fuzzy Systems Applications in Engineering & Technology, 15(3, 4) (2004) 149-164.
[55] Călin-Adrian Popa, Octonion-valued neural networks, Artificial Neural Networks and Machine Learning-ICANN, 2016.
[56] Călin-Adrian Popa, Global Exponential Stability of Neutral-Type Octonion-Valued Neural Networks with Time-Varying Delays, Neurocomputing, 309(2) (2018)177-133.
[57] Călin-Adrian Popa, Global Exponential Stability of Octonion-Valued Neural Networks with leakage delay and mixed delays. Neural Networks, 105 (2018) 277-293.
[58] J. Pearson, D. Bisset, Neural networks in the Clifford domain, In International conference on neural networks, IEEE, 3 (1994) 1465–1469.
[59] S. Buchholz, G. Sommer, On Clifford neurons and Clifford multi-layer perceptrons, Neural Networks, 21(7) (2008) 925–935.
[60] Y. Kuroe, S. Tanigawa, H. Iima, Models of Hopfield-type Clifford neural networks and their energy functions –hyperbolic and dual valued networks –, Proc.int.conf.neural Information Processing, 7062 (2011) 560–569.
[61] D.P. Reichert and T. Serre, Neuronal synchrony in complex-valued deep networks, in: Proceedings of ICLR, 2014.
[62] R. Haensch, O. Hellwich, Complex-valued convolutional neural networks for object detection in PolSAR data, in: Proceedings of EUSAR, 2010, 1-4.
[63] Z. Zhang, H. Wang, F. Xu, Y. Q. Jin, Complex-valued convolutional neural network and its application in polarimetric SAR image classification, IEEE Trans Geosci Remote Sens, 55(12) (2017) 7177-7188.
[64] Călin-Adrian Popa, Complex-valued convolutional neural networks for real-valued image classification, in: Proceedings of the IJCNN, 2017.
[65] C. Trabelsi, O. Bilaniuk, Y. Zhang, et al., Deep Complex Networks, in: Proceedings of ICLR, 2018.
[66] Krizhevsky, Alex, and H. Geoffrey, Learning multiple layers of features from tiny images, Technical report, University of Toronto, 1(4) (2009).
[67] C. Gaudet, A. Maida, Deep Quaternion Networks, in: Proceedings of the IJCNN, 2018.
[68] T. Parcollet, M. Morchid, P. M. Bousquet, et al., Quaterion Neural Networks for Spoken Language Understanding, Spoken Language Technology Workshop IEEE, 2017.
[69] T. Parcollet, Y. Zhang, M. Morchid, et al., Speech Recognition with Quaternion Neural Networks, Conference on Neural Information Processing Systems, 2018.
[70] T. Parcollet, M. Ravanelli, M. Morchid, et al., Quaternion Recurrent Neural Networks, ICLR, 2019.
[71] T. Parcollet, M. Morchid, G. Linarès, et al., Bidirectional Quaternion Long-Short Term Memory Recurrent Neural Networks for Speech Recognition, 2018.
[72] T. Parcollet, M. Morchid, G. Linarès, et al., Quaternion Convolutional Neural Networks for Heterogeneous Image Processing, ICASSP, 2019.
[73] R. Caruana, Multitask Learning. Autonomous Agents and Multi-Agent Systems, 27(1) (1998) 95–133.
[74] S. Ruder, An overview of multi-task learning in deep neural networks, 2017.
[75] J. Baxter, A Bayesian/information theoretic model of learning to learn via multiple task sampling, Machine Learning, 28 (1997) 7–39.
[76] F. Chollet, Xception: Deep learning with depthwise separable convolutions, In Proceedings of the IEEE conference on computer vision and pattern recognition, (2017) 1251-1258.
[77] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, In International conference on artificial intelligence and statistics, 2010.
[78] K. He, X. Zhang, S.Ren, and J. Sun, Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, In Proceedings of the IEEE international conference on computer vision, 2015.
[79] K. He, J. Sun, Convolutional neural networks at constrained time cost, In Proceedings of the IEEE conference on computer vision and pattern recognition, (2015) 5353-5360.

[80] S. Han, J. Pool, J. Tran, et al, Learning both weights and connections for efficient neural network, Advances in neural information processing systems, (2015) 1135-1143.
[81] I. Misra, A. Shrivastava, A. Gupta, et al. Cross-stitch networks for multi-task learning, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, (2016) 3994-4003.
[82] S. Ruder12, J. Bingel, I. Augenstein, et al. Learning what to share between loosely related tasks, arXiv preprint arXiv: 1705.08142, 2017.
[83] MD. Zeiler, R. Fergus, Visualizing and understanding convolutional networks. In ECCV, (2014)818–833.
[84] Y Gao, J Ma, M Zhao, et al., NDDR-CNN: Layerwise feature fusing in multi-task CNNs by neural discriminative dimensionality reduction, In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, (2019) 3205-3214.

Jiasong Wu received the B.S. degree in Biomedical Engineering from the University of South China, Hengyang, China, in 2005, and joint Ph.D. degree with the Laboratory of Image Science and Technology (LIST), Southeast University, Nanjing, China, and Laboratoire Traitement du signal et de l'Image (LTSI), University of Rennes 1, Rennes, France in 2012. He is now working in the LIST as a lecturer. His research interest mainly includes deep learning, fast algorithms of digital signal processing and its applications. Mr. Wu received the Eiffel doctorate scholarship of excellence (2009) from the French Ministry of Foreign Affairs and also the Chinese government award for outstanding self-financed student abroad (2010) from the China Scholarship Council.

Ling Xu received the B.S. degree in Computer Science and technology from Hefei University of Technology, Hefei, China, in 2017. Now she is currently pursuing the M.S. degree in Computer Science and technology, Southeast University. Her research interests lie in deep learning and pattern recognition.

Journal Pre-proof

Fuzhi Wu received the B.S. degree from Anhui Normal University in 2017 and now is studying for Ph.D. degree in School of Computer Science and Engineering, Southeast University. His research interests lie in deep learning and pattern recognition, signal and image processing.

Youyong Kong received the B.S. and M.S. degrees in computer science and engineering from Southeast University, Nanjing, China, in 2008 and 2011, respectively, and the Ph.D. degree in imaging and diagnostic radiology from the Chinese University of Hong Kong, Hong Kong, in 2014.He is currently an Assistant Professor with the College of Computer Science and Engineering, Southeast University. His current research interests include machine learning, and medical image processing and brain network analysis.

Journal Pre-proof

Journal Pre-proof

Lotfi Senhadji received the Ph.D. degree from the University of Rennes 1, Rennes, France, in signal processing and telecommunications in 1993. He is a Professor and the Head of the INSERM Research Laboratory LTSI. His is also Co-Director of the French-Chinese Laboratory CRIBs "Centre de Recherche en Information Biomédicale Sino-Français". His main research efforts are focused on nonstationary signal processing with particular emphasis on wavelet transforms and time-frequency representations for detection, classification, and interpretation of biosignals. He has published more than 80 research papers in journals and conferences, and he contributed to five handbooks. Dr. Senhadji is a senior member of the IEEE EMBS and the IEEE Signal Processing Society.

Huazhong Shu received the B.S. degree in Applied Mathematics from Wuhan University, China, in 1987, and a Ph.D. degree in Numerical Analysis from the University of Rennes 1, Rennes, France in 1992. He is a professor of the LIST Laboratory and the Codirector of the CRIBs. His recent work concentrates on the image analysis, pattern recognition and fast algorithms of digital signal processing. Dr. Shu is a senior member of IEEE Society.

Journal Pre-proof

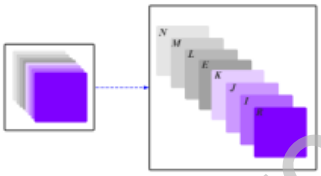
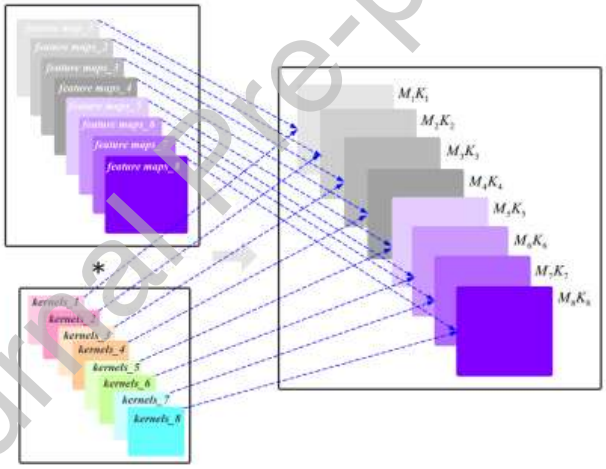
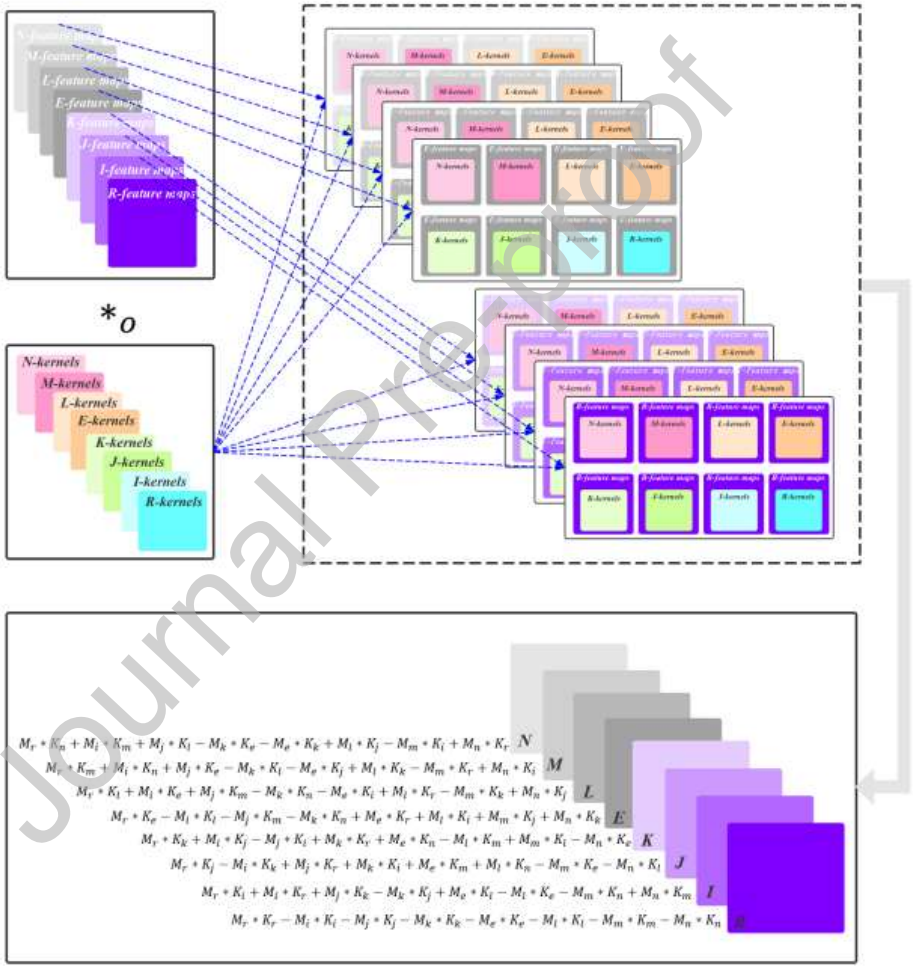


Fig. 1. The octonion internal representation



(a) Real Convolution

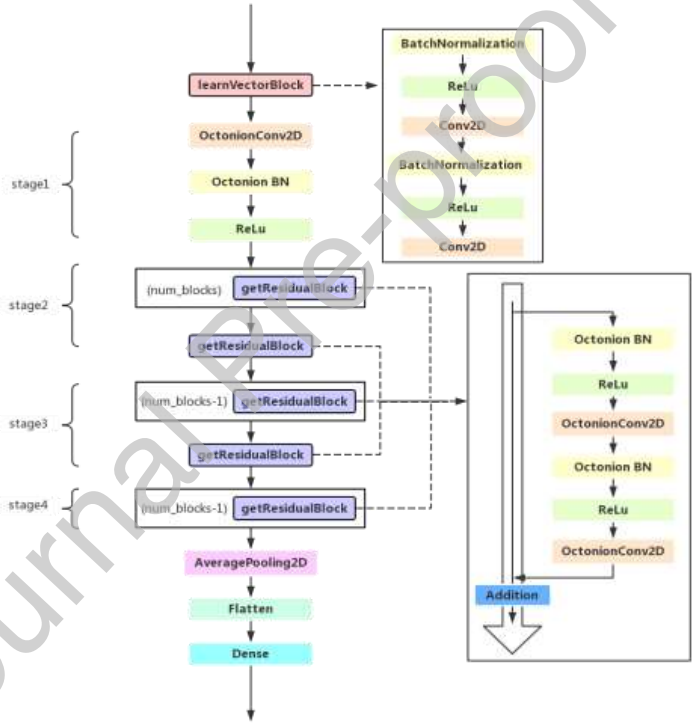
Journal Pre-proof



(b) Octonion Convolution

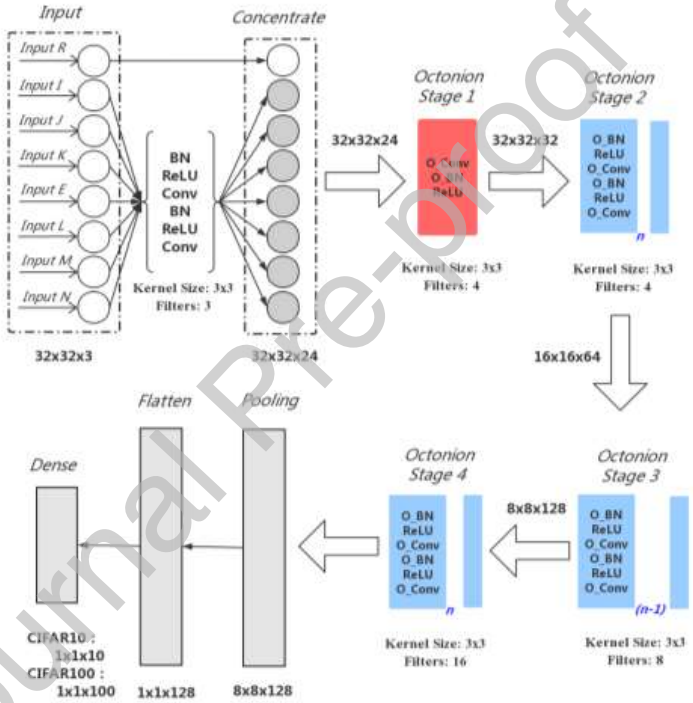
Journal Pre-proof

Fig. 2. Illustration of the real convolution (a) and octonion convolution (b)



Journal Pre-proof

(a) Convolution modules



(b) Network architecture for DONs

Fig. 3. The implementation details of deep octonion networks (DONs)

Journal Pre-proof

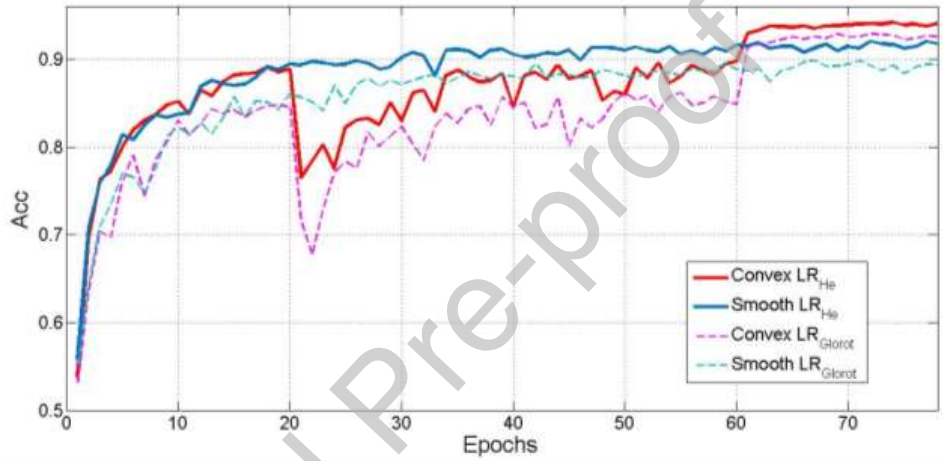


Fig. 4. Results of different learning rate settings

Journal Pre-proof

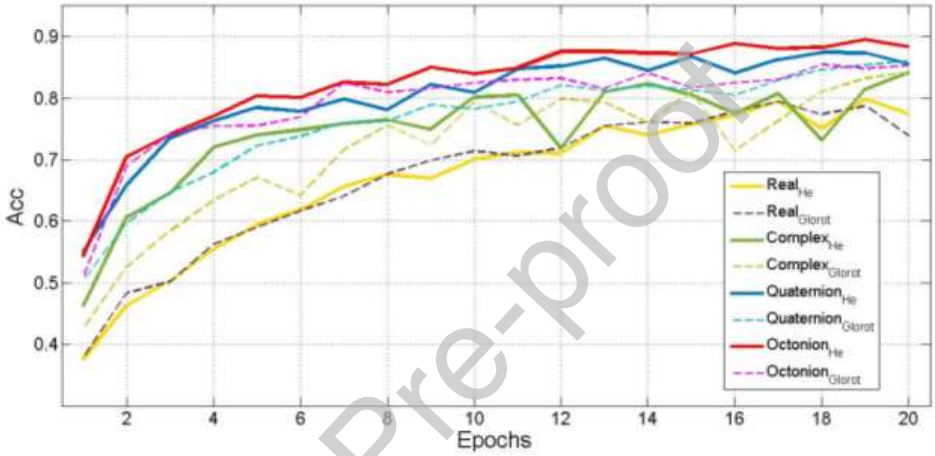
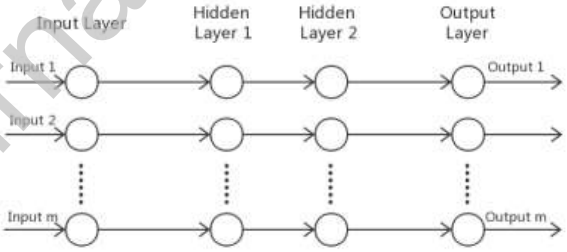
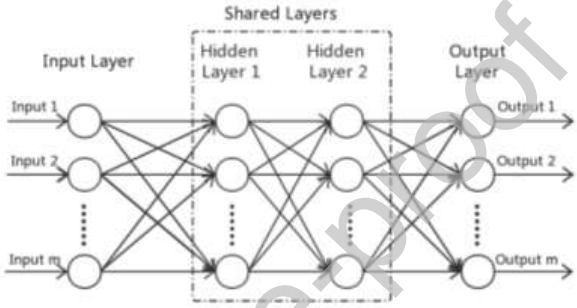


Fig. 5. Accuracy curves of four models in the first 20 epochs

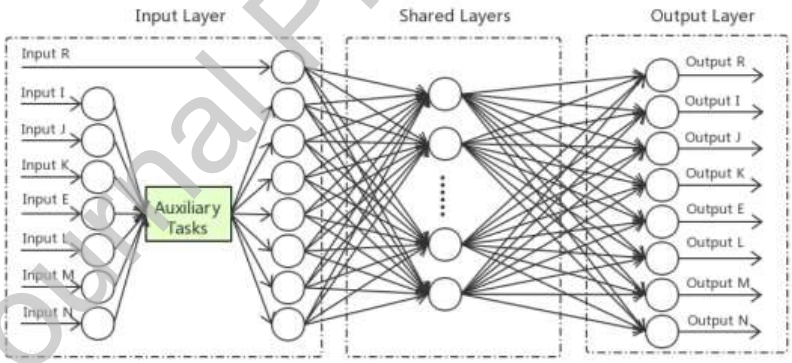


Journal Pre-proof

(a) Single-task learning (STL)



(b) Multi-tasking learning (MTL)



(c) Deep Octonion Networks (DONs)

Journal Pre-proof

Journal Pre-proof

Fig. 6. The comparison of Single-task learning (a), Multi-tasking learning (b) and Deep octonion networks (c)

Table 1
The multiplication table of the unit octonions

		e_j							
	e_ie_j	1	e_1	e_2	e_3	e_4	e_5	e_6	e_7
e_i	1	1	e_1	e_2	e_3	e_4	e_5	e_6	e_7
	e_1	e_1	–1	e_3	$-e_2$	e_5	$-e_4$	$-e_7$	e_6
	e_2	e_2	$-e_3$	–1	e_1	e_6	e_7	$-e_4$	$-e_5$
	e_3	e_3	e_2	$-e_1$	–1	e_7	$-e_6$	e_5	$-e_4$
	e_4	e_4	$-e_5$	$-e_6$	$-e_7$	–1	e_1	e_2	e_3
	e_5	e_5	e_4	$-e_7$	e_6	$-e_1$	–1	$-e_3$	e_2
	e_6	e_6	e_7	e_4	$-e_5$	$-e_2$	e_3	–1	$-e_1$
	e_7	e_7	$-e_6$	e_5	e_4	$-e_3$	$-e_2$	e_1	–1

Table 2
The learning rate (%) of octonion convolution neural network.

Epoch	Learning-rate
(0 , 20)	0.01
(20, 60)	0.1
(60, 80)	0.01
(80,110)	0.001
(110, 120)	0.0001

Journal Pre-proof

Table 3
The classification error rate of three models in two types of datasets. FLOPs and MACCs denote floating point operations and multiply-accumulate operations, respectively.

Architecture	Params	FLOPs	MACCs	CIFAR-10	CIFAR-100
Real [25]	3,619,844	1081,333,248	340,380,416	6.37	-
Complex [65]	1,823,620	541,132,288	270,273,792	5.60	27.09
Quaternion [67]	932,792	271,922,688	135,662,848	5.44	26.01
Octonion	**481,150**	**137,350,144**	**68,368,896**	**5.35**	**24.60**

Appendix 1 The real representations of complex convolution, quaternion convolution, and octonion convolution. e denotes imaginary component, where $e_i^2=-1$, $e_ie_j=-e_je_i\neq e_je_i$, $(e_ie_j)e_k=-e_i(e_je_k)\neq e_ie_je_k$, $\forall i\neq j\neq k,\ 1\le i,j,k\le 7$. $*$, $*_c$, $*_q$, and $*_o$ denote real convolution, complex convolution, quaternion convolution, and octonion convolution, respectively. $\mathcal{R}(\bullet)$ denotes the real components of $\bullet$, $\mathcal{I}(\bullet)$, $\mathcal{J}(\bullet)$, $\mathcal{K}(\bullet)$, $\mathcal{E}(\bullet)$, $\mathcal{L}(\bullet)$, $\mathcal{M}(\bullet)$ and $\mathcal{N}(\bullet)$ denote the different imaginary components of $\bullet$ respectively. $\mathbf{x}_r\in R^N$, $\mathbf{x}_c\in C^N$, $\mathbf{x}_q\in Q^N$, $\mathbf{x}_o\in O^N$, $\mathbf{W}_r\in R^{N\times N}$, $\mathbf{W}_c\in C^{N\times N}$, $\mathbf{W}_q\in Q^N$, $\mathbf{W}_o\in O^N$, $\mathbf{x}_i\in R^N, \mathbf{W}_i\in R^{N\times N}, i=0,1,...,7$, where R, C, Q, and O denote real, complex, quaternion, and octonion domain, respectively.

Multidimensional domains	Input vector	Filter matrix	Various convolutions between **W** and **h**	Implementation of various convolutions in real domain
Real [25]	$\mathbf{x}_r$	$\mathbf{W}_r$	$\mathbf{W}_r*\mathbf{x}_r=\sum_i\sum_j W_r[i,j]x_r[m-i,n-j]$	$\mathcal{R}(\mathbf{W}*\mathbf{x})=\mathbf{W}*\mathbf{x}$
Complex [65]	$\mathbf{x}_c=\mathbf{x}_0+\mathbf{x}_1e_1$	$\mathbf{W}_c=\mathbf{W}_0+\mathbf{W}_1e_1$	$\mathbf{W}_c*_c\mathbf{x}_c=(\mathbf{W}_0+\mathbf{W}_1e_1)*_c(\mathbf{x}_0+\mathbf{x}_1e_1)$	$\begin{bmatrix}\mathcal{R}(\mathbf{W}_c*_c\mathbf{x}_c)\\ \mathcal{I}(\mathbf{W}_c*_c\mathbf{x}_c)\end{bmatrix}=\begin{bmatrix}\mathbf{W}_0 & -\mathbf{W}_1\\ \mathbf{W}_1 & \mathbf{W}_0\end{bmatrix}*\begin{bmatrix}\mathbf{x}_0\\ \mathbf{x}_1\end{bmatrix}$
Quaternion [67]	$\mathbf{x}_q=\mathbf{x}_0+\mathbf{x}_1e_1+\mathbf{x}_2e_2+\mathbf{x}_3e_3$	$\mathbf{W}_q=\mathbf{W}_0+\mathbf{W}_1e_1+\mathbf{W}_2e_2+\mathbf{W}_3e_3$	$\mathbf{W}_q*_q\mathbf{x}_q=(\mathbf{W}_0+\mathbf{W}_1e_1+\mathbf{W}_2e_2+\mathbf{W}_3e_3)*_q(\mathbf{x}_0+\mathbf{x}_1e_1+\mathbf{x}_2e_2+\mathbf{x}_3e_3)$	$\begin{bmatrix}\mathcal{R}(\mathbf{W}_q*_q\mathbf{x}_q)\\ \mathcal{I}(\mathbf{W}_q*_q\mathbf{x}_q)\\ \mathcal{J}(\mathbf{W}_q*_q\mathbf{x}_q)\\ \mathcal{K}(\mathbf{W}_q*_q\mathbf{x}_q)\end{bmatrix}=\begin{bmatrix}\mathbf{W}_0 & -\mathbf{W}_1 & -\mathbf{W}_2 & -\mathbf{W}_3\\ \mathbf{W}_1 & \mathbf{W}_0 & -\mathbf{W}_3 & \mathbf{W}_2\\ \mathbf{W}_2 & \mathbf{W}_3 & \mathbf{W}_0 & -\mathbf{W}_1\\ \mathbf{W}_3 & -\mathbf{W}_2 & \mathbf{W}_1 & \mathbf{W}_0\end{bmatrix}*\begin{bmatrix}\mathbf{x}_0\\ \mathbf{x}_1\\ \mathbf{x}_2\\ \mathbf{x}_3\end{bmatrix}$

Journal Pre-proof

Octonion	$\begin{aligned}\mathbf{x}_o &= \mathbf{x}_0 \\ &+ \mathbf{x}_1 e_1 \\ &+ \mathbf{x}_2 e_2 \\ &+ \mathbf{x}_3 e_3 \\ &+ \mathbf{x}_4 e_4 \\ &+ \mathbf{x}_5 e_5 \\ &+ \mathbf{x}_6 e_6 \\ &+ \mathbf{x}_7 e_7\end{aligned}$	$\begin{aligned}\mathbf{W}_o &= \mathbf{W}_0 \\ &+ \mathbf{W}_1 e_1 \\ &+ \mathbf{W}_2 e_2 \\ &+ \mathbf{W}_3 e_3 \\ &+ \mathbf{W}_4 e_4 \\ &+ \mathbf{W}_5 e_5 \\ &+ \mathbf{W}_6 e_6 \\ &+ \mathbf{W}_7 e_7\end{aligned}$	$\begin{aligned}&\mathbf{W}_o *_o \mathbf{x}_o \\ &= \left(\mathbf{W}_0 + \mathbf{W}_1 e_1 + \mathbf{W}_2 e_2 + \mathbf{W}_3 e_3\right. \\ &\left. + \mathbf{W}_4 e_4 + \mathbf{W}_5 e_5 + \mathbf{W}_6 e_6 + \mathbf{W}_7 e_7\right) \\ &*_o \left(\mathbf{x}_0 + \mathbf{x}_1 e_1 + \mathbf{x}_2 e_2 + \mathbf{x}_3 e_3\right. \\ &\left. + \mathbf{x}_4 e_4 + \mathbf{x}_5 e_5 + \mathbf{x}_6 e_6 + \mathbf{x}_7 e_7\right)\end{aligned}$	$\begin{bmatrix}\mathcal{R}(\mathbf{W}_o *_o \mathbf{x}_o) \\ \mathcal{I}(\mathbf{W}_o *_o \mathbf{x}_o) \\ \mathcal{J}(\mathbf{W}_o *_o \mathbf{x}_o) \\ \mathcal{K}(\mathbf{W}_o *_o \mathbf{x}_o) \\ \mathcal{E}(\mathbf{W}_o *_o \mathbf{x}_o) \\ \mathcal{L}(\mathbf{W}_o *_o \mathbf{x}_o) \\ \mathcal{M}(\mathbf{W}_o *_o \mathbf{x}_o) \\ \mathcal{N}(\mathbf{W}_o *_o \mathbf{x}_o)\end{bmatrix} = \begin{bmatrix}\mathbf{W}_0 & -\mathbf{W}_1 & -\mathbf{W}_2 & -\mathbf{W}_3 & -\mathbf{W}_4 & -\mathbf{W}_5 & -\mathbf{W}_6 & -\mathbf{W}_7 \\ \mathbf{W}_1 & \mathbf{W}_0 & -\mathbf{W}_3 & \mathbf{W}_2 & -\mathbf{W}_5 & \mathbf{W}_4 & \mathbf{W}_7 & -\mathbf{W}_6 \\ \mathbf{W}_2 & \mathbf{W}_3 & \mathbf{W}_0 & -\mathbf{W}_1 & -\mathbf{W}_6 & -\mathbf{W}_7 & \mathbf{W}_4 & \mathbf{W}_5 \\ \mathbf{W}_3 & -\mathbf{W}_2 & \mathbf{W}_1 & \mathbf{W}_0 & -\mathbf{W}_7 & \mathbf{W}_6 & -\mathbf{W}_5 & \mathbf{W}_4 \\ \mathbf{W}_4 & \mathbf{W}_5 & \mathbf{W}_6 & \mathbf{W}_7 & \mathbf{W}_0 & -\mathbf{W}_1 & -\mathbf{W}_2 & -\mathbf{W}_3 \\ \mathbf{W}_5 & -\mathbf{W}_4 & \mathbf{W}_7 & -\mathbf{W}_6 & \mathbf{W}_1 & \mathbf{W}_0 & \mathbf{W}_3 & -\mathbf{W}_2 \\ \mathbf{W}_6 & -\mathbf{W}_7 & -\mathbf{W}_4 & \mathbf{W}_5 & \mathbf{W}_2 & -\mathbf{W}_3 & \mathbf{W}_0 & \mathbf{W}_1 \\ \mathbf{W}_7 & \mathbf{W}_6 & -\mathbf{W}_5 & -\mathbf{W}_4 & \mathbf{W}_3 & \mathbf{W}_2 & -\mathbf{W}_1 & \mathbf{W}_0\end{bmatrix} * \begin{bmatrix}\mathbf{x}_0 \\ \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \\ \mathbf{x}_4 \\ \mathbf{x}_5 \\ \mathbf{x}_6 \\ \mathbf{x}_7\end{bmatrix}$

Journal Pre-proof

Appendix 2 The matrix **U**:

$$U_{rr}=\sqrt{V_{rr}};\ U_{ri}=\frac{V_{ri}}{U_{rr}};\ U_{rj}=\frac{V_{rj}}{U_{rr}};\ U_{rk}=\frac{V_{rk}}{U_{rr}};\ U_{re}=\frac{V_{re}}{U_{rr}};\ U_{rl}=\frac{V_{rl}}{U_{rr}};\ U_{rm}=\frac{V_{rm}}{U_{rr}};\ U_{rn}=\frac{V_{rn}}{U_{rr}};$$

$$U_{ii}=\sqrt{V_{ii}-U_{ri}^2};\ U_{ij}=\frac{1}{U_{ii}}(V_{ij}-(U_{ri}\,U_{rj}));\ U_{ik}=\frac{1}{U_{ii}}(V_{ik}-(U_{rj}\,U_{rk}));\ U_{ie}=\frac{1}{U_{ii}}(V_{ie}-(U_{rk}\,U_{re}));\ U_{il}=\frac{1}{U_{ii}}*(V_{il}-(U_{re}\,U_{rl}));$$

$$U_{im}=\frac{1}{U_{ii}}(V_{im}-(U_{rl}U_{rm}));\ U_{in}=\frac{1}{U_{ii}}(V_{in}-(U_{rm}\,U_{rn}))$$

$$U_{jj}=\sqrt{V_{jj}-(U_{ij}^2+U_{rj}^2)};\ U_{jk}=\frac{1}{U_{jj}}(V_{jk}-(U_{ij}\,U_{ik}+U_{rj}\,U_{rk}));\ U_{je}=\frac{1}{U_{jj}}(V_{je}-(U_{ik}\,U_{ie}+U_{rk}\,U_{re}));\ U_{jl}=\frac{1}{U_{jj}}(V_{jl}-(U_{ie}\,U_{il}+U_{re}\,U_{rl}));$$

$$U_{jm}=\frac{1}{U_{jj}}(V_{jm}-(U_{il}\,U_{im}+U_{rl}\,U_{rm}));\ U_{jn}=\frac{1}{U_{jj}}(V_{jn}-(U_{im}\,U_{in}+U_{rm}\,U_{rn}));$$

$$U_{kk}=\sqrt{V_{kk}-(U_{jk}^2+U_{ik}^2+U_{rk}^2)};\ U_{ke}=\frac{1}{U_{kk}}(V_{ke}-(U_{jk}\,U_{je}+U_{ik}\,U_{ie}+U_{rk}\,U_{re}));\ U_{kl}=\frac{1}{U_{kk}}(V_{kl}-(U_{je}\,U_{jl}+U_{ie}\,U_{il}+U_{re}\,U_{rl}));$$

$$U_{km}=\frac{1}{U_{kk}}(V_{km}-(U_{jl}\,U_{jm}+U_{il}\,U_{im}+U_{rl}\,U_{rm}));\ U_{kn}=\frac{1}{U_{kk}}(V_{kn}-(U_{jm}\,U_{jn}+U_{im}\,U_{in}+U_{rm}\,U_{rn}));$$

$$U_{ee}=\sqrt{V_{ee}-(U_{ke}^2+U_{je}^2+U_{ie}^2+U_{re}^2)};\ U_{el}=\frac{1}{U_{ee}}(V_{el}-(U_{ke}\,U_{kl}+U_{je}\,U_{jl}+U_{ie}\,U_{il}+U_{re}\,U_{rl}));$$

$$U_{em}=\frac{1}{U_{ee}}(V_{em}-(U_{kl}\,U_{km}+U_{jl}\,U_{jm}+U_{il}\,U_{im}+U_{rl}\,U_{rm}));\ U_{en}=\frac{1}{U_{ee}}(V_{en}-(U_{km}\,U_{kn}+U_{jm}\,U_{jn}+U_{im}\,U_{in}+U_{rm}\,U_{rn}));$$

$$U_{ll}=\sqrt{V_{ll}-(U_{el}^2+U_{kl}^2+U_{jl}^2+U_{il}^2+U_{rl}^2)};\ U_{lm}=\frac{1}{U_{ll}}(V_{lm}-(U_{el}\,U_{em}+U_{kl}\,U_{km}+U_{jl}\,U_{jm}+U_{il}\,U_{im}+U_{rl}\,U_{rm}));$$

$$U_{ln}=\frac{1}{U_{ll}}(V_{ln}-(U_{em}\,U_{en}+U_{km}\,U_{kn}+U_{jm}\,U_{jn}+U_{im}\,U_{in}+U_{rm}\,U_{rn}));$$

$$U_{mm}=\sqrt{V_{mm}-(U_{lm}^2+U_{em}^2+U_{km}^2+U_{jm}^2+U_{im}^2+U_{rm}^2)};\ U_{mn}=\frac{1}{U_{mm}}(V_{mn}-(U_{lm}\,U_{ln}+U_{em}\,U_{en}+U_{km}\,U_{kn}+U_{jm}\,U_{jn}+U_{im}\,U_{in}+U_{rm}\,U_{rn}));$$

$$U_{nn}=\sqrt{V_{nn}-(U_{mn}^2+U_{ln}^2+U_{en}^2+U_{kn}^2+U_{jn}^2+U_{in}^2+U_{rn}^2)}.$$

Journal Pre-proof