Quaternion Rational Activation Functions

***Abstract*—Quaternion Neural Networks (QNNs) extend traditional deep learning models by operating in the four-dimensional quaternion sub-space, offering a compact and naturally expressive framework for representing multidimensional data. At the heart of neural networks lies the activation function, which introduces non-linearity into the forward process. Due to quaternion algebra, activation functions known as "split" activation functions are predominantly used, as they act on each hypercomplex component independently. In this paper, we introduce for the first time two new activation functions based on the Rational Activation Function (RAF), a class of learnable activation functions characterized by their trainable parameters. Specifically, we propose one variant where the learnable parameters are shared across all hypercomplex components, and another activation function where each component has its own distinct set of learnable parameters. These new Quaternion Rational Activation Functions (QRAF) are evaluated across different neural architectures and tasks, including natural language processing and image processing and show promising reults.**

***Index Terms*—Quaternion neural networks, Rational activation function,**

I. Introduction

Quaternion-valued neural networks (QNN) have been an important research area for the past decade [1] [2], with application on different real-life tasks, including image processing [3] [4], moving object modelling [5] [6] or natural language processing [7] [8]. More architectures of neural networks operating in real numbers have been extensively adapted to quaternion's algebra [2]. Architectures such as long-short-term memory (LSTM), transformers, convolutional neural networks (CNN), and others have their equivalents in quaternion neural networks [9] [10] [4] with a promising gain in terms of number of neural parameters as well as in terms of performances observed.

One of the main challenge in QNN lies with the choice of activation function. In a same way as for their real-valued counterpart, activation functions in QNN introduce non-linearity within the computing process [11], and thus enhance expressiveness and allows for better data representation within the hidden vectorial sub-spaces. The choice of QNN activation functions is non-trivial. As explained in [2], quaternion activation functions are divided into two parts; "whole" activation functions and "split" activation functions. Per their name, whole activation functions operate on the whole quaternion, with intra-component interactions due to the Hamilton dot product. In contrast, split activation functions operate on each component independently, without regard to the other ones. Whole activation functions and split activation functions have been investigated [12] [13]and even if whole activation functions can reach better results on some tasks, split activation functions are often preferred, since the former requires careful training due to numerous singularities that can significantly affect QNN performances [2] [14]. This reason led authors to investigate original split quaternion activation functions which could still retain some aspects of intra-components interactions within the quaternion.

Activation functions for neural networks in general are diverse and rooted in various mathematical disciplines— such as probability, analysis, and statistics— [15]. As highlighted in [16], interest in activation functions has grown significantly in recent years. In particular, activation functions with learnable parameters have attracted considerable attention in neural network research [16], as they enhance the ability to capture hidden patterns, reduce the total number of learnable parameters, lower memory consumption, and improve generative performance. Several activation functions, including the well-known ReLU [17], have shown their effectiveness across a broad spectrum of real-world related tasks. However, linear layers alongside with ReLU requires a large number of parameters, especially in high-dimensional settings such as word representations (e.g., word embeddings [18]), leading to high memory usage.

Among recent developments, trainable "Rational Activation Functions" (RAF) have emerged as a promising alternative [19] to mere ReLU and other non-learnable activation functions. RAFs are defined as the ratio of two polynomials of different degrees, with all coefficients treated as learnable parameters. As demonstrated in [19], such functions can closely approximate the behaviour of ReLU-based neural networks and have shown strong performances on various NLP tasks.

QNN with trainable activation functions is a relatively unexplored area of research. To the authors' knowledge, only [11] have evaluated trainable Bessel activation functions on Quaternion Convolutional Neural Networks (QCNN). However, their proposed activation function with trainable parameters strictly follows the classical split activation structure, as each component of the quaternion is subjected to the same activation function without considering algebra properties of quaternions.

The aim of this paper is to investigate benefit of Quaternion Rational Activation Functions (QRAF) for different quaternion neural architectures, and introduce an original QRAF with component specific dedicated RAF, to instill some aspects of intra-components interactions within the quaternion learning phase.

II. Quaternions Neural Networks

To evaluate the effectiveness of the proposed quaternion RAF, different quaternion-valued neural networks are employed such as QMLP (section II-B), QRNN (section II-C), QLSTM (section II-D) and QCNN (section II-E) on different tasks. This section gives first some basics of quaternion numbers (section II-A) and highlights these models and the differences with casual real-valued neural networks.

A. Quaternions Algebra

Quaternions are an extension of the complex numbers. A quaternion q is defined as follows:

$$q = r + \mathbf{i}x + \mathbf{j}y + \mathbf{k}z \tag{1}$$

Where r, x, y and z are real numbers. $\mathbf{i}, \mathbf{j}$ and $\mathbf{k}$ follows the quaternion fundamental formula:

$$\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -1 \tag{2}$$

The quaternion space is a four-dimension associative division algebra over the real numbers, written $\mathbb{H}$. Quaternions are composed of two parts, $q = r + \overrightarrow{S}$, where r is the real component and $\overrightarrow{S} = \mathbf{i}x + \mathbf{j}y + \mathbf{k}z$ is the hypercomplex component. Quaternions can be represented in the real space $\mathbb{R}$ using the injective morphism:

$$f_{\mathbb{R}}(q) = \begin{pmatrix} r & -x & -y & -z \\ x & r & -z & y \\ y & z & x & -y \\ z & -y & x & r \end{pmatrix} \tag{3}$$

And in the complex space $\mathbb{C}$:

$$f_{\mathbb{C}}(q) = \begin{pmatrix} r + \mathbf{i}x & -y - \mathbf{i}z \\ y - \mathbf{i}z & r - \mathbf{i}x \end{pmatrix} \tag{4}$$

The conjugate q^* is an involution of q:

$$q^* = r - \mathbf{i}x - \mathbf{j}y - \mathbf{k}z. \tag{5}$$

The reciprocal of a non-zero quaternion is

$$q^{-1} = \frac{q^*}{||q||^2} \tag{6}$$

with $||.||$ being the euclidean norm. In the $\mathbb{H}$ space, the euclidean norm of a quaternion is defined as $||q|| = \sqrt{r^2 + x^2 + y^2 + z^2}$.

According to the Froebenius theorem [20]:

Theorem 1 (Froebenius)**.** *All finite-dimensional associative division algebras over the real numbers are isomorphic to either the real number space ($\mathbb{R}$), the complex space ($\mathbb{C}$) or the quaternion space ($\mathbb{H}$).*

Using the quaternion fundamental formula, multiplying two quaternions $q_1 = r_1 + \mathbf{i}x_1 + \mathbf{j}y_1 + \mathbf{k}z_1$ and $q_2 = r_2 + \mathbf{i}x_2 + \mathbf{j}y_2 + \mathbf{k}z_2$ is called the Hamilton product, and is define as follows:

$$\begin{aligned} q_1 \otimes q_2 = {} & r_1 r_2 - x_1 x_2 - y_1 y_2 - z_1 z_2 \\ & (r_1 x_2 + x_1 r_2 + y_1 z_2 - z_1 y_2)\mathbf{i} \\ & (r_1 y_2 - x_1 z_2 + y_1 r_2 + z_1 x_2)\mathbf{j} \\ & (r_1 z_2 + x_1 y_2 - y_1 x_2 + z_1 r_2)\mathbf{k} \end{aligned} \tag{7}$$

The Hamilton product is non-commutative (figure 1) and a quaternion has a polar decomposition:

$$q = ||q||.e^{\eta.\phi} = ||q||.(cos(\phi) + \eta.sin(\phi)) \tag{8}$$

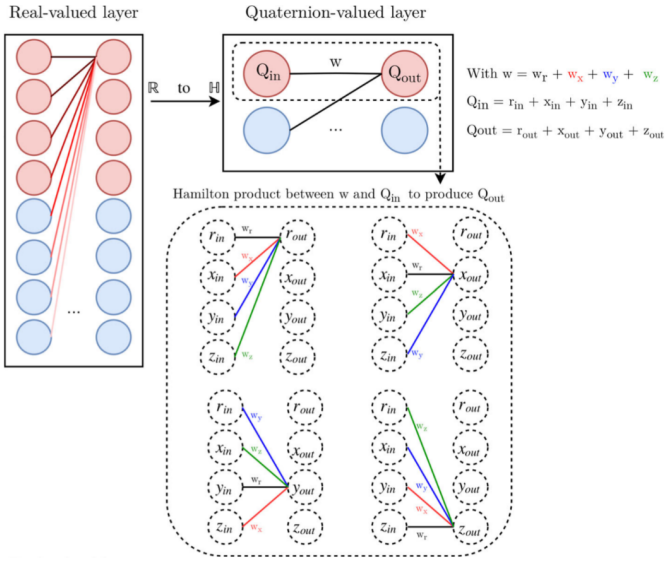

Fig. 1. Illustration of the input features (Q_{in}) latent relations learning ability of a quaternion-valued layer (right) due to the quaternion weight sharing of the Hamilton product (7), compared to a standard real-valued layer (left) [9]

B. Quaternion Multilayer Perceptron

A Quaternion-valued Multilayer Perceptron (QMLP) operates in the same way as a real-valued Multilayer Perceptron (MLP) but within the quaternion algebra. All of its inputs, weights and biases are quaternion numbers. This subsection details the QMLP's forward phase first introduced by [21]. Let M be the number of layers, composed of N nodes, where N may vary from one layer to another. Let x be the input of a node. Let N_l be the number of neurons in the layer l, with $1 < l < M$. The bias of the neuron n, with $1 < n < N_l$, from the layer l, is b_n^l. Let P be a set of quaternion input patterns x_p with $1 < p < P$ and t_p a label attached to an input x_p. The output γ_n^l, with $\gamma_n^0 = x_p^n$, is given by:

$$\gamma_n^l = \alpha(S_n^l) \tag{9}$$

where α is a quaternion-valued activation function and with:

$$S_n^l = \sum_{m=0}^{N_{l-1}} w_{n,m}^l \otimes \gamma_m^{l-1} + b_n^l \tag{10}$$

where $w_{n,m}^l$ is a quaternion-valued weight. In QNN, the split activation function is the most commonly used type of activation function. Split-type activation functions are defined as follows:

$$\alpha(q) = f(r) + \mathbf{i}f(x) + \mathbf{j}f(y) + \mathbf{k}f(z) \tag{11}$$

with f being a real-valued differential function. The split function is convenient to use for QNN, as the differentiability of these functions are easily proven, and the computing of these functions can be done. As we can see from (11), no Hamiltonian product is involved for the computing of a split-activation function.

C. Quaternion Recurrent Neural Network

Quaternion Recurrent Neural Network extends the architecture of Recurrent Neural Networks (RNN) to the quaternion algebra. First introduced in [9], QRNNs have exhibited good performances for rigid body dynamics [5] or convex optimisation [22]. Like RNNs, QRNNs transmit information across sequence steps through internal hidden states, enabling them to capture temporal dependencies and contextual patterns.

Consider a Quaternion Recurrent Neural Network (QRNN) with a hidden state consisting of H neurons. Let w_{hh}, $w_{h\gamma}$, and $w_{\gamma h}$ denote the hidden-to-hidden, input-to-hidden, and hidden-to-output weight matrices, respectively. The hidden state h_{tl}^{n} of neurons n at time step t and layer l is computed as follows:

$$h_n^{t,l} = \alpha\left(\sum_{m=0}^{H} w_{nm,hh}^{l} \otimes h_m^{t-1,l} + \sum_{m=0}^{N_{l-1}} w_{nm,h\gamma} \otimes \gamma_m^{t,l-1} + b_n^l\right) \quad (12)$$

With α defined as in (11). The output of the neuron n is

$$\gamma_n^{t,l} = \beta\left(\sum_{m=0}^{N_{l-1}} w_{nm,h\gamma} \otimes h_n^{t,l-1} + b_n^l\right) \quad (13)$$

With β defined as in (11).

D. Quaternion Long-Short Term Memory

LSTM is a type of RNN that has been firstly introduced by [23]. LSTM networks are a variant of RNN specifically designed to effectively model long-term dependencies in sequential data. They achieve this through a gating mechanism—comprising input, forget, and output gates — that regulates the flow of information, determining which data to retain, discard, or output at each time step.

Quaternion Long-Short Term Memory (QLSTM) has been investigated by [24]. QLSTMs have been employed for tasks such as human action recognition [25] or sensor fusion [26]. Let f_t, i_t, o_t, c_t and h_t be the forget, input, output gates, cell states and the hidden state of a QLSTM cell at time-step t in this order.

$$f_t = \sigma(W_f \otimes x_t + R_f \otimes h_{t-1} + b_f) \quad (14)$$

$$i_t = \sigma(W_i \otimes x_t + R_i \otimes h_{t-1} + b_i) \quad (15)$$

$$c_t = f_t * c_{t-1} + i_t * \alpha(W_c x_t + R_c h_{t-1} + b_c) \quad (16)$$

$$o_t = \sigma(W_o \otimes x_t + R_o \otimes h_{t-1} + b_o) \quad (17)$$

$$h_t = o_t * \alpha(c_t) \quad (18)$$

With σ and α the sigmoid and tanh quaternion split activations:

$$\sigma(q) = \frac{1}{1+e^{-r}} + \frac{1}{1+e^{-x}}\mathbf{i} + \frac{1}{1+e^{-y}}\mathbf{j} + \frac{1}{1+e^{-z}}\mathbf{k} \quad (19)$$

$$\alpha(q) = \frac{e^{r}-e^{-r}}{e^{r}+e^{-r}} + \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}\mathbf{i} + \frac{e^{y}-e^{-y}}{e^{y}+e^{-y}}\mathbf{j} + \frac{e^{z}-e^{-z}}{e^{z}+e^{-z}}\mathbf{k} \quad (20)$$

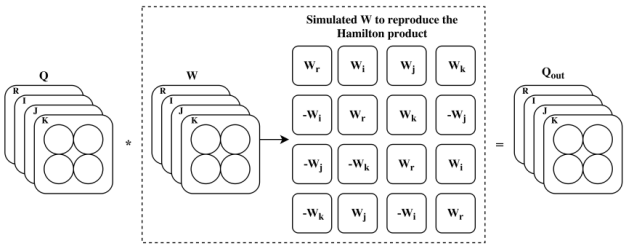

Fig. 2. Illustration of the quaternion convolution [9]

E. Quaternion Convolutional Neural Network

Convolutional Neural Networks (CNNs) are a type of deep learning architecture particularly effective for analysing grid-structured data, such as images. They rely on convolutional layers that apply filters (also known as kernels) to extract local features like edges, textures, or shapes. As more layers are stacked, CNNs progressively learn more abstract and complex representations - starting from basic patterns in early layers to full object representations in deeper ones. Typically, they also include pooling layers to downsample spatial dimensions and fully connected layers for producing final predictions or classifications. Quaternion Convolutional Neural Networks (QCNN) have been introduced in [4] and shown good results for heterogeneous image processing [3], colour image classification and de-noising tasks [4].

As for previous architecture, QCNN fully operates in the quaternion algebra. QCNN's forward phase is defined as follows: with $S_{a,b}^{l}$ the pre-activation output at layer l and at the indexes (a, b) of the new feature map and w the weight-filter map of size $f * f$:

$$\gamma_{a,b}^{l} = \alpha(S_{a,b}^{l}) \quad (21)$$

With:

$$S_{a,b}^{l} = \sum_{c=0}^{f-1}\sum_{d=0}^{f-1} w^{l} \otimes \gamma_{a+c,b+d}^{l-1} \quad (22)$$

And α defined as in (11).

III. Real- and Quaternion-valued Rational activation functions

The previous section described the quaternion-valued neural networks employed during the experiments using the hitherto proposed real-valued RAF (section III-B) and the proposed quaternion-valued RAF (section III-C).

A. Background In Learnable Activation Functions

Activation functions can be broadly categorised into two types: non-trainable and trainable. As the name suggests, non-trainable activation functions perform fixed transformations on the input data and remain unchanged throughout the learning process. In contrast, trainable activation functions incorporate learnable parameters that are adjusted during training, typically through back propagation using gradient descent. One of

the earliest examples of such functions is the Adjustable Generalized Sigmoid (AGSig) [27], defined as:

$$\text{AGSig}(x) = \frac{\alpha}{1 + \exp(-\beta x)} \tag{23}$$

where α and β are learnable parameters. Since then, numerous trainable activation functions have been proposed, many of which are extensions of traditional non-trainable functions. Notable examples include the sigmoidal selector [28] and the Flexible ReLU [29].

B. Rational activation functions

Rational Activation Functions (RAFs), introduced by [19], are defined as follows:

$$F(x) \quad = \frac{P(x)}{Q(x)} = \frac{\sum_{i=0}^{r_p} a_i x^i}{1 + \left|\sum_{j=1}^{r_q} b_j x^j\right|}, \tag{24}$$

where r_p and r_q denote the degrees of the numerator and denominator polynomials, respectively, with the constraint $r_p \geq r_q$. The coefficients a_i and b_j in the polynomials $P(x)$ and $Q(x)$ are learnable parameters, optimised during training. Incorporating such trainable components allows neural networks to more effectively capture complex features. Networks employing these activation functions are referred to as rational neural networks. As demonstrated in [19], RAFs can approximate any ReLU-based neural network. The polynomial degrees r_p and r_q serve as hyper-parameters that can be tuned for specific tasks. RAFs have shown strong performances on various benchmark tasks, including MNIST classification [19] and natural language processing architectures such as attention-based transformers [30]. To this author's knowledge, RAF have never been evaluated as activation functions in QNN.

C. Quaternion Rational Activation Functions

This subsection introduces two QRAF for QNN. The first is modelled on the split activation function structure:

$$Q_F(q) = F(r) + \mathbf{i}F(x) + \mathbf{j}F(y) + \mathbf{k}F(z) \tag{25}$$

with F a RAF defined as in (24). The second QRAF introduced has component specific RAF:

$$QCS(q) = F_1(r) + \mathbf{i}F_2(x) + \mathbf{j}F_3(y) + \mathbf{k}F_4(z) \tag{26}$$

with F_1, F_2, F_3 and F_4 RAFs defined as in (24), each having the same polynomial degrees. In the following, function (25) will be referred as QRAF and (26) as Component-Specific Quaternion Rational Activation Function (CQRAF).

D. Intra-component Interactions

As mentioned in the introduction of this paper, our goal is to introduce aspects of intra-component interactions during the quaternion learning process. This subsection provides justification for the presence of such interactions in our proposed CQRAF. Let Δ_n^l denote the gradients of the hidden layer parameters. Following [31], we have:

$$\Delta_n^l = \sum_{h=1}^{N^{l+1}} \omega_{h,n}^{*^{l+1}} \otimes (\Delta_h^{l+1} \odot \alpha'(S_n^{l+1})) \tag{27}$$

Substituting (26) into (27), we obtain:

$$\Delta_n^l = \sum_{h=1}^{N^{l+1}} \omega_{h,n}^{l+1*} \otimes \left(\Delta_h^{l+1} \odot \begin{pmatrix} F_1'\left(S_{h_\mathbf{r}}^{l+1}\right) \\ F_2'\left(S_{h_\mathbf{i}}^{l+1}\right) \\ F_3'\left(S_{h_\mathbf{j}}^{l+1}\right) \\ F_4'\left(S_{h_\mathbf{k}}^{l+1}\right) \end{pmatrix} \right) \tag{28}$$

Expanding the Hadamard product in (28), Δ_n^l can be expressed as:

$$\Delta_n^l = \sum_{h=1}^{N^{l+1}} \omega_{h,n}^{l+1*} \otimes \begin{pmatrix} (\Delta_h^{l+1})_\mathbf{r} \cdot F_1'\left(S_{h_\mathbf{r}}^{l+1}\right) \\ (\Delta_h^{l+1})_\mathbf{i} \cdot F_2'\left(S_{h_\mathbf{i}}^{l+1}\right) \\ (\Delta_h^{l+1})_\mathbf{j} \cdot F_3'\left(S_{h_\mathbf{j}}^{l+1}\right) \\ (\Delta_h^{l+1})_\mathbf{k} \cdot F_4'\left(S_{h_\mathbf{k}}^{l+1}\right) \end{pmatrix} \tag{29}$$

By expanding the Hamilton product (which involves non-trivial notation), we observe that each of the derivatives F_1', F_2', F_3', and F_4' contributes to **every** component of the resulting quaternion gradient Δ_n^l. This supports our claim that CQRAF introduces intra-component interactions within quaternion neural networks (QNNs).

E. Functional Properties and Requirements Satisfaction

This subsection briefly outlines the theoretical justifications underlying QRAF and CQRAF. [12] presents a set of requirements for activation functions, which are typically best satisfied by whole activation functions. Our proposed CQRAF fulfills several of these requirements, specifically:

1: Sufficient non-linearity,
2: Suitability for gradient-based optimization,
3: Proper gradient flow with maximized sensitivity.

All three of these criteria are satisfied by both QRAF and CQRAF. While [12] also highlights requirements that are generally unattainable by split-type activation functions, our CQRAF, under the condition that $r_p = r_q + 1$, can also satisfy a relaxed version of Requirement 5: preservation of the ratios between the respective quaternion components.

This relaxed requirement can be derived using partial fraction decomposition. Let q be a quaternion as defined in (1) with non-zero components, and let QCS denote a CQRAF defined as in (1). We can express the ratio $t = \frac{y}{x}$ and evaluate this ratio through $QCS(q)$ as follows:

$$\begin{aligned} T &= \frac{F_3(y)}{F_2(x)} = \frac{P_3(y)}{Q_3(y)} \cdot \frac{Q_2(x)}{P_2(x)} = \frac{P_3(y)}{Q_3(y)} \cdot \frac{x \cdot Q_2(x)}{x \cdot P_2(x)} \\ &= (y + S_2(y)) \cdot \frac{1}{x} \cdot (\theta + S_3(x)) \end{aligned}$$

Here, $S_2(y)$ and $S_3(x)$ represent the sum of rational functions obtained from the partial fraction decomposition over $\mathbb{R}$ of $\frac{P_3(y)}{Q_3(y)}$ and $\frac{Q_2(x)}{P_2(x)}$, respectively. The terms y and the real constant θ are the quotients resulting from the Euclidean division during decomposition. We can then simplify:

$$T = \frac{\theta y}{x} + o\left(\frac{y}{x}\right) \simeq \frac{\theta y}{x} = \theta \cdot t \tag{30}$$

In summary, QRAF and CQRAF satisfy the three primary activation function requirements described above. Furthermore, under the condition ($r_p = r_q + 1$), our CQRAF is also capable of approximately satisfying a relaxed version of the fifth requirement: the preservation of the ratio between corresponding quaternion components. This behavior underscores CQRAF's ability to maintain proportional relationships among quaternion components, a property rarely achieved by split-type activation functions according to [12].

IV. Experimental Protocol

This section describes the experimental protocol as well as the models employed during the experiments. The multiple QNNs architectures presented before all have their usual activation functions. This subsection introduces alternatives architectures based on QRAF and CQRAF introduced in subsection III-C.

A. Proposed QMLP

- QMLP$_{Relu}$ is a QMLP composed of 3 quaternion linear layers of size [4, 40], [40, 40] and [40, 4] respectively. The activation function used for each layer is the Rectified Linear Unit (ReLU). This model has 564 learnable parameters.

- QMLP$_{QRAF}$ is composed like QMLP$_{Relu}$ but with QRAF for activation functions instead of ReLU. The degrees of those QRAF are $r_p = 5$ and $r_q = 4$. This model has 584 learnable parameters.

- QMLP$_{CQRAF}$ is composed like QMLP$_{Relu}$ but with CQRAF for activation functions instead of ReLU. The degrees of those CQRAF are $r_p = 5$ and $r_q = 4$. This model has 644 learnable parameters.

B. Proposed QRNN

- QRNN$_{Tanh}$ is composed of a data normalisation layer, a QRNN with an input size of 252 and a hidden state of size 80. The activation function used is the Hyperbolic Tangent (Tanh). A linear layer of size [80, 8] with a Softmax activation function is added for classification. This model has 27380 learnable parameters.

- QRNN$_{QRAF}$ is composed like QRNN$_{Tanh}$ but with QRAF for activation functions instead of Tanh. The degrees of those QRAF are $r_p = 5$ and $r_q = 4$. This model has 27390 learnable parameters.

- QRNN$_{CQRAF}$ is composed like QRNN$_{Tanh}$ but with CQRAF for activation functions instead of Tanh. The degrees of those QRAF are $r_p = 5$ and $r_q = 4$. This model has 27540 learnable parameters.

C. Proposed QLSTM

- QLSTM$_{Tanh/Sig}$ is composed of a data normalisation layer, a QLSTM with an input size of 252 and a hidden state of size 80. The activation function used is Tanh and the Sigmoid function, as described in equations (14), (15), (16), (17) and (18). A linear layer of size [80, 8] with a Softmax activation function is added for classification. This model has 47460 learnable parameters.

- QLSTM$_{QRAF}$ is composed like QLSTM$_{Tanh/Sig}$ but with QRAF for activation functions instead of Tanh and Sigmoid. The degrees of those QRAF are $r_p = 5$ and $r_q = 4$. This model has 47480 learnable parameters.

- QLSTM$_{CQRAF}$ is composed like QLSTM$_{Tanh/Sig}$ but with CQRAF for activation functions instead of Tanh and Sigmoid. The degrees of those QRAF are $r_p = 5$ and $r_q = 4$. This model has 47540 learnable parameters.

D. Proposed QCNN

- QCNN$_{Relu}$ is a stacked QCNN designed for 4-channel input data of size (4, 32, 32). It begins with a quaternion convolutional layer ([4,6] channels, 5×5 kernel, stride 1) followed by a ReLU activation and 2×2 max pooling, reducing the spatial dimensions to [6, 14, 14]. A second quaternion convolutional layer ([6, 16] channels, 5×5 kernel) with ReLU and pooling further reduces the feature map to [16, 5, 5]. The output is flattened to a 400-dimensional vector and passed through three quaternion linear layers of sizes [400, 120], [120, 84], and [84, 40], each followed by ReLU activations except the final layer. This model has 16104 learnable parameters.

- QCNN$_{QRAF}$ is composed like QCNN$_{Relu}$ but with QRAF for activation functions instead of ReLU. The degrees of those QRAF are $r_p = 5$ and $r_q = 4$. This model has 16144 learnable parameters.

- QCNN$_{CQRAF}$ is composed like QCNN$_{Relu}$ but with CQRAF for activation functions instead of ReLU. The degrees of those QRAF are $r_p = 5$ and $r_q = 4$. This model has 16264 learnable parameters.

The choice made to set all of the QRAF and CQRAF degrees to the same $r_p = 5$ and $r_q = 4$ is motivated by [30], where it has shown promising results with transformers.

V. Experimental Setups

This section outlines the experimental setups used to compare traditional QNN architectures with QNN based on QRAF and QNN based on CQRAF. The aim of these experiments is **not** to reach state -of-the-art performances, but to compare the performances of the simple architectures introduced before. As established in [32] and [33], testing original learnable activation functions for neural networks on small architectures is common practice, and allows a better comprehension and insight on the inner workings of learnable activation functions.

A. Chaotic Function Approximation

The objective of this experiment is to compare the performances of QRAF CSQRAF and split quaternion activation functions. In this experiment, the task involves predicting the future states of a chaotic system based on its past states. The chaotic system in question is modelled by the differential equations of Chua's circuit, defined as:

$$u = (x(t), y(t), z(t))$$
$$\text{and } \frac{\partial u}{\partial x} = (\alpha \cdot (y - x - f(x)),\ x - y + z,\ -\beta \cdot y)$$
$$\text{with } f(x) = C \cdot x + \frac{1}{2}(R - C)(|x + 1| - |x - 1|)$$
$$\alpha = 15.395,\ \beta = 28,\ R = -1.143,\ \text{and } C = -0.714$$

Chua's chaotic system prediction has been used to evaluate QNNs architectures in [34]. The simulation time t spans from 0 to 10 seconds, with a time step of 1×10^{-1} seconds. Solutions to this system of differential equations were computed using the ode45 solver [35] from the SciPy Python package. The prediction task consists of using each state at even time steps $t \equiv 0\,[2] : (x_t, y_t, z_t)$ to forecast the state at the subsequent odd time step $t \equiv 1\,[2] : (x_{t+1}, y_{t+1}, z_{t+1})$.

To process these states using a quaternion-valued neural network, a zero was prepended to each state vector, resulting in the input vectors $X = (0, x_t, y_t, z_t)$ and corresponding targets $Y = (0, x_{t+1}, y_{t+1}, z_{t+1})$. The architectures used for the quaternion-valued neural network are the three QMLP described in Section IV-A. For the training process, the hyper-parameters are set as follows: training was conducted across 1000 epochs, with a learning rate set as $5e^{-5}$ and a batch size of 1. The optimiser used is SGD and the loss function employed is the L1 loss. Results are shown and discussed in VI-A.

B. Classification of Noisy Spoken Dialogues

This experiment involves a conversation classification task based on spoken dialogues from the DECODA corpus. The task focuses on the automatic analysis of telephone conversations [36] between a customer and an agent at the call centre of the Paris public transport authority (RATP) [37]. The DECODA corpus comprises 1,242 French-language agent/customer telephone conversations, representing approximately 74 hours of audio. These recordings were transcribed using the LIA-Speeral Automatic Speech Recognition (ASR) system [38] in order to preserve the noisy, real-world conditions of spoken dialogue. Each conversation was also manually transcribed and annotated with a single thematic label, chosen from a set of eight possible categories corresponding to the main topic of the exchange. The dataset is partitioned into training (730 conversations), development (171 conversations), and test (321 conversations) subsets. LIA-Speeral, a high-error ASR system, was intentionally used to maintain realistic speech conditions, thereby enabling a more effective assessment of the robustness of QRAF in the presence of noisy input during model training. A 3-gram language model (LM) is created by adapting a base LM using the transcriptions from the training set. Automatic transcriptions yield word error rates (WERs) of 33.8% on the training set, 45.2% on the development set, and 49.0% on the test set. These high error rates are primarily attributed to speech disfluencies from casual users and challenging acoustic conditions in environments such as metro stations and streets. DECODA has served as a benchmark dataset for evaluating quaternion neural network architectures. QCNN [9], QMLP [7], and Quaternion Encoder-decoder [39] all have been evaluated on DECODA's classification. The embedding size was set to 252. The architectures used for the quaternion-valued neural networks are the three QRNN described in Section IV-B and the three QLSTM described in Section IV-C. For the training process, the hyper-parameters are set as follows: training was conducted across 100 epochs, with a learning rate set as $6.25e^{-6}$ and a batch size of 8. The optimiser used is the Adam optimiser and the loss function employed is the cross-entropy. Results are shown and discussed in VI-B.

C. Image Classification

This task focuses on the CIFAR10 [40] dataset. The CIFAR-10 dataset consists of 60,000 colour images, each with a resolution of 32×32 pixels and three RGB colour channels. These images are evenly distributed across 10 distinct object classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck, with 6,000 images per class. CIFAR10 has also been previously used to test Deep Quaternion Neural Networks [41]. Since quaternions are hyper-complex numbers consisting of one real and three distinct imaginary components, quaternions are well-suited for representing three- or four-dimensional feature vectors—such as the (R, G, B) colour channels in image processing. QNN can model and learn internal dependencies between these components using the Hamilton product [2]. The architectures used for the quaternion-valued neural network are the three QCNN described in Section IV-D. As each image is coded according to (R, G, B), each pixel was coded as a pure quaternion, which is a quaternion with a zero real part:

$$q_{\text{pure}} = 0 + \mathbf{i}x + \mathbf{j}y + \mathbf{k}z \tag{31}$$

For the training process, the hyper-parameters are set as follows: training was conducted across 20 epochs, with a learning rate set as $1.00e^{-3}$ and a batch size of 4. The optimiser used is SGD and the loss function employed is the cross-entropy. Results are shown and discussed in VI-C.

VI. Results

A. Chaotic Function Approximation

TABLE I
LOSS VALUE AFTER TRAINING OF THE THREE QMLP FOR THE CHAOS FUNCTION APPROXIMATION.

Model	Loss
QMLP_{Relu}	27.209
QMLP_{QRAF}	15.608
QMLP_{CQRAF}	**9.824**

Table IV presents the final training losses of three QMLP trained to approximate a Chuas chaotic function, each differing in the activation function used. The model using standard

ReLU activation (QMLP$_{Relu}$) achieves the highest loss at 27.209 points, indicating limited effectiveness in capturing the complex dynamics of the target function. Replacing ReLU with a QRAF significantly improves performance, reducing the loss to 15.608 points. The best results are obtained with the CQRAF. Indeed, QMLP$_{CQRAF}$ yields the lowest loss of 9.824 points. Regarding the number of parameters, QMLP$_{QRAF}$ reach half the loss value of QMLP$_{Relu}$ with only 3.54% more parameters and QMLP$_{CQRAF}$ reach a third of QMLP$_{Relu}$'s loss with only 14.18% more parameters. These results highlight the importance of using component-specific activation functions, which are specifically tailored to quaternion representations. Indeed, they demonstrate that CQRAF substantially enhances the network's ability to model chaotic behaviour, outperforming both conventional and quaternion-specific alternatives. These promising results for a simple MLP architectures led us to test other types of QNN.

B. Classification of Noisy Spoken Dialogues

TABLE II
RESULTS OF THE DECODA CLASSIFICATION TASK FOR THE THREE QRNN.

Model	Train acc	Test acc	Dev acc	Loss
QRNN$_{Tanh}$	1.697	70%	69%	64%
QRNN$_{QRAF}$	1.674	71%	71%	66%
QRNN$_{CQRAF}$	**1.638**	**74%**	**74%**	**70%**

Table II reports the performances of our three QRNN on the DECODA classification task, comparing these architectures in terms of training loss and classification accuracy on the train, test, and development sets. The baseline model, QRNN$Tanh$, which uses the standard hyperbolic tangent activation, achieves a training loss of 1.697 points with corresponding accuracies of 70% (train), 69% (test), and 64% (dev). Replacing Tanh with the QRAF in QRNN$QRAF$ leads to a slight improvement across all metrics, reducing the loss to 1.674 points and achieving up to 71% accuracy. The best-performing model is QRNN$_{CQRAF}$, with the CQRAF and achieves the lowest loss with 1.638 points alongside the highest accuracies—74% on both the test and development sets, and 70% on the dev set. Regarding the number of parameters, QRNN$_{QRAF}$ and QRNN$_{CQRAF}$ have 0.036% and 0.58% more parameters of QRNN$_{Tanh}$ respectively. These results highlight the benefit of using component-specific activations as CQRAF, which consistently improves the model's generalisation and classification performances in real-world, disfluent speech scenarios of the the DECODA corpus.

TABLE III
RESULTS OF THE DECODA CLASSIFICATION TASK FOR THE THREE QLSTM.

Model	Loss	Train acc	Test acc	Dev acc
QLSTM$_{Tanh/Sig}$	**1.964**	49%	47%	39%
QLSTM$_{QRAF}$	2.068	63%	60%	47%
QLSTM$_{CQRAF}$	2.067	**66%**	**66%**	**62%**

Table III presents the performances results of three QLSTM on the DECODA classification task, each using a different activation function. The baseline model, QLSTM$_{Tanh/Sig}$, which employs the conventional Tanh and Sigmoid activations, achieves the lowest training loss at 1.964 points, but its classification accuracy is limited—49% on the training set, 47% on the test set, and 39% on the development set. In contrast, QLSTM$_{QRAF}$, which uses the QRAF, shows a modest increase in training loss at 2.068 points but significantly improves accuracy to 63%, 60%, and 47% on the train, test, and development sets, respectively. The best performance is obtained with QLSTM$_{CQRAF}$, which incorporates CQRAF, maintaining a similar loss at 2.067 points while achieving the highest accuracy across all sets: 66% on both train and test, and 62% on the development set. Regarding the number of parameters, QLSTM$_{QRAF}$ and QLSTM$_{CQRAF}$ have 0.042% and 0.16% more parameters of QLSTM$Tanh/Sig$ respectively. These results suggest that while traditional activations may minimise loss in a QLSTM framework, component-specific activations — as CQRAF—offer superior classification performance and generalisation in the context of disfluent and noisy real-world speech data.

C. Image Classification

TABLE IV
LOSS VALUE AND CLASSIFICATION ACCURACY AFTER TRAINING OF THE THREE QCNN FOR THE CIFAR10 IMAGE CLASSIFICATION.

Model	Loss	Accuracy
QCNN$_{Relu}$	650.299	56.398%
QCNN$_{QRAF}$	555.596	60.41%
QCNN$_{CQRAF}$	**517.481**	**62.09%**

Table IV presents the training loss and classification accuracy of the three QCNN introduced in IV-D evaluated on the CIFAR-10 image classification task, each employing a different activation function. The baseline model, QCNN$_{Relu}$, which uses the standard ReLU activation, shows the highest loss at 650.299 points and the lowest accuracy at 56.40%, indicating limited ability to exploit the quaternion structure of the data. The introduction of a QRAF in QCNN$_{QRAF}$ results in a notable performance boost, with a reduced loss of 555.596 points and improved accuracy of 60.41%. The best results are obtained with QCNN$_{CQRAF}$, which incorporates a QRAF and achieves the lowest loss at 517.481 and the highest classification accuracy at 62.09%. Regarding the number of parameters, QCNN$_{QRAF}$ and QCNN$_{CQRAF}$ have 0.24% and 0.99% more parameters of QCNN$Relu$ respectively. These findings confirm that quaternion-specific activations, particularly CQRAF, enhance the network's representational capacity and learning efficiency, leading to superior performance on challenging visual recognition tasks like CIFAR-10.

VII. CONCLUSIONS

Conclusions: Experiments across different tasks, such as conversation classification, image classification and chaotic function predictions have shown promising results for QRAF and CQRAF. QRAF and CQRAF have almost always outperformed the classically used activation functions, such as ReLU, Tanh and Sigmoid. Especially, CQRAF have exhibited high prediction and classifications capacity across all

the experiments. As stated in VI-A, VI-B and VI-C these better performances are reached with a negligible increase in learnable parameters for the different QNN.

Future Works: Future works will focus on other architectures and tasks, such as attention-based quaternion neural networks. These works will also investigate the impact of varying degrees of QRAF, as this paper focused on testing component-specific rational activation functions. Other works will investigate parameters reduction for QNN using QRAF, by trying to reduce the size of neural layers by leveraging QRAF. Further theoretical perspective on this is offered through Sobolev spaces. Extending the framework established by [19], we are inclined to think that certain convergence properties can be transferred to the quaternion domain by treating quaternions as elements in a 4-dimensional real space.

REFERENCES

[1] D. García-Retuerta, R. Casado-Vara, A. Martin-del Rey, F. De la Prieta, J. Prieto, and J. M. Corchado, "Quaternion neural networks: state-of-the-art and research challenges," in *International Conference on Intelligent Data Engineering and Automated Learning*, pp. 456–467, Springer, 2020.

[2] T. Parcollet, M. Morchid, and G. Linarès, "A survey of quaternion neural networks," *Artificial Intelligence Review*, vol. 53, no. 4, pp. 2957–2982, 2020.

[3] T. Parcollet, M. Morchid, and G. Linarès, "Quaternion convolutional neural networks for heterogeneous image processing," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 8514–8518, IEEE, 2019.

[4] X. Zhu, Y. Xu, H. Xu, and C. Chen, "Quaternion convolutional neural networks," in *Proceedings of the European conference on computer vision (ECCV)*, pp. 631–647, 2018.

[5] J. Pöppelbaum and A. Schwung, "Predicting rigid body dynamics using dual quaternion recurrent neural networks with quaternion attention," *IEEE Access*, vol. 10, pp. 82923–82943, 2022.

[6] D. Pavllo, C. Feichtenhofer, M. Auli, and D. Grangier, "Modeling human motion with quaternion-based neural networks," *International Journal of Computer Vision*, vol. 128, pp. 855–872, 2020.

[7] T. Parcollet, M. Morchid, P.-M. Bousquet, R. Dufour, G. Linarès, and R. De Mori, "Quaternion neural networks for spoken language understanding," in *2016 IEEE Spoken Language Technology Workshop (SLT)*, pp. 362–368, IEEE, 2016.

[8] T. Parcollet, M. Morchid, and G. Linares, "Deep quaternion neural networks for spoken language understanding," in *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pp. 504–511, IEEE, 2017.

[9] T. Parcollet, M. Ravanelli, M. Morchid, G. Linarès, C. Trabelsi, R. De Mori, and Y. Bengio, "Quaternion recurrent neural networks," *arXiv preprint arXiv:1806.04418*, 2018.

[10] X. Yang, W. Cao, Y. Lu, and Y. Zhou, "Qtn: Quaternion transformer network for hyperspectral image classification," *IEEE Transactions on Circuits and Systems for Video Technology*, 2023.

[11] N. Vieira, "Quaternionic convolutional neural networks with trainable bessel activation functions," *Complex Analysis and Operator Theory*, vol. 17, no. 6, p. 82, 2023.

[12] J. Pöppelbaum and A. Schwung, "Improving quaternion neural networks with quaternionic activation functions," *Knowledge-Based Systems*, vol. 305, p. 112619, 2024.

[13] G. Altamirano-Gómez and C. Gershenson, "Statistical analysis of the impact of quaternion components in convolutional neural networks," *arXiv preprint arXiv:2409.00140*, 2024.

[14] B. C. Ujang, C. Jahanchahi, C. C. Took, and D. Mandic, "Quaternion valued neural networks and nonlinear adaptive filters," *IEEE Trans. Neural Netw*, 2010.

[15] H. ZHANG, Q. ZHANG, and J. YU, "Overview of the development of activation function and its nature analysis," *Journal of Xihua University (Natural Science Edition)*, vol. 40, no. 4, pp. 1–10, 2021.

[16] A. Apicella, F. Donnarumma, F. Isgrò, and R. Prevete, "A survey on modern trainable activation functions," *Neural Networks*, vol. 138, pp. 14–32, 2021.

[17] J. Schmidt-Hieber, "Nonparametric regression using deep neural networks with relu activation function," 2020.

[18] S. Li and B. Gong, "Word embedding and text classification based on deep learning methods," in *MATEC Web of Conferences*, vol. 336, p. 06022, EDP Sciences, 2021.

[19] N. Boullé, Y. Nakatsukasa, and A. Townsend, "Rational neural networks," *Advances in neural information processing systems*, vol. 33, pp. 14243–14253, 2020.

[20] R. S. Palais, "The classification of real division algebras," *The American Mathematical Monthly*, vol. 75, no. 4, pp. 366–368, 1968.

[21] P. Arena, L. Fortuna, G. Muscato, and M. G. Xibilia, "Multilayer perceptrons to approximate quaternion valued functions," *Neural Networks*, vol. 10, no. 2, pp. 335–342, 1997.

[22] Y. Liu, Y. Zheng, J. Lu, J. Cao, and L. Rutkowski, "Constrained quaternion-variable convex optimization: A quaternion-valued recurrent neural network approach," *IEEE transactions on neural networks and learning systems*, vol. 31, no. 3, pp. 1022–1035, 2019.

[23] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[24] T. Parcollet, M. Morchid, G. Linarès, and R. De Mori, "Bidirectional quaternion long short-term memory recurrent neural networks for speech recognition," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 8519–8523, IEEE, 2019.

[25] B. Meng, X. Liu, and X. Wang, "Human action recognition based on quaternion spatial-temporal convolutional neural network and lstm in rgb videos," *Multimedia Tools and Applications*, vol. 77, no. 20, pp. 26901–26918, 2018.

[26] U. Onyekpe, V. Palade, S. Kanarachos, and S.-R. G. Christopoulos, "A quaternion gated recurrent unit neural network for sensor fusion," *Information*, vol. 12, no. 3, p. 117, 2021.

[27] Z. Hu, "The study of neural network control system," *Control and Decision*, vol. 7, pp. 361–366, 1992.

[28] Y. Singh and P. Chandra, "A class+ 1 sigmoidal activation functions for ffanns," *Journal of Economic Dynamics and Control*, vol. 28, no. 1, pp. 183–187, 2003.

[29] S. Qiu, X. Xu, and B. Cai, "Frelu: Flexible rectified linear units for improving convolutional neural networks," in *2018 24th international conference on pattern recognition (icpr)*, pp. 1223–1228, IEEE, 2018.

[30] H. Fang, J.-U. Lee, N. S. Moosavi, and I. Gurevych, "Transformers with learnable activation functions," *arXiv preprint arXiv:2208.14111*, 2022.

[31] T. Parcollet, M. Ravanelli, M. Morchid, G. Linarès, C. Trabelsi, R. De Mori, *et al.*, "Quaternion neural networks," in *International conference on learning representations*, 2019.

[32] A. Apicella, F. Isgro, and R. Prevete, "A simple and efficient architecture for trainable activation functions," *Neurocomputing*, vol. 370, pp. 1–15, 2019.

[33] E. Trentin, "Networks with trainable amplitude of activation functions," *Neural Networks*, vol. 14, no. 4-5, pp. 471–493, 2001.

[34] A. B. Greenblatt and S. S. Agaian, "Introducing quaternion multi-valued neural networks with numerical examples," *Information Sciences*, vol. 423, pp. 326–342, 2018.

[35] V. Leek and L. Eriksson, "Accurate simulation for numerical optimal control," *Scandinavian Simulation Society*, pp. 148–155, 2022.

[36] J. J. Godfrey, E. C. Holliman, and J. McDaniel, "Switchboard: Telephone speech corpus for research and development," in *Acoustics, Speech, and Signal Processing, 1992. ICASSP-92., 1992 IEEE International Conference on*, vol. 1, pp. 517–520, IEEE, 1992.

[37] F. Bechet, B. Maza, N. Bigouroux, T. Bazillon, M. El-Beze, R. De Mori, and E. Arbillot, "Decoda: a call-centre human-human spoken conversation corpus.," in *LREC*, pp. 1343–1347, 2012.

[38] G. Linares, P. Nocéra, D. Massonie, and D. Matrouf, "The lia speech recognition system: from 10xrt to 1xrt," in *International Conference on Text, Speech and Dialogue*, pp. 302–308, Springer, 2007.

[39] T. Parcollet, M. Morchid, and G. Linarès, "Quaternion denoising encoder-decoder for theme identification of telephone conversations," in *Interspeech 2017*, pp. 3325–3328, ISCA, 2017.

[40] A. Krizhevsky, G. Hinton, *et al.*, "Learning multiple layers of features from tiny images," 2009.

[41] C. J. Gaudet and A. S. Maida, "Deep quaternion networks," in *2018 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, IEEE, 2018.