I.J. Intelligent Systems and Applications, **2018, 7, 11-26**
Published Online July 2018 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijisa.2018.07.02

On the Root-Power Mean Aggregation Based Neuron in Quaternionic Domain

Sushil Kumar
Harcourt Butler Technical University/Department of Computer Science & Engineering, Kanpur, 208002, India
E-mail: sushil0402k5@gmail.com

Bipin K. Tripathi
Harcourt Butler Technical University/Department of Computer Science & Engineering, Kanpur, 208002, India
E-mail: abkt.iitk@gmail.com

Received: 27 June 2017; Accepted: 15 September 2017; Published: 08 July 2018

***Abstract*—**This paper illustrates the new structure of artificial neuron based on root-power means (RPM) for quaternionic-valued signals and also presented an efficient learning process of neural networks with quaternionic-valued root-power means neurons (ℍ-RPMN). The main aim of this neuron is to present the potential capability of a nonlinear aggregation operation on the quaternionic-valued signals in neuron cell. A wide spectrum of aggregation ability of RPM in between minima and maxima has a beautiful property of changing its degree of compensation in the natural way which emulates the various existing neuron models as its special cases. Further, the quaternionic resilient propagation algorithm (ℍ-RPROP) with error-dependent weight backtracking step significantly accelerates the training speed and exhibits better approximation accuracy. The wide spectrums of benchmark problems are considered to evaluate the performance of proposed quaternionic root-power mean neuron with ℍ-RPROP learning algorithm.

***Index Terms*—**Quasi-arithmetic means, Root-power means in quaternionic domain (ℍ), Quaternionic-valued multilayer perceptron, Quaternionic-valued backpropagation, Quaternionic resilient propagation, 3D face recognition.

I. INTRODUCTION

The information processing in cell body is an important function of a neuron, which emulates the computational power of a neuron [1-4]. In last few years, various neuro-computing researchers have confirmed the computational capability of a neuron with nonlinear aggregation operations on synaptic inputs [1, 2, 5-8] and presented various higher order neurons based on the nonlinear correlation among different impinging signals. These attempts resulted in the various class of neural structure as pi-sigma [9, 10] second order neuron [11], compensatory neuron [12], and other higher order neurons [13-16, 54]. However, the higher order neurons have proved to be efficient, but they face the problem of explosion of terms as the number of inputs increases hence demanding sparseness in representation. The problem worsens when neurons are implemented in high dimension. It is highly demanding to investigate a neuron model like a conventional neuron in higher dimension but is free from the problem of higher order neurons. This paper presents a neuron model with a complete specification for quaternionic-valued that employs the nonlinear correlation among input components, but it is free from above problem even when there is an increase in the degree of approximation. The corresponding neural network with learning algorithm in the quaternionic domain (ℍ) provides a better learning and generalization opportunity for problems in three or four dimension. The weighted root-power mean covers the various classes of aggregation in the interval between minima to maxima operations [18, 19]. It provides the flexibility to approximate appropriate operation in the wide range of aggregation through variation of power coefficient. The weighted root-power mean as an aggregation function of the proposed neuron model with quaternionic-valued signals exhibits the natural and general model that presents the various existing neuron models as its special cases, depending on the domain of input signals and value of power coefficient. However, the quaternionic-valued networks with conventional neurons are used in PolSAR Land classification [55] and spoken language understanding [53].

The backpropagation (BP) learning algorithm has gained popularly due to its simplicity, but the slower convergence and getting stuck into local minima are the major weaknesses for degrading its performance. Therefore, some of other proposals have been given, like modified error function [20, 21], addition of variable learning rate [22, 23], addition of momentum [24, 25], delta-bar-delta algorithm [26, 27], Levenberg Marquardt (LM) algorithm [51], GA-MLP hybrid algorithm [56], and quick prop [28], to overcome the above issues, but they have not accelerated the convergence to a significant amount. The fast convergence with efficacious performance along with less complexity of neural network is the important matter for the variety of applications. The resilient propagation algorithm

(RPROP) has been shown the extreme learning capability in real [29-31], complex domain [17, 32], and quaternionic domain [52]. The RPROP was developed for faster convergence which proved its learning and generalization capabilities in many applications such as weighted geometric dilution of precision and mobile location [33], speech quality prediction [34]. This algorithm eliminates the harmful influences of the size of the partial derivative of error function on weight update because adaptation depends on the signs of consecutive partial derivatives.

The RPROP in the quaternionic domain (ℍ-RPROP) has been thoroughly investigated with proposed quaternionic-valued root power mean neuron (ℍ-RPMN) and compared with conventional neuron and BP learning procedure. In the derivation of learning rules, a bounded but non-holomorphic activation function in the quaternionic domain is incorporated. The fusion of quaternionic domain efficient training algorithm (ℍ-RPROP) with ℍ-RPMN demonstrates the drastic reduction in learning cycles along with better generalization in simulation results. This paper is organized as follows. Section II presents a new structure of neuron in the quaternionic domain that covers the many neuron models as its special cases. The weight update rules for a three layer network based on ℍ-RPMN are derived by quaternionic backpropagation (ℍ-BP). In Section III, the weight update rules for a three layer network based on ℍ-RPMN are derived by quaternionic backpropagation (ℍ-BP). Section IV presents the modified ℍ-RPROP algorithm with pseudo code. In Section V, the comparative performance capability the proposed algorithm over existing algorithms is demonstrated and also considered 3D human face recognition problem as a biometric application. Finally, the conclusion and future scope are presented in Section VI.

II. Root Power Mean Neuron Model

In the variety of researches, it is observed that the approximation capabilities of an artificial neuron are governed through spatial aggregation input signals [1, 2, 5-7]. This paper focuses on the design and assessment of a generalized artificial neuron whose aggregation operation provides a wide spectrum of approximation in between minima to maxima. The aggregation operation for proposed neuron is conceptually based on weighted root-power means of input signals, which belongs to the family of quasi-arithmetic means [35, 36] and expressed as:

$$\mathcal{M}_{\varphi}\left(x_1,\dots,x_n;\omega_1,\dots,\omega_n\right)=\varphi^{-1}\left(\sum_{i=1}^{n}\omega_i\varphi\left(x_i\right)\right) \qquad (1)$$

where, $\varphi:\mathbb{R}^n\to\mathbb{R}$ is a continuous and strictly monotonic function which is a generator of quasi-arithmetic mean ($\mathcal{M}_{\varphi}$). $\mathbb{R}$ is the set of real numbers. The φ^{-1} is the inverse of function $\varphi: x\to x^{\alpha}$, where, $\alpha\in\mathbb{R}-\{0\}$, thus weight root-power mean [18] is defined as

$$\mathcal{M}\left(x_1,\dots,x_n;\omega_1,\dots,\omega_n;\alpha\right)=\left(\sum_{i=1}^{n}\omega_i x_i^{\alpha}\right)^{\frac{1}{\alpha}}. \qquad (2)$$

This function models a compensative operation to adjust the degree of approximation by varying the power coefficient α, from $-\infty$ (minimum) to $+\infty$ (maximum). The specific power coefficients determine various classical means. When $\alpha\to 0$ then $\mathcal{M}$ converges to geometric mean and when $\alpha=-1,1,2$ then $\mathcal{M}$ acts as harmonic, arithmetic, and quadratic means, respectively. Most of the common averaging operations come under the family of root-power mean (2), which motivated us to define new aggregation function for a neuron in a quaternionic domain (ℍ). This neuron is named as ℍ-RPMN whose net potential (V) is defined as follows:

$$V\left(q_0,\dots,q_n;w_0,\dots,w_n;\alpha\right)=\left(\sum_{i=1}^{n}w_i\otimes q_i^{\alpha}\right)^{\frac{1}{\alpha}} \qquad (3)$$

where, $V,w_i,q_i\in\mathbb{H}$ and $\alpha\in\mathbb{R}$. The ⊗ symbol denotes the quaternion multiplication which satisfies Hamilton's properties as $i^2=j^2=k^2=ijk=-1$, $ij=-ji=k$, $jk=-kj=i$, and $ki=-ik=j$ [37]. The net potential through quaternionic variables is further transformed through quaternionic-valued split activation function $f_{\mathbb{H}}\left(f_{\mathbb{H}}:\mathbb{H}\to\mathbb{H}\right)$. This presents the four-dimensional extension with the suitable real activation function f ('split-quaternion') [38]. This idea has been motivated from split-type action function in the complex domain whose approximation capabilities has been thoroughly justified [12, 17, 39]. The split-type activation function in the quaternionic domain is bounded but non-holomorphic since Cauchy–Riemann-Fueter (CRF) condition does not hold for it [40]. Let V be a quaternionic variable, expressed as:

$$V=\Re(V)+\Im_1(V)i+\Im_2(V)j+\Im_3(V),$$

then $f_{\mathbb{H}}(V)$ is defined as:

$$\begin{aligned}f_{\mathbb{H}}(V)&=f\left(\Re(V)\right)+f\left(\Im_1(V)\right)i\\&+f\left(\Im_2(V)\right)j+f\left(\Im_3(V)\right)k\end{aligned} \qquad (4)$$

where $\Re$, $\Im_1$, $\Im_2$, and $\Im_3$ denote a real and other three imaginary components of a quaternionic variable respectively. The various types of nonlinear real valued activation functions f have been defined in the various literatures. The choice of a suitable activation function depends on the intended applications. The output of proposed ℍ-RPMN can be defined as:

$$Y\left(q_0,..,q_n;w_0,..,w_n;\alpha\right)=f_{\mathbb{H}}\left(\left(\sum_{i=0}^{n}w_i\otimes q_i^{\alpha}\right)^{\frac{1}{\alpha}}\right) \quad (5)$$

where, the power coefficient α provides the wide ranging functionality of ℍ-RPMN. The many existing neuron models can be realized as special cases of (5) by substituting the specific value of power coefficient. Some special types of existing neuron models are extracted from (5). On substituting $\alpha=1$ in (5), then the outputs of neuron belong to the conventional type of neuron model proposed in the real [41], complex [39, 42], and quaternionic domain [43] as cases apply in consideration of imaginary components. If all the parameters are in the complex domain then the output of a neuron belongs to the ℂ-RPMN model in the complex domain, whose functional capabilities are investigated in [17]. The multiplicative neuron model [44], whose capability has been proven there, can be realized by (5) when $\alpha\to 0$. The harmonic neuron model [45] can be obtained when substituting $\alpha=-1$ in (5). Similarly, when $\alpha=2$ and all variables are in the real domain, then the output from (5) is conceptually similar to the quadratic neuron model proposed in [46].

III. Backpropagation Learning Of ℍ-rpmn Model

The multilayer network with proposed neuron model is similar to the network of the conventional neuron. Consider a three-layer (*L*-*M*-*N*) network of ℍ-RPMN where the first layer possesses *L* (*l*=1,…,*L*) inputs, and second and third layer contain *M* (*m*=1,…,*M*) and *N* (*n*=1,…,*N*) neurons respectively. All weights, biases, and input-output signals are quaternionic numbers. Let $Q=[q_1,q_2,\ldots,q_L]$ be the vector of quaternionic input q and $\bar{q}$ be the conjugate. Let f be the real-valued activation function and $f^{'}$ be its derivative and $\eta\in[0,1]$ be the learning rate. Let weight w_{lm} be from l th input to m th hidden neuron and w_{mn} be from m th hidden neuron to n th output neuron of the network. Let w_0 be the bias weight and q_0 be the bias input. Let V be net internal potential of a neuron and output can be computed through the split-quaternion activation function $f_{\mathbb{H}}$. For m th hidden neuron V_m, then output

$$Y_m=f_{\mathbb{H}}\left(V_m\right)=f_{\mathbb{H}}\left(\left(\sum_{l=0}^{L}w_{lm}\otimes q_l^{\alpha}\right)^{\frac{1}{\alpha}}\right). \quad (6)$$

Let $u_m=\sum_{l=0}^{L}w_{lm}\otimes q_l^{\alpha}$ and $\vec{V}_{(u_m)}$ is the vector component of u_m then u_m can be expressed as a real and its vector part together as $\Re\left(u_m\right)+\vec{V}_{(u_m)}$ and it can be represented in the polar form of a quaternion [47] as

$$\left\|u_m\right\|\left(\cos\left(\theta_{(u_m)}\right)+\frac{\vec{V}_{(u_m)}}{\left\|\vec{V}_{(u_m)}\right\|}\sin\left(\theta_{(u_m)}\right)\right) \quad (7)$$

where,

$$\theta_{(u_m)}=\tan^{-1}\left(\left\|\vec{V}_{(u_m)}\right\|\big/\Re\left(u_m\right)\right),$$

$$\left\|u_m\right\|^2=\left(\Re\left(u_m\right)\right)^2+\left\|\vec{V}_{(u_m)}\right\|^2, \text{ and}$$

$$\left\|\vec{V}_{(u_m)}\right\|^2=\sum_{i=1}^{3}\left(\Im_i\left(u_m\right)\right)^2.$$

$\left\|u_m\right\|$ and $\left\|\vec{V}_{(u_m)}\right\|$ denote the norm and the magnitude of the vector component ($\vec{V}_{(u_m)}$) of u_m respectively. The net internal potential V_m of m th hidden neuron can be obtained by applying De Moivre's theorem on u_m [48] as

$$V_m=\left\|u_m\right\|^{\frac{1}{\alpha}}\left(\cos\left(\frac{\theta_{(u_m)}}{\alpha}\right)+\frac{\vec{V}_{(u_m)}}{\left\|\vec{V}_{(u_m)}\right\|}\sin\left(\frac{\theta_{(u_m)}}{\alpha}\right)\right). \quad (8)$$

The output of the n th neuron in output layer can be expressed as similar to (6) as

$$Y_n=f_{\mathbb{H}}\left(V_n\right)=f_{\mathbb{H}}\left(\left(\sum_{m=0}^{M}w_{mn}\otimes Y_m^{\alpha}\right)^{\frac{1}{\alpha}}\right). \quad (9)$$

Let $u_n=\sum_{m=0}^{M}w_{mn}\otimes Y_m^{\alpha}$ and V_n be the net internal potential of n th neuron which can be derived as similar to (8) as

$$V_n=\left\|u_n\right\|^{\frac{1}{\alpha}}\left(\cos\left(\frac{\theta_{(u_n)}}{\alpha}\right)+\frac{\vec{V}_{(u_n)}}{\left\|\vec{V}_{(u_n)}\right\|}\sin\left(\frac{\theta_{(u_n)}}{\alpha}\right)\right). \quad (10)$$

The gradient-descent-based error backpropagation learning scheme for the feed-forward neural network has been extended in the quaternionic domain. Let error $e_n=D_n-Y_n$ be the difference between desired (D_n) and actual (Y_n) output of n th neuron at the output layer. The weight update formula can be derived by minimizing the real-valued error function (E) as

$$\begin{aligned} E &= \frac{1}{2N}\sum_{n=1}^{N}\|e_n\|^2 \\ &= \frac{1}{2N}\sum_{n=1}^{N}\left\{\left(\Re(e_n)\right)^2 + \sum_{i=1}^{3}\left(\Im_i(e_n)\right)^2\right\} \end{aligned} . \quad (11)$$

The real-valued error function does not follow the Cauchy-Riemann condition; therefore, it is not holomorphic. The error function is minimized by recursively altering the weight coefficients as:

$$w^{new} = w^{old} - \eta\nabla_w(E) \quad (12)$$

where, $\nabla_w(E)$ presents the gradient of the error function which is derived with respect to real and other three imaginary components of quaternionic weights. The weight update (Δw) is proportional to the negative gradient of the error function with respect to quaternionic weight as

$$\begin{aligned} \Delta w = -\eta\nabla_w(E) = -\eta\left\{\frac{\partial E}{\partial\Re(w)} + \frac{\partial E}{\partial\Im_1(w)}i\right. \\ \left. + \frac{\partial E}{\partial\Im_2(w)}j + \frac{\partial E}{\partial\Im_3(w)}k\right\} \end{aligned} . \quad (13)$$

For the weight ($w = w_{mn}$) that connects m th hidden neuron to n th output neuron, the weight update is obtained using chain rule of derivation as:

$$\begin{aligned} \Delta w_{mn} = \frac{\eta}{N}\left\{\Re(e_n) f'\left(\Re(V_n)\right)\nabla_{w_{mn}}\left(\Re(V_n)\right)\right. \\ \left. + \sum_{i=1}^{3}\Im_i(e_n) f'\left(\Im_i(V_n)\right)\nabla_{w_{mn}}\left(\Im_i(V_n)\right)\right\} \end{aligned} . \quad (14)$$

The weight update (Δw_{mn}) depends on gradients of each component of the net potential (V_n) of n th output neuron with respect to weight (w_{mn}). The gradient of the real component of the net potential can be obtained using real part of (8) e.g. $\|u_n\|^{\frac{1}{\alpha}}\cos\left(\frac{\theta_{(u_n)}}{\alpha}\right)$ as

$$\begin{aligned} \nabla_{w_{mn}}\left(\Re(V_n)\right) = \frac{\|u_n\|^{\frac{1}{\alpha}-2}}{\alpha}\left\{A_1\Re(u_n)\right. \\ \left. \nabla_{w_{mn}}\left(\Re(u_n)\right) + B_1\Im_j(u_n)\nabla_{w_{mn}}\left(\Im_j(u_n)\right)\right\} \end{aligned} \quad (15)$$

and the gradients of three imaginary components of the net potential ($\Im_i(V_n), i = 1,2,3$) can be obtained as:

$$\begin{aligned} \nabla_{w_{mn}}\left(\Im_i(V_n)\right) = \frac{\|u_n\|^{\frac{1}{\alpha}-2}}{\alpha\left\|\vec{V}_{(u_n)}\right\|}\Im_i(u_n)\left\{A_2\Re(u_n)\right. \\ \left. \times\nabla_{w_{mn}}\left(\Re(u_n)\right) + B_2\sum_{j=1}^{3}\Im_j(u_n)\nabla_{w_{mn}}\left(\Im_j(u_n)\right)\right\} \\ + \frac{\|u_n\|^{\frac{1}{\alpha}}}{\left\|\vec{V}_{(u_n)}\right\|}\sin\left(\frac{\theta_{(u_n)}}{\alpha}\right)\nabla_{w_{mn}}\left(\Im_i(u_n)\right) \\ - \frac{\|u_n\|^{\frac{1}{\alpha}}}{\left\|\vec{V}_{(u_n)}\right\|^3}\Im_i(u_n)\sin\left(\frac{\theta_{(u_n)}}{\alpha}\right) \\ \times\left\{\sum_{j=1}^{3}\Im_j(u_n)\nabla_{w_{mn}}\left(\Im_j(u_n)\right)\right\} \end{aligned} \quad (16)$$

where,

$$\|u_n\|^2 = \left(\Re(u_n)\right)^2 + \left\|\vec{V}_{(u_n)}\right\|^2,$$

$$\left\|\vec{V}_{(u_n)}\right\|^2 = \sum_{i=1}^{3}\left(\Im_i(u_n)\right)^2,$$

$$A_1 = \cos\left((1-1/\alpha)\theta_{(u_n)}\right)\Big/\cos\left(\theta_{(u_n)}\right),$$

$$B_1 = \sin\left((1-1/\alpha)\theta_{(u_n)}\right)\Big/\sin\left(\theta_{(u_n)}\right),$$

$$A_2 = -\sin\left((1-1/\alpha)\theta_{(u_n)}\right)\Big/\cos\left(\theta_{(u_n)}\right),$$

$$B_2 = \cos\left((1-1/\alpha)\theta_{(u_n)}\right)\Big/\sin\left(\theta_{(u_n)}\right), \text{ and}$$

$$\theta_{(u_n)} = \tan^{-1}\left(\left\|\vec{V}_{(u_n)}\right\|\Big/\Re(u_n)\right).$$

The gradients of real and three imaginary components of u_n with respect to the weight (w_{mn}) can be obtained as:

$$\nabla_{w_{mn}}\left(\Re(u_n)\right) = \left(\bar{Y}_m\right)^\alpha; \quad (17)$$

$$\nabla_{w_{mn}}\left(\Im_1(u_n)\right) = i\otimes\left(\bar{Y}_m\right)^\alpha; \quad (18)$$

$$\nabla_{w_{mn}}\left(\Im_2(u_n)\right) = j\otimes\left(\bar{Y}_m\right)^\alpha; \quad (19)$$

$$\nabla_{w_{mn}}\left(\Im_3(u_n)\right) = k\otimes\left(\bar{Y}_m\right)^\alpha. \quad (20)$$

For the weight ($w = w_{lm}$) that connects l th input to m th hidden neuron, the weight update

$$\Delta w_{lm} = \frac{\eta}{N}\sum_{n=1}^{N}\Big\{\Re(e_n)\, f'\big(\Re(V_n)\big)\nabla_{w_{lm}}\big(\Re(V_n)\big) + \sum\nolimits_{i=1}^{3}\Im_i(e_n)\, f'\big(\Im_i(V_n)\big)\nabla_{w_{lm}}\big(\Im_i(V_n)\big)\Big\}. \tag{21}$$

The weight update (Δw_{lm}) between l th input and m th hidden neuron depends on gradients of each component of the net potential (V_n) of n th output neuron with respect to weight (w_{lm}). The gradient of the real component of the net potential can be obtained as:

$$\nabla_{w_{lm}}\big(\Re(V_n)\big) = \frac{\|u_n\|^{\frac{1}{\alpha}-2}}{\alpha}\Big\{A_1\Re(u_n)\nabla_{w_{lm}}\big(\Re(u_n)\big) + B_1\sum\nolimits_{j=1}^{3}\Im_j(u_n)\nabla_{w_{lm}}\big(\Im_j(u_n)\big)\Big\} \tag{22}$$

and the gradients of three imaginary components of the net potential ($\Im_i(V_n), i = 1,2,3$) can be obtained as:

$$\begin{aligned}\nabla_{w_{lm}}\big(\Im_i(V_n)\big) &= \frac{\|u_n\|^{\frac{1}{\alpha}-2}}{\alpha\|\vec{V}_{(u_n)}\|}\Im_i(u_n)\Big\{A_2\Re(u_n) \\ &\times\nabla_{w_{lm}}\big(\Re(u_n)\big) + B_2\sum\nolimits_{j=1}^{3}\Im_j(u_n)\nabla_{w_{lm}}\big(\Im_j(u_n)\big)\Big\} \\ &+\frac{\|u_n\|^{\frac{1}{\alpha}}}{\|\vec{V}_{(u_n)}\|}\sin\left(\frac{\theta_{(u_n)}}{\alpha}\right)\nabla_{w_{lm}}\big(\Im_i(u_n)\big) \\ &-\frac{\|u_n\|^{\frac{1}{\alpha}}}{\|\vec{V}_{(u_n)}\|^3}\Im_i(u_n)\sin\left(\frac{\theta_{(u_n)}}{\alpha}\right) \\ &\times\Big\{\sum\nolimits_{j=1}^{3}\Im_j(u_n)\nabla_{w_{lm}}\big(\Im_j(u_n)\big)\Big\}\end{aligned} \tag{23}$$

The gradients of real and three imaginary components of u_n with respect to the weight (w_{lm}) can be obtained as:

$$\begin{aligned}\nabla_{w_{lm}}\big(\Re(u_n)\big) &= \Re(w_{mn})\nabla_{w_{lm}}\big(\Re(Y_m^{\alpha})\big) \\ &-\Im_1(w_{mn})\nabla_{w_{lm}}\big(\Im_1(Y_m^{\alpha})\big) \\ &-\Im_2(w_{mn})\nabla_{w_{lm}}\big(\Im_2(Y_m^{\alpha})\big) \\ &-\Im_3(w_{mn})\nabla_{w_{lm}}\big(\Im_3(Y_m^{\alpha})\big)\end{aligned} \tag{24}$$

$$\begin{aligned}\nabla_{w_{lm}}\big(\Im_1(u_n)\big) &= \Re(w_{mn})\nabla_{w_{lm}}\big(\Im_1(Y_m^{\alpha})\big) \\ &+\Im_1(w_{mn})\nabla_{w_{lm}}\big(\Re(Y_m^{\alpha})\big) \\ &+\Im_2(w_{mn})\nabla_{w_{lm}}\big(\Im_3(Y_m^{\alpha})\big) \\ &-\Im_3(w_{mn})\nabla_{w_{lm}}\big(\Im_2(Y_m^{\alpha})\big)\end{aligned} \tag{25}$$

$$\begin{aligned}\nabla_{w_{lm}}\big(\Im_2(u_n)\big) &= \Re(w_{mn})\nabla_{w_{lm}}\big(\Im_2(Y_m^{\alpha})\big) \\ &-\Im_1(w_{mn})\nabla_{w_{lm}}\big(\Im_3(Y_m^{\alpha})\big) \\ &+\Im_2(w_{mn})\nabla_{w_{lm}}\big(\Re(Y_m^{\alpha})\big) \\ &+\Im_3(w_{mn})\nabla_{w_{lm}}\big(\Im_1(Y_m^{\alpha})\big)\end{aligned} \tag{26}$$

$$\begin{aligned}\nabla_{w_{lm}}\big(\Im_3(u_n)\big) &= \Re(w_{mn})\nabla_{w_{lm}}\big(\Im_3(Y_m^{\alpha})\big) \\ &+\Im_1(w_{mn})\nabla_{w_{lm}}\big(\Im_2(Y_m^{\alpha})\big) \\ &-\Im_2(w_{mn})\nabla_{w_{lm}}\big(\Im_1(Y_m^{\alpha})\big) \\ &+\Im_3(w_{mn})\nabla_{w_{lm}}\big(\Re(Y_m^{\alpha})\big)\end{aligned} \tag{27}$$

The gradient of real component of Y_m^{α} can be obtained as:

$$\begin{aligned}\nabla_{w_{lm}}\big(\Re(Y_m^{\alpha})\big) &= \alpha\|Y_m\|^{\alpha-2}\Big\{A_3\Re(Y_m)\, f'\big(\Re(V_m)\big) \\ &\times\nabla_{w_{lm}}\big(\Re(V_m)\big) + B_3 \\ &\times\sum\nolimits_{j=1}^{3}\Im_j(Y_m)\, f'\big(\Im_j(V_m)\big)\nabla_{w_{lm}}\big(\Im_j(V_m)\big)\Big\}\end{aligned} \tag{28}$$

and the gradients of three imaginary components of the net potential ($\Im_i(Y_m^{\alpha}), i = 1,2,3$) can be obtained as:

$$\begin{aligned}\nabla_{w_{lm}}\big(\Im_i(Y_m^{\alpha})\big) &= \frac{\alpha\|Y_m\|^{\alpha-2}}{\|\vec{V}_{(Y_m)}\|}\Im_i(Y_m) \\ &\times\left\{\begin{aligned}&A_4\Re(Y_m)\, f'\big(\Re(V_m)\big)\nabla_{w_{lm}}\big(\Re(V_m)\big) \\ &+B_4\sum_{j=1}^{3}\Im_j(Y_m)\, f'\big(\Im_j(V_m)\big)\nabla_{w_{lm}}\big(\Im_j(V_m)\big)\end{aligned}\right\} \\ &+\frac{\|Y_m\|^{\alpha}}{\|\vec{V}_{(Y_m)}\|}\sin\big(\alpha\theta_{(Y_m)}\big)\, f'\big(\Im_i(V_m)\big)\nabla_{w_{lm}}\big(\Im_i(V_m)\big) \\ &-\frac{\|Y_m{}^{\alpha}\|}{\|\vec{V}_{(Y_m)}\|^3}\Im_i(Y_m)\sin\big(\alpha\theta_{(Y_m)}\big) \\ &\times\Big\{\sum\nolimits_{j=1}^{3}\Im_j(Y_m)\, f'\big(\Im_j(V_m)\big)\nabla_{w_{lm}}\big(\Im_j(V_m)\big)\Big\}\end{aligned} \tag{29}$$

where,

$$\|Y_m\|^2 = \big(\Re(Y_m)\big)^2 + \|\vec{V}_{(Y_m)}\|^2,$$

$$\|\vec{V}_{(Y_m)}\|^2 = \sum\nolimits_{i=1}^{3}\big(\Im_i(Y_m)\big)^2,$$

$$A_3 = \cos\left(\left(1-\alpha\right)\theta_{(Y_m)}\right)/\cos\left(\theta_{(Y_m)}\right),$$

$$B_3 = \sin\left(\left(1-\alpha\right)\theta_{(Y_m)}\right)/\sin\left(\theta_{(Y_m)}\right),$$

$$A_4 = -\sin\left(\left(1-\alpha\right)\theta_{(Y_m)}\right)/\cos\left(\theta_{(Y_m)}\right),$$

$$A_4 = -\sin\left(\left(1-\alpha\right)\theta_{(Y_m)}\right)\Big/\cos\left(\theta_{(Y_m)}\right),$$

$$B_4 = \cos\left(\left(1-\alpha\right)\theta_{(Y_m)}\right)\Big/\sin\left(\theta_{(Y_m)}\right), \text{ and}$$

$$\theta_{(Y_m)} = \tan^{-1}\left(\left\|\vec{V}_{(Y_m)}\right\|\Big/\Re\left(Y_m\right)\right).$$

The gradient of the real component of the net potential (V_m) can be obtained as:

$$\nabla_{w_{lm}}\left(\Re\left(V_m\right)\right) = \frac{\left\|u_m\right\|^{\frac{1}{\alpha}-2}}{\alpha}\left\{A_5\Re\left(u_m\right)\nabla_{w_{lm}}\left(\Re\left(u_m\right)\right)\right. \\ \left. + B_5\sum_{j=1}^{3}\Im_j\left(u_m\right)\nabla_{w_{lm}}\left(\Im_j\left(u_m\right)\right)\right\} \quad (30)$$

and the gradients of three imaginary components of the net potential ($\Im_i\left(V_m\right), i = 1,2,3$) can be obtained as:

$$\nabla_{w_{lm}}\left(\Im_i\left(V_m\right)\right) = \frac{\left\|u_m\right\|^{\frac{1}{\alpha}-2}}{\alpha\left\|\vec{V}_{(u_m)}\right\|}\Im_i\left(u_m\right) \\ \times\begin{Bmatrix} A_6\Re\left(u_m\right)\nabla_{w_{lm}}\left(\Re\left(u_m\right)\right) \\ +B_6\sum_{j=1}^{3}\Im_j\left(u_m\right)\nabla_{w_{lm}}\left(\Im_j\left(u_m\right)\right)\end{Bmatrix} \\ + \frac{\left\|u_m\right\|^{\frac{1}{\alpha}}}{\left\|\vec{V}_{(u_m)}\right\|}\sin\left(\frac{\theta_{(u_m)}}{\alpha}\right)\nabla_{w_{lm}}\left(\Im_i\left(u_m\right)\right) \\ - \frac{\left\|u_m\right\|^{\frac{1}{\alpha}}}{\left\|\vec{V}_{(u_m)}\right\|^3}\Im_i\left(u_m\right)\sin\left(\frac{\theta_{(u_m)}}{\alpha}\right) \\ \times\left\{\sum_{j=1}^{3}\Im_j\left(u_m\right)\nabla_{w_{lm}}\left(\Im_j\left(u_m\right)\right)\right\} \quad (31)$$

where,

$$\left\|u_m\right\|^2 = \left(\Re\left(u_m\right)\right)^2 + \left\|\vec{V}_{(u_m)}\right\|^2,$$

$$\left\|\vec{V}_{(u_m)}\right\|^2 = \sum_{i=1}^{3}\left(\Im_i\left(u_m\right)\right)^2,$$

$$A_5 = \cos\left(\left(1-1/\alpha\right)\theta_{(u_m)}\right)\Big/\cos\left(\theta_{(u_m)}\right),$$

$$B_5 = \sin\left(\left(1-1/\alpha\right)\theta_{(u_m)}\right)\Big/\sin\left(\theta_{(u_m)}\right),$$

$$A_6 = -\sin\left(\left(1-1/\alpha\right)\theta_{(u_m)}\right)\Big/\cos\left(\theta_{(u_m)}\right),$$

$$B_6 = \cos\left(\left(1-1/\alpha\right)\theta_{(u_m)}\right)\Big/\sin\left(\theta_{(u_m)}\right), \text{ and}$$

$$\theta_{(u_m)} = \tan^{-1}\left(\left\|\vec{V}_{(u_m)}\right\|\Big/\Re\left(u_m\right)\right).$$

The gradients of real and three imaginary components of u_m with respect to the weight (w_{lm}) can be obtained as:

$$\nabla_{w_{lm}}\left(\Re\left(u_m\right)\right) = \left(\bar{q}_l\right)^{\alpha}; \quad (32)$$

$$\nabla_{w_{lm}}\left(\Im_1\left(u_m\right)\right) = i\otimes\left(\bar{q}_l\right)^{\alpha}; \quad (33)$$

$$\nabla_{w_{lm}}\left(\Im_2\left(u_m\right)\right) = j\otimes\left(\bar{q}_l\right)^{\alpha}; \quad (34)$$

$$\nabla_{w_{lm}}\left(\Im_3\left(u_m\right)\right) = k\otimes\left(\bar{q}_l\right)^{\alpha}. \quad (35)$$

IV. ℍ-RPROP Learning of ℍ-RPMN Model

The resilient propagation (RPROP) learning algorithm in the real domain is very popular to exhibit fast and robust learning scheme because only the sign of the partial derivatives in successive steps is used to perform learning and adaptation. It accomplishes the local adaptation of weight updates (Δw) according to the nature error at each iteration to overcome the disadvantages of pure gradient descent approach [29]. The further improvement [30, 31] in this learning process has been obtained by associating *error-dependent weight reversal* step; it appears more logical on the evolution of error when error increases even weight updates have caused changes to the sign of partial derivatives. Thus, in improved RPROP algorithm the individual information about error surface (sign of gradient with respect to a weight) is combined with more global information to settle down for each weight, which individually modifies the real and imaginary components of complex weights by an amount $\Re\left(\Delta(t)\right)$ and $\Im\left(\Delta(t)\right)$ where Δ is update value) to decrease the overall error.

The ℍ-RPROP algorithm in the quaternionic domain is investigated with a goal to modify the real and other three imaginary components of the quaternionic weights by an amount $\Re\left(\Delta(t)\right)$, $\Im_1\left(\Delta(t)\right)$, $\Im_2\left(\Delta(t)\right)$, and $\Im_3\left(\Delta(t)\right)$ with a view to decrease the overall error in learning cycles. Each update value determines the size of weight update. The sign of the partial derivative of the error function with respect to each component of quaternionic weight determines the direction of weight update, where partial derivatives are the gradient summation over all patterns of the pattern set. Let the symbol $\aleph$ denotes the generalized notation for real and other three imaginary components of quaternionic weight. The weight updates for all components are determined as

$$\aleph\left(\Delta w(t)\right) = -\mathrm{sign}\left(\frac{\partial E(t)}{\partial \aleph(w)}\right)\aleph\left(\Delta(t)\right). \tag{36}$$

If the sign of partial derivate changes in successive steps, the previous weight update will be reverted. But the important element is required to investigate whether the error is increasing or decreasing, caused by weight update. This important element has not considered in [29]. This backtracking step does not seem proper especially when the overall error has decreased. Hence, the previous weight update is reverted only when it has caused a change of sign in the corresponding partial derivative in the case of an overall error increase. The results of this algorithm demonstrate the excellent performance than ℍ-BP algorithm. If the partial derivatives in successive step are opposite in sign and the overall error increases then only the previous weight update is reverted as:

$$\begin{aligned} &\text{if}\left\{\frac{\partial E(t-1)}{\partial \aleph(w)} \times \frac{\partial E(t)}{\partial \aleph(w)} < 0 \quad \text{and}\, E(t) > E(t-1)\right\} \\ &\text{then}\ \aleph\left(w(t+1)\right) = \aleph\left(w(t)\right) - \aleph\left(w(t-1)\right). \end{aligned} \tag{37}$$

The partial derivative of error is set to zero after weight reversal because it avoids the update of step size in the next iteration. The ℍ-RPROP algorithm defines various parameters to yield the global minimum of the error function with faster convergence. The parameters, initial step size (Δ_0), increase factor (μ^+), decrease factor (μ^-), minimum step size ($\Delta_{\min}$), and maximum step size ($\Delta_{\max}$), are set at the beginning of ℍ-RPROP. The step size (update value) is modifying according to gradient direction, as given in algorithm. The abstract pseudo code of ℍ-RPROP algorithm is presented as follows:

```
1.  $t = 1; 0 < \mu^- < \mu^+ < 1.2$
2.  $\aleph(\Delta(t)) = \Delta_0; \partial E(t-1)/\partial \aleph(w) = 0;$
    $\aleph(\Delta_{\max}) = \Delta_{\max}; \aleph(\Delta_{\min}) = \Delta_{\min};$
3.  Repeat {Compute $\partial E(t)/\partial \aleph(w)$ for all quaternionic
    weights.
4.    Repeat {
5.      If $\left(\frac{\partial E(t-1)}{\partial \aleph(w)} \times \frac{\partial E(t)}{\partial \aleph(w)} > 0\right)$ then
6.        { $(\Delta(t)) = \min(\mu^+ \aleph(\Delta(t-1)), \aleph(\Delta_{\max}));$
7.        $\aleph(w(t+1)) = \aleph(w(t) - \mathrm{sign}\left(\frac{\partial E(t)}{\partial \aleph(w)}\right)$ }
          $\times \aleph(\Delta(t));$
8.      If $\left(\frac{\partial E(t-1)}{\partial \aleph(w)} \times \frac{\partial E(t)}{\partial \aleph(w)} < 0\right)$ then
9.    { $\aleph(\Delta(t)) = \max(\mu^- \aleph(\Delta(t-1)), \aleph(\Delta_{\min}));$
10.         If $(E(t) > E(t-1))$ then
11.           $\aleph(w(t+1)) = \aleph(w(t) - \aleph(\Delta w(t-1));$
12.     $\frac{\partial E(t)}{\partial \aleph(w)} = 0;$ }
13.     If $\left(\frac{\partial E(t-1)}{\partial \aleph(w)} \times \frac{\partial E(t)}{\partial \aleph(w)} = 0\right)$ then
14.     $\aleph(w(t+1)) = \aleph(w(t) - \mathrm{sign}\left(\frac{\partial E(t)}{\partial \aleph(w)}\right)$
        $\times \aleph(\Delta(t));$
15. } until (completed all components of weight)
16.   $t = t + 1;$
17. } until (converged).
```

All components of quaternionic weights are modified individually according to the algorithm in each learning cycle and such cycle is repeated till error converges to designated level. In this algorithm, the weights and update values are changing every time for the new training set.

V. Performance Evaluations Through Benchmark Problems

In this paper, we present the effectiveness of proposed neuron ℍ-RPMN and algorithm ℍ-RPMN through a wide spectrum of benchmark problems. Comparative evaluation is done by networks designed by conventional neurons (MLP) and root-power-mean neurons along with BP and RPROP learning algorithms in the quaternionic domain. Four components of all quaternionic weights and biases for both networks are randomly initialized in the range -1 to 1. The quaternionic variable $q_0 = 1 + i + j + k$ is assumed as bias input and the hyperbolic tangent function is used as the activation function. The comparison of training and testing performance through function approximation is thoroughly evaluated by statistical parameters like error variance, correlation, and AIC [49]. Another class of benchmark problem is the learning of linear transformations (rotation, scaling, and translation and their combinations) through few set of points lying on line whose generalization abilities are tested over the complicated 3D geometric structure. In the last subsection, two primary experiments are presented for 3D face recognition which surely it will be stepping stone for prospective researchers.

A. Function Approximation of Model for Spread of Tuberculosis

The spread of tuberculosis model [50] is the system of four differential equations with respect to time along with four variables that can be denoted together as a quaternion number instead of four real numbers. This model needs the intelligent behavior and automated analysis of bacterial effect to reach equilibrium from different initial conditions. A mathematical model aims to analyze the effect of the accumulation of bacteria which survive due to conducive ecological factors such as flower pots, plants, grasses, human clothes, etc. in the habitat, acting as a reservoir, on the spread of tuberculosis (TB) in the human population. The total population ($N(t)$) is categorized into classes, susceptible ($S(t)$) and infective ($T(t)$). $B(t)$ governs the bacteria density in the environment and $E(t)$ is the cumulative density of ecological factors which is conducive to the accumulation of bacteria population. All these factors of this model are treated as quaternion ($S(t) + T(t)i + B(t)j + E(t)k$) with respect to time. In this model, it is assumed that TB is spread by direct contacts with infective in the population and indirect contacts with bacteria which is emitted by infective in the habitat. The dynamics of the spread of TB is governed by the system of nonlinear differential equations as:

$$\begin{aligned}
\frac{dS}{dt} &= A-\beta ST-\lambda SB-dS+\nu T \\
\frac{dT}{dt} &= \beta ST+\lambda SB-(\nu+\alpha+d)T \\
\frac{dB}{dt} &= sT-s_0B+s_1BE \\
\frac{dE}{dt} &= \gamma E-\gamma_0E^2+\gamma_1(S+T)E \\
& S(0)>0; T(0)\ge 0; B(0)\ge 0; E(0)\ge 0
\end{aligned} \tag{38}$$

where, A is the immigration rate of susceptibles. β and λ are the transmission coefficients of TB by contact of susceptibles with infectives and by inhalation of bacteria from the environment; d is the natural death rate, ν is the therapeutic treatment rate of infected individuals and α is the death rate due to TB infection. The parameter s is the release rate of bacteria from the TB infected individuals, s_0 is the decrement coefficient due to natural factors and s_1 is the rate of survival and accumulation of bacteria population due to conducive ecological factors in the habitat. γ is the growth rate, γ/γ_0 is the carrying capacity in the habitat and γ_1 is the increase coefficient due to the total human population. All these parameters in this model are assumed only for positive values.

The TB model presented in (38) with parameters [$A=500$, $\beta=0.0003$, $\lambda=0.0001$, $d=0.15$, $\nu=0.01$, $\alpha=0.2$, $s=0.1$, $s_0=0.3$, $s_1=0.0001$, $\gamma=25$, $\gamma_0=0.1$, and $\gamma_1=0.002$] generates four datasets containing 200 data points in 100 hours with a time interval $\Delta t=0.5$ and four initial conditions are given as:

(i) $S(0)=1400$, $T(0)=1600$, $B(0)=358$, $E(0)=290$

(ii) $S(0)=200$, $T(0)=1300$, $B(0)=358$, $E(0)=290$

(iii) $S(0)=2600$, $T(0)=400$, $B(0)=358$, $E(0)=290$

(iv) $S(0)=600$, $T(0)=400$, $B(0)=358$, $E(0)=290$

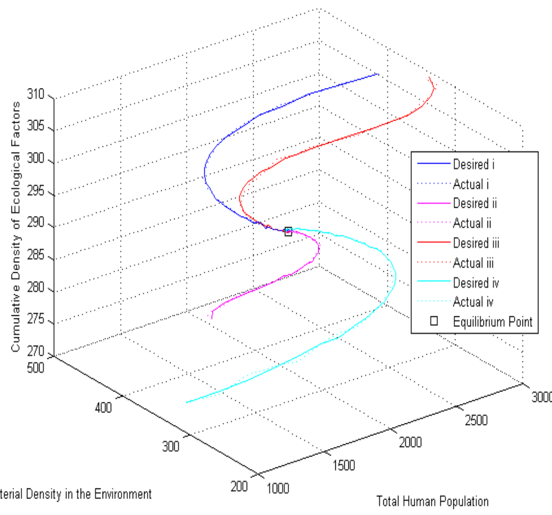


Fig.1. Comparison of testing result of ℍ-RPMN based network trained by ℍ-RPROP algorithm with normalized desired.

The system (38) reaches to an equilibrium point from four different initial conditions. This system generates four datasets containing four components ($S(t)$, $T(t)$, $B(t)$, and $E(t)$) using four initial conditions. All four components are encoded in a single quaternion. Datasets are further normalized between -0.9 to 0.9 and its 80 data points are used for training by ℍ-BP and ℍ-RPROP learning algorithms. The normalized datasets containing 200 points are used for testing of networks trained by both algorithms.

The comparative analysis of training and testing data through conventional and proposed neuron based networks performed by both the algorithms are presented in Table 1. It clearly shows that ℍ-RPROP with ℍ-RPMN has significantly faster convergence and has better testing results in terms of error, variance, correlation, and AIC. Figure 1 demonstrates the testing results by ℍ-RPROP with ℍ-RPMN and compares with the desired result in 3D for different initial conditions, where the normalized total human population ($S(t)+T(t)$), bacterial population density ($B(t)$) and cumulative population density ($E(t)$) are in x, y, and z direction respectively. The overall training and testing performance infer the superiority of ℍ-RPROP algorithm with ℍ-RPMN.

Table 1. Comparison of Training and Testing Performance for the Spread of TB Model

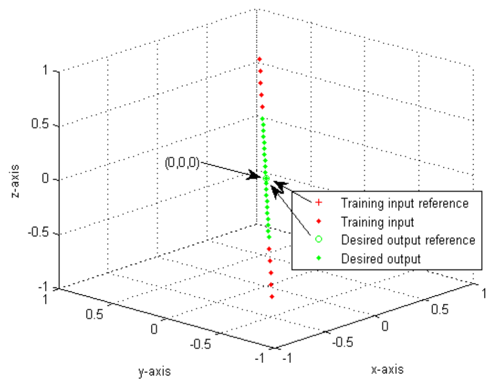
	ℍ-MLP		ℍ-RPMN (α =0.94)	
Learning Algorithm	ℍ-BP (η=0.001)	ℍ-RPROP	ℍ-BP (η=0.001)	ℍ-RPROP
Network	4-8-4	4-8-4	4-8-4	4-8-4
Parameters	76	76	76	76
MSE Training	0.0005	0.0003	0.0004	0.0001
Learning Cycles	6000	2100	5000	1200
Average Training Time (in Minutes)	70	50	60	35
MSE Testing	3.4615e-04	2.1137e-04	2.9362e-04	1.5213e-04
Error Variance	5.8870e-04	6.0118e-04	5.0581e-04	4.9042e-04
Correlation	0.9965	0.9978	0.9970	0.9983
AIC	-6.6409	-7.3517	-7.0640	-8.4503

B. *Linear Transformations*

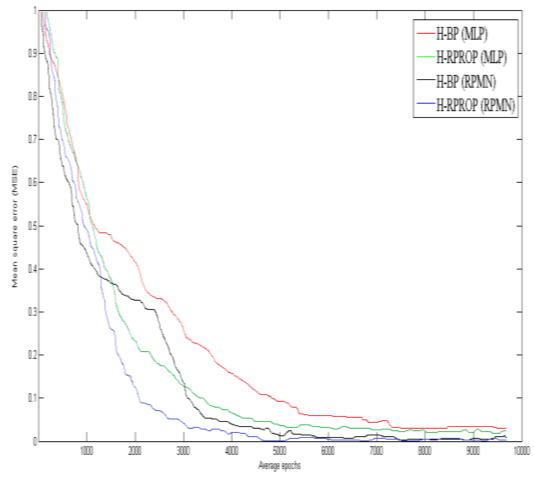
This experiment presents the capability to learn 3D motion patterns through a training set containing points on a line and motion or transformation generalization over complicated geometrical structures in space. As a benchmark problem, this section presents the learning and generalization of linear transformations (rotation, scaling, and translation and their combinations) through ℍ-RPROP and ℍ-BP algorithm for the network based on ℍ-RPMN and ℍ-MLP. This facilitates the viewing of 3D objects from different orientations as well as the interpretation of their motion in space.

We have considered a three layer network (2-4-2) and its learning process for input-output mapping governed by a straight line containing a reference point (like mid of the line) in 3D space for all experiments. The first input receives a set of points that lies on a straight line and second input passes the reference point. The simulation results show that the ℍ-RPROP algorithm with ℍ-RPMN drastically reduces the number of training epochs and also able to generalize more accurately as compared to other algorithms.

The learning of a three layer network is performed for the different class of transformations, which are as follows: The input-output mapping for scaling with factor ½ is shown in Fig. 2(a); scaling with factor ½ followed by 0.3 unit translation along the positive z-direction is shown in Fig. 4(a); and scaling with factor ½ followed by 0.3 unit translation along the positive z-direction and π/2 radian rotation around the unit vector (i) is shown in Fig. 6(a). This mapping is defined by the straight line containing 21 points and referenced at (0, 0, 0), as shown in Figs. 2(a), 4(a), and 6(a). Figures 2(b), 4(b), and 6(b) show the faster learning capability of proposed ℍ-RPROP as compared to ℍ-BP in all three classes of transformations. The network trained by ℍ-RPROP converges to MSE = 0.0007 after 5000 average epochs in case of scaling, 5050 average epochs in case of scaling and translation, and 6020 average epochs in case of scaling, translation and rotation of straight line, but the training through ℍ-BP with conventional neural network requires comparatively larger average epochs then ℍ-BP with ℍ-RPMN neuron to achieve similar MSE, as shown in Figs. 2(b), 4(b), and 6(b), and presented in Tables 2, 3, and 4 respectively. Thus, the convergence of proposed algorithm is extremely faster over conventional ℍ-BP.

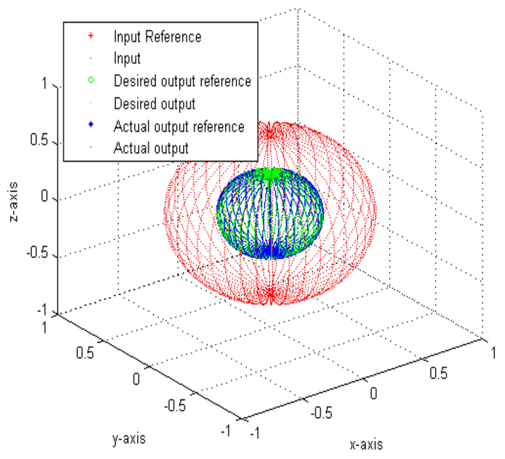


(a)

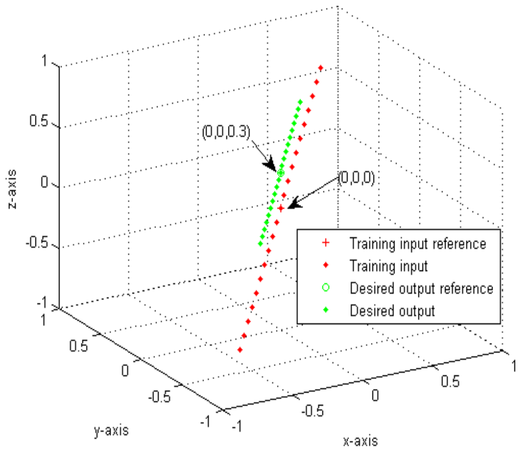


(b)

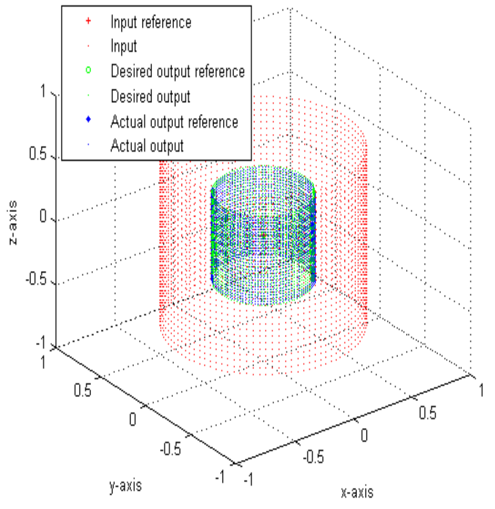
Fig.2. (a) Training with input-output mapping of straight line with scaling factor ½; (b) Convergence of MSE curves for scaling in the network with ℍ-BP (MLP) (red), ℍ-RPROP (MLP) (green), ℍ-BP (RPMN) (black) and ℍ-RPROP (RPMN) (blue) algorithms.



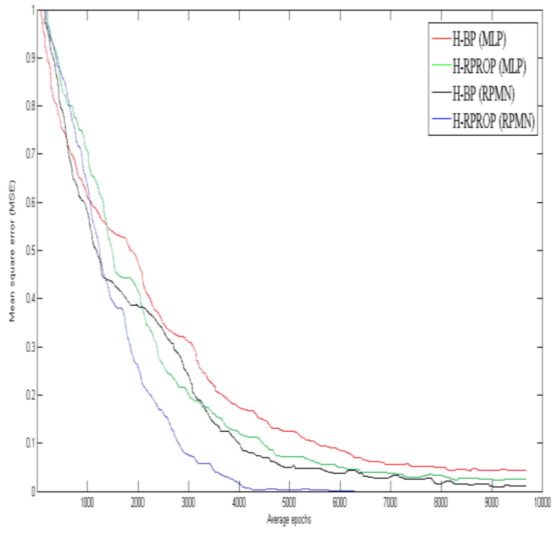
(a)



(a)

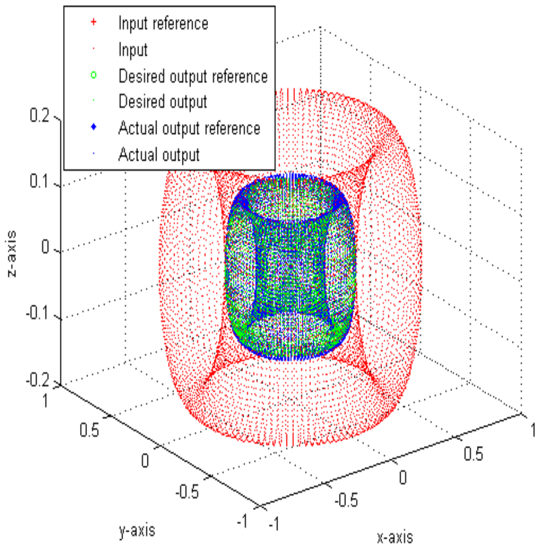


(b)



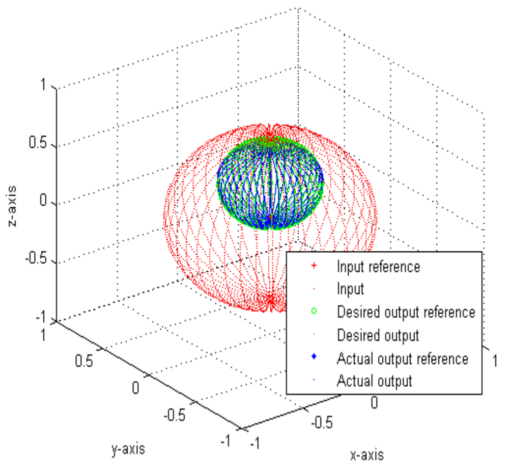
(b)

Fig.4. (a) Training with input-output mapping over straight line with scaling factor ½ and 0.3 unit translation in positive z-direction; (b) Convergence of MSE curves for scaling in the network with ℍ-BP (MLP) (red), ℍ-RPROP (MLP) (green), ℍ-BP (RPMN) (black) and ℍ-RPROP (RPMN) (blue) algorithms.

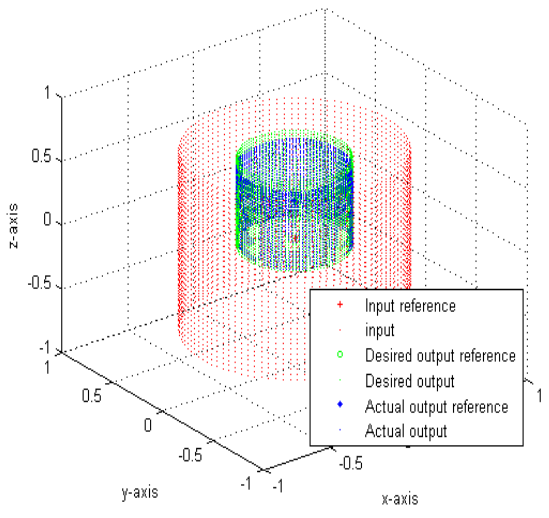


(c)

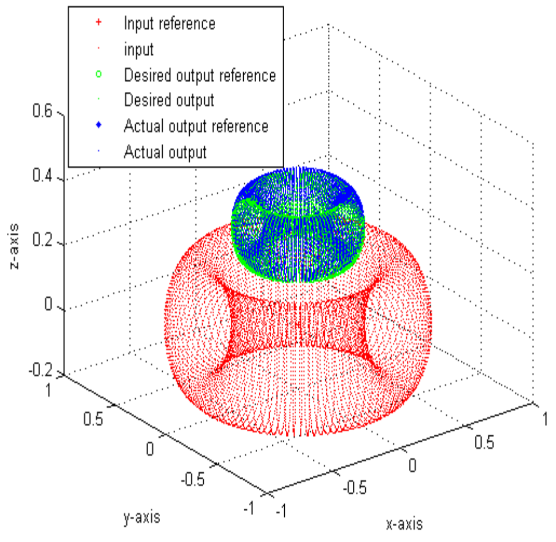
Fig.3. The generalization through ℍ-RPROP algorithm: Transformations with scaling factor ½; over (a) Sphere (b) Cylinder and (c) Torus.



(a)



(b)



(c)

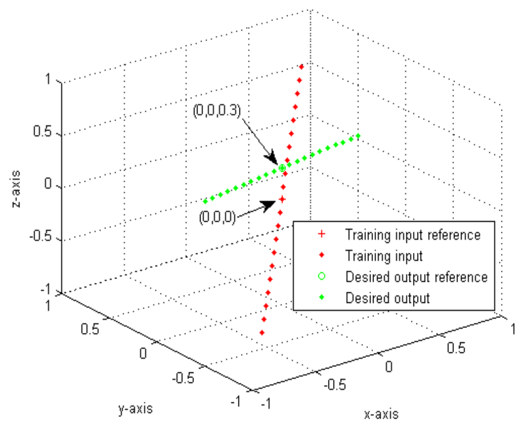
Fig.5. The generalization through ℍ-RPROP algorithm: Transformations with scaling factor ½ and 0.3 unit translation in positive z-direction; over (a) Sphere (b) Cylinder and (c) Torus.

Table 2. Comparison of Training and Testing Performance for Scaling

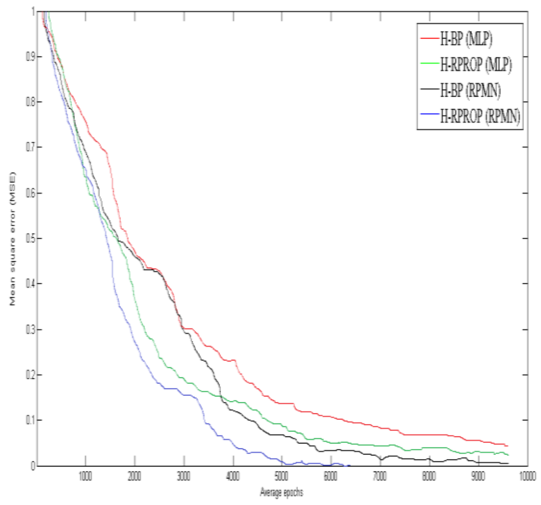
		ℍ-MLP		ℍ-RPMN (α = 0.95)	
Algorithm		ℍ-BP	ℍ-RPROP	ℍ-BP	ℍ-RPROP
Network		2-4-2	2-4-2	2-4-2	2-4-2
Parameters		88	88	88	88
MSE training through straight line		0.0007	0.0007	0.0007	0.0007
Average Learning cycles		25000	11000	8000	5000
Average Training Time (in Minutes)		170	110	95	70
MSE testing through	Sphere	0.0052	0.0033	0.0017	0.0015
	Cylinder	0.0034	0.0027	0.0014	0.0011
	Torus	0.0098	0.0052	0.0033	0.0026

Table 3. Comparison of Training and Testing Performance for Scaling and Translation

		ℍ-MLP		ℍ-RPMN (α = 0.95)	
Algorithm		ℍ-BP	ℍ-RPROP	ℍ-BP	ℍ-RPROP
Network		2-4-2	2-4-2	2-4-2	2-4-2
Parameters		88	88	88	88
MSE training through straight line		0.0007	0.0007	0.0007	0.0007
Average Learning Cycles		28000	12000	8040	5050
Average Training Time (in Minutes)		180	130	110	85
MSE testing through	Sphere	0.0063	0.0035	0.0021	0.0012
	Cylinder	0.0054	0.0031	0.0016	0.0009
	Torus	0.0095	0.0064	0.0041	0.0027

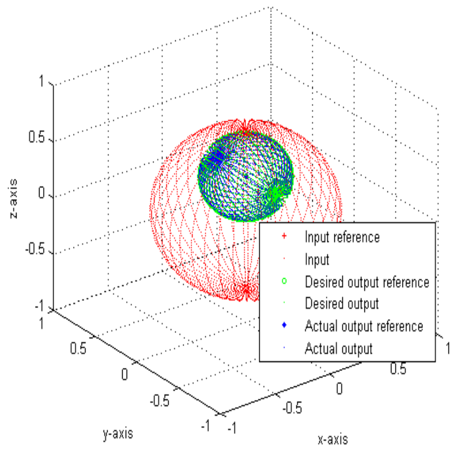


(a)

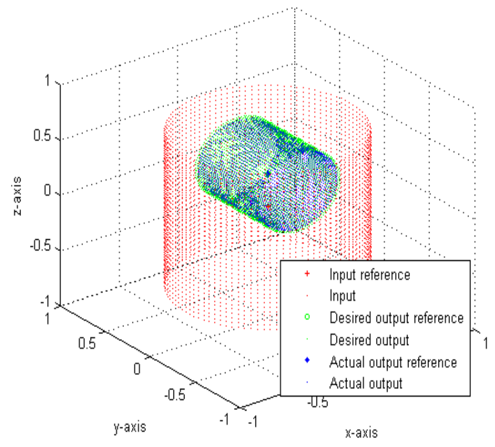


(b)

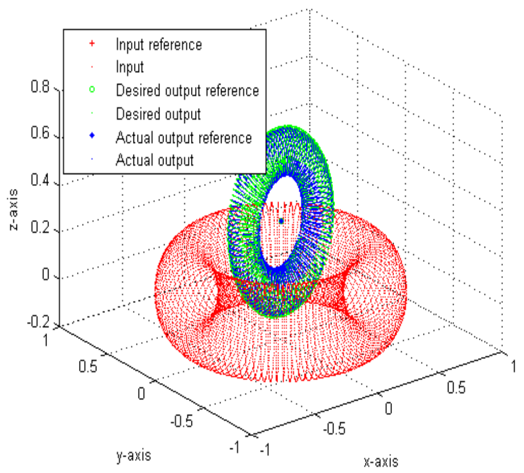
Fig.6. (a) Training with input-output mapping of straight line with scaling factor ½, 0.3 unit translation in positive z-direction, and π/2 radian rotation around the unit vector (i); (b) Convergence of MSE curves for scaling in the network with ℍ-BP (MLP) (red), ℍ-RPROP (MLP) (green), ℍ-BP (RPMN) (black), and ℍ-RPROP (RPMN) (blue) algorithms.



(a)

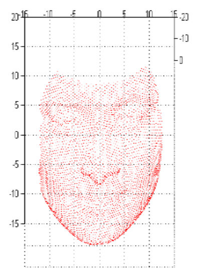


(b)

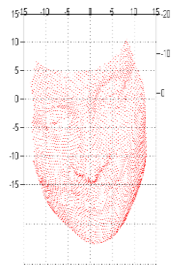


(c)

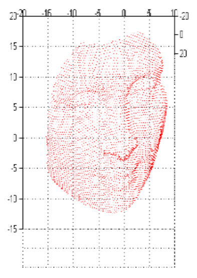
Fig.7. The generalization through ℍ-RPROP algorithm: Transformations with scaling factor ½, 0.3 unit translation in positive z-direction, and π/2 radian rotation around the unit vector (i); over (a) Sphere (b) Cylinder and (c) Torus.



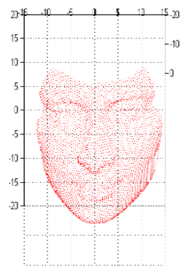
(a)



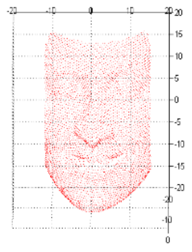
(b)



(c)



(d)



(e)

Fig.8. Five 3D faces of same person with different orientation and poses.

The generalization of trained networks has been performed over complicated 3D objects like sphere (4141 data points), cylinder (2929 data points) and torus (10201 data points). The ℍ-RPROP with ℍ-RPMN (Figs. 3, 5, and 7) shows excellent generalization for all three cases of transformations over rest of the algorithms. Tables 3, 4, and 5 clearly demonstrate the superiority of ℍ-RPROP with RPMN in all experiments.

Table 4. Comparison of Training and Testing Performance for Scaling, Translation and Rotation

		ℍ-MLP		ℍ-RPMN (α = 0.95)	
Algorithm		ℍ-BP	ℍ-RPROP	ℍ-BP	ℍ-RPROP
Network		2-4-2	2-4-2	2-4-2	2-4-2
Parameters		88	88	88	88
MSE training through straight line		0.0007	0.0007	0.0007	0.0007
Average Learning Cycles		30000	15000	8500	6020
Average Training Time (in Minutes)		195	143	118	103
MSE testing through	Sphere	0.0062	0.0058	0.0025	0.0016
	Cylinder	0.0035	0.0028	0.0018	0.0013
	Torus	0.0088	0.0069	0.0047	0.0025

C. *3D Face Recognition*

3D face identification is used as biometrics application through proposed methodology and compared it with related methods. The two human face datasets of 3D points cloud containing variable head position, orientation, and facial expressions, have been considered for training and testing. The first set consists of five faces of the same person and other set have five faces of different persons. The two experiments are conducted for face identification and in both experiments, one face has been used for training of the 1-2-1 network and the rest for testing. Thus, it is a basic and primitive experiment that learns the complex geometrical surface of one face and classify the rest of the faces through quaternionic-valued neural networks.

The first experiment is performed on first dataset containing 05 faces of the same person with different orientation and poses; the learning of NN in quaternionic domain is done with one face (Fig. 8(a)) and testing with all faces where each 3D face consists of 4654 points cloud data. Table 5 presents the training and testing analysis of faces through learning algorithms of each face. Table 5 also presents the comparative analysis of threshold MSEs with respect to average epochs for all algorithms. The threshold MSE reaches significantly faster during training in case of RPMN model (power coefficient α = 0.90) in both ℍ-RPROP and ℍ-BP. This table shows that the testing error of all five faces are less comparable to each other for all algorithms which demonstrate they are faces of same person irrespective of minor variations in face orientation and poses. These results infer the learning and generalization capability of neural network in quaternionic domain.

Similarly, the second experiment is performed on another dataset containing 05 faces of different persons; the learning of NN in quaternionic domain is done with one face (Fig. 9(a)) and testing with all faces where each 3D face consists of 6397 points cloud data. Table 6 presents the training and testing analysis of faces through learning algorithms. Table 6 also presents the comparative analysis of threshold MSEs with respect to average epochs for all algorithms. The threshold MSE reaches significantly faster during training in case of RPMN model (power coefficient α = 0.90) in both ℍ-RPROP and ℍ-BP. For all algorithms, the table shows the testing error of all five faces but MSE of other four faces are much higher in comparison to the face (Fig. 9(a)) which is used in training of the network. This demonstrates that the network classifies the faces of same or different person with different orientation and poses. These results also infer the learning and generalization capability of neural network.

Table 5. Comparison of Testing Error of Each Face of Same Person with Different Orientation and Poses

		ℍ-MLP		ℍ-RPMN (α = 0.90)	
Algorithm		ℍ-BP	ℍ-RPROP	ℍ-BP	ℍ-RPROP
Network		1-2-1	1-2-1	1-2-1	1-2-1
Parameters		28	28	28	28
MSE training through Fig. 8(a)		0.0001	0.0001	0.0001	0.0001
Average Epochs		28000	16000	12000	8000
Average Training Time (in Minutes)		210	176	160	132
MSE testing through Figure	8(a)	2.7214e-04	2.3941e-04	2.4842e-04	1.5192e-04
	8(b)	3.5431e-03	8.1822e-04	3.8822e-04	1.9213e-04
	8(c)	5.1153e-03	1.7821e-03	3.13943-04	1.8628e-04
	8(d)	4.5212e-04	3.2961e-04	4.8824e-04	1.6781e-04
	8(e)	3.9148e-04	2.9016e-04	3.6904e-04	1.9937e-04

Table 6. Comparison of Testing Error of Each Face of Different Person

		ℍ-MLP		ℍ-RPMN (α = 0.90)	
Algorithm		ℍ-BP	ℍ-RPROP	ℍ-BP	ℍ-RPROP
Network		1-2-1	1-2-1	1-2-1	1-2-1
Parameters		28	28	28	28
MSE Training through Fig. 9(a)		0.0001	0.0001	0.0001	0.0001
Average Epochs		29000	16500	13000	8500
Average Training Time (in Minutes)		228	186	174	145
MSE testing through Figure	9(a)	1.8521e-04	1.6883e-04	1.7721e-04	1.5817e-04
	9(b)	8.7296e-01	7.9562e-01	8.2840e-01	7.8825e-01
	9(c)	3.5742e-00	3.4861e-00	3.3772e-00	3.1930e-00
	9(d)	6.2996e-02	6.0299e-02	5.5721e-02	5.8551e-02
	9(e)	3.9274e-01	3.6462e-01	3.7327e-01	3.5332e-01

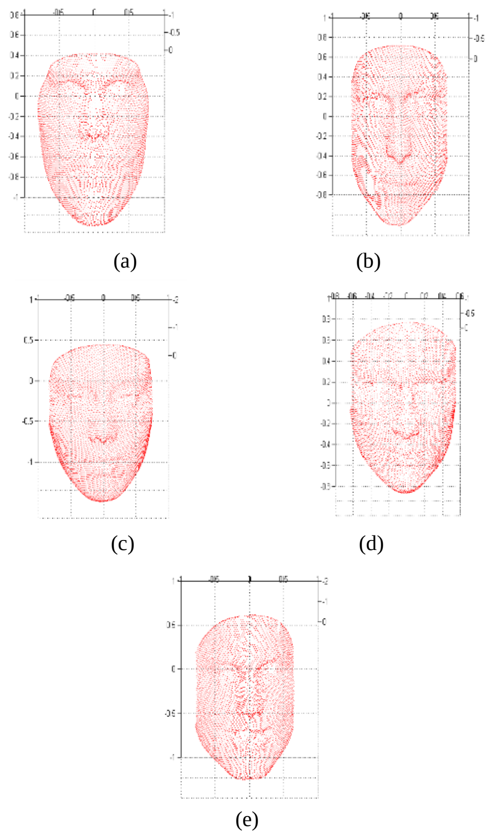


Fig.9. Five 3D faces of different persons.

VI. Conclusions

This paper presents to design an efficient neuron model with nonlinear aggregation function of quaternionic-valued signals. The modified resilient propagation (RPROP) learning algorithm is also presented in quaternionic domain. The proposed methodology is systematically evaluated and compared with convention summing neuron in quaternion domain through a wide spectrum of 3D and 4D problems. The root-power mean of quaternionic signal is conceptually used as an aggregation function of the proposed neuron and its performance is far better than quaternionic-valued conventional neuron in terms of convergence rate. The main drawbacks of backpropagation (BP) algorithm are slow convergence and getting stuck to local minima. This algorithm needs significant improvement in the training cycle to adjust synaptic weights as fast as possible. Therefore, the modified resilient propagation algorithm in quaternionic domain (ℍ-RPROP) has been developed to overcome the problems of ℍ-BP algorithm. The quicker convergence with better performance is the significant advantage of this algorithm which always reveals to reduce the training iterations. Its computational power is also demonstrated through various benchmark problems (function approximation, linear transformation, and 3D face recognition). The power coefficient (α) has an important parameter in the quaternionic-valued root-power mean neuron (ℍ-RPMN), which exhibits the better approximation of the functional capabilities. The development of an algorithm to select power coefficient of ℍ-RPMN will be interesting problem for future work. The computational capabilities of the proposed neuron in quaternionic domain can also be verified by the theoretical proof.

References

[1] B. W. Mel. Information processing in dendritic trees. *Neural Comput.*, vol. 6, no. 6, pp. 1031–1085, Nov. 1994.

[2] C. Koch and I. Segev. The role of single neurons in information processing. *Nat. Neurosci.*, 3(Suppl), pp. 1171–1177, Nov. 2000.

[3] A. Polsky, B. W. Mel, and J. Schiller. Computational subunits in thin dendrites of pyramidal cells. *Nat. Neurosci.*, 7, pp. 621–627, May, 2004.

[4] K. Sidiropoulou, E. K. Pissadaki, and P. Poirazi. Inside the brain of a neuron. *EMBO Rep.*, vol. 7, no. 9, pp. 886–892, Sep. 2006.

[5] M. Lavzin, S. Rapoport, A. Polsky, L. Garion, and J. Schiller. Nonlinear dendritic processing determines angular tuning of barrel cortex neurons *in vivo*. *Nature*, vol. 490, no. 7420, pp. 397–401, Sep. 2012.

[6] Y. Todo, H. Tamura, K. Yamashita, and Z. Tang. Unsupervised learnable neuron model with nonlinear interaction on dendrites. *Neural Netw.*, vol. 60, pp. 96–103, Dec. 2014.

[7] T Jiang, D. Wang, J. Ji, Y. Todo, and S. Gao. Single dendritic neuron with nonlinear computation capacity: a case study on XOR problem. in *Proc. Int. Conf. on Progress in Informatics and Computing (PIC)*, pp. 20–24, Dec. 2015.

[8] W. Chen, J. Sun, S. Gao, J.-J. Cheng, J. Wang, and Y. Todo. Using a single dendritic neuron to forecast tourist arrivals to japan. *IEICE Trans. Inf. & Syst.*, vol. E100–D, no. 1, pp. 190–202, Jan. 2017.

[9] Y. Xiong, W. Wu, X. Kang, and C. Zhang. Training pi-sigma network by online gradient algorithm with penalty for small weight update. *Neural Comput.*, vol. 19, no. 12, 3356–3368, Jan. 2008.

[10] C.-K. Li. A sigma-pi-sigma neural network (SPSNN). *Neural Process. Lett.*, vol. 17, no. 1, pp. 1–19, Mar. 2003.

[11] N. Homma and M. M. Gupta. A general second-order neural unit. *Bull. Coll. Med. Sci. Tohoku Univ.*, vol. 11, no. 1, pp. 1–6, 2002.

[12] B. K. Tripathi and P. K. Kalra. The novel aggregation function-based neuron models in complex domain. *Soft Comput.*, vol. 14, no. 10, pp. 1069–1081, Aug. 2010.

[13] C. L. Giles and T. Maxwell. Learning, invariance, and generalization in high-order neural networks. *Appl. Opt.*, vol. 26, no. 23, pp. 4972–4978, 1987.

[14] M. Zhang, S. Xu, and J. Fulcher. Neuron-adaptive higher order neural network models for automated financial data modeling. *IEEE Trans. Neural Netw.*, vol. 13, no. 1, pp. 188–204, Jan. 2002.

[15] S. Xu. Adaptive higher order neural network models and their applications in business. *IGI Global*, pp. 314–329, 2009.

[16] E. B. Kosmatopoulos, M. M. Polycarpou, M. A. Christodoulou, and P. A. Ioannou. High-order neural network structures for identification of dynamical systems. *IEEE Trans. Neural Netw.*, vol. 6, no. 2, pp. 422–431, Mar. 1995.

[17] B. K. Tripathi and P. K. Kalra. On efficient learning machine with root-power mean neuron in complex domain. *IEEE Trans. Neural netw.*, vol. 22, no. 5, pp. 727–738, May 2011.

[18] H. Dyckhoff and W. Pedrycz. Generalized means as model of compensative connectives. *Fuzzy Sets Syst.*, vol. 14, no. 2, pp. 143–154, Nov. 1984.

[19] R. R. Yager. Generalized OWA aggregation operators. *Fuzzy Optim. Decis. Ma.*, vol. 3, no. 1, pp. 93–107, Mar. 2004.

[20] A. V. Ooyen and B. Nienhuis. Improving the convergence of the backpropagation algorithm. *Neural Netw.*, vol. 5, no. 3, pp. 465–472, 1992.

[21] X. Chen, Z. Tang, and S. Li. An modified error function for the complex-value backpropagation neural network. *Neural Inf. Process.*, vol. 8, no. 1, pp. 1–8, Jul. 2005.

[22] G. D. Magoulas, N. V. Michael, and S. A. George. Effective backpropagation training with variable stepsize. *Neural Netw.*, vol. 10, no.1, pp. 69–82, 1997.

[23] T. P. Vogl, J. K. Mangis, A. K. Rigler, W. T. Zink, and D. L. Alkon. Accelerating the convergence of the back-propagation method. *Biol. Cybern.*, vol. 59, no. 4, pp. 257–263, 1988.

[24] C. C. Yu and D. B. Liu. A backpropagation algorithm with adaptive learning rate and momentum coefficient. in *Proc. IEEE Int. Jt. Conf. Neural Netw.*, vol. 2, 2002, pp. 1218–1223.

[25] E. Istook and T. Martinez. Improved backpropagation learning in neural networks with windowed momentum. *Int. J. Neural Sys.*, vol. 12, no. 3 and 4, pp. 303–318, Jan. 2002.

[26] R. A. Jacobs. Increased rates of convergence through learning rate adaptation. *Neural Netw.*, vol. 1, no. 4, pp. 295–307, 1988.

[27] A. A. Minai and R. D. Williams. Back-propagation heuristics: a study of the extended delta-bar-delta algorithm. in *Proc. Int. Jt. Conf. Neural Netw.*, June 1990, pp. 595–600.

[28] S. E. Fahlman. An empirical study of learning speed in backpropagation networks. *Tech. Rep.* CMU-CS-88-162, Sep. 1988.

[29] M. Riedmiller and H. Braun. A direct adaptive method for faster backpropagation learning: The RPROP algorithm. in *Proc. IEEE Int. Conf. Neural Netw.*, San Francisco, CA, Apr. 1993.

[30] C. Igel and M. Husken. Empirical evaluation of the improved Rprop learning algorithms. *Neurocomputing*, vol. 50, pp. 105–123, Jan. 2003.

[31] C. Igel, M. Toussaint, and W. Weishui. Rprop using the natural gradient. in *Trends and Applications in Constructive Approximation*, Ed. by D. Mache, J. Szabados, and M. De Bruin (Int. Series Numerical Math. (ISNM), Birkhauser, Basel), vol. 151, pp. 259–272, 2005.

[32] A. Kantsila, M. Lehtokangas, and J. Saarinen. Complex RPROP-algorithm for neural network equalization of GSM data bursts. *Neurocomputing*, vol. 61, pp. 339–360, Oct. 2004.

[33] C.-S. Chen, J.-M. Lin, and C.-T. Lee. Neural network for WGDOP approximation and mobile location. *Mathematical Problems in Engineering*, vol. 2013, Article ID 369694, 11 pages, 2013.

[34] L. Orcik, M. Voznak, J. Rozhon, F. Rezac, J. Slachta, H. T. Cruz, and J. C.-W. Lin. Prediction of speech quality based on resilient backpropagation artificial neural network. *Wireless Personal Communications*, doi:10.1007/s11277-016-3746-2, Oct. 2016.

[35] A. N. Kolmogoroff. Sur la notion de la moyenne. *Acad.*

Naz. Lincei Mem. Cl. Sci. Fis. Mat. Natur. Sez., vol. 12, no. 6, pp. 388–391, 1930.

[36] M. Nagumo. Über eine klasse der mittelwerte. *Jpn. J. Math.*, vol. 7, pp. 71–79, 1930.

[37] W. R. Hamilton. On a new species of imaginary quantities connected with a theory of quaternions. in *Proc. Royal Irish Academy*, vol. 2, no. 1843, pp. 424–434, Nov. 1844.

[38] B. C. Ujang, C. C. Took, and D. P. Mandic. Split quaternion nonlinear adaptive filtering. *Neural Netw.*, vol. 23, no. 3, pp. 426–434, Apr. 2010.

[39] T. Nitta. An extension of the back-propagation algorithm to complex numbers. *Neural Netw*., vol. 10, no. 8, pp. 1391–1415, Nov. 1997.

[40] B. K. Tripathi. *High dimensional neurocomputing: growth, appraisal and applications*. India: Springer, 2015.

[41] W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bull. Math. Biophys.*, vol. 4, no. 4, pp. 115–133, Dec. 1943.

[42] A. Hirose. *Complex-valued neural networks: theories and applications.* vol. 5., World Scientific, 2003.

[43] T. Nitta. A quaternary version of the back-propagation algorithm. in *Proc. Int. Conf. Neural Netw.*, vol. 5, Nov. 1995, pp. 2753–2756.

[44] M. Schmitt. On the complexity of computing and learning with multiplicative neural networks. *Neural Comput.*, vol. 14, no. 2, pp. 241–301, Feb. 2002.

[45] P. K. Kalra, B. Chandra, and M. Shiblee. New neuron model for blind source separation. in *Proc. Int. Conf. Neural Inf. Process.*, Auckland, New Zealand, Nov. 2008, pp. 27–36.

[46] G. M. Georgiou. Exact interpolation and learning in quadratic neuralnetworks. in *Proc. Int. Joint Conf. Neural Netw.*, Vancouver, BC, Canada, Jul. 2006, pp. 230–234.

[47] S. J. Sangwine and N. L. Bihan. Quaternion polar representation with a complex modulus and complex argument inspired by the Cayley-Dickson form. *Advances in Applied Clifford Algebras*, vol. 20, no. 1, pp. 111–120, Mar. 2010.

[48] E. Cho. De moivre's formula for quaternions. *Appl. Math. Lett.*,vol. 11, no. 6, pp. 33–35, Nov. 1998.

[49] D. B. Foggel. An information criterion for optimal neural network selection. *IEEE Trans. Neural Netw.*, vol. 2, no. 5, pp. 490-497, Sep.1991.

[50] R. Naresh, S. Pandey, and J. B. Shukla. Modeling the cumulative effect of ecological factors in the habitat on the spread of tuberculosis. *Int. J. Biomath.*, vol. 2, no. 3, pp. 339-355, Sep. 2009.

[51] R. Kaur and A. K. Narula. Artificial neural network based design of modified shaped patch antenna. *IJISA*, vol. 9, no. 4, pp. 32-38, Apr. 2017.

[52] S. Kumar and B. K. Tripathi. Machine learning with resilient propagation in quaternionic domain. *IJIES*, vol. 10, no. 4, pp. 205-216, Aug. 2017.

[53] T. Parcollet, M. Morchid, and P. M. Bousquet, R. Dufour, G. Linarès, and R. De Mori. Quaternion neural networks for spoken language understanding. In *Proc. IEEE Spoken Language Technology Workshop (SLT)*, San Diego, CA, 2016, pp. 362-368.

[54] S. C. Nayak. Development and performance evaluation of adaptive hybrid higher order neural networks for exchange rate prediction. *IJISA*, vol. 9, no. 8, pp. 71-85, Aug. 2017.

[55] F. Shang and A. Hirose. Quaternion neural-network-based PolSAR land classification in Poincare-Sphere-Parameter space. *IEEE Trans. on Geoscience and Remote Sensing*, vol. 52, no. 9, pp. 5693-5703, Sep. 2014.

[56] D. K. Choubey and S. Paul. GA_MLP NN: A hybrid intelligent system for diabetes disease diagnosis. *IJISA*, vol. 8, no. 1, pp. 49-59, Jan. 2016.

Authors' Profiles

Sushil Kumar is pursuing his PhD in Computational Intelligence from HBTU Kanpur, India and completed M. Tech in Modelling and Simulation from DIAT-DRDO Pune, India. He is currently research scholar in Department of Computer Science and Engineering of HBTU Kanpur, India. He is associated with the Nature-inspired Computational Intelligence Research Group (NCIRG) at HBTU. His areas of research include high-dimensional neurocomputing, computational intelligence, machine learning and computer vision focused on biometrics and 3D Imaging. He has published several research papers in these areas.

Bipin K. Tripathi completed his PhD in Computational Intelligence from IIT Kanpur, India and M. Tech in Computer Science and Engineering from IIT Delhi, India. Dr. Tripathi is currently serving as a Professor in the Department of Computer Science and Engineering of HBTU Kanpur, India. He is also leading the Nature-inspired Computational Intelligence Research Group (NCIRG) at HBTI. His areas of research include high-dimensional neurocomputing, computational neuroscience, intelligent system design, machine learning and computer vision focused on biometrics and 3D Imaging. He has published several research papers in these areas in many peer reviewed journals including IEEE Transaction, Elsevier, Springer and other international conferences. He has also contributed book chapters in different international publications and patent in his area. He is continuously serving as PC for many international conferences and as a reviewer of several international journals.

How to cite this paper: Sushil Kumar, Bipin K. Tripathi, "On the Root-Power Mean Aggregation Based Neuron in Quaternionic Domain", International Journal of Intelligent Systems and Applications(IJISA), Vol.10, No.7, pp.11-26, 2018. DOI: 10.5815/ijisa.2018.07.02