IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 1 August 2023, accepted 22 August 2023, date of publication 5 September 2023, date of current version 9 October 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3312234

RESEARCH ARTICLE

Quaternion and Split Quaternion Neural Networks for Low-Light Color Image Enhancement

EDUARDO DE JESÚS DÁVILA-MEZA AND EDUARDO JOSE BAYRO-CORROCHANO, (Senior Member, IEEE)
Center of Research and Advanced Studies (CINVESTAV), National Polytechnic Institute (IPN), Guadalajara Campus, Zapopan 45019, Mexico
Corresponding author: Eduardo de Jesús Dávila-Meza (eduardo.davila@cinvestav.mx)

This work was supported by the National Council of Humanities, Sciences and Technologies (CONAHCYT), Mexico, under Grant CVU.854352.

ABSTRACT In this study, two models of multilayer quaternionic feedforward neural networks are presented. Whereas the first model is based on quaternion algebra, the second model uses *split quaternion* algebra. For both quaternionic neural networks, a learning algorithm was derived using an adaptation of the extended Kalman filter. In addition, to analyze the performance of these two neural network models, they were applied to address the problem of enhancing low-light color images, which for this work consists particularly in the recovery of illuminated color images by quaternionic neural network processing from underexposed images. The quaternion neural network enhances images in the RGB color space (Euclidean metric), whereas the *split quaternion* neural network enhances images in the HSV color space (Minkowski metric). From the results, we can observe that the *split quaternion* neural network using the HSV color model shows advantages that were not previously published and were not shown by the quaternion neural network using the RGB color model. Therefore, this article introduces a novel quaternionic neural network that uses the Minkowski metric for color image processing, which can be advantageously used by practitioners interested in working with the HSV color model.

INDEX TERMS Extended Kalman filter, low-light image enhancement, quaternion, quaternion neural network, RGB and HSV color space, *split quaternion*, *split quaternion* neural network, underexposed image.

I. INTRODUCTION

Image enhancement procedures consist of a set of techniques or operations aimed at improving the visual appeal of an image or converting an image into a form that is more suitable for analysis by a human viewer or machine processing. In an image enhancement system, there are no conscious efforts to increase the fidelity of the reproduced image in relation to a desired image form, as is the case in image restoration. Thus, there is currently no general unifying theory for image enhancement because there is no general image quality standard that can serve as a design criterion for an image enhancement processor [1]. In this sense, enhancing color images becomes a more difficult task, not only due to the extra dimension of the data compared to grayscale images, but also due to the added complexity of color perception [2].

On the other hand, the algebra of quaternions, first described by the mathematician William Rowan Hamilton in 1843, has recently attracted the interest of researchers in areas such as color image processing, automatic control, aerospace, satellite tracking, and body motion tracking. An attractive feature of quaternion algebra is that it minimizes the number of parameters while improving computational complexity and functional simplicity [3], [4].

In addition, in recent years, quaternion neural networks (QNNs) have been used to address a variety of engineering challenges, including color image compression, control problems, inertial body sensing, and wind profile modeling. QNNs are distinguished by the fact that their input, weights, activation functions, and output are all quaternion-valued; therefore, they can process quaternion-valued data directly.

The associate editor coordinating the review of this manuscript and approving it for publication was Qiangqiang Yuan.

Thus, quaternions have reduced the number of parameters and operations required for these applications [5].

Furthermore, neural network literature has been flooded with studies that offer several training methods that purportedly outperform traditional backpropagation and similar approaches in terms of generalization, mapping accuracy, overall performance, and training time. In this sense, the extended Kalman filter (EKF) constitutes the principal component of a second-order neural network training approach that is both feasible and successful compared with other second-order methods. The essence of the recursive EKF technique is that, during training, an approximation error covariance matrix storing second-order information concerning the training problem is maintained and evolved, and the weights of the network architecture evolve sequentially. Singhal and Wu [6] introduced the global EKF training algorithm in the late 1980s and has served as the basis for the formulation and improvement of a set of methods for computational training of neural networks. This has enabled the application of feedforward and recurrent neural networks to control problems, pattern recognition, and signal processing [7].

Therefore, in this context, our work aims to design, develop, and apply quaternionic neural networks using quaternion and *split quaternion* algebras, and an adaptation of the EKF as a learning algorithm. Since quaternion neural networks have become very popular, as we have seen, the goal of this paper is to analyze their role in color image processing for enhancement by using different geometric spatial metrics and color models.

The relevant literature related to this study is briefly introduced in the following. These previous works focused on the EKF for multilayered perceptron training and, similarly, on *Quaternion Algebra* and its use in neural networks for image processing.

In 1992, Iiguni et al. [8] used the extended Kalman filter technique as a learning algorithm for a multilayer perceptron neural network. They demonstrated that the convergence performance of their learning method outperformed the backward error propagation strategy using the steepest descent technique.

In 1999, Rughooputh et al. [9] adapted the extended Kalman filter technique for a hypercomplex multilayer perceptron neural network. The weights and biases of this neural network are quaternions, and in the same form, the input and output signals.

In 2002, Puskorius and Feldkamp presented in [7] a brief discussion of the feedforward network architecture that they considered for training using the EKF method. Therefore, they presented a *global* EKF training approach, followed by parameter setting suggestions, including the relationship between learning rate selection and initialization of the error covariance matrix.

In 2004, Matsui et al. [10] showed through tests that the quaternion version of the backpropagation procedure performs correct geometric transformations in the three-dimensional space and color space for the image compression problem, whereas real-valued backpropagation fails. The most important contribution to the development of this work is the appendix of the article, where the gradient of the neural network output, concerning all adjustable network parameters, is computed for the backpropagation algorithm.

Also in 2004, Kusamichi et al. [11] presented a neural network for extracting color information from low-light images. They adopted a quaternion feedforward neural network, in which the neuronal parameters were presented as four-dimensional vectors. The network was trained by setting a low-light image as the input and its original image as the target using a quaternion version of the backpropagation algorithm.

In 2009, Isokawa et al. [12] presented a type of quaternion neural network model. This model is a multilayer perceptron configured on geometric transformations using quaternions, where these transformations are operators in a three-dimensional space: dilation, spatial rotation, and translation. Color-night vision and color image compression problems were used to demonstrate the usefulness of this neural network.

Thus, as has already been noticed, the use of quaternions for neural networks is not new. However, the significance of this work is the use of *split quaternion* algebra to design a neural network model based on the geometric operations of *split quaternions*, and the application of the extended Kalman filter technique to adapt a learning algorithm for this neural network model that can be advantageously used by practitioners interested in working with the HSV color model, since it enhances color images using the Minkowski metric.

The remainder of this paper is organized as follows. Section II provides the mathematical foundation necessary for developing the subsequent content. Section III describes the quaternionic neural network model using quaternion and *split quaternion* algebras and describes the learning algorithm for these two quaternionic neural networks. Section IV introduces the application of both quaternionic neural networks to the low-light color image enhancement problem. In Section V, the training and testing results are reported. Finally, conclusions are drawn in Section VI. The partial derivatives for the gradient of the *split quaternion* neural network model are calculated in the Appendix. To visualize the gradient calculation of the quaternion neural network model, refer to [10].

II. PRELIMINARIES

A. QUATERNION ALGEBRA

Quaternions are an associative but not commutative algebra over $\mathbb{R}$. Next, this subsection specifies some definitions, properties, and notations of quaternion algebra according to [3], [4], [5], [10], [11], [12], [13], [14], [15], and [16].

Definition 1: A quaternion is defined as a vector x in a four-dimensional vector space, which is

$$x = x^{(e)} + x^{(i)}i + x^{(j)}j + x^{(k)}k, \tag{1}$$

where $x^{(e)}, x^{(i)}, x^{(j)}, x^{(k)} \in \mathbb{R}$, and i, j and k are the three imaginary units.

Definition 2: Quaternion algebra is an associative and noncommutative algebra over $\mathbb{R}$, defined by the set of all formal expressions given by (1) that is

$$\begin{aligned}\mathbb{H} &= span\{1, i, j, k\} \\ &= \{x^{(e)} + x^{(i)}i + x^{(j)}j + x^{(k)}k \\ &\quad | \; x^{(e)}, x^{(i)}, x^{(j)}, x^{(k)} \in \mathbb{R}\},\end{aligned} \tag{2}$$

which is called the set of *real quaternions* or *Hamilton numbers*. Likewise, the imaginary bases in quaternion algebra satisfy the following rules:

$$\begin{aligned} i^2 = j^2 = k^2 = ijk = -1, \\ ij = -ji = k, \quad jk = -kj = i, \quad ki = -ik = j. \end{aligned} \tag{3}$$

From these rules, we can see that multiplication in $\mathbb{H}$ is associative and noncommutative.

Definition 3: The conjugate of the quaternion $x = x^{(e)} + x^{(i)}i + x^{(j)}j + x^{(k)}k$ is defined as

$$x^* = x^{(e)} - x^{(i)}i - x^{(j)}j - x^{(k)}k. \tag{4}$$

It can also be defined by the notation $\overline{x}$.

Definition 4: A quaternion with the real part equal to zero has the form

$$x = x^{(i)}i + x^{(j)}j + x^{(k)}k, \tag{5}$$

and it is defined as *pure quaternion*.

Let p and q be quaternions defined as $p = p^{(e)} + p^{(i)}i + p^{(j)}j + p^{(k)}k$ and $q = q^{(e)} + q^{(i)}i + q^{(j)}j + q^{(k)}k$, respectively. Then, we can define a series of operations for the quaternions that will be used in this study:

Definition 5: Addition and subtraction ($\pm$):

$$\begin{aligned} p \pm q = &\left(p^{(e)} \pm q^{(e)}\right) + \left(p^{(i)} \pm q^{(i)}\right)i \\ &+ \left(p^{(j)} \pm q^{(j)}\right)j + \left(p^{(k)} \pm q^{(k)}\right)k. \end{aligned} \tag{6}$$

Definition 6: Element-wise product ($\odot$):

$$p \odot q = p^{(e)}q^{(e)} + p^{(i)}q^{(i)}i + p^{(j)}q^{(j)}j + p^{(k)}q^{(k)}k. \tag{7}$$

Definition 7: Quaternion product ($\otimes$):

$$\begin{aligned} p \otimes q = &\left(p^{(e)}q^{(e)} - p^{(i)}q^{(i)} - p^{(j)}q^{(j)} - p^{(k)}q^{(k)}\right) \\ &+ \left(p^{(e)}q^{(i)} + p^{(i)}q^{(e)} + p^{(j)}q^{(k)} - p^{(k)}q^{(j)}\right)i \\ &+ \left(p^{(e)}q^{(j)} - p^{(i)}q^{(k)} + p^{(j)}q^{(e)} + p^{(k)}q^{(i)}\right)j \\ &+ \left(p^{(e)}q^{(k)} + p^{(i)}q^{(j)} - p^{(j)}q^{(i)} + p^{(k)}q^{(e)}\right)k. \end{aligned} \tag{8}$$

Definition 8: Product between scalar and quaternion ($\cdot$): The product of the scalar λ and the quaternion $x = x^{(e)} + x^{(i)}i + x^{(j)}j + x^{(k)}k$ is defined by

$$\lambda \cdot x = \lambda x^{(e)} + \lambda x^{(i)}i + \lambda x^{(j)}j + \lambda x^{(k)}k, \tag{9}$$

where $\lambda, x^{(e)}, x^{(i)}, x^{(j)}, x^{(k)} \in \mathbb{R}$.

Definition 9: Norm ($\|.\|$): The norm of the quaternion $x = x^{(e)} + x^{(i)}i + x^{(j)}j + x^{(k)}k$ is defined by

$$\begin{aligned}\|x\| &= \sqrt{x \otimes x^*} \\ &= \sqrt{x^{(e)2} + x^{(i)2} + x^{(j)2} + x^{(k)2}}.\end{aligned} \tag{10}$$

B. SPLIT QUATERNION ALGEBRA

Split quaternions are also an associative and noncommutative algebra over $\mathbb{R}$. This subsection provides some definitions, properties, and notations for *split quaternion* algebra according to [3], [4], [5], [10], [11], [12], [13], [14], [15], [16], and [17].

Definition 10: A *split quaternion* is defined as a vector x in a four-dimensional vector space, which is

$$x = x^{(e)} + x^{(i)}i + x^{(j)}j + x^{(k)}k, \tag{11}$$

where $x^{(e)}, x^{(i)}, x^{(j)}, x^{(k)} \in \mathbb{R}$, and i, j and k are the three bases.

Definition 11: Split quaternion algebra is an associative, noncommutative, and nondivided ring over $\mathbb{R}$, defined by the set of all formal expressions given by (11) that is

$$\begin{aligned}\mathbb{H}_s &= span\{1, i, j, k\} \\ &= \{x^{(e)} + x^{(i)}i + x^{(j)}j + x^{(k)}k \\ &\quad | \; x^{(e)}, x^{(i)}, x^{(j)}, x^{(k)} \in \mathbb{R}\},\end{aligned} \tag{12}$$

which is called the set of *real split quaternions*. Similarly, the bases in *split quaternion* algebra satisfy the following rules:

$$\begin{aligned} i^2 = -1, \qquad j^2 = k^2 = ijk = 1 \\ ij = -ji = k, \quad jk = -kj = -i, \quad ki = -ik = j. \end{aligned} \tag{13}$$

From these rules, we can see that multiplication in $\mathbb{H}_s$ is associative and noncommutative.

Definition 12: The conjugate of the *split quaternion* $x = x^{(e)} + x^{(i)}i + x^{(j)}j + x^{(k)}k$ is defined as

$$x^* = x^{(e)} - x^{(i)}i - x^{(j)}j - x^{(k)}k. \tag{14}$$

It can also be defined by the notation $\overline{x}$.

Definition 13: A *split quaternion* with the real part equal to zero has the form

$$x = x^{(i)}i + x^{(j)}j + x^{(k)}k, \tag{15}$$

and it is defined as *pure split quaternion*.

Let p and q be *split quaternions* defined as $p = p^{(e)} + p^{(i)}i + p^{(j)}j + p^{(k)}k$, and $q = q^{(e)} + q^{(i)}i + q^{(j)}j + q^{(k)}k$, respectively. We can then define a series of operations for the *split quaternions* that will be used in this study:

Definition 14: Addition and subtraction ($\pm$):

$$\begin{aligned} p \pm q = &\left(p^{(e)} \pm q^{(e)}\right) + \left(p^{(i)} \pm q^{(i)}\right)i \\ &+ \left(p^{(j)} \pm q^{(j)}\right)j + \left(p^{(k)} \pm q^{(k)}\right)k. \end{aligned} \tag{16}$$

Definition 15: Element-wise product ($\odot$):

$$p \odot q = p^{(e)}q^{(e)} + p^{(i)}q^{(i)}i + p^{(j)}q^{(j)}j + p^{(k)}q^{(k)}k. \tag{17}$$

Definition 16: Split quaternion product ($\otimes$):

$$\begin{aligned} p \otimes q = &\left(p^{(e)}q^{(e)} - p^{(i)}q^{(i)} + p^{(j)}q^{(j)} + p^{(k)}q^{(k)}\right) \\ &+ \left(p^{(e)}q^{(i)} + p^{(i)}q^{(e)} - p^{(j)}q^{(k)} + p^{(k)}q^{(j)}\right)i \\ &+ \left(p^{(e)}q^{(j)} - p^{(i)}q^{(k)} + p^{(j)}q^{(e)} + p^{(k)}q^{(i)}\right)j \\ &+ \left(p^{(e)}q^{(k)} + p^{(i)}q^{(j)} - p^{(j)}q^{(i)} + p^{(k)}q^{(e)}\right)k. \end{aligned} \tag{18}$$

Definition 17: Product between scalar and *split quaternion* $(\cdot)$: The product of the scalar λ and the *split quaternion* $x = x^{(e)} + x^{(i)}i + x^{(j)}j + x^{(k)}k$ is defined by

$$\lambda \cdot x = \lambda x^{(e)} + \lambda x^{(i)}i + \lambda x^{(j)}j + \lambda x^{(k)}k, \tag{19}$$

where $\lambda, x^{(e)}, x^{(i)}, x^{(j)}, x^{(k)} \in \mathbb{R}$.

Definition 18: Norm $(\|.\|)$: The norm of the *split quaternion* $x = x^{(e)} + x^{(i)}i + x^{(j)}j + x^{(k)}k$ is defined by

$$\begin{aligned} \|x\| &= \sqrt{|x \otimes x^*|} \\ &= \sqrt{\left|x^{(e)2} + x^{(i)2} - x^{(j)2} - x^{(k)2}\right|}. \end{aligned} \tag{20}$$

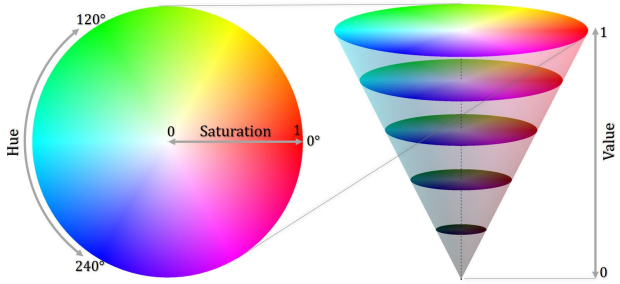


FIGURE 1. HSV color model. This figure was created using MATLAB software.

C. HSV COLOR SPACE USING SPLIT QUATERNIONS

The HSV color space is a three-dimensional representation of color based on the components of hue, saturation, and brightness, as defined in 1978 by Alvy Ray Smith [18]. It is also a more accurate representation of how people perceive colors and their quality. Color tones are grouped, unlike in the RGB scenario, in which colors are not sorted in a coherent manner. The color space created by this model corresponds to a cone, as shown in Fig. 1. This color model is based on hue, saturation, and value (HSV), which are described below:

- Hue refers to the dominant pure color or wavelength of light in color. It is represented by an angle around the vertical axis (0°,..., 360°), but also in the modular range [0.0, 1.0].
- Saturation represents the relative purity of the hue. It ranges from 0.0 to 1.0, where the maximum purity is represented by $S = 1.0$, whereas with $S = 0.0$ we have the grayscale.
- Value corresponds to the relative brightness. It ranges from 0.0 to 1.0, where zero stands at the bottom of the cone and represents black, whereas the maximum value corresponds to the top of the cone, with white in the center and pure colors in the perimeter, since they are *hues* and *primary* colors with $V = 1.0$ and $S = 1.0$.

The equations that relate the components of the HSV model to those of the RGB model are as follows:

$$V = \max(R, G, B), \tag{21}$$

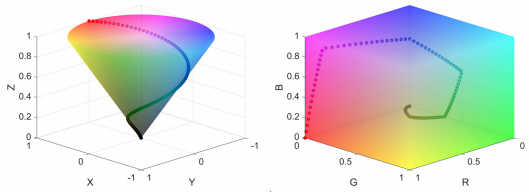
$$X = \min(R, G, B), \tag{22}$$

$$S = \begin{cases} 0, & \text{if V = 0} \\ \frac{V-X}{V}, & \text{otherwise,} \end{cases} \tag{23}$$

$$H = \begin{cases} \text{not defined,} & \text{if V = X} \\ \frac{1}{6}\left(5 + \frac{V-B}{V-X}\right), & \text{if R = V and G = X} \\ \frac{1}{6}\left(1 - \frac{V-G}{V-X}\right), & \text{if R = V and B = X} \\ \frac{1}{6}\left(1 + \frac{V-R}{V-X}\right), & \text{if G = V and B = X} \\ \frac{1}{6}\left(3 - \frac{V-B}{V-X}\right), & \text{if G = V and R = X} \\ \frac{1}{6}\left(3 + \frac{V-G}{V-X}\right), & \text{if B = V and R = X} \\ \frac{1}{6}\left(5 - \frac{V-R}{V-X}\right), & \text{if B = V and G = X} \end{cases}. \tag{24}$$

Next, we explain why quaternion models are more advantageous for color image processing, particularly for the HSV color model using *split quaternion* algebra. A representation of color theory using the HSV color space mapped onto a convex cone can be made using *split quaternions* that have a space-time Minkowski metric; therefore, the space has a pseudo-Euclidean metric $\mathbb{R}^{3,1}$. In this sense, considering a color sequence, such as the one depicted in Fig. 2a, we observe that the color vector $q(x, y, z)$ in the color cone moves along a smooth path, as shown in Fig. 2b. On the contrary, we observe that the color vector $q(x, y, z)$ in the RGB space, tracking the same color sequence, moves along an erratic path, as shown in Fig. 2c. In the experimental results of Section V, we present an analysis of both methods for enhancing color images.

(a) Color change sequence.



(b) Sequence in the HSV color space.

(c) Sequence in the RGB color space.

FIGURE 2. Color change sequence from black to red. This figure was created using MATLAB software.

D. MEAN IMAGE VIA RGB COLOR SPACE

The computation of the mean image M between the RGB images A and B (both images of width w, height h, and three channels) in the RGB color space is shown in the pseudocode

of Algorithm 1. The RGB image is returned as a numeric array of $h \times w \times 3$, where the third dimension defines the red, green, and blue channels for each pixel.

Algorithm 1 RGB-Mean Image

forall *pixels* (u, v, w) *in* M **do**
  $M(u, v, w) = \dfrac{A(u, v, w) + B(u, v, w)}{2}$;
end

In Section V, note that image A is computed in the RGB color space and image B is computed using the HSV color model; thus, to compute the mean of both images (and for storage and display), we convert the resulting HSV image to RGB.

E. MEAN IMAGE VIA HSV COLOR SPACE

The Computation of the mean image M between the RGB images A and B (both images of width w, height h, and three channels) in the HSV color space is shown in the pseudocode in Algorithm 2. The function `rgb2hsv` converts the red, green, and blue values of an RGB image into the hue, saturation, and value components of an HSV image, as described in (21)–(24). This HSV image is returned as a numeric array of $h \times w \times 3$ with values in the range [0.0, 1.0], where the third dimension defines the hue, saturation, and value of each pixel. In contrast, the function `hsv2rgb` converts the hue, saturation, and value components of an HSV image to the red, green, and blue values of an RGB image.

In Section V, note that image A is computed in the RGB color model and image B, using the HSV color space; however, to compute the mean image, we need to convert both images to HSV, since they are stored in RGB for display.

Algorithm 2 HSV-Mean Image

$A = \texttt{rgb2hsv}(A_{rgb})$;
$B = \texttt{rgb2hsv}(B_{rgb})$;
forall *pixels* (u, v, w) *in* M **do**
  if $w == 1$ **then**
    $d = \left|A(u, v, w) - B(u, v, w)\right|$;
    if $d > 0.5$ **then**
      $d = 1 - d$;
      if $A(u, v, w) < B(u, v, w)$ **then**
        $M(u, v, w) = \left|A(u, v, w) - 0.5d\right|$;
      else
        $M(u, v, w) = \left|B(u, v, w) - 0.5d\right|$;
      end
    else
      if $A(u, v, w) < B(u, v, w)$ **then**
        $M(u, v, w) = A(u, v, w) + 0.5d$;
      else
        $M(u, v, w) = B(u, v, w) + 0.5d$;
      end
    end
  else
    $M(u, v, w) = \dfrac{A(u, v, w) + B(u, v, w)}{2}$;
  end
end
$M_{rgb} = \texttt{hsv2rgb}(M)$;

III. QUATERNIONIC NEURAL NETWORKS

A. QUATERNIONIC NEURAL NETWORK MODEL

The multilayer quaternionic neural network is built by arranging the nodes or neurons in layers, allowing each neuron in a layer to take only the output of nodes that are in the former layer or in the external input as input. If there are two layers of neurons in the network, the network is called a two-layer network, and so on. Due to its structure, this network is commonly referred to as a *feedforward network* [19]. Fig. 3 shows an example of this architecture.

The neural network model is then expressed as a geometric operation using quaternions or *split quaternions*. The following list of notation defines the parameters used in the very following equations:

- x_j Input of the j^{th} neuron in the input layer.
- y_l Output of the l^{th} neuron in the output layer.
- $\{a\}$ Set of a elements.
- $L_{j,i}$ Input from the j^{th} neuron in the i^{th} layer for neurons in the $(i+1)^{th}$ layer. If $i = 1$, then $L_{j,i} = x_j$.

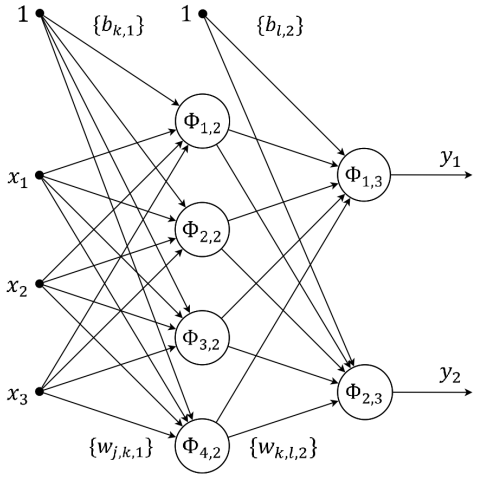


FIGURE 3. A fully connected two-layer quaternionic feedforward network with three inputs, four hidden neurons, and two outputs. This figure was created using Dia software.

- $\Phi_{k,i}$ Activation function of the k^{th} neuron in the i^{th} layer. If the i^{th} layer is the output layer, then $\Phi_{k,i} = y_l$, otherwise, $\Phi_{k,i} = L_{k,i}$.
- $s_{k,i}$ Sum in the k^{th} neuron in the i^{th} layer; this is the argument of the activation function $\Phi_{k,i}$.
- φ Activation function for each part of the quaternionic argument $s_{k,i}$ of $\Phi_{k,i}(s_{k,i})$.

$w_{j,k,i}$ Synaptic weight between the j^{th} neuron, in the i^{th} layer, and the k^{th} neuron in the $(i+1)^{th}$ layer.
$b_{k,i}$ Bias from i^{th} layer, for the k^{th} neuron in the $(i+1)^{th}$ layer.
N_i Total number of neurons in the i^{th} layer.

According to Fig. 3, we can generalize the neural network model for the output $\Phi_{k,i+1}$ of the quaternionic neuron k in layer $(i+1)$, which is the input $L_{k,i+1}$ for the neurons in the $(i+2)^{th}$ layer, as follows:

$$L_{k,i+1} = \Phi_{k,i+1}(s_{k,i+1}) = \Phi_{k,i+1}\left(\sum_{j=1}^{N_i} \frac{w_{j,k,i} \otimes L_{j,i} \otimes w^*_{j,k,i}}{\|w_{j,k,i}\|} + b_{k,i}\right). \quad (25)$$

Similarly, the output of each neuron is determined by the output of the activation function Φ, which is defined as

$$\Phi(s) = \varphi(s^{(e)}) + \varphi(s^{(i)})i + \varphi(s^{(j)})j + \varphi(s^{(k)})k, \quad (26)$$

where $\varphi(a) = \tanh(a)$, for the work of this paper.

Therefore, the output y_l of neural network (NN) of the quaternionic neuron l is defined as

$$y_l = \Phi_{l,3}(s_{l,3}) = \Phi_{l,3}\left(\sum_{k=1}^{N_2} \frac{w_{k,l,2} \otimes L_{k,2} \otimes w^*_{k,l,2}}{\|w_{k,l,2}\|} + b_{l,2}\right); \quad (27)$$

and the output $\Phi_{k,2}$ of the quaternionic neuron k in the hidden layer, which is the input $L_{k,2}$ of the neurons in the output layer, is expressed as

$$L_{k,2} = \Phi_{k,2}(s_{k,2}) = \Phi_{k,2}\left(\sum_{j=1}^{N_1} \frac{w_{j,k,1} \otimes x_j \otimes w^*_{j,k,1}}{\|w_{j,k,1}\|} + b_{k,1}\right). \quad (28)$$

Since bias can be understood as a weight operating on a clamped input to 1, the joint term ''weight'' has frequently been used to include both weights and biases.

In general, multilayer neural networks use a certain number of hidden layers depending on the classification or regression task being addressed. Therefore, we tested up to three hidden layers but found that one hidden layer was sufficient, as the efficiency of the neural networks did not improve with an increase in the number of hidden layers. In addition, what matters is the metric used, as is the case of the *split quaternion* neural network that utilizes the Minkowski metric of the *split quaternion* algebra in a four-dimensional vector space. Interestingly, we can change the metric in a quaternionic neural network to the Euclidean (quaternion algebra) or Minkowski (*split quaternion* algebra) metric, which opens up great versatility in the use of these neural network models.

B. THE EXTENDED KALMAN FILTER PROCEDURE

The Kalman filter is widely known for its ability to estimate the states of a linear system with additive state and output disturbances. The network weights become the states to be estimated while training the neural network using the Kalman filter. Due to the nonlinearity of the neural network mapping, an extended Kalman filter (EKF) algorithm is necessary. Therefore, training aims to find the optimal weight values of the network to reduce the prediction error. The following nonlinear discrete-time system can represent the behavior of a neural network [7]:

$$W(\kappa+1) = W(\kappa) + \omega(\kappa), \quad (29)$$

$$D(\kappa) = h(W(\kappa), u(\kappa)) + \nu(\kappa) = Y(\kappa) + \nu(\kappa). \quad (30)$$

The first of these equations, generally recognized as the process equation, simply specifies that the state of the ideal neural network system is defined as a stationary process which is represented by the values of the network weight parameter $W(\kappa)$, and is altered by the noise of the process $\omega(\kappa)$. The second equation is the observation or measurement equation, which describes the desired response vector $D(\kappa)$ of the network as a nonlinear function of the weight parameter vector $W(\kappa)$ and the input vector $u(\kappa)$, which is disturbed by random measurement noise $\nu(\kappa)$. Additionally, process noise $\omega(\kappa)$ is commonly defined as white noise of zero mean with covariance matrix $Q_\omega(\kappa)$ specified by $Q_\omega(\kappa) = E[\omega(\kappa)\omega^T(\kappa)]$. Similarly, measurement noise $\nu(\kappa)$ is defined as white noise of zero mean with covariance matrix $R_\nu(\kappa)$ provided by $R_\nu(\kappa) = E[\nu(\kappa)\nu^T(\kappa)]$.

C. LEARNING ALGORITHM

The training task for a quaternionic neural network applying the EKF is defined to find the estimate of the minimum mean squared error of the quaternionic state $W^{(p)}$ by utilizing all the data observed up to that moment. Consequently, the following real-time recursion provides the EKF solution to the training problem [7], [9]:

$$A^{(p)}(\kappa) = \left[R^{(p)} + H^{(p)T}(\kappa)P^{(p)}(\kappa)H^{(p)}(\kappa)\right]^{-1}, \quad (31)$$

$$K^{(p)}(\kappa) = P^{(p)}(\kappa)H^{(p)}(\kappa)A^{(p)}(\kappa), \quad (32)$$

$$\hat{W}^{(p)}(\kappa+1) = \hat{W}^{(p)}(\kappa) + \alpha K^{(p)}(\kappa)\xi^{(p)}(\kappa), \quad (33)$$

$$P^{(p)}(\kappa+1) = P^{(p)}(\kappa) - K^{(p)}(\kappa)H^{(p)T}(\kappa)P^{(p)}(\kappa) + Q^{(p)}, \quad (34)$$

where $p = \{e, i, j, k\}$. The scaling matrix $A^{(p)}(\kappa)$ is determined from the measurement noise covariance matrix $R^{(p)}$, approximation error covariance matrix $P^{(p)}(\kappa)$, and network output derivative matrix $H^{(p)}(\kappa)$ related to all trainable weight parameters. For feedforward networks, matrix $H^{(p)}(\kappa)$ is computed using static backpropagation. The Kalman gain matrix $K^{(p)}(\kappa)$ is determined by multiplying the matrices $P^{(p)}(\kappa)$, $H^{(p)}(\kappa)$, and $A^{(p)}(\kappa)$. The vector $\hat{W}^{(p)}(\kappa)$ contains the estimated states or weights of the system at the update step κ and is determined by the learning rate α, the matrix $K^{(p)}(\kappa)$, and the error vector $\xi^{(p)}(\kappa) = d^{(p)}(\kappa) - y^{(p)}(\kappa)$, where $d^{(p)}(\kappa)$ is the target vector and $y^{(p)}(\kappa)$ is the output vector for the κ^{th} exposure to a training pattern. Finally, with the estimation of the weight vector, the matrix $P^{(p)}(\kappa)$ evolves recursively as a function of the matrices $K^{(p)}(\kappa)$ and $H^{(p)}(\kappa)$, and the

covariance matrix of the process noise $Q^{(p)}(\kappa)$. Therefore, it stores the second-derivative information in relation to the training problem. This algorithm seeks the optimal weight values that minimize the average sum of squared errors

$$\frac{1}{4}\sum_{p=\{e,i,j,k\}}\xi^{(p)T}(\kappa)\xi^{(p)}(\kappa). \tag{35}$$

It should be noted that the algorithm requires that the process and measurement noise covariance matrices, $Q^{(p)}$ and $R^{(p)}$, be provided for all training instances and that they are generally diagonal matrices. Similarly, the estimated error covariance matrix $P^{(p)}(\kappa)$ must be established at the beginning of training [7].

Since matrix $H^{(p)}(\kappa)$ is equivalent to the gradient of the output y_l concerning all the weights and biases of the network at step κ, which is expressed as

$$H^{(p)}(\kappa)=\left[\frac{\partial y_l(\kappa)}{\partial w(\kappa)}\right]^{(p)T}_{w(\kappa)=\{w_{j,k,i}(\kappa),b_{k,i}(\kappa)\in\hat{W}(\kappa)\}}, \tag{36}$$

its calculation depends on the quaternionic neural network model that is used. Thus, for the quaternion neural network (QNN) model, the partial derivatives are defined as

$$\partial^{y_l}_{b_{l,2}}=\Phi'_{l,3}(s_{l,3}), \tag{37}$$

$$\partial^{y_l}_{w_{k,l,2}}=\frac{1}{\left\|w_{k,l,2}\right\|}\left(2\partial^{y_l}_{b_{l,2}}\otimes w_{k,l,2}\otimes L^*_{k,2}-\partial^{y_l}_{b_{l,2}}\odot\frac{w_{k,l,2}\otimes L_{k,2}\otimes w^*_{k,l,2}}{\left\|w_{k,l,2}\right\|^2}\otimes w_{k,l,2}\right), \tag{38}$$

$$\partial^{y_l}_{b_{k,1}}=\left(\sum_{l=1}^{N_3}\frac{w^*_{k,l,2}\otimes\partial^{y_l}_{b_{l,2}}\otimes w_{k,l,2}}{\left\|w_{k,l,2}\right\|}\right)\odot\Phi'_{k,2}(s_{k,2}), \tag{39}$$

$$\partial^{y_l}_{w_{j,k,1}}=\frac{1}{\left\|w_{j,k,1}\right\|}\left(2\partial^{y_l}_{b_{k,1}}\otimes w_{j,k,1}\otimes L^*_{j,1}-\partial^{y_l}_{b_{k,1}}\odot\frac{w_{j,k,1}\otimes L_{j,1}\otimes w^*_{j,k,1}}{\left\|w_{j,k,1}\right\|^2}\otimes w_{j,k,1}\right). \tag{40}$$

Note that $\frac{\partial y(k)}{\partial x(k)}=\partial^y_x$, for compactness in the notation and distinguishing these partial derivatives from other derivatives. In addition, note that $\partial^{y_l}_{b_{l,2}}(k)$, $\partial^{y_l}_{w_{k,l,2}}(k)$, $\partial^{y_l}_{b_{k,1}}(k)$, $\partial^{y_l}_{w_{j,k,1}}(k)\in\mathbb{H}$.

Similarly, for the *split quaternion* neural network (SQNN) model, these partial derivatives are defined as

$$\partial^{y_l}_{b_{l,2}}=\Phi'_{l,3}(s_{l,3}), \tag{41}$$

$$\begin{aligned}\partial^{y_l}_{w_{k,l,2}}=\frac{1}{\left\|w_{k,l,2}\right\|}\Bigg[2\Big(&w_{k,l,2}\otimes L^*_{k,2}\otimes\partial^{y_l(i)}_{b_{l,2}}i\\&+\partial^{y_l(j)}_{b_{l,2}}j\otimes L^{*(i)}_{k,2}i\otimes w_{k,l,2}\\&+i\otimes L^{*(j)}_{k,2}j\otimes w_{k,l,2}\otimes\partial^{y_l(j)}_{b_{l,2}}j\otimes i\\&+L^{*(k)}_{k,2}k\otimes w_{k,l,2}\otimes\partial^{y_l(j)}_{b_{l,2}}j\\&+\partial^{y_l(k)}_{b_{l,2}}k\otimes L^{*(i)}_{k,2}i\otimes w_{k,l,2}\\&+L^{*(j)}_{k,2}j\otimes w_{k,l,2}\otimes\partial^{y_l(k)}_{b_{l,2}}k\\&+i\otimes L^{*(k)}_{k,2}k\otimes w_{k,l,2}\otimes\partial^{y_l(k)}_{b_{l,2}}k\otimes i\Big)\\&-\partial^{y_l}_{b_{l,2}}\odot\frac{w_{k,l,2}\otimes L_{k,2}\otimes w^*_{k,l,2}}{\left\|w_{k,l,2}\right\|^4}\otimes\lambda_1q_1\Bigg],\end{aligned} \tag{42}$$

$$\partial^{y_l}_{b_{k,1}}=\left(\sum_{l=1}^{N_3}\frac{w_{k,l,2}\otimes\partial^{y_l}_{b_{l,2}}\otimes w^*_{k,l,2}}{\left\|w_{k,l,2}\right\|}\right)\odot\Phi'_{k,2}(s_{k,2}), \tag{43}$$

$$\begin{aligned}\partial^{y_l}_{w_{j,k,1}}=\frac{1}{\left\|w_{j,k,1}\right\|}\Bigg[2\Big(&w_{j,k,1}\otimes L^*_{j,1}\otimes\partial^{y_l(i)}_{b_{k,1}}i\\&+\partial^{y_l(j)}_{b_{k,1}}j\otimes L^{*(i)}_{j,1}i\otimes w_{j,k,1}\\&+i\otimes L^{*(j)}_{j,1}j\otimes w_{j,k,1}\otimes\partial^{y_l(j)}_{b_{k,1}}j\otimes i\\&+L^{*(k)}_{j,1}k\otimes w_{j,k,1}\otimes\partial^{y_l(j)}_{b_{k,1}}j\\&+\partial^{y_l(k)}_{b_{k,1}}k\otimes L^{*(i)}_{j,1}i\otimes w_{j,k,1}\\&+L^{*(j)}_{j,1}j\otimes w_{j,k,1}\otimes\partial^{y_l(k)}_{b_{k,1}}k\\&+i\otimes L^{*(k)}_{j,1}k\otimes w_{j,k,1}\otimes\partial^{y_l(k)}_{b_{k,1}}k\otimes i\Big)\\&-\partial^{y_l}_{b_{k,1}}\odot\frac{w_{j,k,1}\otimes L_{j,1}\otimes w^*_{j,k,1}}{\left\|w_{j,k,1}\right\|^4}\otimes\lambda_2q_2\Bigg].\end{aligned} \tag{44}$$

where:

$$\begin{aligned}\lambda_1&=w^{(e)2}_{k,l,2}+w^{(i)2}_{k,l,2}-w^{(j)2}_{k,l,2}-w^{(k)2}_{k,l,2}\mid\lambda_1\in\mathbb{R},\\q_1&=w^{(e)}_{k,l,2}+w^{(i)}_{k,l,2}i-w^{(j)}_{k,l,2}j-w^{(k)}_{k,l,2}k\mid q_1\in\mathbb{H}_s,\\\lambda_2&=w^{(e)2}_{j,k,1}+w^{(i)2}_{j,k,1}-w^{(j)2}_{j,k,1}-w^{(k)2}_{j,k,1}\mid\lambda_2\in\mathbb{R},\\q_2&=w^{(e)}_{j,k,1}+w^{(i)}_{j,k,1}i-w^{(j)}_{j,k,1}j-w^{(k)}_{j,k,1}k\mid q_2\in\mathbb{H}_s.\end{aligned}$$

Again, note that $\frac{\partial y(k)}{\partial x(k)}=\partial^y_x$, for compactness in the notation and distinguishing these partial derivatives from other derivatives. In this case, note that $\partial^{y_l}_{b_{l,2}}(k)$, $\partial^{y_l}_{w_{k,l,2}}(k)$, $\partial^{y_l}_{b_{k,1}}(k)$, $\partial^{y_l}_{w_{j,k,1}}(k)\in\mathbb{H}_s$.

IV. LOW-LIGHT IMAGE ENHANCEMENT

A. PROBLEM STATEMENT

Consider the schematic represented in Fig. 4. In this scheme, a quaternionic feedforward neural network is used to extract pixel values from an input low-light image and obtain new values as the output of this quaternionic neural network to finally reconstruct an output image where the original scene is appreciated with brightness. Therefore, to train this neural network, a pair of images of the same scene is necessary. Whereas the first image is selected so that there is sufficient illumination to appreciate the scene, the second image is selected so that the original scene is difficult to appreciate due to a low illumination condition. In this order, the first image is set as the target image or the desired output, and the second image is set as the input for the neural network.

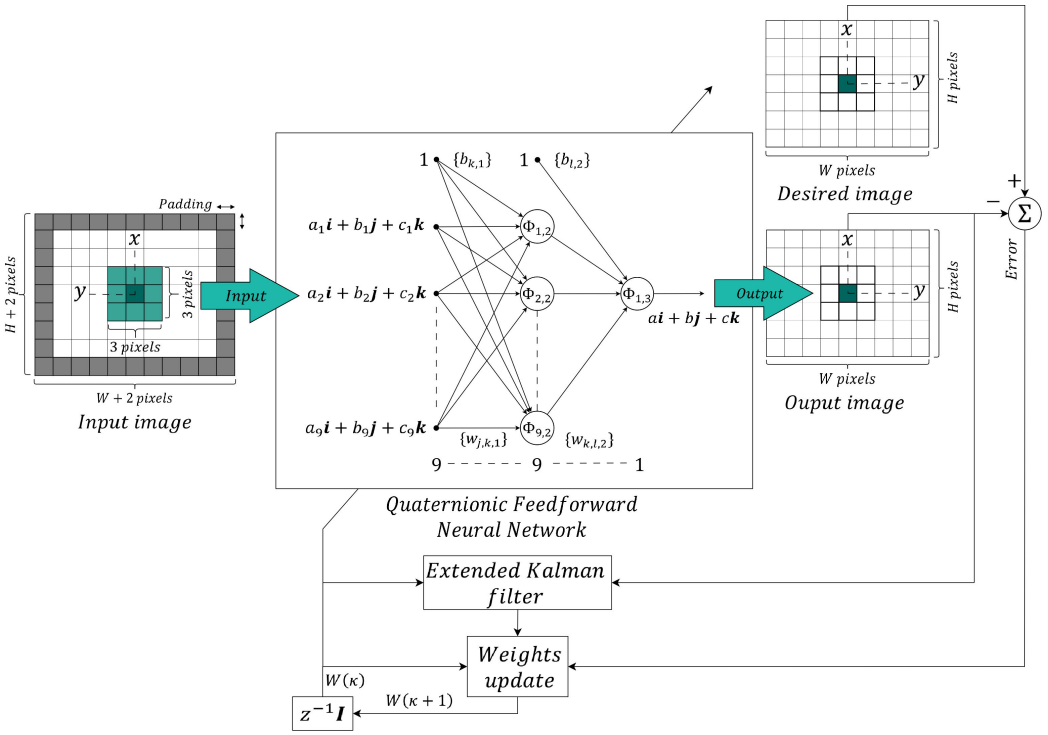


FIGURE 4. Schematic of the training framework for the quaternion feedforward neural network and the *split quaternion* feedforward neural network. This figure was created using Dia and PowerPoint software.

This training method is based on the results reported by Kusamichi et al. in [11], where a quaternion feedforward neural network is trained by the backpropagation algorithm and used in their proposed method for the implementation of a color night vision system.

Thus, nine pixels of the input image are used as input for the neural network. The pixel values of the pixel located at the coordinates (x, y) are set as input, together with the pixel values of the immediate neighboring pixels located at $(x + i, y + j)$, where the pair $(i, j) \in \mathbb{Z}$, such that $\{(i, j)\} = \{(-1, -1), (0, -1), (1, -1), (-1, 0), (1, 0), (-1, 1), (0, 1), (1, 1)\}$. Thus, the output of the neural network corresponds to the pixel values of the pixel located at coordinates (x, y) in the output image, and the desired output corresponds to the pixel values of the pixel located at coordinates (x, y) in the desired image.

Considering that one quaternionic input is necessary for the three components of a single input pixel, and one quaternionic output is sufficient for the three components of the output pixel, the number of quaternionic neurons for the input and output of the neural network is then equal to the number of previously established input and output pixels, respectively.

It should be noted that multilayer neural networks use a certain number of hidden layers, depending on the classification or regression task. To address this problem, we found that one hidden layer was sufficient because the efficiency of the neural network did not improve with an increase in the number of hidden layers. In addition, what really matters is that the quaternionic neural network model, even with one hidden layer, uses either the Minkowski metric or the Euclidean metric in a four-dimensional space.

In this manner, the structure of the quaternionic neural network used corresponds to a fully connected three-layer feedforward network with nine input neurons, one output neuron, and, according to the preliminary tests carried out, one hidden layer with nine neurons.

Hence, to address this color enhancement problem, two quaternionic feedforward neural networks were used. These two versions were trained using the extended Kalman filter (EKF) algorithm, which was described for both neural network models in the previous section. Thus, two different cases were configured to show the results according to the neural network model and the color space used to treat the images. The two cases are a Quaternion Feedforward Neural Network (QFFNN) using images in the RGB color space, and a *Split Quaternion* Feedforward Neural Network (SQFFNN) using images in the HSV color space.

Note that the input and output of the neural network for this problem are *pure quaternions* (or *pure split quaternions*), where each imaginary part or base stores one component of the color triplet for a single pixel. In the first case, where the RGB color space is used, the coefficient of the imaginary unit i corresponds to the red channel, the coefficient of the imaginary unit j belongs to the green channel, and the coefficient of the imaginary unit k refers to the blue channel. And in the case where the HSV color space is

used, it is necessary to change the HSV values in Cartesian coordinates to adequately treat these values, since they are commonly measured in the closed interval [0.0, 1.0]. Thus, the conversion is as follows:

$$\theta = 2\pi \cdot H \text{ (for the Hue)}, \tag{45}$$
$$\rho = S \cdot V \text{ (for the Saturation)}, \tag{46}$$
$$z = V \text{ (for the Value)}, \tag{47}$$
$$X = \rho \cdot \cos(\theta), \tag{48}$$
$$Y = \rho \cdot \sin(\theta). \tag{49}$$
$$Z = z. \tag{50}$$

The necessity of this conversion can be observed in Fig. 5.

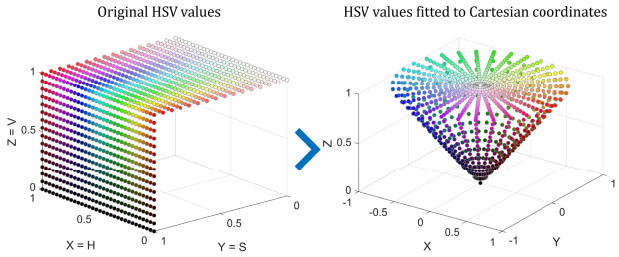


FIGURE 5. Graphic representation of the change from HSV values to Cartesian coordinates. This figure was created using MATLAB and PowerPoint software.

Therefore, for the second case, where SQFFNN is used, the coefficient of the imaginary unit i corresponds to the Z channel, the coefficient of the base j belongs to the Y channel, and the coefficient of the base k refers to the X channel. Hence, we obtain the following color vector:

$$q(x, y, z) = Zi + Yj + Xk, \tag{51}$$

where $q \in \mathbb{H}_S$, and $X, Y, Z \in \mathbb{R}$.

B. DATASET FOR TRAINING

The CIFAR-10 dataset [20] was used to train the two neural network models. This dataset consists of 60,000 32×32 pixel color images in 10 classes (airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck), with 6000 images per class, where there are 50,000 training images and 10,000 test images. However, we used only the training set and other images with a better resolution for testing.

Consequently, we used two versions for each image. The first version corresponds to the original image, where there is enough light to appreciate the scene, and the second version is configured so that there is a low-light condition to visualize the original scene adequately.

We determined that using this dataset is advantageous because it has a wider spectrum of colors and illumination levels since every scene of each class might possess a landscape, room, sky, or sea as background in addition to the object or animal of the class. Furthermore, due to the low resolution of the images, the loading and transformation of each image would be faster during training.

In this manner, two versions of each image were used for training. The first version is equivalent to the original image, and the second version is configured so that the values of all pixels decrease 80% their level in the value channel of the HSV color map, and consequently, a darkened image is obtained. The two versions were images of 32×32 pixels, whose values are represented in a 64-bit format. In this order, they were set as the desired image and the input image for the training method, respectively.

Since 1024 (32×32) pixels constitute each image for training, performing an iteration using all pixels would represent a high computational cost, in addition to the redundancy that would occur due to the processing of too much similar information. Subsequently, four pixels were used for each training iteration as an arbitrary and suitable choice. These four pixels are selected in a random manner, whose distribution function is uniform, considering from the pixel at the leftmost and uppermost location to the pixel at the rightmost and lowermost location in the randomly selected image for the current training epoch.

C. COST FUNCTION FOR TRAINING

The two neural networks were trained using the EKF algorithm, where the training cost function at iteration κ is given by

$$J(\kappa) = \frac{1}{3}\left(\frac{1}{4}\sum_{n=1}^{4}\left(d_n^{(i)} - y_n^{(i)}\right)^2 + \frac{1}{4}\sum_{n=1}^{4}\left(d_n^{(j)} - y_n^{(j)}\right)^2 + \frac{1}{4}\sum_{n=1}^{4}\left(d_n^{(k)} - y_n^{(k)}\right)^2\right) \tag{52}$$

where $d_n^{(l)}$ is the desired output in the base l for the random pixel n; $y_n^{(l)}$ is the neural network output in the base l for the random pixel n; $\frac{1}{4}\sum_{n=1}^{4}\left(d_n^{(l)} - y_n^{(l)}\right)^2$ is the mean squared error in the base l of the four random pixels selected for the iteration κ; and thus, $J(\kappa)$ is basically the average of the mean squared errors, of the three bases, at iteration κ.

For the two cases, the training was set to end when the EKF minimized the cost function to a value less than $1 \cdot 10^{-4}$.

D. A PARTICULAR IMAGE FOR TESTING

To evaluate the performance of each neural network, we created and selected a distinctive image with very particular content. Therefore, the first image corresponds to the transformation of the values of the HSV color space into a plane. The space of this model is represented by a three-dimensional region whose shape is a cone. The alignment is more closely correlated with the way human vision perceives and distinguishes the colors of the visible light spectrum. In general, this model can be more useful for choosing a specific color. Fig. 6 shows the cone of the HSV

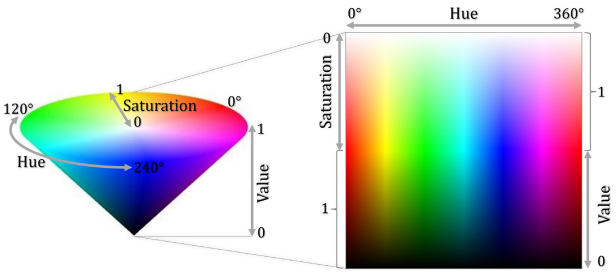


FIGURE 6. Transformation of the HSV space into a plane. This figure was created using MATLAB and PowerPoint software.

color space and the image that results from the transformation of its external values into a plane.

Using this image for testing could be more advantageous because it has a wider color spectrum than any other image that might have a landscape, room, person, animal, or any other object in the scene. Thus, we could observe a more general performance for each neural network model along the color spectrum.

V. RESULTS

This section presents the results that each neural network model produced using its specific quaternionic algebra, its specific color space for image processing, and the image reconstruction scheme explained in Section IV-A.

It is worth mentioning that each neural network configuration required different epochs to complete its training and minimize its cost function to a value less than $1 \cdot 10^{-4}$. Quaternion Feedforward Neural Network (QFFNN) with RGB color space required 1012 iterations, whereas *Split Quaternion* Feedforward Neural Network (SQFFNN) with HSV color space required 1208 iterations. In both cases, a learning rate of 0.1 was used.

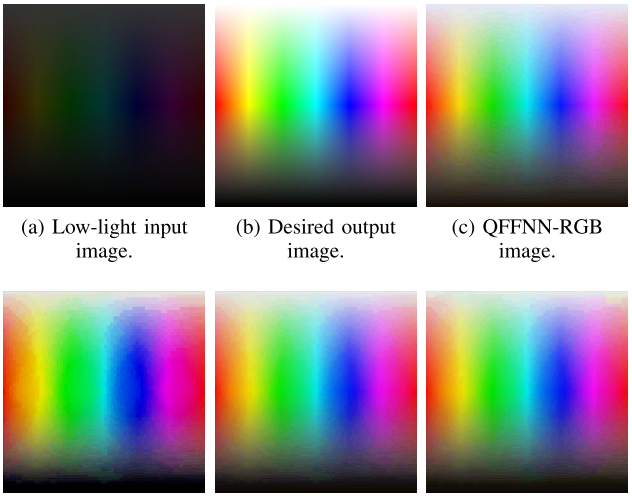
A. EXPERIMENTAL RESULTS ON LOW-LIGHT CONDITIONED IMAGES

Figs. 7, 8, 9, 10, and 11 show the results produced by each neural network model on several images. In them, Fig. x.a shows the input image; Fig. x.b, the desired image; Fig. x.c, the image reconstructed by QFFNN using the RGB color space; Fig. x.d, the image reconstructed by SQFFNN using the HSV color space; Fig. x.e, the image obtained by computing the average image through the RGB color space, as defined in Algorithm 1, using Figs. x.c and x.d; and Fig. x.f, the image obtained by computing the mean image through the HSV color space, as defined in Algorithm 2, using Figs. x.c and x.d again.

The original images used in these figures were chosen for the following reasons. The image in Fig. 7b was preferred for the reason explained in Section IV-D. The images in Figs. 8b, 9b, and 10b were selected because they correspond to three of the ten classes that the CIFAR-10 dataset considers in its content. And Fig. 11b was elected because we wanted to evaluate the performance of the neural network models on human skin tone, because the dataset used for their training does not contain a class on people.

As was done during training, a darkened version of each original image was configured for testing. This version corresponds to the one shown in Fig. x.a of the aforementioned figures and was conditioned so that the values of all pixels decreased 80% their level in the value channel of the HSV color space. The two versions, as well as those consequently reconstructed, are images with a resolution of 1024×1024 pixels, whose values are represented in a 64-bit format.

In addition, each subsequent table (Table 1 after Fig. 7, Table 2 after Fig. 8, Table 3 after Fig. 9, Table 4 after Fig. 10, and Table 5 after Fig. 11) shows how quantitatively similar the input and reconstructed images are to the original image with respect to the *mean squared error (MSE)*, *peak signal-to-noise ratio (PSNR)*, *structural similarity (SSIM)*, *Chi-Square*, and *Intersection* metrics. The *MSE* represents the mean squared error value calculated based on the difference in pixels between the reference and reconstructed images. The *PSNR* determines the quality of the image reconstruction by



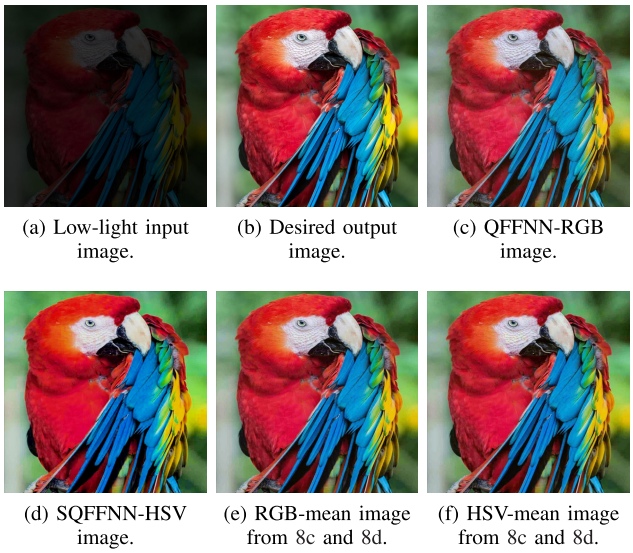
(a) Low-light input image. (b) Desired output image. (c) QFFNN-RGB image.

(d) SQFFNN-HSV image. (e) RGB-mean image from 7c and 7d. (f) HSV-mean image from 7c and 7d.

FIGURE 7. Comparison of the original input (7a) and output (7b) images with those reconstructed by QFFNN with RGB (7c), SQFFNN with HSV (7d), and those obtained by computing the mean image through the RGB (7e) and HSV (7f) color spaces, respectively, for test image 1 (7b). Source of the original image (7b): Primary.

TABLE 1. Quantitative comparison of the enhancement effect according to the results obtained in Fig. 7 for test image 1 (7b).

Fig.	MSE↓	PSNR↑	SSIM↑	Chi-Square↓	Intersection↑
7a)	17051.7	5.8	0.39	335.0	7.0
7b)	0	inf	1	0	341.3
7c)	⋆162.1	⋆26.0	•0.94	404.7	3.3
7d)	1069.2	17.8	0.81	⋆339.8	1.8
7e)	•372.7	•22.4	⋆0.941	371.3	⋆4.2
7f)	383.5	22.3	0.936	•369.4	•3.6



(a) Low-light input image. (b) Desired output image. (c) QFFNN-RGB image.

(d) SQFFNN-HSV image. (e) RGB-mean image from 8c and 8d. (f) HSV-mean image from 8c and 8d.

FIGURE 8. Comparison of the original input (8a) and output (8b) images with those reconstructed by QFFNN with RGB (8c), SQFFNN with HSV (8d), and those obtained by computing the mean image through the RGB (8e) and HSV (8f) color spaces, respectively, for test image 2 (8b). Source of the original image (8b): Adapted from [24].

TABLE 2. Quantitative comparison of the enhancement effect according to the results obtained in Fig. 8 for test image 2 (8b).

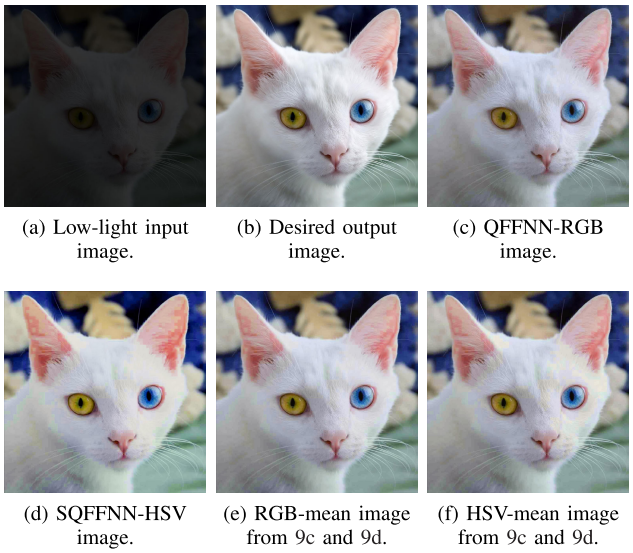
Fig.	MSE↓	PSNR↑	SSIM↑	Chi-Square↓	Intersection↑
8a)	9409.8	8.4	0.34	6805.3	50.3
8b)	0	inf	1	0	1667.1
8c)	⋆126.3	⋆27.12	0.83	3840.0	80.6
8d)	325.5	23.0	0.79	⋆1857.9	48.6
8e)	•126.5	•27.11	⋆0.845	•2194.8	⋆111.0
8f)	129.6	27.01	•0.842	2374.2	•109.3

TABLE 3. Quantitative comparison of the enhancement effect according to the results obtained in Fig. 9 for test image 3 (9b).

Fig.	MSE↓	PSNR↑	SSIM↑	Chi-Square↓	Intersection↑
9a)	16735.4	5.9	0.35	1962.6	0.9
9b)	0	inf	1	0	689.9
9c)	⋆108.8	⋆27.8	⋆0.91	5064.3	23.8
9d)	385.4	22.3	0.88	⋆2254.0	17.3
9e)	•125.9	•27.1	•0.909	•3384.7	⋆28.7
9f)	129.0	27.0	0.908	3704.4	•24.5

computing the error between the corresponding pixels, that is, the *MSE* [21]. The *SSIM* defines structural information from the perspective of the composition of the image, reflecting the structural attributes of the scene and considering the mean to estimate the brightness, standard deviation for contrast, and covariance to compute the degree of structural similarity [22]. The *Chi-Square* and *Intersection* methods obtain a numerical parameter that expresses how well the two histograms of the images match each other [23].

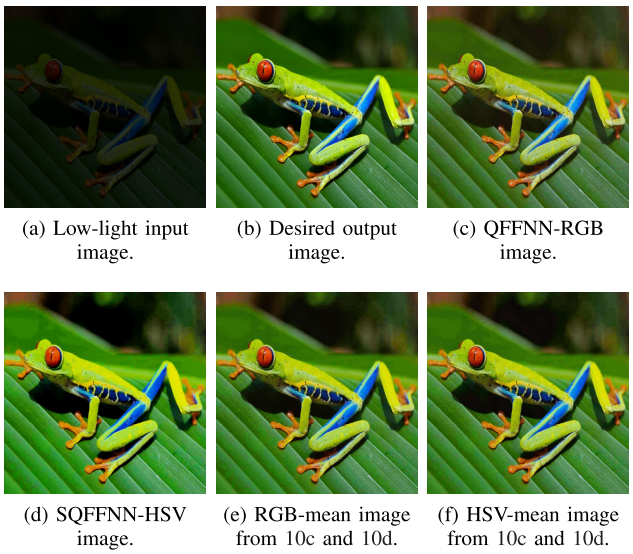
In the same tables (Tables 1, 2, 3, 4, and 5), the ↓ marker means that the lower the metric, the better the match, which is the case for the *MSE* and *Chi-Square* methods; and the



(a) Low-light input image. (b) Desired output image. (c) QFFNN-RGB image.

(d) SQFFNN-HSV image. (e) RGB-mean image from 9c and 9d. (f) HSV-mean image from 9c and 9d.

FIGURE 9. Comparison of the original input (9a) and output (9b) images with those reconstructed by QFFNN with RGB (9c), SQFFNN with HSV (9d), and those obtained by computing the mean image through the RGB (9e) and HSV (9f) color spaces, respectively, for test image 3 (9b). Source of the original image (9b): Adapted from [25].



(a) Low-light input image. (b) Desired output image. (c) QFFNN-RGB image.

(d) SQFFNN-HSV image. (e) RGB-mean image from 10c and 10d. (f) HSV-mean image from 10c and 10d.

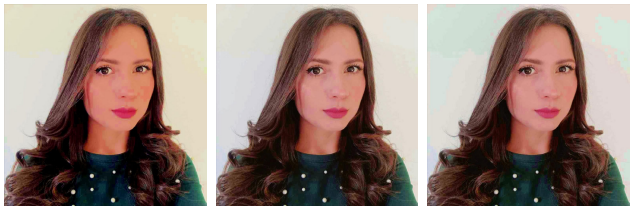
FIGURE 10. Comparison of the original input (10a) and output (10b) images with those reconstructed by QFFNN with RGB (10c), SQFFNN with HSV (10d), and those obtained by computing the mean image through the RGB (10e) and HSV (10f) color spaces, respectively, for test image 4 (10b). Source of the original image (10b): Adapted from [26].

↑ marker indicates that the higher the metric, the more accurate the match, which is the case for the *PSNR*, *SSIM* and *Intersection* methods. In this sense, we can observe or expect a perfect match when we compare the original or base image with itself, so that the parameter obtained serves as a reference to compare the quality of the reconstructed images according to each of the metrics used. Additionally, for each metric, we can notice that the ⋆ marker indicates which model

TABLE 4. Quantitative comparison of the enhancement effect according to the results obtained in Fig. 10 for test image 4 (10b).

Fig.	MSE↓	PSNR↑	SSIM↑	Chi-Square↓	Intersection↑
10a)	4905.5	11.2	0.42	2014.5	6.8
10b)	0	inf	1	0	197.1
10c)	137.9	26.7	0.81	12531.9	7.3
10d)	310.7	23.2	0.807	⋆353.0	3.4
10e)	⋆116.6	⋆27.5	⋆0.84	•8375.6	•13.6
10f)	•122.7	•27.2	•0.83	9133.1	⋆15.0

(a) Low-light input image. (b) Desired output image. (c) QFFNN-RGB image.

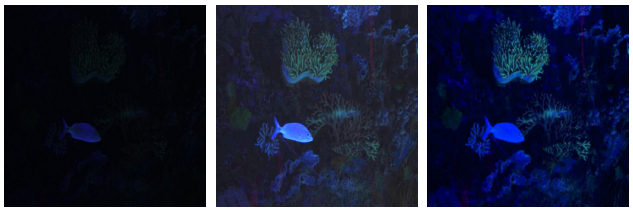


(d) SQFFNN-HSV image. (e) RGB-mean image from 11c and 11d. (f) HSV-mean image from 11c and 11d.

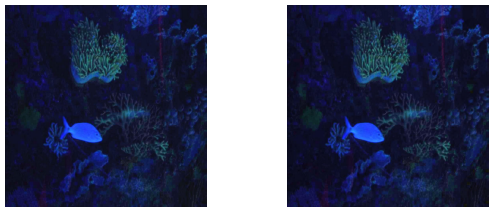
FIGURE 11. Comparison of the original input (11a) and output (11b) images with those reconstructed by QFFNN with RGB (11c), SQFFNN with HSV (11d), and those obtained by computing the mean image through the RGB (11e) and HSV (11f) color spaces, respectively, for test image 5 (11b). Source of the original image (11b): Primary.

TABLE 5. Quantitative comparison of the enhancement effect according to the results obtained in Fig. 11 for test image 5 (11b).

Fig.	MSE↓	PSNR↑	SSIM↑	Chi-Square↓	Intersection↑
11a)	17366.3	5.7	0.32	10268.5	1.8
11b)	0	inf	1	0	64.8
11c)	⋆219.6	⋆24.7	0.819	⋆917.6	3.5
11d)	398.1	22.1	0.76	•1345.4	2.5
11e)	•228.1	•24.5	⋆0.828	7825.2	⋆4.8
11f)	263.1	23.9	•0.821	1914.7	•4.4



(a) Low-light input image. (b) QFFNN-RGB image. (c) SQFFNN-HSV image.



(d) RGB-mean image from 12b and 12c. (e) HSV-mean image from 12b and 12c.

FIGURE 12. Comparison between the original input image (12a) and those reconstructed by QFFNN with RGB (12b), SQFFNN with HSV (12c), and those obtained by computing the mean image through the RGB (12d) and HSV (12e) color spaces, respectively, for test image 6 (12a). Source of the original image (12a): Adapted from [27].

(a) Low-light input image. (b) QFFNN-RGB image. (c) SQFFNN-HSV image.

(d) RGB-mean image from 13b and 13c. (e) HSV-mean image from 13b and 13c.

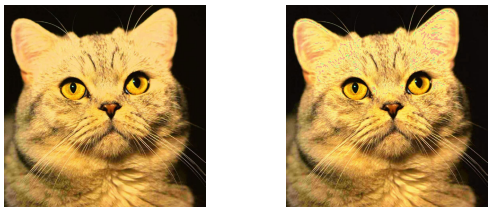
FIGURE 13. Comparison between the original input image (13a) and those reconstructed by QFFNN with RGB (13b), SQFFNN with HSV (13c), and those obtained by computing the mean image through the RGB (13d) and HSV (13e) color spaces, respectively, for test image 7 (13a). Source of the original image (13a): Adapted from [28].

or reconstructed image obtained the best result, whereas the · marker indicates the second-best result.

From the graphic results that we observed in the last five figures (Figs. 7, 8, 9, 10, and 11), as well as from the following ones presented in the next subsection (Figs. 12, 13, 14, 15, 16, 17, 18, and 19), we can realize that the QFFNN model works on the intensity in each of the RGB channels, whereas the SQFFNN model works not only on the value channel (brightness) but also on the saturation channel of the HSV color space. We can also observe that the adjustment made in the red, green, and blue components in the RGB color space by QFFNN, as well as the adjustment in the saturation and value components in the HSV color space by SQFFNN, is not precisely to the same extent.

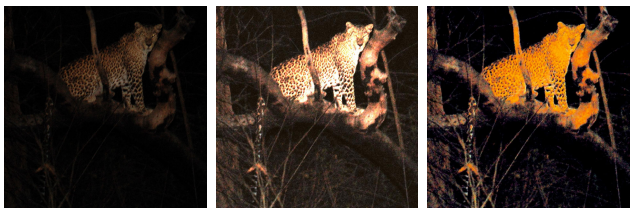
Moreover, from the numerical results in Tables 1, 2, 3, 4, and 5, we can observe several results that generally favor a certain image reconstruction method based on the evaluation metric used. For the *MSE* metric, and consequently for the *PSNR* metric, the images reconstructed by QFFNN (Figs. x.c), followed by the RGB-mean images (Figs. x.e), obtained a better quantitative match in most of the results. For the *SSIM* metric, the RGB-mean images (Figs. x.e) produced

(a) Low-light input image. (b) QFFNN-RGB image. (c) SQFFNN-HSV image.



(d) RGB-mean image from 14b and 14c. (e) HSV-mean image from 14b and 14c.

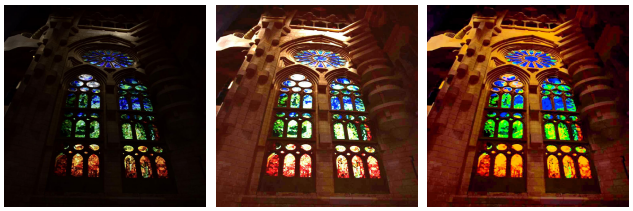
FIGURE 14. Comparison between the original input image (14a) and those reconstructed by QFFNN with RGB (14b), SQFFNN with HSV (14c), and those obtained by computing the mean image through the RGB (14d) and HSV (14e) color spaces, respectively, for test image 8 (14a). Source of the original image (14a): Adapted from [29].



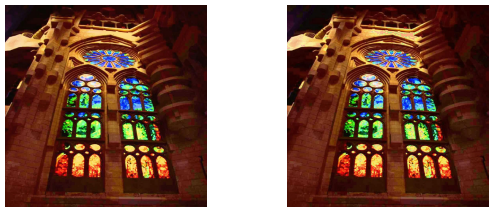
(a) Low-light input image. (b) QFFNN-RGB image. (c) SQFFNN-HSV image.

(d) RGB-mean image from 15b and 15c. (e) HSV-mean image from 15b and 15c.

FIGURE 15. Comparison between the original input image (15a) and those reconstructed by QFFNN with RGB (15b), SQFFNN with HSV (15c), and those obtained by computing the mean image through the RGB (15d) and HSV (15e) color spaces, respectively, for test image 9 (15a). Source of the original image (15a): Adapted from [30].

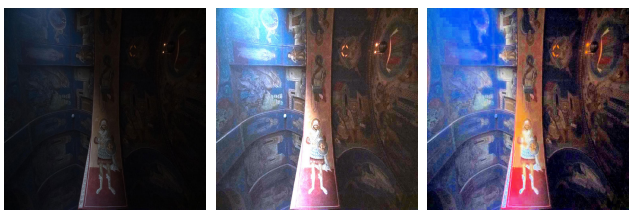


(a) Low-light input image. (b) QFFNN-RGB image. (c) SQFFNN-HSV image.

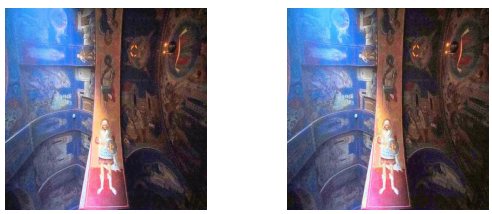


(d) RGB-mean image from 16b and 16c. (e) HSV-mean image from 16b and 16c.

FIGURE 16. Comparison between the original input image (16a) and those reconstructed by QFFNN with RGB (16b), SQFFNN with HSV (16c), and those obtained by computing the mean image through the RGB (16d) and HSV (16e) color spaces, respectively, for test image 10 (16a). Source of the original image (16a): Adapted from [31].



(a) Low-light input image. (b) QFFNN-RGB image. (c) SQFFNN-HSV image.



(d) RGB-mean image from 17b and 17c. (e) HSV-mean image from 17b and 17c.

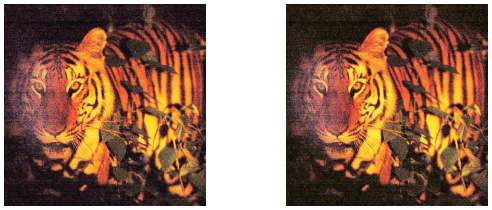
FIGURE 17. Comparison between the original input image (17a) and those reconstructed by QFFNN with RGB (17b), SQFFNN with HSV (17c), and those obtained by computing the mean image through the RGB (17d) and HSV (17e) color spaces, respectively, for test image 11 (17a). Source of the original image (17a): Adapted from [32].

a better quantitative match in most cases. For the *Chi-Square* metric, SQFFNN (Figs. x.d) obtained a better quantitative match with respect to its image reconstruction quality in most of the results. And for the *Intersection* metric, the similarity parameter increased when a mean image was computed in the RGB color space (Figs. x.e), and even in the HSV color space (Figs. x.f), through both images reconstructed by each of the quaternionic neural network models (Figs. x.c and x.d).

B. EXPERIMENTAL RESULTS ON IMAGES WITH NATURAL LOW-LIGHT CONDITIONS

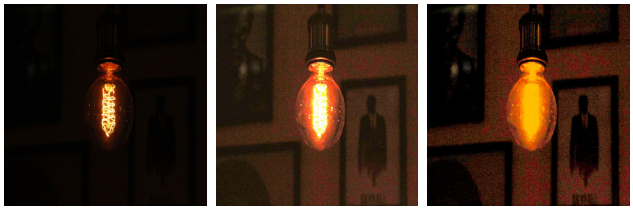
Figs. 12, 13, 14, 15, 16, 17, 18, and 19 show the results produced by each neural network model on several images with natural low-light conditions. In them, Fig. x.a shows the original input image; Fig. x.b, the image reconstructed by QFFNN using the RGB color space; Fig. x.c, the image reconstructed by SQFFNN using the HSV color space; and

(a) Low-light input image. (b) QFFNN-RGB image. (c) SQFFNN-HSV image.

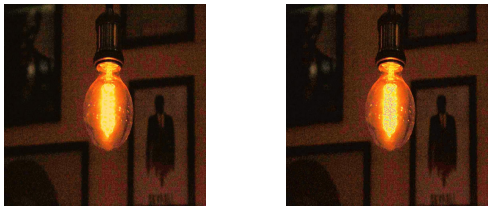


(d) RGB-mean image from 18b and 18c. (e) HSV-mean image from 18b and 18c.

FIGURE 18. Comparison between the original input image (18a) and those reconstructed by QFFNN with RGB (18b), SQFFNN with HSV (18c), and those obtained by computing the mean image through the RGB (18d) and HSV (18e) color spaces, respectively, for test image 12 (18a). Source of the original image (18a): Adapted from [33].



(a) Low-light input image. (b) QFFNN-RGB image. (c) SQFFNN-HSV image.



(d) RGB-mean image from 19b and 19c. (e) HSV-mean image from 19b and 19c.

FIGURE 19. Comparison between the original input image (19a) and those reconstructed by QFFNN with RGB (19b), SQFFNN with HSV (19c), and those obtained by computing the mean image through the RGB (19d) and HSV (19e) color spaces, respectively, for test image 13 (19a). Source of the original image (19a): Adapted from [34].

Figs. x.d and x.e display the results of computing an average image through the RGB and HSV color spaces, respectively, using Figs. x.b and x.c.

The original images used in these figures (Figs. 12a, 13a, 14a, 15a, 16a, 17a, 18a, and 19a) were particularly picked due to their various degrees of natural low light intensity, in addition to the fact that in each of them there is no homogeneous light condition throughout the entire image. Furthermore, not all correspond precisely to any of the classes considered by the CIFAR-10 dataset in its content. These underexposed input images, as well as those consequently reconstructed, are images with a resolution of 1024×1024 pixels, whose values are represented in a 64-bit format.

From the graphic results observed in the last eight figures (Figs. 12, 13, 14, 15, 16, 17, 18, and 19), it is more evident that the quaternion neural network model works on the intensity in each of the RGB channels, whereas the *split quaternion* neural network model works not only on the value channel (brightness), but also on the saturation channel of the HSV color space. Similarly, we can observe that both quaternionic neural networks treat in a very particular way those regions in the images that had better illumination before their processing, thus obtaining a supersaturated region in terms of intensity for the case of QFFNN, and in terms of color saturation for the case of SQFFNN. Note that this effect is to be expected because both neural network models were trained with input scenes in which the low-light condition was homogeneous or uniform throughout the entire image.

If we wished to quantitatively analyze the differences in these results, as was done in the previous subsection using the match metrics briefly described in Section V-A, we would need a base image with a ''good'' or adequate exposure. However, since the original images (Figs. x.a) correspond to underexposed photographs, we would have to generate a new version of each one as the ground truth through an improvement in brightness and shadows, for example, either manually or by another method of color adjustment.

VI. CONCLUSION

In this study, a Quaternion Feedforward Neural Network (QFFNN) and *Split Quaternion* Feedforward Neural Network (SQFFNN) were used to enhance low-light images. This method extracts information from the scene of a low-light or underexposed image to produce an image in which the original scene is appreciated with improved illumination.

From the figures and tables presented in Section V, we can observe several experimental results that could generally favor a certain quaternionic neural network model, depending on the quality of the resulting image or the evaluation metric used. In that sense, particularly for the *MSE* metric (and consequently the *PSNR* metric), QFFNN has a better quantitative performance, which is understandable because the RGB color model focuses on intensity, which is the essence of these metrics that, without a doubt, favor the improvement of intensity. Now, considering the *Chi-Square* metric, SQFFNN has a better quantitative match with respect to its image reconstruction quality, indicating that the histogram comparison of this metric suggests to be more sensitive to hue and saturation. However, the same results reveal that taking into account the *Intersection* metric, the similarity parameter increases when a mean image is computed in the RGB color space, and even in the HSV color space, through both images reconstructed by each quaternionic neural network, which indicates that the images reconstructed by QFFNN could be improved in terms of

color saturation, since this model only works on the intensity of RGB channels, whereas SQFFNN works on the value (brightness) and saturation components.

However, a conclusive theoretical explanation for qualifying the QFFNN model (based on the RGB color space or the Euclidean metric) and the QFFNN model (based on the HSV color space or the Minkowski metric) is difficult, as we are trying to approach the subject of human perception because, as a cognitive process, there is no clear metric that describes it, nor a general unifying theory for image enhancement that can serve as a design criterion.

It should be remembered that the images presented in Section V-A were obtained by setting low-light input images with lighting similar to that used in the images for training, whereas the images presented in Section V-B were particularly selected due to their various degrees of natural low light intensity.

Although both quaternionic neural network models perform image reconstruction by improving the illumination of images that are underexposed or could benefit from light, we note that these images could still be improved in aspects such as sharpening, contrast, denoising, and color balance adjustment. Even, in particular, we note that the results obtained in Fig. 11 could still be improved to reproduce the skin tone more faithfully, since humans relate to skin tones more critically than other colors; water, grass, environment, sky, animals, or any other object might look off or saturated without concern, but if the skin tones look off or saturated, the human subject might seem sick or dead, for example. However, these considerations are beyond the scope of the present study.

The experimental results suggest the utility and performance that multilayer quaternionic feedforward neural network models can exhibit against other more complex or deeper neural network structures. In particular, what makes this work significant is the use of *split quaternion* algebra to design a neural network model based on the geometric operations of *split quaternions* for color image processing, as well as the application of the extended Kalman filter technique to design a learning algorithm for this neural network model. In addition, we can observe that the *split quaternion* neural network using the HSV color model shows advantages not previously published, which are not shown by the quaternion neural network using the RGB color model. Therefore, this work presents a novel quaternionic neural network using the Minkowski metric for color processing, which can be advantageously used by practitioners interested in working with the HSV color model.

APPENDIX A
MATRIX REPRESENTATION OF SPLIT QUATERNIONS

Let $x = x^{(e)} + x^{(i)}i + x^{(j)}j + x^{(k)}k$ be a *split quaternion* and $y = y^{(i)}i + y^{(j)}j + y^{(k)}k$ a *pure split quaternion*. These *split quaternions* can then be represented in vector form using their real coefficients as elements for a column vector [10], [14], [15], [17], such as follows:

$$x = \begin{bmatrix} x^{(e)} \\ x^{(i)} \\ x^{(j)} \\ x^{(k)} \end{bmatrix}, \quad y = \begin{bmatrix} y^{(i)} \\ y^{(j)} \\ y^{(k)} \end{bmatrix}. \tag{53}$$

Similarly, we can represent the *split quaternion* product as a multiplication of matrices. In this form, the *split quaternion* product of $\boldsymbol{i}, \boldsymbol{j}$ and $\boldsymbol{k}$ with x, respectively, is

$$\begin{aligned} i \otimes \begin{bmatrix} x^{(e)} \\ x^{(i)} \\ x^{(j)} \\ x^{(k)} \end{bmatrix} &= I_L \begin{bmatrix} x^{(e)} \\ x^{(i)} \\ x^{(j)} \\ x^{(k)} \end{bmatrix} = \begin{bmatrix} -x^{(i)} \\ x^{(e)} \\ -x^{(k)} \\ x^{(j)} \end{bmatrix}, \\ \textit{where} \quad I_L &= \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 \end{bmatrix}; \end{aligned} \tag{54}$$

$$\begin{aligned} j \otimes \begin{bmatrix} x^{(e)} \\ x^{(i)} \\ x^{(j)} \\ x^{(k)} \end{bmatrix} &= J_L \begin{bmatrix} x^{(e)} \\ x^{(i)} \\ x^{(j)} \\ x^{(k)} \end{bmatrix} = \begin{bmatrix} x^{(j)} \\ -x^{(k)} \\ x^{(e)} \\ -x^{(i)} \end{bmatrix}, \\ \textit{where} \quad J_L &= \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \\ 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \end{bmatrix}; \end{aligned} \tag{55}$$

$$\begin{aligned} k \otimes \begin{bmatrix} x^{(e)} \\ x^{(i)} \\ x^{(j)} \\ x^{(k)} \end{bmatrix} &= K_L \begin{bmatrix} x^{(e)} \\ x^{(i)} \\ x^{(j)} \\ x^{(k)} \end{bmatrix} = \begin{bmatrix} x^{(k)} \\ x^{(j)} \\ x^{(i)} \\ x^{(e)} \end{bmatrix}, \\ \textit{where} \quad K_L &= \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}. \end{aligned} \tag{56}$$

The *split quaternion* product of x with $\boldsymbol{i}$, $\boldsymbol{j}$ and $\boldsymbol{k}$, respectively, is

$$\begin{aligned} \begin{bmatrix} x^{(e)} \\ x^{(i)} \\ x^{(j)} \\ x^{(k)} \end{bmatrix} \otimes i &= I_R \begin{bmatrix} x^{(e)} \\ x^{(i)} \\ x^{(j)} \\ x^{(k)} \end{bmatrix} = \begin{bmatrix} -x^{(i)} \\ x^{(e)} \\ x^{(k)} \\ -x^{(j)} \end{bmatrix}, \\ \textit{where} \quad I_R &= \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \end{bmatrix}; \end{aligned} \tag{57}$$

$$\begin{bmatrix} x^{(e)} \\ x^{(i)} \\ x^{(j)} \\ x^{(k)} \end{bmatrix} \otimes j = J_R \begin{bmatrix} x^{(e)} \\ x^{(i)} \\ x^{(j)} \\ x^{(k)} \end{bmatrix} = \begin{bmatrix} x^{(j)} \\ x^{(k)} \\ x^{(e)} \\ x^{(i)} \end{bmatrix},$$

$$\text{where} \quad J_R = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}; \tag{58}$$

$$\begin{bmatrix} x^{(e)} \\ x^{(i)} \\ x^{(j)} \\ x^{(k)} \end{bmatrix} \otimes k = K_R \begin{bmatrix} x^{(e)} \\ x^{(i)} \\ x^{(j)} \\ x^{(k)} \end{bmatrix} = \begin{bmatrix} x^{(k)} \\ -x^{(j)} \\ -x^{(i)} \\ x^{(e)} \end{bmatrix},$$

$$\text{where} \quad K_R = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \\ 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}. \tag{59}$$

In general, we can also represent the *split quaternion* product of a *split quaternion* x as a matrix. Let $Q_L(x)$ and $Q_R(x)$ be matrices that represent the product of x from the left-hand and from the right-hand sides, respectively. These are expressed as follows:

$$\begin{aligned} Q_L(x) &= x^{(e)}U + x^{(i)}I_L + x^{(j)}J_L + x^{(k)}K_L \\ &= \begin{bmatrix} x^{(e)} & -x^{(i)} & x^{(j)} & x^{(k)} \\ x^{(i)} & x^{(e)} & x^{(k)} & -x^{(j)} \\ x^{(j)} & x^{(k)} & x^{(e)} & -x^{(i)} \\ x^{(k)} & -x^{(j)} & x^{(i)} & x^{(e)} \end{bmatrix}, \end{aligned} \tag{60}$$

$$\begin{aligned} Q_R(x) &= x^{(e)}U + x^{(i)}I_R + x^{(j)}J_R + x^{(k)}K_R \\ &= \begin{bmatrix} x^{(e)} & -x^{(i)} & x^{(j)} & x^{(k)} \\ x^{(i)} & x^{(e)} & -x^{(k)} & x^{(j)} \\ x^{(j)} & -x^{(k)} & x^{(e)} & x^{(i)} \\ x^{(k)} & x^{(j)} & -x^{(i)} & x^{(e)} \end{bmatrix}, \end{aligned} \tag{61}$$

where U is the identity matrix.

Products of the *split quaternions* x and y satisfy the following properties:

$$\begin{aligned} Q_L(x)Q_L(y) &= Q_L(xy), \\ Q_R(y)Q_R(x) &= Q_R(xy), \\ Q_R(y)Q_L(x) &= Q_L(x)Q_R(y). \end{aligned} \tag{62}$$

Any rotation is defined by

$$y = a \otimes x \otimes a^* \tag{63}$$

This operation can also be represented as a matrix. Applying (60) and (61), the operation can be rewritten as follows:

$$\begin{bmatrix} 0 \\ y^{(i)} \\ y^{(j)} \\ y^{(k)} \end{bmatrix} = Q_R(a^*)Q_L(a) \begin{bmatrix} 0 \\ x^{(i)} \\ x^{(j)} \\ x^{(k)} \end{bmatrix} \tag{64}$$

where

$$Q_R(a^*)Q_L(a) = \begin{bmatrix} x^{(e)^2} + x^{(i)^2} - x^{(j)^2} - x^{(k)^2} & 0 & 2\left(x^{(e)}x^{(k)} + x^{(i)}x^{(j)}\right) & 2\left(x^{(i)}x^{(k)} - x^{(e)}x^{(j)}\right) \\ 0 & x^{(e)^2} + x^{(i)^2} + x^{(j)^2} + x^{(k)^2} & 2\left(x^{(e)}x^{(k)} - x^{(i)}x^{(j)}\right) & 2\left(x^{(e)}x^{(i)} + x^{(j)}x^{(k)}\right) \\ 0 & 2\left(x^{(e)}x^{(k)} + x^{(i)}x^{(j)}\right) & x^{(e)^2} - x^{(i)^2} - x^{(j)^2} + x^{(k)^2} & 2\left(x^{(e)}x^{(i)} - x^{(j)}x^{(k)}\right) \\ 0 & 2\left(x^{(e)}x^{(k)} - x^{(i)}x^{(j)}\right) & -2\left(x^{(e)}x^{(j)} + x^{(i)}x^{(k)}\right) & x^{(e)^2} - x^{(i)^2} + x^{(j)^2} - x^{(k)^2} \end{bmatrix}. \tag{65}$$

Therefore, the matrix representation of the *split quaternions* is used to improve the readability of the equations in the remainder of this paper.

APPENDIX B
DERIVATIVES OF SQNN FOR EKF ALGORITHM

Recalling the *split quaternion* neural model, the sum $s_{k,i+1}$ of the neuron k, in the i^{th} layer, is expressed as

$$s_{k,i+1} = \sum_{j=1}^{N_i} \frac{w_{j,k,i} \otimes L_{j,i} \otimes w^*_{j,k,i}}{\|w_{j,k,i}\|} + b_{k,i}. \tag{66}$$

So, by applying (65), $s_{k,i+1}$ can be represented in matrix form as:

$$\begin{aligned} s_{k,i+1} = \sum_{j=1}^{N_i} \frac{1}{\|w_{j,k,i}\|} & \begin{bmatrix} w^{(e)^2}_{j,k,i} + w^{(i)^2}_{j,k,i} + w^{(j)^2}_{j,k,i} + w^{(k)^2}_{j,k,i} & 0 & 2\left(w^{(e)}_{j,k,i}w^{(k)}_{j,k,i} + w^{(i)}_{j,k,i}w^{(j)}_{j,k,i}\right) & 2\left(w^{(i)}_{j,k,i}w^{(k)}_{j,k,i} - w^{(e)}_{j,k,i}w^{(j)}_{j,k,i}\right) \\ 0 & w^{(e)^2}_{j,k,i} + w^{(i)^2}_{j,k,i} + w^{(j)^2}_{j,k,i} + w^{(k)^2}_{j,k,i} & 2\left(w^{(e)}_{j,k,i}w^{(k)}_{j,k,i} - w^{(i)}_{j,k,i}w^{(j)}_{j,k,i}\right) & 2\left(w^{(e)}_{j,k,i}w^{(i)}_{j,k,i} + w^{(j)}_{j,k,i}w^{(k)}_{j,k,i}\right) \\ 0 & 2\left(w^{(e)}_{j,k,i}w^{(k)}_{j,k,i} + w^{(i)}_{j,k,i}w^{(j)}_{j,k,i}\right) & w^{(e)^2}_{j,k,i} - w^{(i)^2}_{j,k,i} - w^{(j)^2}_{j,k,i} + w^{(k)^2}_{j,k,i} & 2\left(w^{(e)}_{j,k,i}w^{(i)}_{j,k,i} - w^{(j)}_{j,k,i}w^{(k)}_{j,k,i}\right) \\ 0 & 2\left(w^{(e)}_{j,k,i}w^{(k)}_{j,k,i} - w^{(i)}_{j,k,i}w^{(j)}_{j,k,i}\right) & -2\left(w^{(e)}_{j,k,i}w^{(j)}_{j,k,i} + w^{(i)}_{j,k,i}w^{(k)}_{j,k,i}\right) & w^{(e)^2}_{j,k,i} - w^{(i)^2}_{j,k,i} + w^{(j)^2}_{j,k,i} - w^{(k)^2}_{j,k,i} \end{bmatrix} \begin{bmatrix} L^{(i)}_{j,i} \\ L^{(j)}_{j,i} \\ L^{(k)}_{j,i} \end{bmatrix} \\ & + \begin{bmatrix} b^{(i)}_{k,i} \\ b^{(j)}_{k,i} \\ b^{(k)}_{k,i} \end{bmatrix}. \end{aligned} \tag{67}$$

Updates of the network parameters are determined by the EKF learning algorithm, where the gradient of y_l is necessary with respect to the network parameters $\{w_{j,k,i}\}$ and $\{b_{k,i}\}$ for matrix H in (36). Then, to calculate the gradient of the *split*

quaternion neural network (SQNN), we can base our calculus on the gradient for the output y_l of a Real-valued Neural Network (RNN). Thus, for an RNN, we have

$$\frac{\partial y_l}{\partial b_{l,2}} = \frac{\partial \Phi_{l,3}}{\partial s_{l,3}} \frac{\partial s_{l,3}}{\partial b_{l,2}}, \tag{68}$$

$$\frac{\partial y_l}{\partial w_{k,l,2}} = \frac{\partial \Phi_{l,3}}{\partial s_{l,3}} \frac{\partial s_{l,3}}{\partial w_{k,l,2}}, \tag{69}$$

$$\frac{\partial y_l}{\partial b_{k,1}} = \frac{\partial \Phi_{l,3}}{\partial s_{l,3}} \frac{\partial s_{l,3}}{\partial \Phi_{k,2}} \frac{\partial \Phi_{k,2}}{\partial s_{k,2}} \frac{\partial s_{k,2}}{\partial b_{k,1}}, \tag{70}$$

$$\frac{\partial y_l}{\partial w_{j,k,1}} = \frac{\partial \Phi_{l,3}}{\partial s_{l,3}} \frac{\partial s_{l,3}}{\partial \Phi_{k,2}} \frac{\partial \Phi_{k,2}}{\partial s_{k,2}} \frac{\partial s_{k,2}}{\partial w_{j,k,1}}. \tag{71}$$

Hence, for SQNN, since $\partial^{y_l}_{b_{l,2}}, \partial^{y_l}_{w_{k,l,2}}, \partial^{y_l}_{b_{k,1}}$ and $\partial^{y_l}_{w_{j,k,1}} \in \mathbb{H}$, the derivative is calculated as follows:

$$\frac{\partial y_l}{\partial b_{l,2}} = \partial^{y_l}_{b_{l,2}} = \begin{bmatrix} \frac{\partial y_l}{\partial b^{(i)}_{l,2}} \\ \frac{\partial y_l}{\partial b^{(j)}_{l,2}} \\ \frac{\partial y_l}{\partial b^{(k)}_{l,2}} \end{bmatrix} = \begin{bmatrix} \frac{\partial \Phi^{(i)}_{l,3}}{\partial s^{(i)}_{l,3}} \frac{\partial s^{(i)}_{l,3}}{\partial b^{(i)}_{l,2}} \\ \frac{\partial \Phi^{(j)}_{l,3}}{\partial s^{(j)}_{l,3}} \frac{\partial s^{(j)}_{l,3}}{\partial b^{(j)}_{l,2}} \\ \frac{\partial \Phi^{(k)}_{l,3}}{\partial s^{(k)}_{l,3}} \frac{\partial s^{(k)}_{l,3}}{\partial b^{(k)}_{l,2}} \end{bmatrix}$$

$$= \begin{bmatrix} \Phi'^{(i)}_{l,3}(s^{(i)}_{l,3}) \cdot 1 \\ \Phi'^{(j)}_{l,3}(s^{(j)}_{l,3}) \cdot 1 \\ \Phi'^{(k)}_{l,3}(s^{(k)}_{l,3}) \cdot 1 \end{bmatrix} = \Phi'_{l,3}(s_{l,3}). \tag{72}$$

where

$$\Phi'(s) = \frac{\partial \varphi(s^{(i)})}{\partial s^{(i)}} i + \frac{\partial \varphi(s^{(j)})}{\partial s^{(j)}} j + \frac{\partial \varphi(s^{(k)})}{\partial s^{(k)}} k. \tag{73}$$

The derivative of the output y_l with respect to the synaptic weight between the k^{th} neuron, in the hidden layer, and the l^{th} neuron in the output layer, $\frac{\partial y_l}{\partial w_{k,l,2}}$, is determined as

$$\frac{\partial y_l}{\partial w_{k,l,2}} = \begin{bmatrix} \frac{\partial y_l}{\partial w^{(e)}_{k,l,2}} \\ \frac{\partial y_l}{\partial w^{(i)}_{k,l,2}} \\ \frac{\partial y_l}{\partial w^{(j)}_{k,l,2}} \\ \frac{\partial y_l}{\partial w^{(k)}_{k,l,2}} \end{bmatrix}$$

$$= \begin{bmatrix} \frac{\partial \Phi^{(i)}_{l,3}}{\partial s^{(i)}_{l,3}} \frac{\partial s^{(i)}_{l,3}}{\partial w^{(e)}_{k,l,2}} + \frac{\partial \Phi^{(j)}_{l,3}}{\partial s^{(j)}_{l,3}} \frac{\partial s^{(j)}_{l,3}}{\partial w^{(e)}_{k,l,2}} + \frac{\partial \Phi^{(k)}_{l,3}}{\partial s^{(k)}_{l,3}} \frac{\partial s^{(k)}_{l,3}}{\partial w^{(e)}_{k,l,2}} \\ \frac{\partial \Phi^{(i)}_{l,3}}{\partial s^{(i)}_{l,3}} \frac{\partial s^{(i)}_{l,3}}{\partial w^{(i)}_{k,l,2}} + \frac{\partial \Phi^{(j)}_{l,3}}{\partial s^{(j)}_{l,3}} \frac{\partial s^{(j)}_{l,3}}{\partial w^{(i)}_{k,l,2}} + \frac{\partial \Phi^{(k)}_{l,3}}{\partial s^{(k)}_{l,3}} \frac{\partial s^{(k)}_{l,3}}{\partial w^{(i)}_{k,l,2}} \\ \frac{\partial \Phi^{(i)}_{l,3}}{\partial s^{(i)}_{l,3}} \frac{\partial s^{(i)}_{l,3}}{\partial w^{(j)}_{k,l,2}} + \frac{\partial \Phi^{(j)}_{l,3}}{\partial s^{(j)}_{l,3}} \frac{\partial s^{(j)}_{l,3}}{\partial w^{(j)}_{k,l,2}} + \frac{\partial \Phi^{(k)}_{l,3}}{\partial s^{(k)}_{l,3}} \frac{\partial s^{(k)}_{l,3}}{\partial w^{(j)}_{k,l,2}} \\ \frac{\partial \Phi^{(i)}_{l,3}}{\partial s^{(i)}_{l,3}} \frac{\partial s^{(i)}_{l,3}}{\partial w^{(k)}_{k,l,2}} + \frac{\partial \Phi^{(j)}_{l,3}}{\partial s^{(j)}_{l,3}} \frac{\partial s^{(j)}_{l,3}}{\partial w^{(k)}_{k,l,2}} + \frac{\partial \Phi^{(k)}_{l,3}}{\partial s^{(k)}_{l,3}} \frac{\partial s^{(k)}_{l,3}}{\partial w^{(k)}_{k,l,2}} \end{bmatrix}$$

$$= \partial^{y_l(i)}_{b_{l,2}} \begin{bmatrix} \frac{\partial s^{(i)}_{l,3}}{\partial w^{(e)}_{k,l,2}} \\ \frac{\partial s^{(i)}_{l,3}}{\partial w^{(i)}_{k,l,2}} \\ \frac{\partial s^{(i)}_{l,3}}{\partial w^{(j)}_{k,l,2}} \\ \frac{\partial s^{(i)}_{l,3}}{\partial w^{(k)}_{k,l,2}} \end{bmatrix} + \partial^{y_l(j)}_{b_{l,2}} \begin{bmatrix} \frac{\partial s^{(j)}_{l,3}}{\partial w^{(e)}_{k,l,2}} \\ \frac{\partial s^{(j)}_{l,3}}{\partial w^{(i)}_{k,l,2}} \\ \frac{\partial s^{(j)}_{l,3}}{\partial w^{(j)}_{k,l,2}} \\ \frac{\partial s^{(j)}_{l,3}}{\partial w^{(k)}_{k,l,2}} \end{bmatrix}$$

$$+ \partial^{y_l(k)}_{b_{l,2}} \begin{bmatrix} \frac{\partial s^{(k)}_{l,3}}{\partial w^{(e)}_{k,l,2}} \\ \frac{\partial s^{(k)}_{l,3}}{\partial w^{(i)}_{k,l,2}} \\ \frac{\partial s^{(k)}_{l,3}}{\partial w^{(j)}_{k,l,2}} \\ \frac{\partial s^{(k)}_{l,3}}{\partial w^{(k)}_{k,l,2}} \end{bmatrix}$$

$$= \partial^{y_l(i)}_{b_{l,2}} T_1 + \partial^{y_l(j)}_{b_{l,2}} T_2 + \partial^{y_l(k)}_{b_{l,2}} T_3. \tag{74}$$

Next, we develop each term of (74) separately. The first term, $\partial^{y_l(i)}_{b_{l,2}} T_1$, is then calculated by applying the quotient rule as follows:

$$\partial^{y_l(i)}_{b_{l,2}} T_1 = \partial^{y_l(i)}_{b_{l,2}} \left(\frac{1}{\|w_{k,l,2}\|} \begin{bmatrix} \frac{\partial \left(w_{k,l,2} \otimes L_{k,2} \otimes w^*_{k,l,2} \right)^{(i)}}{\partial w^{(e)}_{k,l,2}} \\ \frac{\partial \left(w_{k,l,2} \otimes L_{k,2} \otimes w^*_{k,l,2} \right)^{(i)}}{\partial w^{(i)}_{k,l,2}} \\ \frac{\partial \left(w_{k,l,2} \otimes L_{k,2} \otimes w^*_{k,l,2} \right)^{(i)}}{\partial w^{(j)}_{k,l,2}} \\ \frac{\partial \left(w_{k,l,2} \otimes L_{k,2} \otimes w^*_{k,l,2} \right)^{(i)}}{\partial w^{(k)}_{k,l,2}} \end{bmatrix} \right.$$

$$\left. - \frac{\left(w_{k,l,2} \otimes L_{k,2} \otimes w^*_{k,l,2} \right)^{(i)}}{\|w_{k,l,2}\|^2} \begin{bmatrix} \frac{\partial \left(\|w_{k,l,2}\| \right)}{\partial w^{(e)}_{k,l,2}} \\ \frac{\partial \left(\|w_{k,l,2}\| \right)}{\partial w^{(i)}_{k,l,2}} \\ \frac{\partial \left(\|w_{k,l,2}\| \right)}{\partial w^{(j)}_{k,l,2}} \\ \frac{\partial \left(\|w_{k,l,2}\| \right)}{\partial w^{(k)}_{k,l,2}} \end{bmatrix} \right)$$

$$= \partial^{y_l(i)}_{b_{l,2}} (A_1 - A_2) \tag{75}$$

So, for the calculus of the terms A_1 and A_2, we have

$$A_1 = \frac{1}{\|w_{k,l,2}\|} \begin{bmatrix} 2w^{(e)}_{k,l,2} L^{(i)}_{k,2} + 2w^{(k)}_{k,l,2} L^{(j)}_{k,2} - 2w^{(j)}_{k,l,2} L^{(k)}_{k,2} \\ 2w^{(i)}_{k,l,2} L^{(i)}_{k,2} - 2w^{(j)}_{k,l,2} L^{(j)}_{k,2} - 2w^{(k)}_{k,l,2} L^{(k)}_{k,2} \\ 2w^{(j)}_{k,l,2} L^{(i)}_{k,2} - 2w^{(i)}_{k,l,2} L^{(j)}_{k,2} - 2w^{(e)}_{k,l,2} L^{(k)}_{k,2} \\ 2w^{(k)}_{k,l,2} L^{(i)}_{k,2} + 2w^{(e)}_{k,l,2} L^{(j)}_{k,2} - 2w^{(i)}_{k,l,2} L^{(k)}_{k,2} \end{bmatrix}$$

$$= \frac{2}{\|w_{k,l,2}\|}\left(L_{k,2}^{(i)} \begin{bmatrix} w_{k,l,2}^{(e)} \\ w_{k,l,2}^{(i)} \\ w_{k,l,2}^{(j)} \\ w_{k,l,2}^{(k)} \end{bmatrix} + L_{k,2}^{(j)} \begin{bmatrix} w_{k,l,2}^{(k)} \\ -w_{k,l,2}^{(j)} \\ -w_{k,l,2}^{(i)} \\ w_{k,l,2}^{(e)} \end{bmatrix} + L_{k,2}^{(k)} \begin{bmatrix} -w_{k,l,2}^{(j)} \\ -w_{k,l,2}^{(k)} \\ -w_{k,l,2}^{(e)} \\ -w_{k,l,2}^{(i)} \end{bmatrix} \right)$$

$$= \frac{2}{\|w_{k,l,2}\|}\left(L_{k,2}^{(i)} \begin{bmatrix} w_{k,l,2}^{(e)} \\ w_{k,l,2}^{(i)} \\ w_{k,l,2}^{(j)} \\ w_{k,l,2}^{(k)} \end{bmatrix} \otimes i \otimes (-i) + L_{k,2}^{(j)} \begin{bmatrix} w_{k,l,2}^{(e)} \\ w_{k,l,2}^{(i)} \\ w_{k,l,2}^{(j)} \\ w_{k,l,2}^{(k)} \end{bmatrix} \otimes j \otimes (-i) + L_{k,2}^{(k)} \begin{bmatrix} w_{k,l,2}^{(e)} \\ w_{k,l,2}^{(i)} \\ w_{k,l,2}^{(j)} \\ w_{k,l,2}^{(k)} \end{bmatrix} \otimes k \otimes (-i) \right)$$

$$= \frac{2}{\|w_{k,l,2}\|} \begin{bmatrix} w_{k,l,2}^{(e)} \\ w_{k,l,2}^{(i)} \\ w_{k,l,2}^{(j)} \\ w_{k,l,2}^{(k)} \end{bmatrix} \otimes L_{k,2} \otimes (-i)$$

$$= \frac{2}{\|w_{k,l,2}\|} w_{k,l,2} \otimes L_{k,2} \otimes (-i)$$

$$= \frac{2}{\|w_{k,l,2}\|} w_{k,l,2} \otimes L_{k,2}^{*} \otimes i; \tag{76}$$

$$A_2 = \frac{\left(w_{k,l,2} \otimes L_{k,2} \otimes w_{k,l,2}^{*}\right)^{(i)}}{\|w_{k,l,2}\|^2} \begin{bmatrix} \frac{\partial\left(\|w_{k,l,2}\|\right)}{\partial w_{k,l,2}^{(e)}} \\ \frac{\partial\left(\|w_{k,l,2}\|\right)}{\partial w_{k,l,2}^{(i)}} \\ \frac{\partial\left(\|w_{k,l,2}\|\right)}{\partial w_{k,l,2}^{(j)}} \\ \frac{\partial\left(\|w_{k,l,2}\|\right)}{\partial w_{k,l,2}^{(k)}} \end{bmatrix}$$

$$= \frac{\left(w_{k,l,2} \otimes L_{k,2} \otimes w_{k,l,2}^{*}\right)^{(i)}}{\left|w_{k,l,2}^{(e)2} + w_{k,l,2}^{(i)2} - w_{k,l,2}^{(j)2} - w_{k,l,2}^{(k)2}\right|^{5/2}} \begin{bmatrix} w_{k,l,2}^{(e)}\left(w_{k,l,2}^{(e)2} + w_{k,l,2}^{(i)2} - w_{k,l,2}^{(j)2} - w_{k,l,2}^{(k)2}\right) \\ w_{k,l,2}^{(i)}\left(w_{k,l,2}^{(e)2} + w_{k,l,2}^{(i)2} - w_{k,l,2}^{(j)2} - w_{k,l,2}^{(k)2}\right) \\ -w_{k,l,2}^{(j)}\left(w_{k,l,2}^{(e)2} + w_{k,l,2}^{(i)2} - w_{k,l,2}^{(j)2} - w_{k,l,2}^{(k)2}\right) \\ -w_{k,l,2}^{(k)}\left(w_{k,l,2}^{(e)2} + w_{k,l,2}^{(i)2} - w_{k,l,2}^{(j)2} - w_{k,l,2}^{(k)2}\right) \end{bmatrix}$$

$$= \frac{\left(w_{k,l,2} \otimes L_{k,2} \otimes w_{k,l,2}^{*}\right)^{(i)}}{\|w_{k,l,2}\|^5} \otimes \lambda_1 q_1, \tag{77}$$

where

$$\lambda_1 = w_{j,k,i}^{(e)2} + w_{j,k,i}^{(i)2} - w_{j,k,i}^{(j)2} - w_{j,k,i}^{(k)2} \mid \lambda_1 \in \mathbb{R},$$
$$q_1 = w_{k,l,2}^{(e)} + w_{k,l,2}^{(i)} i - w_{k,l,2}^{(j)} j - w_{k,l,2}^{(k)} k \mid q_1 \in \mathbb{H}_s.$$

Consequently, we obtain the first term of (74) as

$$\begin{aligned} \partial_{b_{l,2}}^{y_l(i)} T_1 &= \frac{\partial_{b_{l,2}}^{y_l(i)}}{\|w_{k,l,2}\|}\left(2\, w_{k,l,2} \otimes L_{k,2}^{*} \otimes i - \frac{\left(w_{k,l,2} \otimes L_{k,2} \otimes w_{k,l,2}^{*}\right)^{(i)}}{\|w_{k,l,2}\|^4} \otimes \lambda_1 q_1 \right) \\ &= \frac{1}{\|w_{k,l,2}\|}\left(2\, w_{k,l,2} \otimes L_{k,2}^{*} \otimes \partial_{b_{l,2}}^{y_l(i)} i - \partial_{b_{l,2}}^{y_l(i)} \frac{\left(w_{k,l,2} \otimes L_{k,2} \otimes w_{k,l,2}^{*}\right)^{(i)}}{\|w_{k,l,2}\|^4} \otimes \lambda_1 q_1 \right). \end{aligned} \tag{78}$$

Therefore, a similar technique can be applied to develop the second and third terms of (74), resulting for $\partial_{b_{l,2}}^{y_l(j)} T_2$ in

$$\begin{aligned} \partial_{b_{l,2}}^{y_l(j)} T_2 = \partial_{b_{l,2}}^{y_l(j)} \Bigg(& \frac{1}{\|w_{k,l,2}\|} \begin{bmatrix} \frac{\partial\left(w_{k,l,2} \otimes L_{k,2} \otimes w_{k,l,2}^{*}\right)^{(j)}}{\partial w_{k,l,2}^{(e)}} \\ \frac{\partial\left(w_{k,l,2} \otimes L_{k,2} \otimes w_{k,l,2}^{*}\right)^{(j)}}{\partial w_{k,l,2}^{(i)}} \\ \frac{\partial\left(w_{k,l,2} \otimes L_{k,2} \otimes w_{k,l,2}^{*}\right)^{(j)}}{\partial w_{k,l,2}^{(j)}} \\ \frac{\partial\left(w_{k,l,2} \otimes L_{k,2} \otimes w_{k,l,2}^{*}\right)^{(j)}}{\partial w_{k,l,2}^{(k)}} \end{bmatrix} \\ & - \frac{\left(w_{k,l,2} \otimes L_{k,2} \otimes w_{k,l,2}^{*}\right)^{(j)}}{\|w_{k,l,2}\|^2} \begin{bmatrix} \frac{\partial\left(\|w_{k,l,2}\|\right)}{\partial w_{k,l,2}^{(e)}} \\ \frac{\partial\left(\|w_{k,l,2}\|\right)}{\partial w_{k,l,2}^{(i)}} \\ \frac{\partial\left(\|w_{k,l,2}\|\right)}{\partial w_{k,l,2}^{(j)}} \\ \frac{\partial\left(\|w_{k,l,2}\|\right)}{\partial w_{k,l,2}^{(k)}} \end{bmatrix} \Bigg) \\ = \partial_{b_{l,2}}^{y_l(j)} (B_1 - B_2). & \end{aligned} \tag{79}$$

Then, for the calculus of terms B_1 and B_2, we have the following:

$$
\begin{aligned}
B_1 = & \frac{1}{\left\|w_{k,l,2}\right\|} \\
& \begin{bmatrix} 2w^{(k)}_{k,l,2}L^{(i)}_{k,2} + 2w^{(e)}_{k,l,2}L^{(j)}_{k,2} - 2w^{(i)}_{k,l,2}L^{(k)}_{k,2} \\ 2w^{(j)}_{k,l,2}L^{(i)}_{k,2} - 2w^{(i)}_{k,l,2}L^{(j)}_{k,2} - 2w^{(e)}_{k,l,2}L^{(k)}_{k,2} \\ 2w^{(i)}_{k,l,2}L^{(i)}_{k,2} - 2w^{(j)}_{k,l,2}L^{(j)}_{k,2} - 2w^{(k)}_{k,l,2}L^{(k)}_{k,2} \\ 2w^{(e)}_{k,l,2}L^{(i)}_{k,2} + 2w^{(k)}_{k,l,2}L^{(j)}_{k,2} - 2w^{(j)}_{k,l,2}L^{(k)}_{k,2} \end{bmatrix} \\
= & \frac{2}{\left\|w_{k,l,2}\right\|} \left(L^{(i)}_{k,2} \begin{bmatrix} w^{(k)}_{k,l,2} \\ w^{(j)}_{k,l,2} \\ w^{(i)}_{k,l,2} \\ w^{(e)}_{k,l,2} \end{bmatrix} + L^{(j)}_{k,2} \begin{bmatrix} w^{(e)}_{k,l,2} \\ -w^{(i)}_{k,l,2} \\ -w^{(j)}_{k,l,2} \\ w^{(k)}_{k,l,2} \end{bmatrix} \right. \\
& \left. + L^{(k)}_{k,2} \begin{bmatrix} -w^{(i)}_{k,l,2} \\ -w^{(e)}_{k,l,2} \\ -w^{(k)}_{k,l,2} \\ -w^{(j)}_{k,l,2} \end{bmatrix} \right) \\
= & \frac{2}{\left\|w_{k,l,2}\right\|} \left((-j) \otimes i \otimes L^{(i)}_{k,2} \begin{bmatrix} w^{(e)}_{k,l,2} \\ w^{(i)}_{k,l,2} \\ w^{(j)}_{k,l,2} \\ w^{(k)}_{k,l,2} \end{bmatrix} \right. \\
& + i \otimes j \otimes L^{(j)}_{k,2} \begin{bmatrix} w^{(e)}_{k,l,2} \\ w^{(i)}_{k,l,2} \\ w^{(j)}_{k,l,2} \\ w^{(k)}_{k,l,2} \end{bmatrix} \otimes (-j) \otimes i \\
& \left. + k \otimes L^{(k)}_{k,2} \begin{bmatrix} w^{(e)}_{k,l,2} \\ w^{(i)}_{k,l,2} \\ w^{(j)}_{k,l,2} \\ w^{(k)}_{k,l,2} \end{bmatrix} \otimes (-j) \right) \\
= & \frac{2}{\left\|w_{k,l,2}\right\|} \left((-j) \otimes L^{(i)}_{k,2} i \otimes w_{k,l,2} \right. \\
& + i \otimes L^{(j)}_{k,2} j \otimes w_{k,l,2} \otimes (-j) \otimes i \\
& \left. + L^{(k)}_{k,2} k \otimes w_{k,l,2} \otimes (-j) \right) \\
= & \frac{2}{\left\|w_{k,l,2}\right\|} \left(j \otimes L^{*(i)}_{k,2} i \otimes w_{k,l,2} \right. \\
& + i \otimes L^{*(j)}_{k,2} j \otimes w_{k,l,2} \otimes j \otimes i \\
& \left. + L^{*(k)}_{k,2} k \otimes w_{k,l,2} \otimes j \right); \qquad (80)
\end{aligned}
$$

$$
\begin{aligned}
B_2 = & \frac{\left(w_{k,l,2} \otimes L_{k,2} \otimes w^*_{k,l,2}\right)^{(j)}}{\left\|w_{k,l,2}\right\|^2} \begin{bmatrix} \frac{\partial\left(\left\|w_{k,l,2}\right\|\right)}{\partial w^{(e)}_{k,l,2}} \\ \frac{\partial\left(\left\|w_{k,l,2}\right\|\right)}{\partial w^{(i)}_{k,l,2}} \\ \frac{\partial\left(\left\|w_{k,l,2}\right\|\right)}{\partial w^{(j)}_{k,l,2}} \\ \frac{\partial\left(\left\|w_{k,l,2}\right\|\right)}{\partial w^{(k)}_{k,l,2}} \end{bmatrix} \\
= & \frac{\left(w_{k,l,2} \otimes L_{k,2} \otimes w^*_{k,l,2}\right)^{(j)}}{\left|w^{(e)2}_{k,l,2} + w^{(i)2}_{k,l,2} - w^{(j)2}_{k,l,2} - w^{(k)2}_{k,l,2}\right|^{5/2}} \\
& \begin{bmatrix} w^{(e)}_{k,l,2}\left(w^{(e)2}_{k,l,2} + w^{(i)2}_{k,l,2} - w^{(j)2}_{k,l,2} - w^{(k)2}_{k,l,2}\right) \\ w^{(i)}_{k,l,2}\left(w^{(e)2}_{k,l,2} + w^{(i)2}_{k,l,2} - w^{(j)2}_{k,l,2} - w^{(k)2}_{k,l,2}\right) \\ -w^{(j)}_{k,l,2}\left(w^{(e)2}_{k,l,2} + w^{(i)2}_{k,l,2} - w^{(j)2}_{k,l,2} - w^{(k)2}_{k,l,2}\right) \\ -w^{(k)}_{k,l,2}\left(w^{(e)2}_{k,l,2} + w^{(i)2}_{k,l,2} - w^{(j)2}_{k,l,2} - w^{(k)2}_{k,l,2}\right) \end{bmatrix} \\
= & \frac{\left(w_{k,l,2} \otimes L_{k,2} \otimes w^*_{k,l,2}\right)^{(j)}}{\left\|w_{k,l,2}\right\|^5} \otimes \lambda_1 q_1, \qquad (81)
\end{aligned}
$$

Consequently, we obtain the second term of (74) as

$$
\begin{aligned}
\partial^{y_l(j)}_{b_{l,2}} T_2 = & \frac{\partial^{y_l(j)}_{b_{l,2}}}{\left\|w_{k,l,2}\right\|} \left(2j \otimes L^{*(i)}_{k,2} i \otimes w_{k,l,2} \right. \\
& + 2i \otimes L^{*(j)}_{k,2} j \otimes w_{k,l,2} \otimes j \otimes i \\
& + 2L^{*(k)}_{k,2} k \otimes w_{k,l,2} \otimes j \\
& \left. - \frac{\left(w_{k,l,2} \otimes L_{k,2} \otimes w^*_{k,l,2}\right)^{(j)}}{\left\|w_{k,l,2}\right\|^4} \otimes \lambda_1 q_1 \right) \\
= & \frac{1}{\left\|w_{k,l,2}\right\|} \left(2\partial^{y_l(j)}_{b_{l,2}} j \otimes L^{*(i)}_{k,2} i \otimes w_{k,l,2} \right. \\
& + 2i \otimes L^{*(j)}_{k,2} j \otimes w_{k,l,2} \otimes \partial^{y_l(j)}_{b_{l,2}} j \otimes i \\
& + 2L^{*(k)}_{k,2} k \otimes w_{k,l,2} \otimes \partial^{y_l(j)}_{b_{l,2}} j \\
& \left. - \partial^{y_l(j)}_{b_{l,2}} \frac{\left(w_{k,l,2} \otimes L_{k,2} \otimes w^*_{k,l,2}\right)^{(j)}}{\left\|w_{k,l,2}\right\|^4} \otimes \lambda_1 q_1 \right). \qquad (82)
\end{aligned}
$$

and resulting for $\partial^{y_l(k)}_{b_{l,2}} T_3$ as follows:

$$
\partial^{y_l(k)}_{b_{l,2}} T_3 = \partial^{y_l(k)}_{b_{l,2}} \left(\frac{1}{\left\|w_{k,l,2}\right\|} \begin{bmatrix} \frac{\partial\left(w_{k,l,2} \otimes L_{k,2} \otimes w^*_{k,l,2}\right)^{(k)}}{\partial w^{(e)}_{k,l,2}} \\ \frac{\partial\left(w_{k,l,2} \otimes L_{k,2} \otimes w^*_{k,l,2}\right)^{(k)}}{\partial w^{(i)}_{k,l,2}} \\ \frac{\partial\left(w_{k,l,2} \otimes L_{k,2} \otimes w^*_{k,l,2}\right)^{(k)}}{\partial w^{(j)}_{k,l,2}} \\ \frac{\partial\left(w_{k,l,2} \otimes L_{k,2} \otimes w^*_{k,l,2}\right)^{(k)}}{\partial w^{(k)}_{k,l,2}} \end{bmatrix} \right.
$$

$$-\frac{\left(w_{k,l,2}\otimes L_{k,2}\otimes w^{*}_{k,l,2}\right)^{(k)}}{\left\|w_{k,l,2}\right\|^{2}}\begin{bmatrix}\frac{\partial\left(\left\|w_{k,l,2}\right\|\right)}{\partial w^{(e)}_{k,l,2}}\\ \frac{\partial\left(\left\|w_{k,l,2}\right\|\right)}{\partial w^{(i)}_{k,l,2}}\\ \frac{\partial\left(\left\|w_{k,l,2}\right\|\right)}{\partial w^{(j)}_{k,l,2}}\\ \frac{\partial\left(\left\|w_{k,l,2}\right\|\right)}{\partial w^{(k)}_{k,l,2}}\end{bmatrix}\Bigg)$$

$$=\partial^{y_l(k)}_{b_{l,2}}(C_1-C_2). \tag{83}$$

So, for the calculus of the terms C_1 and C_2, we have

$$C_1=\frac{1}{\left\|w_{k,l,2}\right\|}\begin{bmatrix}-2w^{(j)}_{k,l,2}L^{(i)}_{k,2}+2w^{(i)}_{k,l,2}L^{(j)}_{k,2}+2w^{(e)}_{k,l,2}L^{(k)}_{k,2}\\ 2w^{(k)}_{k,l,2}L^{(i)}_{k,2}+2w^{(e)}_{k,l,2}L^{(j)}_{k,2}-2w^{(i)}_{k,l,2}L^{(k)}_{k,2}\\ -2w^{(e)}_{k,l,2}L^{(i)}_{k,2}-2w^{(k)}_{k,l,2}L^{(j)}_{k,2}+2w^{(j)}_{k,l,2}L^{(k)}_{k,2}\\ 2w^{(i)}_{k,l,2}L^{(i)}_{k,2}-2w^{(j)}_{k,l,2}L^{(j)}_{k,2}-2w^{(k)}_{k,l,2}L^{(k)}_{k,2}\end{bmatrix}$$

$$=\frac{2}{\left\|w_{k,l,2}\right\|}\Bigg(L^{(i)}_{k,2}\begin{bmatrix}-w^{(j)}_{k,l,2}\\ w^{(k)}_{k,l,2}\\ -w^{(e)}_{k,l,2}\\ w^{(i)}_{k,l,2}\end{bmatrix}+L^{(j)}_{k,2}\begin{bmatrix}w^{(i)}_{k,l,2}\\ w^{(e)}_{k,l,2}\\ -w^{(k)}_{k,l,2}\\ -w^{(j)}_{k,l,2}\end{bmatrix}+L^{(k)}_{k,2}\begin{bmatrix}w^{(e)}_{k,l,2}\\ -w^{(i)}_{k,l,2}\\ w^{(j)}_{k,l,2}\\ -w^{(k)}_{k,l,2}\end{bmatrix}\Bigg)$$

$$=\frac{2}{\left\|w_{k,l,2}\right\|}\Bigg((-k)\otimes i\otimes L^{(i)}_{k,2}\begin{bmatrix}w^{(e)}_{k,l,2}\\ w^{(i)}_{k,l,2}\\ w^{(j)}_{k,l,2}\\ w^{(k)}_{k,l,2}\end{bmatrix}+j\otimes L^{(j)}_{k,2}\begin{bmatrix}w^{(e)}_{k,l,2}\\ w^{(i)}_{k,l,2}\\ w^{(j)}_{k,l,2}\\ w^{(k)}_{k,l,2}\end{bmatrix}\otimes(-k)+i\otimes k\otimes L^{(k)}_{k,2}\begin{bmatrix}w^{(e)}_{k,l,2}\\ w^{(i)}_{k,l,2}\\ w^{(j)}_{k,l,2}\\ w^{(k)}_{k,l,2}\end{bmatrix}\otimes(-k)\otimes i\Bigg)$$

$$=\frac{2}{\left\|w_{k,l,2}\right\|}\Big((-k)\otimes L^{(i)}_{k,2}i\otimes w_{k,l,2}+L^{(j)}_{k,2}j\otimes w_{k,l,2}\otimes(-k)+i\otimes L^{(k)}_{k,2}k\otimes w_{k,l,2}\otimes(-k)\otimes i\Big)$$

$$=\frac{2}{\left\|w_{k,l,2}\right\|}\Big(k\otimes L^{*(i)}_{k,2}i\otimes w_{k,l,2}+L^{*(j)}_{k,2}j\otimes w_{k,l,2}\otimes k+i\otimes L^{*(k)}_{k,2}k\otimes w_{k,l,2}\otimes k\otimes i\Big); \tag{84}$$

$$C_2=\frac{\left(w_{k,l,2}\otimes L_{k,2}\otimes w^{*}_{k,l,2}\right)^{(k)}}{\left\|w_{k,l,2}\right\|^{2}}\begin{bmatrix}\frac{\partial\left(\left\|w_{k,l,2}\right\|\right)}{\partial w^{(e)}_{k,l,2}}\\ \frac{\partial\left(\left\|w_{k,l,2}\right\|\right)}{\partial w^{(i)}_{k,l,2}}\\ \frac{\partial\left(\left\|w_{k,l,2}\right\|\right)}{\partial w^{(j)}_{k,l,2}}\\ \frac{\partial\left(\left\|w_{k,l,2}\right\|\right)}{\partial w^{(k)}_{k,l,2}}\end{bmatrix}$$

$$=\frac{\left(w_{k,l,2}\otimes L_{k,2}\otimes w^{*}_{k,l,2}\right)^{(k)}}{\left|w^{(e)2}_{k,l,2}+w^{(i)2}_{k,l,2}-w^{(j)2}_{k,l,2}-w^{(k)2}_{k,l,2}\right|^{5/2}}\begin{bmatrix}w^{(e)}_{k,l,2}\left(w^{(e)2}_{k,l,2}+w^{(i)2}_{k,l,2}-w^{(j)2}_{k,l,2}-w^{(k)2}_{k,l,2}\right)\\ w^{(i)}_{k,l,2}\left(w^{(e)2}_{k,l,2}+w^{(i)2}_{k,l,2}-w^{(j)2}_{k,l,2}-w^{(k)2}_{k,l,2}\right)\\ -w^{(j)}_{k,l,2}\left(w^{(e)2}_{k,l,2}+w^{(i)2}_{k,l,2}-w^{(j)2}_{k,l,2}-w^{(k)2}_{k,l,2}\right)\\ -w^{(k)}_{k,l,2}\left(w^{(e)2}_{k,l,2}+w^{(i)2}_{k,l,2}-w^{(j)2}_{k,l,2}-w^{(k)2}_{k,l,2}\right)\end{bmatrix}$$

$$=\frac{\left(w_{k,l,2}\otimes L_{k,2}\otimes w^{*}_{k,l,2}\right)^{(k)}}{\left\|w_{k,l,2}\right\|^{5}}\otimes\lambda_1 q_1, \tag{85}$$

Consequently, we obtain the third term of (74) as

$$\partial^{y_l(k)}_{b_{l,2}}T_3=\frac{\partial^{y_l(k)}_{b_{l,2}}}{\left\|w_{k,l,2}\right\|}\Bigg(2k\otimes L^{*(i)}_{k,2}i\otimes w_{k,l,2}+2L^{*(j)}_{k,2}j\otimes w_{k,l,2}\otimes k+2i\otimes L^{*(k)}_{k,2}k\otimes w_{k,l,2}\otimes k\otimes i-\frac{\left(w_{k,l,2}\otimes L_{k,2}\otimes w^{*}_{k,l,2}\right)^{(k)}}{\left\|w_{k,l,2}\right\|^{4}}\otimes\lambda_1 q_1\Bigg)$$

$$=\frac{1}{\left\|w_{k,l,2}\right\|}\Bigg(2\partial^{y_l(k)}_{b_{l,2}}k\otimes L^{*(i)}_{k,2}i\otimes w_{k,l,2}+2L^{*(j)}_{k,2}j\otimes w_{k,l,2}\otimes\partial^{y_l(k)}_{b_{l,2}}k+2i\otimes L^{*(k)}_{k,2}k\otimes w_{k,l,2}\otimes\partial^{y_l(k)}_{b_{l,2}}k\otimes i-\partial^{y_l(k)}_{b_{l,2}}\frac{\left(w_{k,l,2}\otimes L_{k,2}\otimes w^{*}_{k,l,2}\right)^{(k)}}{\left\|w_{k,l,2}\right\|^{4}}\otimes\lambda_1 q_1\Bigg). \tag{86}$$

Hence, we obtain $\frac{\partial y_l}{\partial w_{k,l,2}}$ as

$$\frac{\partial y_l}{\partial w_{k,l,2}}=\frac{1}{\left\|w_{k,l,2}\right\|}\Bigg(2\,w_{k,l,2}\otimes L^{*}_{k,2}\otimes\partial^{y_l(i)}_{b_{l,2}}i-\partial^{y_l(i)}_{b_{l,2}}\frac{\left(w_{k,l,2}\otimes L_{k,2}\otimes w^{*}_{k,l,2}\right)^{(i)}}{\left\|w_{k,l,2}\right\|^{4}}\otimes\lambda_1 q_1\Bigg)$$

$$
\begin{aligned}
&+\frac{1}{\|w_{k,l,2}\|}\Bigg(2\partial_{b_{l,2}}^{y_l(j)} j\otimes L_{k,2}^{*(i)} i\otimes w_{k,l,2}\\
&+2i\otimes L_{k,2}^{*(j)} j\otimes w_{k,l,2}\otimes \partial_{b_{l,2}}^{y_l(j)} j\otimes i\\
&+2L_{k,2}^{*(k)} k\otimes w_{k,l,2}\otimes \partial_{b_{l,2}}^{y_l(j)} j\\
&-\partial_{b_{l,2}}^{y_l(j)}\frac{\left(w_{k,l,2}\otimes L_{k,2}\otimes w_{k,l,2}^*\right)^{(j)}}{\|w_{k,l,2}\|^4}\otimes\lambda_1 q_1\Bigg)\\
&+\frac{1}{\|w_{k,l,2}\|}\Bigg(2\partial_{b_{l,2}}^{y_l(k)} k\otimes L_{k,2}^{*(i)} i\otimes w_{k,l,2}\\
&+2L_{k,2}^{*(j)} j\otimes w_{k,l,2}\otimes \partial_{b_{l,2}}^{y_l(k)} k\\
&+2i\otimes L_{k,2}^{*(k)} k\otimes w_{k,l,2}\otimes \partial_{b_{l,2}}^{y_l(k)} k\otimes i\\
&-\partial_{b_{l,2}}^{y_l(k)}\frac{\left(w_{k,l,2}\otimes L_{k,2}\otimes w_{k,l,2}^*\right)^{(k)}}{\|w_{k,l,2}\|^4}\otimes\lambda_1 q_1\Bigg)\\
&=\frac{1}{\|w_{k,l,2}\|}\Bigg[2\Bigg(w_{k,l,2}\otimes L_{k,2}^*\otimes\partial_{b_{l,2}}^{y_l(i)} i\\
&+\partial_{b_{l,2}}^{y_l(j)} j\otimes L_{k,2}^{*(i)} i\otimes w_{k,l,2}\\
&+i\otimes L_{k,2}^{*(j)} j\otimes w_{k,l,2}\otimes\partial_{b_{l,2}}^{y_l(j)} j\otimes i\\
&+L_{k,2}^{*(k)} k\otimes w_{k,l,2}\otimes\partial_{b_{l,2}}^{y_l(j)} j\\
&+\partial_{b_{l,2}}^{y_l(k)} k\otimes L_{k,2}^{*(i)} i\otimes w_{k,l,2}\\
&+L_{k,2}^{*(j)} j\otimes w_{k,l,2}\otimes\partial_{b_{l,2}}^{y_l(k)} k\\
&+i\otimes L_{k,2}^{*(k)} k\otimes w_{k,l,2}\otimes\partial_{b_{l,2}}^{y_l(k)} k\otimes i\Bigg)\\
&-\partial_{b_{l,2}}^{y_l}\odot\frac{w_{k,l,2}\otimes L_{k,2}\otimes w_{k,l,2}^*}{\|w_{k,l,2}\|^4}\otimes\lambda_1 q_1\Bigg]. \qquad (87)
\end{aligned}
$$

where

$$\lambda_1 = w_{k,l,2}^{(e)2} + w_{k,l,2}^{(i)2} - w_{k,l,2}^{(j)2} - w_{k,l,2}^{(k)2} \mid \lambda_1\in\mathbb{R},$$

$$q_1 = w_{k,l,2}^{(e)} + w_{k,l,2}^{(i)} i - w_{k,l,2}^{(j)} j - w_{k,l,2}^{(k)} k \mid q_1\in\mathbb{H}_s.$$

The derivative of neuron y_l with respect to $b_{k,1}$ is calculated as follows:

$$
\begin{aligned}
\frac{\partial y_l}{\partial b_{k,1}} &= \begin{bmatrix}\frac{\partial y_l}{\partial b_{k,1}^{(i)}}\\ \frac{\partial y_l}{\partial b_{k,1}^{(j)}}\\ \frac{\partial y_l}{\partial b_{k,1}^{(k)}}\end{bmatrix} = \begin{bmatrix}\frac{\partial\Phi_{l,3}}{\partial s_{l,3}}\frac{\partial s_{l,3}}{\partial\Phi_{k,2}^{(i)}}\frac{\partial\Phi_{k,2}^{(i)}}{\partial s_{k,2}^{(i)}}\frac{\partial s_{k,2}^{(i)}}{\partial b_{k,1}^{(i)}}\\ \frac{\partial\Phi_{l,3}}{\partial s_{l,3}}\frac{\partial s_{l,3}}{\partial\Phi_{k,2}^{(j)}}\frac{\partial\Phi_{k,2}^{(j)}}{\partial s_{k,2}^{(j)}}\frac{\partial s_{k,2}^{(j)}}{\partial b_{k,1}^{(j)}}\\ \frac{\partial\Phi_{l,3}}{\partial s_{l,3}}\frac{\partial s_{l,3}}{\partial\Phi_{k,2}^{(k)}}\frac{\partial\Phi_{k,2}^{(k)}}{\partial s_{k,2}^{(k)}}\frac{\partial s_{k,2}^{(k)}}{\partial b_{k,1}^{(k)}}\end{bmatrix}\\
&= \begin{bmatrix}\frac{\partial\Phi_{l,3}}{\partial s_{l,3}}\frac{\partial s_{l,3}}{\partial\Phi_{k,2}^{(i)}}\cdot\Phi_{k,2}^{\prime(i)}\left(s_{k,2}^{(i)}\right)\cdot 1\\ \frac{\partial\Phi_{l,3}}{\partial s_{l,3}}\frac{\partial s_{l,3}}{\partial\Phi_{k,2}^{(j)}}\cdot\Phi_{k,2}^{\prime(j)}\left(s_{k,2}^{(j)}\right)\cdot 1\\ \frac{\partial\Phi_{l,3}}{\partial s_{l,3}}\frac{\partial s_{l,3}}{\partial\Phi_{k,2}^{(k)}}\cdot\Phi_{k,2}^{\prime(k)}\left(s_{k,2}^{(k)}\right)\cdot 1\end{bmatrix}\\
&= \begin{bmatrix}\frac{\partial\Phi_{l,3}}{\partial s_{l,3}}\frac{\partial s_{l,3}}{\partial\Phi_{k,2}^{(i)}}\\ \frac{\partial\Phi_{l,3}}{\partial s_{l,3}}\frac{\partial s_{l,3}}{\partial\Phi_{k,2}^{(j)}}\\ \frac{\partial\Phi_{l,3}}{\partial s_{l,3}}\frac{\partial s_{l,3}}{\partial\Phi_{k,2}^{(k)}}\end{bmatrix}\odot\Phi_{k,2}^{\prime}(s_{k,2})\\
&= W\odot\Phi_{k,2}^{\prime}(s_{k,2}). \qquad (88)
\end{aligned}
$$

where

$$
\begin{aligned}
W &= \begin{bmatrix}\frac{\partial\Phi_{l,3}}{\partial s_{l,3}}\frac{\partial s_{l,3}}{\partial\Phi_{k,2}^{(i)}}\\ \frac{\partial\Phi_{l,3}}{\partial s_{l,3}}\frac{\partial s_{l,3}}{\partial\Phi_{k,2}^{(j)}}\\ \frac{\partial\Phi_{l,3}}{\partial s_{l,3}}\frac{\partial s_{l,3}}{\partial\Phi_{k,2}^{(k)}}\end{bmatrix}\\
&= \begin{bmatrix}\sum_{l=1}^{N_3}\left(\frac{\partial\Phi_{l,3}^{(i)}}{\partial s_{l,3}^{(i)}}\frac{\partial s_{l,3}^{(i)}}{\partial\Phi_{k,2}^{(i)}}+\frac{\partial\Phi_{l,3}^{(j)}}{\partial s_{l,3}^{(j)}}\frac{\partial s_{l,3}^{(j)}}{\partial\Phi_{k,2}^{(i)}}+\frac{\partial\Phi_{l,3}^{(k)}}{\partial s_{l,3}^{(k)}}\frac{\partial s_{l,3}^{(k)}}{\partial\Phi_{k,2}^{(i)}}\right)\\ \sum_{l=1}^{N_3}\left(\frac{\partial\Phi_{l,3}^{(i)}}{\partial s_{l,3}^{(i)}}\frac{\partial s_{l,3}^{(i)}}{\partial\Phi_{k,2}^{(j)}}+\frac{\partial\Phi_{l,3}^{(j)}}{\partial s_{l,3}^{(j)}}\frac{\partial s_{l,3}^{(j)}}{\partial\Phi_{k,2}^{(j)}}+\frac{\partial\Phi_{l,3}^{(k)}}{\partial s_{l,3}^{(k)}}\frac{\partial s_{l,3}^{(k)}}{\partial\Phi_{k,2}^{(j)}}\right)\\ \sum_{l=1}^{N_3}\left(\frac{\partial\Phi_{l,3}^{(i)}}{\partial s_{l,3}^{(i)}}\frac{\partial s_{l,3}^{(i)}}{\partial\Phi_{k,2}^{(k)}}+\frac{\partial\Phi_{l,3}^{(j)}}{\partial s_{l,3}^{(j)}}\frac{\partial s_{l,3}^{(j)}}{\partial\Phi_{k,2}^{(k)}}+\frac{\partial\Phi_{l,3}^{(k)}}{\partial s_{l,3}^{(k)}}\frac{\partial s_{l,3}^{(k)}}{\partial\Phi_{k,2}^{(k)}}\right)\end{bmatrix}\\
&= \sum_{l=1}^{N_3}\Bigg(\partial_{b_{l,2}}^{y_l(i)}\begin{bmatrix}\frac{\partial s_{l,3}^{(i)}}{\partial\Phi_{k,2}^{(i)}}\\ \frac{\partial s_{l,3}^{(i)}}{\partial\Phi_{k,2}^{(j)}}\\ \frac{\partial s_{l,3}^{(i)}}{\partial\Phi_{k,2}^{(k)}}\end{bmatrix}+\partial_{b_{l,2}}^{y_l(j)}\begin{bmatrix}\frac{\partial s_{l,3}^{(j)}}{\partial\Phi_{k,2}^{(i)}}\\ \frac{\partial s_{l,3}^{(j)}}{\partial\Phi_{k,2}^{(j)}}\\ \frac{\partial s_{l,3}^{(j)}}{\partial\Phi_{k,2}^{(k)}}\end{bmatrix}\\
&\quad+\partial_{b_{l,2}}^{y_l(k)}\begin{bmatrix}\frac{\partial s_{l,3}^{(k)}}{\partial\Phi_{k,2}^{(i)}}\\ \frac{\partial s_{l,3}^{(k)}}{\partial\Phi_{k,2}^{(j)}}\\ \frac{\partial s_{l,3}^{(k)}}{\partial\Phi_{k,2}^{(k)}}\end{bmatrix}\Bigg)\\
&= \sum_{l=1}^{N_3}\frac{1}{\|w_{k,l,2}\|}\begin{bmatrix}w_{k,l,2}^{(e)2}+w_{k,l,2}^{(i)2}+w_{k,l,2}^{(j)2}+w_{k,l,2}^{(k)2}\\ 2\left(w_{k,l,2}^{(e)}w_{k,l,2}^{(k)}-w_{k,l,2}^{(i)}w_{k,l,2}^{(j)}\right)\\ -2\left(w_{k,l,2}^{(e)}w_{k,l,2}^{(j)}+w_{k,l,2}^{(i)}w_{k,l,2}^{(k)}\right)\end{bmatrix}\\
&\quad\begin{matrix}2\left(w_{k,l,2}^{(e)}w_{k,l,2}^{(k)}+w_{k,l,2}^{(i)}w_{k,l,2}^{(j)}\right)\\ w_{k,l,2}^{(e)2}-w_{k,l,2}^{(i)2}-w_{k,l,2}^{(j)2}+w_{k,l,2}^{(k)2}\\ -2\left(w_{k,l,2}^{(e)}w_{k,l,2}^{(i)}+w_{k,l,2}^{(j)}w_{k,l,2}^{(k)}\right)\end{matrix}\\
&\quad\begin{bmatrix}-2\left(w_{k,l,2}^{(e)}w_{k,l,2}^{(j)}-w_{k,l,2}^{(i)}w_{k,l,2}^{(k)}\right)\\ 2\left(w_{k,l,2}^{(e)}w_{k,l,2}^{(i)}+w_{k,l,2}^{(j)}w_{k,l,2}^{(k)}\right)\\ w_{k,l,2}^{(e)2}-w_{k,l,2}^{(i)2}+w_{k,l,2}^{(j)2}-w_{k,l,2}^{(k)2}\end{bmatrix}\begin{bmatrix}\partial_{b_{l,2}}^{y_l(i)}\\ \partial_{b_{l,2}}^{y_l(j)}\\ \partial_{b_{l,2}}^{y_l(k)}\end{bmatrix} \qquad (89)
\end{aligned}
$$

So, $\boldsymbol{W}$ can be reformulated as

$$W = \sum_{l=1}^{N_3} \frac{Q_R(w^*_{k,l,2})Q_L(w_{k,l,2})}{\|w_{k,l,2}\|} \begin{bmatrix} \partial^{y_l(i)}_{b_{l,2}} \\ \partial^{y_l(j)}_{b_{l,2}} \\ \partial^{y_l(k)}_{b_{l,2}} \end{bmatrix} = \sum_{l=1}^{N_3} \frac{w_{k,l,2} \otimes \partial^{y_l}_{b_{l,2}} \otimes w^*_{k,l,2}}{\|w_{k,l,2}\|}. \quad (90)$$

Hence, the derivative of neuron y_l with respect to $b_{k,1}$ is

$$\frac{\partial y_l}{\partial b_{k,1}} = \left(\sum_{l=1}^{N_3} \frac{w_{k,l,2} \otimes \partial^{y_l}_{b_{l,2}} \otimes w^*_{k,l,2}}{\|w_{k,l,2}\|} \right) \odot \Phi'_{k,2}(s_{k,2}). \quad (91)$$

Finally

$$\frac{\partial y_l}{\partial w_{j,k,1}} = \begin{bmatrix} \frac{\partial y_l}{\partial w^{(e)}_{j,k,1}} \\ \frac{\partial y_l}{\partial w^{(i)}_{j,k,1}} \\ \frac{\partial y_l}{\partial w^{(j)}_{j,k,1}} \\ \frac{\partial y_l}{\partial w^{(k)}_{j,k,1}} \end{bmatrix} = \begin{bmatrix} \frac{\partial \Phi_{l,3}}{\partial s_{l,3}} \frac{\partial s_{l,3}}{\partial \Phi_{k,2}} \frac{\partial \Phi^{(i)}_{k,2}}{\partial s^{(i)}_{k,2}} \frac{\partial s^{(i)}_{k,2}}{\partial w^{(e)}_{j,k,1}} & \frac{\partial \Phi_{l,3}}{\partial s_{l,3}} \frac{\partial s_{l,3}}{\partial \Phi^{(i)}_{k,2}} \frac{\partial \Phi^{(i)}_{k,2}}{\partial s^{(i)}_{k,2}} \frac{\partial s^{(i)}_{k,2}}{\partial w^{(i)}_{j,k,1}} & \frac{\partial \Phi_{l,3}}{\partial s_{l,3}} \frac{\partial s_{l,3}}{\partial \Phi^{(j)}_{k,2}} \frac{\partial \Phi^{(j)}_{k,2}}{\partial s^{(j)}_{k,2}} \frac{\partial s^{(j)}_{k,2}}{\partial w^{(j)}_{j,k,1}} & \frac{\partial \Phi_{l,3}}{\partial s_{l,3}} \frac{\partial s_{l,3}}{\partial \Phi^{(k)}_{k,2}} \frac{\partial \Phi^{(k)}_{k,2}}{\partial s^{(k)}_{k,2}} \frac{\partial s^{(k)}_{k,2}}{\partial w^{(k)}_{j,k,1}} \end{bmatrix}$$

$$= \begin{bmatrix} \frac{\partial \Phi^{(i)}_{l,3}}{\partial s^{(i)}_{l,3}} \frac{\partial s^{(i)}_{l,3}}{\partial \Phi^{(i)}_{k,2}} \frac{\partial \Phi^{(i)}_{k,2}}{\partial s^{(i)}_{k,2}} \frac{\partial s^{(i)}_{k,2}}{\partial w^{(e)}_{j,k,1}} \\ \frac{\partial \Phi^{(i)}_{l,3}}{\partial s^{(i)}_{l,3}} \frac{\partial s^{(i)}_{l,3}}{\partial \Phi^{(i)}_{k,2}} \frac{\partial \Phi^{(i)}_{k,2}}{\partial s^{(i)}_{k,2}} \frac{\partial s^{(i)}_{k,2}}{\partial w^{(i)}_{j,k,1}} \\ \frac{\partial \Phi^{(i)}_{l,3}}{\partial s^{(i)}_{l,3}} \frac{\partial s^{(i)}_{l,3}}{\partial \Phi^{(i)}_{k,2}} \frac{\partial \Phi^{(i)}_{k,2}}{\partial s^{(i)}_{k,2}} \frac{\partial s^{(i)}_{k,2}}{\partial w^{(j)}_{j,k,1}} \\ \frac{\partial \Phi^{(i)}_{l,3}}{\partial s^{(i)}_{l,3}} \frac{\partial s^{(i)}_{l,3}}{\partial \Phi^{(i)}_{k,2}} \frac{\partial \Phi^{(i)}_{k,2}}{\partial s^{(i)}_{k,2}} \frac{\partial s^{(i)}_{k,2}}{\partial w^{(k)}_{j,k,1}} \end{bmatrix}$$

$$+ \begin{bmatrix} \frac{\partial \Phi^{(j)}_{l,3}}{\partial s^{(j)}_{l,3}} \frac{\partial s^{(j)}_{l,3}}{\partial \Phi^{(j)}_{k,2}} \frac{\partial \Phi^{(j)}_{k,2}}{\partial s^{(j)}_{k,2}} \frac{\partial s^{(j)}_{k,2}}{\partial w^{(e)}_{j,k,1}} \\ \frac{\partial \Phi^{(j)}_{l,3}}{\partial s^{(j)}_{l,3}} \frac{\partial s^{(j)}_{l,3}}{\partial \Phi^{(j)}_{k,2}} \frac{\partial \Phi^{(j)}_{k,2}}{\partial s^{(j)}_{k,2}} \frac{\partial s^{(j)}_{k,2}}{\partial w^{(i)}_{j,k,1}} \\ \frac{\partial \Phi^{(j)}_{l,3}}{\partial s^{(j)}_{l,3}} \frac{\partial s^{(j)}_{l,3}}{\partial \Phi^{(j)}_{k,2}} \frac{\partial \Phi^{(j)}_{k,2}}{\partial s^{(j)}_{k,2}} \frac{\partial s^{(j)}_{k,2}}{\partial w^{(j)}_{j,k,1}} \\ \frac{\partial \Phi^{(j)}_{l,3}}{\partial s^{(j)}_{l,3}} \frac{\partial s^{(j)}_{l,3}}{\partial \Phi^{(j)}_{k,2}} \frac{\partial \Phi^{(j)}_{k,2}}{\partial s^{(j)}_{k,2}} \frac{\partial s^{(j)}_{k,2}}{\partial w^{(k)}_{j,k,1}} \end{bmatrix}$$

$$+ \begin{bmatrix} \frac{\partial \Phi^{(k)}_{l,3}}{\partial s^{(k)}_{l,3}} \frac{\partial s^{(k)}_{l,3}}{\partial \Phi^{(k)}_{k,2}} \frac{\partial \Phi^{(k)}_{k,2}}{\partial s^{(k)}_{k,2}} \frac{\partial s^{(k)}_{k,2}}{\partial w^{(e)}_{j,k,1}} \\ \frac{\partial \Phi^{(k)}_{l,3}}{\partial s^{(k)}_{l,3}} \frac{\partial s^{(k)}_{l,3}}{\partial \Phi^{(k)}_{k,2}} \frac{\partial \Phi^{(k)}_{k,2}}{\partial s^{(k)}_{k,2}} \frac{\partial s^{(k)}_{k,2}}{\partial w^{(i)}_{j,k,1}} \\ \frac{\partial \Phi^{(k)}_{l,3}}{\partial s^{(k)}_{l,3}} \frac{\partial s^{(k)}_{l,3}}{\partial \Phi^{(k)}_{k,2}} \frac{\partial \Phi^{(k)}_{k,2}}{\partial s^{(k)}_{k,2}} \frac{\partial s^{(k)}_{k,2}}{\partial w^{(j)}_{j,k,1}} \\ \frac{\partial \Phi^{(k)}_{l,3}}{\partial s^{(k)}_{l,3}} \frac{\partial s^{(k)}_{l,3}}{\partial \Phi^{(k)}_{k,2}} \frac{\partial \Phi^{(k)}_{k,2}}{\partial s^{(k)}_{k,2}} \frac{\partial s^{(k)}_{k,2}}{\partial w^{(k)}_{j,k,1}} \end{bmatrix}$$

$$= \partial^{y_l(i)}_{b_{k,1}} \begin{bmatrix} \frac{\partial s^{(i)}_{k,2}}{\partial w^{(e)}_{j,k,1}} \\ \frac{\partial s^{(i)}_{k,2}}{\partial w^{(i)}_{j,k,1}} \\ \frac{\partial s^{(i)}_{k,2}}{\partial w^{(j)}_{j,k,1}} \\ \frac{\partial s^{(i)}_{k,2}}{\partial w^{(k)}_{j,k,1}} \end{bmatrix} + \partial^{y_l(j)}_{b_{k,1}} \begin{bmatrix} \frac{\partial s^{(j)}_{k,2}}{\partial w^{(e)}_{j,k,1}} \\ \frac{\partial s^{(j)}_{k,2}}{\partial w^{(i)}_{j,k,1}} \\ \frac{\partial s^{(j)}_{k,2}}{\partial w^{(j)}_{j,k,1}} \\ \frac{\partial s^{(j)}_{k,2}}{\partial w^{(k)}_{j,k,1}} \end{bmatrix} + \partial^{y_l(k)}_{b_{k,1}} \begin{bmatrix} \frac{\partial s^{(k)}_{k,2}}{\partial w^{(e)}_{j,k,1}} \\ \frac{\partial s^{(k)}_{k,2}}{\partial w^{(i)}_{j,k,1}} \\ \frac{\partial s^{(k)}_{k,2}}{\partial w^{(j)}_{j,k,1}} \\ \frac{\partial s^{(k)}_{k,2}}{\partial w^{(k)}_{j,k,1}} \end{bmatrix} \quad (92)$$

and then

$$\begin{aligned} \frac{\partial y_l}{\partial w_{j,k,1}} = \frac{1}{\|w_{j,k,1}\|} \Bigg[2\Bigg(& w_{j,k,1} \otimes L^*_{j,1} \otimes \partial^{y_l(i)}_{b_{k,1}} i \\ & + \partial^{y_l(j)}_{b_{k,1}} j \otimes L^{*(i)}_{j,1} i \otimes w_{j,k,1} \\ & + i \otimes L^{*(j)}_{j,1} j \otimes w_{j,k,1} \otimes \partial^{y_l(j)}_{b_{k,1}} j \otimes i \\ & + L^{*(k)}_{j,1} k \otimes w_{j,k,1} \otimes \partial^{y_l(j)}_{b_{k,1}} j \\ & + \partial^{y_l(k)}_{b_{k,1}} k \otimes L^{*(i)}_{j,1} i \otimes w_{j,k,1} \\ & + L^{*(j)}_{j,1} j \otimes w_{j,k,1} \otimes \partial^{y_l(k)}_{b_{k,1}} k \\ & + i \otimes L^{*(k)}_{j,1} k \otimes w_{j,k,1} \otimes \partial^{y_l(k)}_{b_{k,1}} k \otimes i \Bigg) \\ & - \partial^{y_l}_{b_{k,1}} \odot \frac{w_{j,k,1} \otimes L_{j,1} \otimes w^*_{j,k,1}}{\|w_{j,k,1}\|^4} \otimes \lambda_2 q_2 \Bigg]. \end{aligned} \quad (93)$$

where

$$\lambda_2 = w^{(e)2}_{j,k,1} + w^{(i)2}_{j,k,1} - w^{(j)2}_{j,k,1} - w^{(k)2}_{j,k,1} \mid \lambda_1 \in \mathbb{R},$$
$$q_2 = w^{(e)}_{j,k,1} + w^{(i)}_{j,k,1} i - w^{(j)}_{j,k,1} j - w^{(k)}_{j,k,1} k \mid q_1 \in \mathbb{H}.$$

Therefore, the quaternion-version for the derivatives of y_l, with respect to $\partial^{y_l}_{b_{l,2}}$, $\partial^{y_l}_{w_{k,l,2}}$, $\partial^{y_l}_{b_{k,1}}$ and $\partial^{y_l}_{w_{j,k,1}}$, is summarized as follows:

$$\partial^{y_l}_{b_{l,2}} = \Phi'_{l,3}(s_{l,3}), \quad (94)$$

$$\begin{aligned} \partial^{y_l}_{w_{k,l,2}} = \frac{1}{\|w_{k,l,2}\|} \Bigg[2\Bigg(& w_{k,l,2} \otimes L^*_{k,2} \otimes \partial^{y_l(i)}_{b_{l,2}} i \\ & + \partial^{y_l(j)}_{b_{l,2}} j \otimes L^{*(i)}_{k,2} i \otimes w_{k,l,2} \\ & + i \otimes L^{*(j)}_{k,2} j \otimes w_{k,l,2} \otimes \partial^{y_l(j)}_{b_{l,2}} j \otimes i \\ & + L^{*(k)}_{k,2} k \otimes w_{k,l,2} \otimes \partial^{y_l(j)}_{b_{l,2}} j \end{aligned}$$

$$
\begin{aligned}
&+ \partial_{b_{l,2}}^{y_l(k)} k \otimes L_{k,2}^{*(i)} i \otimes w_{k,l,2} \\
&+ L_{k,2}^{*(j)} j \otimes w_{k,l,2} \otimes \partial_{b_{l,2}}^{y_l(k)} k \\
&+ i \otimes L_{k,2}^{*(k)} k \otimes w_{k,l,2} \otimes \partial_{b_{l,2}}^{y_l(k)} k \otimes i \Bigg) \\
&- \partial_{b_{l,2}}^{y_l} \odot \frac{w_{k,l,2} \otimes L_{k,2} \otimes w_{k,l,2}^*}{\left\| w_{k,l,2} \right\|^4} \otimes \lambda_1 q_1 \Bigg], \qquad (95)
\end{aligned}
$$

$$
\partial_{b_{k,1}}^{y_l} = \left(\sum_{l=1}^{N_3} \frac{w_{k,l,2} \otimes \partial_{b_{l,2}}^{y_l} \otimes w_{k,l,2}^*}{\left\| w_{k,l,2} \right\|} \right) \odot \Phi_{k,2}'(s_{k,2}), \qquad (96)
$$

$$
\begin{aligned}
\partial_{w_{j,k,1}}^{y_l} = \frac{1}{\left\| w_{j,k,1} \right\|} \Bigg[2 \Bigg(& w_{j,k,1} \otimes L_{j,1}^* \otimes \partial_{b_{k,1}}^{y_l(i)} i \\
&+ \partial_{b_{k,1}}^{y_l(j)} j \otimes L_{j,1}^{*(i)} i \otimes w_{j,k,1} \\
&+ i \otimes L_{j,1}^{*(j)} j \otimes w_{j,k,1} \otimes \partial_{b_{k,1}}^{y_l(j)} j \otimes i \\
&+ L_{j,1}^{*(k)} k \otimes w_{j,k,1} \otimes \partial_{b_{k,1}}^{y_l(j)} j \\
&+ \partial_{b_{k,1}}^{y_l(k)} k \otimes L_{j,1}^{*(i)} i \otimes w_{j,k,1} \\
&+ L_{j,1}^{*(j)} j \otimes w_{j,k,1} \otimes \partial_{b_{k,1}}^{y_l(k)} k \\
&+ i \otimes L_{j,1}^{*(k)} k \otimes w_{j,k,1} \otimes \partial_{b_{k,1}}^{y_l(k)} k \otimes i \Bigg) \\
&- \partial_{b_{k,1}}^{y_l} \odot \frac{w_{j,k,1} \otimes L_{j,1} \otimes w_{j,k,1}^*}{\left\| w_{j,k,1} \right\|^4} \otimes \lambda_2 q_2 \Bigg]. \qquad (97)
\end{aligned}
$$

where

$$
\begin{aligned}
\lambda_1 &= w_{k,l,2}^{(e)2} + w_{k,l,2}^{(i)2} - w_{k,l,2}^{(j)2} - w_{k,l,2}^{(k)2} \mid \lambda_1 \in \mathbb{R}, \\
q_1 &= w_{k,l,2}^{(e)} + w_{k,l,2}^{(i)} i - w_{k,l,2}^{(j)} j - w_{k,l,2}^{(k)} k \mid q_1 \in \mathbb{H}_s, \\
\lambda_2 &= w_{j,k,1}^{(e)2} + w_{j,k,1}^{(i)2} - w_{j,k,1}^{(j)2} - w_{j,k,1}^{(k)2} \mid \lambda_2 \in \mathbb{R}, \\
q_2 &= w_{j,k,1}^{(e)} + w_{j,k,1}^{(i)} i - w_{j,k,1}^{(j)} j - w_{j,k,1}^{(k)} k \mid q_2 \in \mathbb{H}_s.
\end{aligned}
$$

ACKNOWLEDGMENT

F. A. Author thanks CONAHCYT, Mexico, for the financial support that has given him during the time he has been a student at the Center for Research and Advanced Studies (CINVESTAV), National Polytechnic Institute (IPN), to carry out this work and obtain his Ph.D. degree under Grant CVU.854352.

REFERENCES

[1] W. K. Pratt, *Digital Image Processing*, 2nd ed. Hoboken, NJ, USA: Wiley, 1991. [Online]. Available: https://www.worldcat.org/oclc/21763433

[2] A. Jain, *Fundamentals of Digital Image Processing* (Prentice-Hall Information and System Sciences Series). Upper Saddle River, NJ, USA: Prentice-Hall, 1989. [Online]. Available: https://books.google.com.mx/books?id=GANSAAAAMAAJ

[3] D. Xu, Y. Xia, and D. P. Mandic, "Optimization in quaternion dynamic systems: Gradient, hessian, and learning algorithms," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 2, pp. 249–261, Feb. 2016, doi: 10.1109/TNNLS.2015.2440473.

[4] D. Xu, C. Jahanchahi, C. C. Took, and D. P. Mandic, "Quaternion derivatives: The GHR calculus," *Roy. Soc. Open Sci.*, vol. 2, no. 8, p. 150255, 2015, doi: 10.1098/rsos.150255.

[5] D. Xu, L. Zhang, and H. Zhang, "Learning algorithms in quaternion neural networks using GHR calculus," *Neural Netw. World*, vol. 27, no. 3, pp. 271–282, 2017, doi: 10.14311/NNW.2017.27.014.

[6] S. Singhal and L. Wu, "Training multilayer perceptrons with the extended Kalman algorithm," in *Proc. Adv. Neural Inf. Process. Syst.*, Jan. 1989, pp. 133–140.

[7] G. Puskorius and L. Feldkamp, "Parameter-based Kalman filter training: Theory and implementation," in *Kalman Filtering and Neural Networks*, S. S. Haykin, Ed. Hoboken, NJ, USA: Wiley, Mar. 2002, pp. 23–67, doi: 10.1002/0471221546.ch2.

[8] Y. Iiguni, H. Sakai, and H. Tokumaru, "A real-time learning algorithm for a multilayered neural network based on the extended Kalman filter," *IEEE Trans. Signal Process.*, vol. 40, no. 4, pp. 959–966, Apr. 1992, doi: 10.1109/78.127966.

[9] H. C. S. Rughooputh and S. D. D. V. Rughooputh, "Extended Kalman filter learning algorithm for hyper-complex multilayer neural networks," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, vol. 3, Jul. 1999, pp. 1824–1828, doi: 10.1109/IJCNN.1999.832656.

[10] N. Matsui, T. Isokawa, H. Kusamichi, F. Peper, and H. Nishimura, "Quaternion neural network with geometrical operators," *J. Intell. Fuzzy Syst.*, vol. 15, nos. 3–4, pp. 149–164, Dec. 2004.

[11] H. Kusamichi, T. Isokawa, and N. Matsui, "A new scheme for color night vision by quaternion neural network," in *Proc. 2nd Int. Conf. Auton. Robots Agents*, Palmerston North, New Zealand, Dec. 2004, pp. 101–106.

[12] T. Isokawa, N. Matsui, and H. Nishimura, "Quaternionic neural networks: Fundamental properties and applications," in *Complex-Valued Neural Networks: Utilizing High-Dimensional Parameters*, vol. 15, nos. 3–4, T. Nitta, Ed. Hershey, PA, USA: IGI Global, Jan. 2009, pp. 411–439, doi: 10.4018/978-1-60566-214-5.ch016.

[13] C. Perwass, *Geometric Algebra With Applications in Engineering* (Geometry and Computing), vol. 4. Berlin, Germany: Springer, 2009, pp. 1–385.

[14] J. J. Rodríguez, "El Álgebra y la Geometría de los Cuaternios y Algunas de sus Aplicaciones," Bachelor thesis, Universidad de Sonora, División de Ciencias Exactas y Naturales, Hermosillo, Mexico, 2010.

[15] F. J. Somma, "Cuaterniones y Ángulos de Euler para Describir Rotaciones en $\mathbb{R}^3$," Bachelor thesis, Universidad Abierta Interamericana, Facultad de Tecnología Informática, Buenos Aires, Argentina, 2018.

[16] J. Vince, *Geometric Algebra: An Algebraic System for Computer Games and Animation*. London, U.K.: Springer, Jan. 2009, doi: 10.1007/978-1-84882-379-2.

[17] Y. Alagöz, K. Oral, and S. Yüce, "Split quaternion matrices," *Miskolc Math. Notes*, vol. 13, no. 2, pp. 223–232, Jun. 2012, doi: 10.18514/MMN.2012.364.

[18] A. R. Smith, "Color gamut transform pairs," *ACM SIGGRAPH Comput. Graph.*, vol. 12, pp. 12–19, Aug. 1978, doi: 10.1145/800248.807361.

[19] M. Nørgaard, O. Ravn, N. K. Poulsen, and L. K. Hansen, *Neural Networks for Modelling and Control of Dynamic Systems: A Practitioner's Handbook*. London, U.K.: Springer, Jan. 2000. [Online]. Available: https://link.springer.com/book/9781852332273

[20] A. Krizhevsky, V. Nair, and G. Hinton, "CIFAR-10 (Canadian institute for advanced research)," 2009. [Online]. Available: http://www.cs.toronto.edu/~kriz/cifar.html

[21] B. Gábor. (2023). *Video Input With OpenCV and Similarity Measurement*. Accessed: Apr. 10, 2023. [Online]. Available: https://docs.opencv.org/4.x/d5/dc4/tutorial_video_input_psnr_ssim.html

[22] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004, doi: 10.1109/TIP.2003.819861.

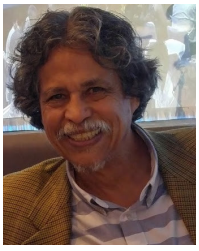
[23] (2023). *OpenCV: Open Source Computer Vision. Histogram Comparison*. Accessed: Apr. 10, 2023. [Online]. Available: https://docs.opencv.org/3.4/d8/dc8/tutorial_histogram_comparison.html

[24] J. Warburton, "Scarlet macaw," [Image] from 500px by the Archive Team (detail page). This file is licensed under the Creative Commons Attribution 3.0 Unported license, May 2012. Accessed: Apr. 10, 2023. [Online]. Available: https://commons.wikimedia.org/wiki/File:Scarlet_Macaw_(17275987).jpeg

[25] K. Kissel, "June's multi-colored eyes," [Image] from flick.com. This file is licensed under the Creative Commons Attribution 2.0 Generic license, Feb. 2012. Accessed: Apr. 10, 2023. [Online]. Available: https://commons.wikimedia.org/wiki/File:June_odd-eyed-cat_cropped.jpg

[26] C. J. Balboa, "Red-eyed tree frog (agalychnis callidryas)," [Image] this work has been released into the public domain by its author, Carey-jamesbalboa at English Wikipedia. This applies worldwide, Aug. 2007. Accessed: Apr. 10, 2023. [Online]. Available: https://commons.wikimedia.org/wiki/File:Red_eyed_tree_frog_edit2.jpg
[27] D. Baeza, "Acuario 033," [Image] this file is licensed under the Creative Commons Attribution-Share Alike 3.0 Unported license, Jun. 2014. Accessed: Jul. 10, 2023. [Online]. Available: https://commons.wikimedia.org/wiki/File:Acuario_033.JPG
[28] N. More, "Camping in tent alone under the sky in night with dog (2) 10," [Image] this file is licensed under the Creative Commons Attribution-Share Alike 4.0 International license, Sep. 2020. Accessed: Jul. 10, 2023. [Online]. Available: https://commons.wikimedia.org/wiki/File:Camping_in_tent_alone_under_the_sky_in_night_with_dog_(2)_10.jpg
[29] Blümchenkäfer, "Cat in the dark," [Image] this file is licensed under the Creative Commons Attribution-Share Alike 4.0 International license, Jan. 2021. Accessed: Jul. 10, 2023. [Online]. Available: https://commons.wikimedia.org/wiki/File:Cat_in_the_dark.jpg
[30] K. Varma, "Leopard at night DSC 9503," [Image] this file is licensed under the Creative Commons Attribution-Share Alike 4.0 International license, Mar. 2005. Accessed: Jul. 10, 2023. [Online]. Available: https://commons.wikimedia.org/wiki/File:Leopard_at_night_DSC_9503.jpg
[31] A. Kumar, "Stained glass paintings at Sagrada Familia, Barcelona (Ank Kumar) 05," [Image] this file is licensed under the Creative Commons Attribution-Share Alike 4.0 International license, Sep. 2014, Accessed: Jul. 10, 2023. [Online]. Available: https://commons.wikimedia.org/wiki/File:Stained_Glass_paintings_at_Sagrada_Familia,_Barcelona_(_Ank_Kumar)_05.jpg
[32] M. Meraji, "Tbilisi City—Urban Photos—Georgia Tourism 11," [Image] this file is licensed under the Creative Commons Attribution-Share Alike 3.0 Unported license, Oct. 2017. Accessed: Jul. 10, 2023. [Online]. Available: https://commons.wikimedia.org/wiki/File:Tbilisi_City_-_Urban_Photos_-_Georgia_Tourism_11.jpg
[33] K. Varma, "Tiger in the night DSC 5138," [Image] this file is licensed under the Creative Commons Attribution-Share Alike 4.0 International license, Jul. 2005. Accessed: Jul. 10, 2023. [Online]. Available: https://commons.wikimedia.org/wiki/File:Tiger_in_the_night_DSC_5138.jpg
[34] P. Rads, "Tungsten light," [Image] this file is licensed under the Creative Commons Attribution-Share Alike 4.0 International license, Apr. 2017. Accessed: Jul. 10, 2023. [Online]. Available: https://commons.wikimedia.org/wiki/File:Tungsten_light.jpg

EDUARDO DE JESÚS DÁVILA-MEZA was born in Guadalajara, Mexico, in 1994. He received the B.Sc. degree in mechatronics engineering from the University of Guadalajara, Guadalajara, in 2017, and the M.Sc. degree from the Center of Research and Advanced Studies (CINVESTAV), National Polytechnic Institute (IPN), Guadalajara Campus, Zapopan, Mexico, in 2019, where he is currently pursuing the Ph.D. degree. His main research interests include the application of artificial neural networks, convolutional neural networks, and computer vision oriented to the detection and recognition of objects, and medical image processing.



EDUARDO JOSE BAYRO-CORROCHANO (Senior Member, IEEE) received the Ph.D. degree in systems engineering from the University of Wales, Cardiff, in 1994. From 1995 to 1999, he was a Researcher and a Lecturer with the Institute of Computer Science, Christian Albrechts University of Kiel, Germany, on Clifford geometric algebra applications to cognitive systems. From 2007 to 2008, he was a DFG Mercator Gastprofessor with the KIT, Technische Hochschule, Karlsruhe, Germany. From 2014 to 2015, he was a Visiting Full Professor with the Media Laboratory, MIT, Boston, MA, USA. He is currently a Full Professor of geometric cybernetics with the Department of Electrical Engineering and Computer Science, CINVESTAV, IPN, Campus Guadalajara, Jalisco, Mexico. He is the author of seven Springer-Verlag books and has published more than 230 refereed journal articles, book chapters, and conference papers. He is a fellow of the International Association of Pattern Recognition Society and the Asian Artificial Intelligence Association. He was the General Chair of ICPR2016, in December, Cancun, Mexico, and the IEEE/RAS Humanoids 2016, in November, Cancun. He was an Associate Editor of the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, *Journal of Mathematical Imaging and Vision*, and *Journal of Pattern Recognition*. He is an Associate Editor of the *Journal of Robotics* and an Editor of the ICRA Conference.