PHNNs: Lightweight Neural Networks via Parameterized Hypercomplex Convolutions

Eleonora Grassucci, *Graduate Student Member, IEEE*, Aston Zhang, and Danilo Comminiello, *Senior Member, IEEE*

***Abstract*— Hypercomplex neural networks have proven to reduce the overall number of parameters while ensuring valuable performance by leveraging the properties of Clifford algebras. Recently, hypercomplex linear layers have been further improved by involving efficient parameterized Kronecker products. In this article, we define the parameterization of hypercomplex convolutional layers and introduce the family of parameterized hypercomplex neural networks (PHNNs) that are lightweight and efficient large-scale models. Our method grasps the convolution rules and the filter organization directly from data without requiring a rigidly predefined domain structure to follow. PHNNs are flexible to operate in any user-defined or tuned domain, from 1-D to nD regardless of whether the algebra rules are preset. Such a malleability allows processing multidimensional inputs in their natural domain without annexing further dimensions, as done, instead, in quaternion neural networks (QNNs) for 3-D inputs like color images. As a result, the proposed family of PHNNs operates with $1/n$ free parameters as regards its analog in the real domain. We demonstrate the versatility of this approach to multiple domains of application by performing experiments on various image datasets and audio datasets in which our method outperforms real and quaternion-valued counterparts. Full code is available at: https://github.com/eleGAN23/HyperNets.**

***Index Terms*— Efficient models, hypercomplex neural networks, Kronecker decomposition, lightweight neural networks, quaternions.**

I. Introduction

RECENT state-of-the-art convolutional models achieved astonishing results in various fields of application by large-scaling the overall parameters' amount [1], [2], [3], [4]. Simultaneously, hypercomplex algebra applications are gaining increasing attention in diverse spheres of research such as signal processing [5], [6], [7], [8] or deep learning [9], [10], [11], [12], [13], [14], [15], [16], [17]. Indeed, hypercomplex and quaternion neural networks (QNNs) demonstrated to significantly reduce the number of parameters while still obtaining comparable performance [18], [19], [20], [21], [22], [23], [24]. These models exploit hypercomplex algebra properties, including the Hamilton product, to painstakingly design interactions among the imaginary units, thus involving $1/4$ or $1/8$ of free parameters with respect to real-valued models. Furthermore, thanks to the modeled interactions, hypercomplex networks capture internal latent relationships in multidimensional inputs and preserve preexisting correlations among input dimensions [25], [26], [27], [28], [29]. Therefore, the quaternion domain is particularly appropriate for processing 3D or 4D data, such as color images or (up to) four-channel signals [30], while the octonion one is suitable for 8-D inputs. Unfortunately, most common color image datasets contain RGB images and some tricks are required to process this data type with QNNs. Among them, the most used are padding a zero-channel to the input to encapsulate the image in the four quaternion components, or remodeling the QNN layer with the help of vector maps [31]. In addition, while quaternion neural operations are widespread and easy to be integrated in preexisting models, very few attempts have been made to extend models to different domain orders. Accordingly, the development of hypercomplex convolutional models for larger multidimensional inputs, such as magnitudes and phases of multichannel audio signals or 16-band satellite images, still remains painful. Moreover, despite the significantly lower number of parameters, these models are often slightly slow with respect to real-valued baselines [32] and ad hoc algorithms may be necessary to improve efficiency [22], [33].

Recently, a novel literature branch aims at compress neural networks leveraging Kronecker product decomposition [34], [35], gaining considerable results in terms of model efficiency [36]. Lately, a parameterization of hypercomplex multiplications have been proposed to generalize hypercomplex fully connected (FC) layers by the sum of Kronecker products [37]. The latter method obtains high performance in various natural language processing tasks by also reducing the number of overall parameters. Other works extended this approach to graph neural networks [38] and transfer learning [39], proving the effectiveness of Kronecker product decomposition for hypercomplex operations. However, no solution exists for convolutional layers yet, which remain the most used layers when dealing with multidimensional inputs, such as images and audio signals [40], [41].

In this article, we devise the family of parameterized hypercomplex neural networks (PHNNs), which are lightweight

Manuscript received 14 February 2022; revised 19 September 2022 and 27 November 2022; accepted 2 December 2022. This work was supported by "Progetti di Ricerca" of Sapienza University of Rome under Grant RG11916B88E1942F. *(Corresponding author: Eleonora Grassucci.)*

Eleonora Grassucci and Danilo Comminiello are with the Department of Information Engineering, Electronics and Telecommunications (DIET), Sapienza University of Rome, 00185 Roma, Italy (e-mail: eleonora.grassucci@uniroma1.it).

Aston Zhang is with the Amazon Web Services AI, East Palo Alto, CA 94303 USA.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2022.3226772.

Digital Object Identifier 10.1109/TNNLS.2022.3226772

large-scale hypercomplex neural models admitting any multidimensional input, whichever the number of dimensions. At the core of this novel set of models, we propose the parameterized hypercomplex convolutional (PHC) layer. Our method is flexible to operate in domains from 1-D to nD, where n can be arbitrarily chosen by the user or tuned to let the model performance lead to the most appropriate domain for the given input data. Such a malleability comes from the ability of the proposed approach to subsume algebra rules to perform convolution regardless of whether these regulations are preset or not. Thus, neural models endowed with our approach adopt $1/n$ of free parameters with respect to their real-valued counterparts, and the amount of parameter reduction is a user choice. This makes PHNNs adaptable to a plethora of applications in which saving storage memory can be a crucial aspect. In addition, PHNNs' versatility allows processing multidimensional data in its natural domain by simply setting the dimensional hyperparameter n. For instance, color images can be analyzed in their RGB domain by setting $n = 3$ without adding any useless information, contrary to standard processing for quaternion networks with the padded zero-channel. Indeed, PHC layers are able to grasp the proper algebra from input data, while capturing internal correlations among the image channels and saving 66% of free parameters.

On a thorough empirical evaluation on multiple benchmarks, we demonstrate the flexibility of our method that can be adopted in different domains of applications, from images to audio signals. We devise a set of PHNNs for large-scale image classification and sound event detection (SED) tasks, letting them operate in different hypercomplex domain and with various input dimensionality with n ranging from 2 to 16.

The contribution of this article is threefold.

1) We introduce a PHC layer which grasps the convolution rules directly from data via backpropagation exploiting the Kronecker product properties, thus reducing the number of free parameters to $1/n$.
2) We devise the family of PHNNs, lightweight and more efficient large-scale hypercomplex models. Thanks to the proposed PHC layer and to the method in [37] for FC layers, PHNNs can be used with any kind of input and preexisting neural models. To show the latter, we redefine common ResNets, VGGs, and SED networks (SEDnets), operating in any user-defined domain just by choosing the hyperparameter n, which also drives the number of convolutional filters.
3) We show how the proposed approach can be used with any kind of multidimensional data by easily changing the hyperparameter n. Indeed, by setting $n = 3$, a PHNN can process RGB images in their natural domain, while leveraging the properties of hypercomplex algebras, allowing parameter sharing inside the layers and leading to a parameter reduction to 1/3. To the best of our knowledge, this is the first approach that processes color images with hypercomplex-based neural models without adding any padding channel. In addition, multichannel audio signals can be analyzed by simply considering $n = 4$ for standard first-order ambisonics (FOA) (which has four microphone capsules), $n = 8$ for an array of two ambisonics microphones, or even $n = 16$ if we want to include the information of each channel phase.

$n = 2$ $n = 3$ $n = 4$ $n = 5$ $n = 6$ $n = 7$ $n = 8$

$\times$	1	i_1	i_2	i_3	i_4	i_5	i_6	i_7
1	1	i_1	i_2	i_3	i_4	i_5	i_6	i_7
i_1	i_1	-1	i_3	$-i_2$	i_5	$-i_4$	$-i_7$	i_6
i_2	i_2	$-i_3$	-1	i_1	i_6	i_7	$-i_4$	$-i_5$
i_3	i_3	i_2	$-i_1$	-1	i_7	$-i_6$	i_5	$-i_4$
i_4	i_4	$-i_5$	$-i_6$	$-i_7$	-1	i_1	i_2	i_3
i_5	i_5	i_4	$-i_7$	i_6	$-i_1$	-1	$-i_3$	i_2
i_6	i_6	i_7	i_4	$-i_5$	$-i_2$	i_3	-1	$-i_1$
i_7	i_7	$-i_6$	i_5	i_4	$-i_3$	$-i_2$	i_1	-1

Fig. 1. Example of hypercomplex multiplication table for $n = 2$, i.e., complex, among others (green line), $n = 4$, i.e., quaternions, tessarines (blue line), and $n = 8$, i.e., octonions, biquaternions, and so on (red line). While for these domains algebra rules exist and are predefined, no regulations are set for other domains such as $n = 3, 5, 6, 7$ (dashed gray lines). The parameterized hypercomplex approaches are able to learn these missing algebra rules from data, thus defining hypercomplex multiplication and convolution for any desired domain.

The rest of the article is organized as follows. In Section II, we introduce concepts of hypercomplex algebra and we recapitulate real- and quaternion-valued convolutional layers. Section III rigorously introduces the theoretical aspects of the proposed method. Sections IV and V reveal how the approach can be adopted in different neural models and in two different domains, the images and audio one, expounding how to process RGB images with $n = 3$ and multichannel audio with n up to 8. The experimental evaluation is presented in Section VI for image classification and in Section VII for SED. Finally, Section VIII reports the ablation studies we conduct, and in Section IX we draw conclusions.

II. Hypercomplex Neural Networks

A. Hypercomplex Algebra

Hypercomplex neural networks rely on a hypercomplex number system based on the set of hypercomplex numbers $\mathbb{H}$ and their corresponding algebra rules to shape additions and multiplications [24]. These operations should be carefully modeled due to the interactions among imaginary units that may not behave as real-valued numbers. For instance, Fig. 1 reports an example of a multiplication table for complex (green), quaternion (blue), and octonion (red) numbers. However, this is just a small subset of the hypercomplex domain that exists. Indeed, for $n = 4$, there exist quaternions, tessarines, among others, while for $n = 8$ octonions, dual-quaternions, and so on. Each of these domains has different multiplication rules due to dissimilar imaginary units interactions. A generic hypercomplex number is defined as

$$h = h_0 + h_i \hat{\imath}_i + \cdots + h_n \hat{\imath}_n, \quad i = 1, \ldots, n \tag{1}$$

being $h_0, \ldots, h_n \in \mathbb{R}$ and $\hat{\imath}_i, \ldots, \hat{\imath}_n$ imaginary units. Different subsets of the hypercomplex domain exist, including complex, quaternion, and octonion, among others. They are identified by the number of imaginary units they use and by the properties of their vector multiplication. The quaternion domain is one of

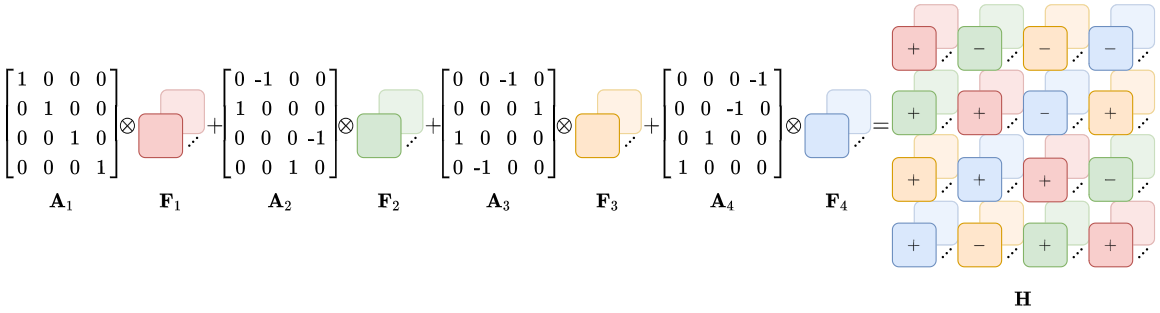


Fig. 2. Quaternion convolution rule can be expressed as the sum of Kronecker products between the matrices $\mathbf{A}_i$ that subsume the algebra rules and the matrices $\mathbf{F}_i$ that contain the convolution filters, with $i = 1, 2, 3, 4$. In this example, the parameters of $\mathbf{A}_i$ are fixed for visualization purposes, but in PHC layers they are learnable parameters.

the most popular for neural networks, thanks to the Hamilton product properties. This domain has its foundations in the quaternion number $q = q_0 + q_1\hat{\imath} + q_2\hat{\jmath} + q_3\hat{\kappa}$, in which $q_c,\ c \in \{0, 1, 2, 3\}$ are real coefficients and $\hat{\imath}, \hat{\jmath}, \hat{\kappa}$ are the imaginary units. A quaternion with its real part q_0 equal to 0 is named *pure quaternion*. The imaginary units comply with the property $\hat{\imath}^2 = \hat{\jmath}^2 = \hat{\kappa}^2 = -1$ and with the noncommutative products $\hat{\imath}\hat{\jmath} = -\hat{\jmath}\hat{\imath};\ \hat{\jmath}\hat{\kappa} = -\hat{\kappa}\hat{\jmath};\ \hat{\kappa}\hat{\imath} = -\hat{\imath}\hat{\kappa}$. Due to the noncommutativity of vector multiplication, the Hamilton product has been introduced to properly model the multiplication between two quaternions.

B. Real- and Quaternion-Valued Convolutional Layers

A generic convolutional layer can be described by

$$\mathbf{y} = \text{Conv}(\mathbf{x}) = \mathbf{W} * \mathbf{x} + \mathbf{b} \tag{2}$$

where the input $\mathbf{x} \in \mathbb{R}^{t \times s}$ is convolved $(*)$ with the filters' tensor $\mathbf{W} \in \mathbb{R}^{s \times d \times k \times k}$ to produce the output $\mathbf{y} \in \mathbb{R}^{d \times t}$, where s is the input channels' dimension, d is the output one, k is the filter size, and t is the input and output dimension. The bias term $\mathbf{b}$ does not heavily influence the number of parameters, and thus, the degrees of freedom for this operation are essentially $\mathcal{O}(sdk^2)$.

Quaternion convolutional layers, instead, build the weight tensor $\mathbf{W} \in \mathbb{R}^{s \times d \times k \times k}$ by following the Hamilton product rule and organize filters according to it:

$$\mathbf{W} * \mathbf{x} = \begin{bmatrix} \mathbf{W}_0 & -\mathbf{W}_1 & -\mathbf{W}_2 & -\mathbf{W}_3 \\ \mathbf{W}_1 & \mathbf{W}_0 & -\mathbf{W}_3 & \mathbf{W}_2 \\ \mathbf{W}_2 & \mathbf{W}_3 & \mathbf{W}_0 & -\mathbf{W}_1 \\ \mathbf{W}_3 & -\mathbf{W}_2 & \mathbf{W}_1 & \mathbf{W}_0 \end{bmatrix} * \begin{bmatrix} \mathbf{x}_0 \\ \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \end{bmatrix} \tag{3}$$

where $\mathbf{W}_0, \mathbf{W}_1, \mathbf{W}_2, \mathbf{W}_3 \in \mathbb{R}^{(s/4) \times (d/4) \times k \times k}$ are the real coefficients of the quaternion weight matrix $\mathbf{W} = \mathbf{W}_0 + \mathbf{W}_1\hat{\imath} + \mathbf{W}_2\hat{\jmath} + \mathbf{W}_3\hat{\kappa}$ and $\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$ are the coefficients of the quaternion input $\mathbf{x}$ with the same structure.

As done for real-valued layers, the bias can be ignored and the degree of freedom computations of the quaternion convolutional layer can be approximated to $\mathcal{O}(sdk^2/4)$. The lower number of parameters with respect to the real-valued operation is due to the reuse of filters performed by the Hamilton product in (3). Also, sharing the parameter submatrices forces to consider and exploit the correlation between the input components [21], [42], [43].

III. Parameterizing Hypercomplex Convolutions

In the following, we delineate the formulation for the proposed PHC layer. We also show that this approach is capable of learning the Hamilton product rule when two quaternions are convolved.

A. Parameterized Hypercomplex Convolutional Layers

The PHC layer is based on the construction, by the sum of Kronecker products, of the weight tensor $\mathbf{H}$ which encapsulates and organizes the filters of the convolution. The proposed method is formally defined as

$$\mathbf{y} = \text{PHC}(\mathbf{x}) = \mathbf{H} * \mathbf{x} + \mathbf{b} \tag{4}$$

whereby $\mathbf{H} \in \mathbb{R}^{s \times d \times k \times k}$ is built by the sum of Kronecker products between two learnable groups of matrices. Here, s is the input dimensionality to the layer, d is the output one, and k is the filter size. More concretely

$$\mathbf{H} = \sum_{i=1}^{n} \mathbf{A}_i \otimes \mathbf{F}_i \tag{5}$$

in which $\mathbf{A}_i \in \mathbb{R}^{n \times n}$ with $i = 1, \ldots, n$ are the matrices that describe the algebra rules, and $\mathbf{F}_i \in \mathbb{R}^{(s/n) \times (d/n) \times k \times k}$ represents the ith batch of filters that are arranged by following the algebra rules to compose the final weight matrix. It is worth noting that $(s/n) \times (d/n) \times k \times k$ holds for squared kernels, while $(s/n) \times (d/n) \times k$ should be considered instead for 1-D kernels. The core element of this approach is the Kronecker product [44], which is a generalization of the vector outer product that can be parameterized by n. The hyperparameter n can be set by the user who wants to operate in a predefined real or hypercomplex domain (e.g., by setting $n = 2$, the PHC layer is defined in the complex domain, or in the quaternion one if n is set equal to 4, as Fig. 2 illustrates), or tuned to obtain the best performance from the model. The matrices $\mathbf{A}_i$ and $\mathbf{F}_i$ are learned during training and their values are reused to build the definitive tensor $\mathbf{H}$.

The degrees of freedom of $\mathbf{A}_i$ and $\mathbf{F}_i$ are n^3 and sdk^2/n, respectively. Usually, real-world applications use a large number of filters in layers $(s, d = 256, 512, \ldots)$ and small values for k. Therefore, frequently $sdk^2 \gg n^3$ holds. Thus, the degrees of freedom for the PHC weight matrix can be approximated to $\mathcal{O}(sdk^2/n)$. Hence, the PHC layer reduces the number of parameters by $1/n$ with respect to a standard convolutional layer in real-world problems.

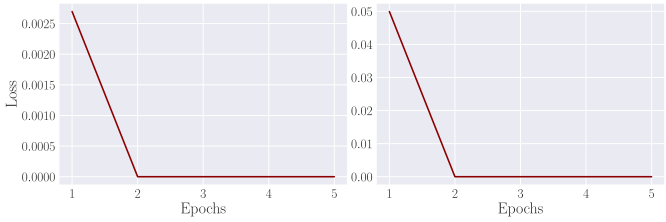


Fig. 3. Loss plots for toy examples. The PHC layer is able to learn the matrix $\mathbf{A}$ describing the convolution rule for pure (left) and full quaternions (right).

Moreover, when processing multidimensional data with correlated channels, such as color images, rather than mulichannel audio or multisensor signals, PHC layers bring benefits due to the weight sharing among different channels. This allows capturing latent intrachannels relationships that standard convolutional networks ignore because of the rigid structure of the weights [20], [45]. The PHC layer is able to subsume hypercomplex convolution rules, and the desired domain is specified by the hyperparameter n. Interestingly, by setting $n = 1$, a real-valued convolutional layer can be represented too. Indeed, standard real layers do not involve parameter sharing, and therefore, the algebra rules are solely described by the single $\mathbf{A} \in \mathbb{R}^{1\times 1}$ and the complete set of filters are included in $\mathbf{F}^{s\times d\times k\times k}$.

Therefore, the PHC layer fills the gaps left by preexisting hypercomplex algebras in Fig. 1 and subsumes the missing algebra rules directly from data, i.e., the dashed gray lines in Fig. 1. Thus, a neural model equipped with PHC layers can grasp the filter organization also for $n = 3, 5, 6, 7$, and so on. Moreover, any convolutional model can be endowed with our approach, since PHC layers easily replace standard convolution/transposed convolution operations and the hyperparameter n gives high flexibility to adapt the layer to any kind of input, such as color images, multichannel audio, or multisensor signals.

B. Learning Tests on Toy Examples

We test the receptive ability of the PHC layer in two toy problems building an artificial dataset. We highly encourage the reader to take a look at the section `tutorials` of the GitHub repository https://github.com/eleGAN23/HyperNets for more insights and results on toy examples, including the learned matrices $\mathbf{A}_i$. The first task aims at learning the right matrix $\mathbf{A}$ to build a quaternion convolutional layer which properly follows the Hamilton rule in (3). That is, we set $n = 4$, and the objective is to learn the four matrices $\mathbf{A}_i$ as they are in the quaternion product in Fig. 2. We build the dataset by performing a convolution with a matrix of filters $\mathbf{W} \in \mathbb{H}$, which are arranged following the regulation in (3), and a quaternion $\mathbf{x} \in \mathbb{H}$ in input. The target is still a quaternion, named $\mathbf{y} \in \mathbb{H}$. As shown in Fig. 3 (right), the MSE loss of the PHC layer converges very fast, meaning that the layer properly learns the matrix $\mathbf{A}$ and the Hamilton convolution.

The second toy example is a modification of the previous dataset target. Here, we want to learn the matrix $\mathbf{A}$ which describes the convolution among two pure quaternions.

$$
\begin{aligned}
\underset{(1\times 1)}{[A]} \otimes \underset{(s\times d\times k\times k)}{\begin{bmatrix} \mathbf{F} \end{bmatrix}} &= \underset{(s\times d\times k\times k)}{\begin{bmatrix} \mathbf{H} \end{bmatrix}} \\
\underset{(2\times 2)}{[\mathbf{A}_1]} \otimes \underset{(\frac{s}{2}\times\frac{d}{2}\times k\times k)}{\begin{bmatrix} \mathbf{F}_1 \end{bmatrix}} + \underset{(2\times 2)}{[\mathbf{A}_2]} \otimes \underset{(\frac{s}{2}\times\frac{d}{2}\times k\times k)}{\begin{bmatrix} \mathbf{F}_2 \end{bmatrix}} &= \underset{(s\times d\times k\times k)}{\begin{bmatrix} \mathbf{H} \end{bmatrix}} \\
&\vdots \\
\underset{(n\times n)}{[\mathbf{A}_1]} \otimes \underset{(\frac{s}{n}\times\frac{d}{n}\times k\times k)}{\begin{bmatrix} \mathbf{F}_1 \end{bmatrix}} + \underset{(n\times n)}{[\mathbf{A}_2]} \otimes \underset{(\frac{s}{n}\times\frac{d}{n}\times k\times k)}{\begin{bmatrix} \mathbf{F}_2 \end{bmatrix}} + \cdots + \underset{(n\times n)}{[\mathbf{A}_n]} \otimes \underset{(\frac{s}{n}\times\frac{d}{n}\times k\times k)}{\begin{bmatrix} \mathbf{F}_n \end{bmatrix}} &= \underset{(s\times d\times k\times k)}{\begin{bmatrix} \mathbf{H} \end{bmatrix}}
\end{aligned}
\tag{6}
$$

Therefore, when setting $n = 4$, the matrix $\mathbf{A}_1$ of a pure quaternion should be complete null. Pure quaternions may be, as an example, an input RGB image and the weights of a hypercomplex convolutional layer since the first channel of RGB images is zero. Fig. 3 (left) displays the convergence of the PHC layer loss during training, proving that the proposed method is able of subsuming hypercomplex convolutional rules when dealing with pure quaternions too (6), as shown at the bottom of the previous page.

C. Demystifying PHC Layers

We provide a formal explanation of the PHC layer to better understand the Kronecker product and how it organizes convolution filters to reduce the overall number of parameters to $1/n$. In (6), we show how the PHC layer generalizes from 1-D to nD domains. When subsuming real-valued convolutions in the first line of (6), the Kronecker product is performed between a scalar A and the filter matrix $\mathbf{F}$, whose dimension is the same as the final weight matrix $\mathbf{H}$, which is $s \times d \times k \times k$.

Considering the complex case with $n = 2$ in the second line of (6), the algebra is defined in $\mathbf{A}_1$ and $\mathbf{A}_2$ while the filters are contained in $\mathbf{F}_1$ and $\mathbf{F}_2$, each of dimension $1/2$ the final matrix $\mathbf{H}$. Therefore, while the size of the weight matrix $\mathbf{H}$ remains unchanged, the parameter size is approximately $1/2$ the real one. In the last line of (6), we can see the generalization of this process, in which the size of matrices $\mathbf{F}_i$, $i = 1, \ldots, n$ is reduced proportionally to n. It is worth noting that while the parameter size is reduced with growing values of n, the dimension of $\mathbf{H}$ remains the same.

IV. Parameterized Hypercomplex Neural Networks for Color Images

In this section, we describe how PHNNs can be applied for processing color images in hypercomplex domains without needing any additional information to the input and we propose examples of parameterized hypercomplex versions of common computer vision models such as VGGs and ResNets. To be consistent with literature, we perform each experiment with a real-valued baseline model, and then we compare it with its complex and quaternion counterparts and with the proposed PHNN. Furthermore, we assess the malleability of the proposed approach testing different values of the hyperparameter n, therefore defining parameterized hypercomplex models in multiple domains.

A. Process Color Images With PHC Layers

Different encodes exist to process color images; however, the most common computer vision datasets comprise three-channel images in $\mathbb{R}^3$. In the quaternion domain, RGB images are enclosed into a quaternion and processed as single elements [42]. The encapsulation is performed by considering the RGB channels as the real coefficients of the imaginary units and by padding a zeros channel as the first real component of the quaternion.

Here, we propose to leverage the high malleability of PHC layers to deal with RGB images in hypercomplex domains without embedding useless information to the input. Indeed, the PHC can directly operate in $\mathbb{R}^3$ by easily setting $n = 3$ and process RGB images in their natural domain while exploiting hypercomplex network properties such as parameters' sharing. Indeed, the great flexibility of PHC layers allows the user to choose whether processing images in $\mathbb{R}^4$ or $\mathbb{R}^3$. On one hand, by setting $n = 4$, the zero-channel is added to the input even so the layer saves the 75% of free parameters. On the other hand, by choosing $n = 3$, the network does not handle any useless information' notwithstanding, it reduces the number of parameters by solely 66%. This is a tradeoff which may depend on the application or on the hardware the user needs. Furthermore, the domain on which processing images can be tuned by letting the performance of the network indicates the best choice for n.

B. Parameterized Hypercomplex VGGs

A family of popular methods for image processing is based on the VGG networks [46] that stack several convolutional layers and a closing FC classifier. To completely define models in the desired hypercomplex domain, we propose to endow the network with PHC layers as convolution components and with parameterized hypercomplex multiplication (PHM) layers [37] as linear classifier. The backbone of our PHVGG is then

$$\begin{aligned} \mathbf{h}_t &= \mathrm{ReLU}(\mathrm{PHC}_t(\mathbf{h}_{t-1})), \quad t = 1, \ldots, j \\ \mathbf{y} &= \mathrm{ReLU}\big(\mathrm{PHM}(\mathbf{h}_j)\big). \end{aligned} \tag{7}$$

C. Parameterized Hypercomplex ResNets

In recent literature, a copious set of high performance in image classification is obtained with models having a residual structure. ResNets [47] pile up manifold residual blocks composed of convolutional layers and identity mappings. A generic PHResNet residual block is defined by

$$\mathbf{y} = \mathcal{F}(\mathbf{x}, \{\mathbf{H}_j\}) + \mathbf{x} \tag{8}$$

whereby $\mathbf{H}_j$ are the PHC weights of layer $j = 1, 2$ in the block, and $\mathcal{F}$ is

$$\mathcal{F}(\mathbf{x}, \{\mathbf{H}_j\}) = \mathrm{PHC}(\mathrm{ReLU}(\mathrm{PHC}(\mathbf{x}))) \tag{9}$$

in which we omit batch normalization to simplify notation. The backward phase of a PHNN reduces to a backpropagation similar to the QNN one, which has been already developed in [42], [48], and [19].

V. Parameterized Hypercomplex Neural Networks for Multichannel Signals

In the following, we expound how PHNNs can be used to deal with multichannel audio signals and we introduce, as an example, the parameterized hypercomplex SEDnets (PHSEDnets).

A. Process Multichannel Audio With PHC Layers

An FOA signal is composed of four microphone capsules, whose magnitude representations can be enclosed in a quaternion [49], [50]. However, the quaternion algebra may be restrictive if more than one microphone is used for registration or whether the phase information has to be included too. Indeed, QNNs badly fit with multidimensional input with more than four channels [51].

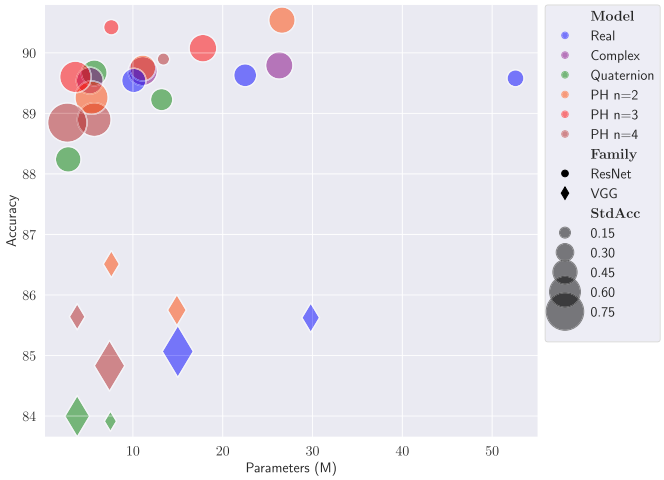


Fig. 4. CIFAR10 accuracy against the number of network parameters for the VGG and ResNet models. The larger the point, the higher the standard deviation over the runs. PHC-based models obtain better accuracies in both the families while far reducing the number of parameters. We do not display complex VGGs as their accuracy is very low with respect to other models.

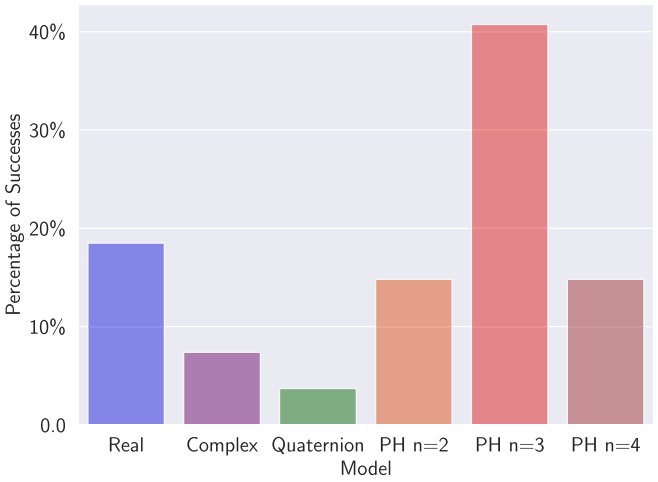


Fig. 5. Bar plot of the number of successes achieved by the models in Table II in each of the runs. The PHC-based models with $n = 3$ (red bar) far exceeds other configurations being the more performing choice for the RGB image classification task.

Conversely, the proposed method can be easily adapted to deal with these additional dimensions by handily setting the hyperparameter n and thus completely leveraging each information in the n-dimensional input.

B. Parameterized Hypercomplex SEDnets

SEDnets [52] comprise a core convolutional component which extracts features from the input spectrogram. The information is then passed to a gated recurrent unit (GRU) module and to a stack of FC layers with a closing sigmoid σ which outputs the probability the sound is in the audio frame. Formally, the PHSEDnet is described by

$$\begin{aligned} \mathbf{h}_t &= \mathrm{PHC}_t(\mathbf{h}_{t-1}), \quad t = 1, \ldots, j \\ \mathbf{y} &= \sigma\big(\mathrm{FC}\big(\mathrm{GRU}\big(\mathbf{h}_j\big)\big)\big). \end{aligned} \tag{10}$$

After the GRU model, we employ standard FC layers, which can also be implemented as PHM layers with $n = 1$, since the so-processed signal loses its multidimensional original structure.

VI. Experimental Evaluation on Image Classification

To begin with, we test the PHC layer on RGB images and we show how, exploiting the correlations among channels, the proposed method saves parameters while ensuring high performance. We perform each experiment with a real-valued baseline model and then we compare it with its complex and quaternion counterparts and with the proposed PHNNs. Furthermore, we assess the malleability of the proposed approach testing different values of the hyperparameter n, therefore defining parameterized hypercomplex models in multiple domains.

A. Experimental Setup

We perform the image classification task with five baseline models. We consider ResNet18, ResNet50, and ResNet152 from the ResNet family and VGG16 and VGG19 from the VGG one. Each hyperparameter is set according to the original articles [47] and [46]. We investigate the performance in four different color images datasets at different scales. We use SVHN, CIFAR10, CIFAR100, and ImageNet, and any kind of data augmentation is applied to these datasets to guarantee a fair comparison.

We modify the number of filters for ResNets to be divisible by 3 and thus having the possibility of testing a configuration with $n = 3$. The modified versions of ResNets are built with an initial convolutional layer of 60 filters. Then, the subsequent blocks have 60, 120, 240, and 516 filters. The number of layers in the blocks depends on the ResNet chosen, whether 18, 50, or 152. Instead, VGG19 convolution component comprises two 24, two 72, four 216, and eight 648 filter layers, with batch normalization. The classifier is composed of three FC layers of 648, 516 and 10, 100, or 1000 depending on the number of classes in the dataset. The rest of the hyperparameters are set as suggested in the original articles. The batch size is fixed to 128 and training is performed via the stochastic gradient descent (SGB) optimizer with momentum equal to 0.9, weight decay $5e^{-4}$, and a cosine annealing scheduler. For ResNets, the initial learning rate is set to 0.1. For VGG, it is equal to 0.01. Models on CIFAR10 and CIFAR100 are trained for 200 epochs, whereas on SVHN networks run for 50 epochs. For the ImageNet dataset, we follow the recipes in [53], so we resize the images for training at 160×160 while keeping the standard size of 224×224 for validation and test. We use a step learning rate decay every 30 epochs with $\gamma = 0.1$, the SGD optimizer, and an initial learning rate of 0.1 with weight decay 0.0001. The training is performed for 300k iterations with a batch size of 256 using four Tesla V100 GPUs.

B. Experimental Results

We execute the initial experiments with VGGs against Quaternion VGGs and two versions of PHVGGs with n equal to 2 and 4. The average and standard deviation accuracy over three runs are reported for the SVHN and CIFAR10

TABLE I

IMAGE CLASSIFICATION RESULTS FOR VGG. THE ACCURACY MEAN AND STANDARD DEVIATION OVER THREE RUNS WITH DIFFERENT SEEDS ARE REPORTED. TRAINING (T) TIME AND INFERENCE (I) TIME REQUIRED ON CIFAR10. FOR TRAINING TIME WE REPORT, IN SECONDS/100 ITERATIONS, THE MEAN AND THE STANDARD DEVIATION OVER THE ITERATIONS IN ONE EPOCH, WHILE THE INFERENCE TIME IS THE TIME REQUIRED TO DECODE THE TEST SET. THE PHNN WITH $n = 4$ OUTPERFORMS THE QUATERNION COUNTERPART BOTH IN TERMS OF ACCURACY AND TIME. PHVGG WITH $n = 2$ FAR EXCEEDS THE REAL-VALUED BASELINE IN THE CONSIDERED DATASETS, WHILE BOTH THE PHVGG19 VERSIONS WITH $n = 2, 4$ ARE MORE EFFICIENT THAN THE REAL AND QUATERNION-VALUED BASELINES AT INFERENCE TIME. p-VALUE UNDER THE T-TEST 0.0002

Model	Params	SVHN	CIFAR10	Time (T)	Time (I)
VGG16	15M	94.364 ± 0.394	85.067 ± 0.765	**2.2 ± 0.02**	**1.2**
Complex VGG16	7.6M (-50%)	93.555 ± 0.392	76.927 ± 0.511	5.2 ± 0.02	1.5
Quaternion VGG16	3.8M (-75%)	93.887 ± 0.292	83.997 ± 0.493	5.2 ± 0.02	2.2
PHVGG16 $n = 2$	7.6M (-50%)	**94.831 ± 0.257**	**86.510 ± 0.216**	3.2 ± 0.02	1.4
PHVGG16 $n = 4$	3.8M (-75%)	94.639 ± 0.121	85.640 ± 0.205	3.2 ± 0.02	1.4
VGG19	29.8M	94.140 ± 0.129	85.624 ± 0.257	**3.2 ± 0.02**	16.0
Complex VGG19	14.8M (-50%)	90.469 ± 0.222	76.979 ± 0.345	5.2 ± 0.02	16.2
Quaternion VGG19	7.5M (-75%)	93.983 ± 0.190	83.914 ± 0.129	6.2 ± 0.02	16.3
PHVGG19 $n = 2$	14.9M (-50%)	**94.553 ± 0.229**	**85.750 ± 0.286**	4.0 ± 0.02	**15.4**
PHVGG19 $n = 4$	7.4M (-75%)	94.169 ± 0.296	84.830 ± 0.733	4.2 ± 0.02	15.5

TABLE II

IMAGE CLASSIFICATION RESULTS WITH THE RESNET MODELS. EACH EXPERIMENT IS RUN THREE TIMES WITH DIFFERENT SEEDS AND MEAN WITH STANDARD DEVIATION IS REPORTED. THE PROPOSED MODELS FAR EXCEED REAL-VALUED AND QUATERNION BASELINES ALMOST IN EACH EXPERIMENT WE CONDUCT. INTERESTINGLY, THE PHNN OUTPERFORMS THE REAL-VALUED COUNTERPART BY 4% POINTS IN THE LARGEST SCALE EXPERIMENT ON CIFAR100. THE TIME IS SIMILAR TO THE CLAIMS IN TABLE I SO WE DO NOT ADD HERE TO AVOID REDUNDANCY

Model	Params	Storage Memory	FLOPs	SVHN	CIFAR10	CIFAR100
ResNet18	10.1M	39MB	1.01G	93.992 ± 1.317	89.543 ± 0.340	62.634 ± 0.600
Complex ResNet18	5.2M (-50%)	20MB (-50%)	1.01G	89.902 ± 0.322	89.541 ± 0.412	60.417 ± 0.811
Quaternion ResNet18	2.8M (-75%)	10MB (-75%)	1.03G	93.661 ± 0.413	88.240 ± 0.377	59.850 ± 0.607
PHResNet18 $n = 2$	5.4M (-50%)	20MB (-50%)	1.03G	**94.359 ± 0.187**	89.260 ± 0.625	60.320 ± 2.249
PHResNet18 $n = 3$	3.6M (-66%)	13MB (-66%)	1.03G	94.303 ± 1.234	**89.603 ± 0.563**	**62.660 ± 1.067**
PHResNet18 $n = 4$	2.7M (-75%)	10MB (-75%)	1.03G	94.234 ± 0.161	88.847 ± 0.874	61.780 ± 0.689
ResNet50	22.5M	86MB	2.36G	94.546 ± 0.269	89.630 ± 0.305	65.514 ± 0.569
Complex ResNet50	11.1M (-50%)	43MB (-50%)	2.36G	89.004 ± 0.215	89.699 ± 0.485	65.104 ± 0.598
Quaternion ResNet50	5.7M (-75%)	22MB (-75%)	2.41G	93.685 ± 0.389	89.670 ± 0.383	63.760 ± 0.717
PHResNet50 $n = 2$	11.1M (-50%)	43MB (-50%)	2.41G	93.849 ± 0.249	89.750 ± 0.386	65.884 ± 0.333
PHResNet50 $n = 3$	7.6M (-66%)	29MB (-65%)	2.41G	93.617 ± 0.497	**90.423 ± 0.145**	**66.497 ± 1.256**
PHResNet50 $n = 4$	5.7M (-75%)	23MB (-74%)	2.41G	**94.558 ± 0.754**	88.897 ± 0.645	66.240 ± 1.165
ResNet152	52.6M	201MB	6.62G	**94.625 ± 0.355**	89.580 ± 0.173	62.053 ± 0.385
Complex ResNet152	26.3M (-50%)	101MB (-50%)	6.62G	90.332 ± 0.129	89.792 ± 0.427	63.125 ± 0.681
Quaternion ResNet152	13.2M (-75%)	51MB (-75%)	6.62G	93.638 ± 0.098	89.227 ± 0.287	61.267 ± 0.784
PHResNet152 $n = 2$	26.6M (-50%)	103MB (-49%)	6.76G	93.915 ± 0.512	**90.540 ± 0.401**	65.817 ± 0.327
PHResNet152 $n = 3$	17.8M (-66%)	70MB (-65%)	6.76G	93.955 ± 0.152	90.077 ± 0.436	66.347 ± 0.567
PHResNet152 $n = 4$	13.4M (-75%)	53 MB (-74%)	6.76G	94.290 ± 0.237	89.897 ± 0.097	**66.437 ± 0.064**

datasets in Table I. We experiment also additional runs but any significant difference emerges as the randomness only affects the network initialization. Both the PHVGG16 and PHVGG19 versions clearly outperform real, complex, and quaternion counterparts while being built with more than a half the number of parameters of the baseline. In addition, the PH-based models extraordinarily reduce the number of training and inference time (computed on an NVIDIA Tesla V100) required with respect to the quaternion model which operates in a hypercomplex domain as well. Furthermore, when scaling up the experiment with VGG19, the proposed methods are more efficient at inference time with respect to the real-valued VGG19. Therefore, PHNNs can be easily adopted in applications with disk memory limitations, due to the reduction in parameters, and for fast inference problems, thanks to the efficiency at testing time. Although the sum of Kronecker products in PHC layers requires additional computations, the increase is insignificant with respect to the FLOPs computed for the whole network, so the overall number of FLOPs is not heavily affected by our method and the count remains almost the same.

Our approach has high malleability, indeed, when dealing with color images, we can the domain in which operating thanks to the hyperparameter n. Therefore, we test PHNNs in the complex ($n = 2$), quaternion ($n = 4$), or $\mathbb{H}^3$ ($n = 3$) domain, where in the latter we do not concatenate any zero padding and process the RGB channels of the image in their natural domain.

Table II presents the average and standard deviation accuracy over three runs with different seeds for the ResNet-based models. We perform extensive experiments, and the PH models with $n = 4$ always outperform the quaternion counterpart gaining a higher accuracy and being more robust. This underlines the effectiveness of the PHC architectural flexibility over the

TABLE III
IMAGENET CLASSIFICATION WITH REAL-VALUED BASELINE AGAINST OUR BEST MODEL PH $n = 3$. OUR APPROACH OUTPERFORMS THE BASELINE WHILE SAVING 66% OF PARAMETERS

Model	Params	ImageNet
ResNet50	25.7M	67.990
PHResNet50 $n = 3$	9.6M (-66%)	**68.584**

predefined and rigid structure of quaternion layers. Furthermore, our method distinctly far exceeds the corresponding real-valued baselines across the experiments while saving from 50% to 75% parameters. Focusing on the latter result, PHResNets with $n = 3$ results to be the most suitable choice in many cases, proving the validity of processing RGB images in their natural domain leveraging hypercomplex algebra. However, performance with $n = 3$ and $n = 4$ is comparable, and thus, the choice of this hyperparameter may depend on the application or on the hardware used. On one hand, $n = 4$ may sometimes lead to lower performance; nevertheless, it allows saving disk memory, as shown in the third column of Table II, and thus it may be more appropriate for edge applications. On the other hand, processing color images with $n = 3$ may bring higher accuracy even so it requires more parameters. Therefore, such a flexibility makes PHNNs adaptable to a large range of applications. Likewise, PHResNets with $n = 2$ gain considerable accuracy scores with respect to the real-valued corresponding models and, due to the larger number of parameters with respect to the PH model with $n = 3$, sometimes outperform it too. Finally, PHResNet with $n = 4$ obtains the overall best accuracy in the largest experiment of this set. Indeed, considering a ResNet152 backbone on CIFAR100, our method exceeds the real-valued baseline by more than 4%. This is the empirical proof that PHNNs well scale to large real-world problems by notably reducing the overall number of parameters. These results are summarized for the ResNets and VGGs models on CIFAR10 in Fig. 4. The plot displays models' accuracies against models' parameters. The PH-based models, either ResNets or VGGs, exceed their real and quaternion-valued baselines while consistently reducing the number of parameters. What is more, in Table II, we also report the memory required to store models' checkpoints for inference. Our method crucially reduces the amount of disk memory demand with respect to the heavier real-valued model.

Furthermore, we perform the image classification task on the ImageNet dataset. We compute the percentage of successes of the ResNet-based models in each run for which we report the average accuracies in Table II. As Fig. 5 shows, the largest percentage of successes is reached by PHResNet with $n = 3$ which has been demonstrated to be the most valuable choice for n when dealing with RGB images. Therefore, we test PHResNet with $n = 3$ against the real-valued counterpart. Table III shows that the proposed method achieves comparable, and even slightly superior, performance than the real-valued baseline, while involving 66% fewer parameters. In addition, in Fig. 6, we provide Grad-CAM visualizations [54] for a sample of predictions by our method in the ImageNet dataset to further prove the correct behavior of PHResNet50 $n = 3$ in this scenario. This proves the robustness of the proposed approach, which can be adopted and implemented in models at different scales.

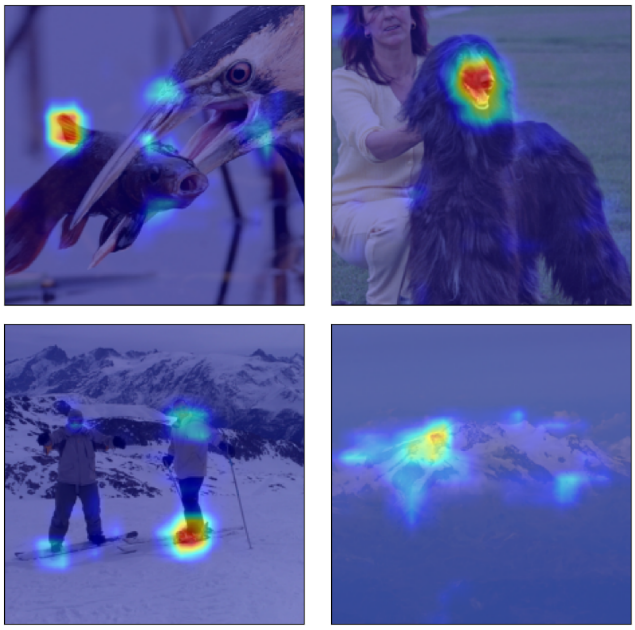


Fig. 6. Grad-CAM visualization for PHResNet50 $n = 3$ on the ImageNet dataset.

VII. EXPERIMENTAL EVALUATION ON SED

SED is the task of recognizing the sounds' classes and at what temporal instances these sounds are active in an audio signal [55]. We prove that the PHC layer is adaptable to n-dimensional input signals and, due to parameter reduction and hypercomplex algebra, is more performing in terms of efficiency and evaluation scores.

A. Experimental Setup

For SED models, we consider the augmented version of SELDnet [49], [52] which was proposed as baseline for the L3DAS21 challenge Task 2 [56] and we perform our experiments with the corresponding released dataset.[1] We consider as our baselines the SEDnet (without the localization part) and its quaternion counterpart. The L3DAS21 Task 2 dataset contains 15 h of MSMP B-format Ambisonics audio recordings, divided into 900 1-min-long data points sampled at a rate of 32 kHz, where up to three acoustic events may overlap. The 14 sounds' classes have been selected from the FSD50K dataset and are representative for an office sounds: *computer keyboard, drawer open/close, cupboard open/close, finger snapping, keys jangling, knock, laughter, scissors, telephone, writing, chink and clink, printer, female speech, and male speech*. In this

[1]L3DAS21 dataset and code are available at: https://github.com/l3das/L3DAS21.

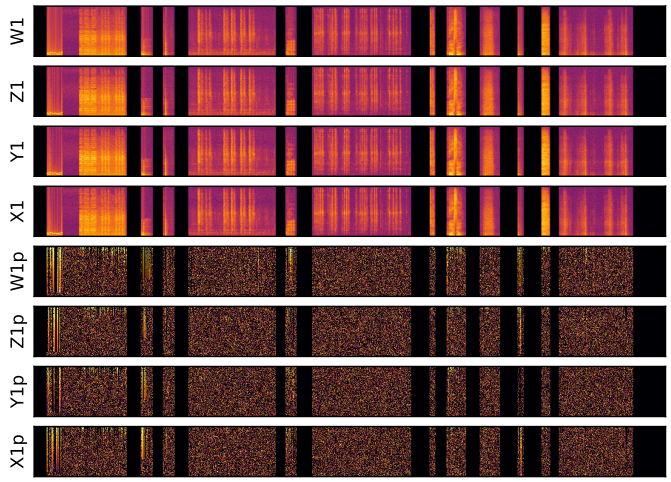


Fig. 7. Sample spectrograms from the L3DAS21 dataset recorded by one microphone with four capsules. The first four figures represent the magnitudes, while the last four contain the corresponding phases information. The black sections represent silent instants.

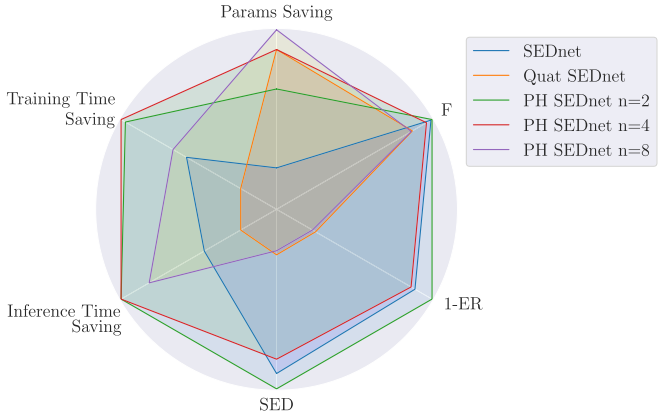


Fig. 8. Radar plot for SEDnets results on the L3DAS21 dataset with two microphones. The larger the area, the better the results. With the same computational time, PHC $n = 2$ gains better scores with respect to PHC $n = 4$ at a cost of more parameters. The real-valued SEDnet, although the discrete SED scores, has a high computational time demand and largest number of parameters.

dataset, the volume difference between the sounds is in the range of 0–20 dB full scale (dBFS). Considering the array of two microphones 1, 2, the channels' order is [W1, Z1, Y1, X1, W2, Z2, Y2, X2], where W, X, Y, and Z are the B-format ambisonics channels if the phase (p) information is not considered. Whether we want to include also this information, the order will be [W1, Z1, Y1, X1, W1p, Z1p, Y1p, X1p, W2, Z2, Y2, X2, W2p, Z2p, Y2p, X2p] up to 16 channels. In Fig. 7, we show the eight-channel input when considering one microphone and phase information. Magnitudes and phases are normalized to be centered in 0 with standard deviation 1.

We perform experiments with multiple configurations of this dataset. We first test the recordings from one microphone considering the magnitudes only (four channels' input), and then we test the networks with the signals recorded by two microphones and magnitudes only (eight channels' input). The extracted features by preprocessing are fed to the four-layer convolutional stack with 64, 128, 256, and 512 filters, with batch normalization, ReLU activation, max pooling, and dropout (probability 0.3), with pooling sizes (8, 2), (8, 2), (2, 2), (1, 1). The bidirectional GRU module has three layers, each with a hidden size of 256. The tail is a four-layer FC classifier with 1024 filters alternated by ReLUs and with a final dropout and a sigmoid activation function. The initial learning rate is set to 0.00001. To be consistent with preexisting literature metrics, we define true positives as TP, false positives as FP, and false negatives as FN. These are computed according to the detection metric [56]. Moreover, to compute the error rate (ER), we consider: $S = \min(\text{FN}, \text{FP})$, $D = \max(0, \text{FN} - \text{FP})$ and $I = \max(0, \text{FP} - \text{FN})$, as in [52] and [55]. Therefore, we consider

$$F_{\text{score}} = \frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}},$$
$$\text{ER} = \frac{S + D + I}{N}$$

whereby N is the total number of active sound event classes in the reference. The $\text{SED}_{\text{score}}$ is defined by

$$\text{SED}_{\text{score}} = \frac{\text{ER} + 1 - F_{\text{score}}}{2}.$$

For ER and $\text{SED}_{\text{score}}$, the lower the scores, the better the performance, while for the F_{score} higher values stand for better accuracy.

B. Experimental Results

We investigate PHSEDnets in complex, quaternion, and octonion domains with $n = 2$, 4, and 8 and train each network for 1000 epochs with a batch size of 16. The proposed parameterized hypercomplex SEDnets distinctly outperform real and quaternion-valued baselines, as reported in Tables IV and V. Indeed, PHSEDnet with $n = 2$ gains the best results for each score and in both one and two microphone datasets, proving that the weights' sharing due to hypercomplex parameterization is able to capture more information regardless of the lower number of parameters. It is interesting to note that PHSEDnet $n = 4$, which operates in the quaternion domain, achieves improved scores with respect to the quaternion SEDnet that follows the rigid predefined algebra rules. Furthermore, the malleability of PHC layers allows gaining comparable performance with respect to the quaternion baseline even so reducing the convolutional parameters by 87%, just setting $n = 8$. In Section VIII-B, we show additional experimental results of the PH models able to save 94% of the convolutional parameters while operating in the sedonion domain by involving $n = 16$.

Furthermore, PHSEDnets are more efficient in terms of time required for training and inference. Table V also shows that each tested version of the proposed method is faster with regards to the real SEDnet and the quaternion one, both at training and inference time. Time efficiency is crucial in audio applications where networks are usually trained for thousands of epochs and datasets are very large and require protracted computations.

TABLE IV

SEDNETS RESULTS WITH ONE MICROPHONE (FOUR CHANNELS' INPUT). SCORES ARE COMPUTED OVER THREE RUNS WITH DIFFERENT SEEDS AND WE REPORT THE MEAN. THE PROPOSED METHOD WITH $n = 2$ FAR EXCEEDS THE BASELINES IN EACH METRIC CONSIDERED

Model	Conv Params	F_{score} ↑	ER ↓	SED_{score} ↓	P ↑	R ↑
SEDnet	1.6M	0.637	0.450	0.406	0.756	0.5505
Quaternion SEDnet	0.4M (-75%)	0.580	0.516	0.468	0.724	0.484
PHSEDnet $n = 2$	0.8M (-50%)	**0.680**	**0.389**	**0.355**	**0.767**	**0.611**
PHSEDnet $n = 4$	0.4M (-75%)	0.638	0.453	0.407	0.765	0.547

TABLE V

SEDNETS RESULTS WITH TWO MICROPHONES (EIGHT CHANNELS' INPUT). SCORES ARE COMPUTED OVER THREE RUNS WITH DIFFERENT SEEDS AND WE REPORT THE MEAN. PHSEDNET $n = 2$ OUTPERFORMS THE BASELINES. FOR TRAINING TIME (SECONDS/ITERATION), THE MEAN AND THE STANDARD DEVIATION OVER ONE EPOCH ARE REPORTED; FOR INFERENCE TIME, WE REPORT THE TIME REQUIRED TO PERFORM AN ITERATION ON THE VALIDATION SET. PH-BASED MODELS FAR EXCEED BASELINES BOTH IN TRAINING AND INFERENCE TIME

Model	Conv Params	FLOPs	F_{score} ↑	ER ↓	SED_{score} ↓	P ↑	R ↑	Time (T)	Time (I)
SEDnet	1.6M	37.3G	0.663	0.428	0.383	**0.788**	0.572	1.242 ± 0.088	1.198
Quaternion SEDnet	0.4M (-75%)	37.3G	0.559	0.556	0.499	0.754	0.444	1.308 ± 0.088	1.298
PHSEDnet $n = 2$	0.8M (-50%)	37.3G	**0.669**	**0.406**	**0.368**	0.767	**0.594**	**1.091 ± 0.074**	1.085
PHSEDnet $n = 4$	0.4M (-75%)	37.3G	0.638	0.433	0.397	0.729	0.567	**1.091 ± 0.032**	**1.077**
PHSEDnet $n = 8$	0.2M (-87%)	37.3G	0.553	0.560	0.503	0.747	0.439	1.142 ± 0.042	1.173

TABLE VI

EXPERIMENTS ON THE SVHN DATASET WITH THE SMALLEST NETWORKS FROM EACH FAMILY, RESNET20 AND VGG11, THE LATTER WITH MODIFIED NUMBER OF FILTERS TO BE DIVIDED BY EACH VALUE OF n AND FC LAYERS IN THE CLOSING CLASSIFIER. WE ALSO TEST THE PHNN WITH $n = 1$ TO REPLICATE THE REAL DOMAIN WHICH OUTPERFORMS THE REAL-VALUED RESNET20

Model	Params	SVHN
ResNet20	0.27M	90.463
Quaternion ResNet20	0.07M (-75%)	93.535
PHResNet20 $n = 1$	0.27M	**93.796**
PHResNet20 $n = 2$	0.14M (-50%)	93.708
PHResNet20 $n = 4$	0.07M (-75%)	93.669
VGG11	13.8M	93.488
Quaternion VGG11	3.9M (-71%)	92.888
PHVGG11 $n = 2$	7.2M (-48%)	**93.958**
PHVGG11 $n = 3$	5.0M (-64%)	93.804
PHVGG11 $n = 4$	3.9M (-71%)	93.919

TABLE VII

FIRST LINES REPORT VGG16 RESULTS WITH REAL-VALUED CLASSIFIER FOR QUATERNION AND PHNNS. EXTENSION OF TABLE I. ADDITIONAL EXPERIMENTS WITH RESNET56 AND RESNET110, THE LATTER WITH MODIFIED NUMBER OF FILTERS TO BE DIVIDED BY EACH VALUE OF n. ACCURACY SCORE IS THE MEAN OVER THREE RUNS WITH DIFFERENT SEEDS

Model	Params	SVHN	CIFAR10
Quaternion VGG16	4.2M (-72%)	94.086	84.126
PHVGG16 $n = 2$	7.9M (-62%)	**94.885**	**86.147**
PHVGG16 $n = 4$	4.2M (-72%)	94.562	85.710
ResNet56	0.9M	94.116	**83.700**
Quaternion ResNet56	0.2M (-75%)	93.664	81.687
PHResNet56 $n = 2$	0.4M (-50%)	93.722	83.413
PHResNet56 $n = 4$	0.2 (-75%)	**94.122**	82.720
ResNet110	16.7M	93.461	84.810
Quaternion ResNet110	4.2M (-75%)	92.788	83.920
PHResNet110 $n = 2$	8.4M (-50%)	93.746	83.220
PHResNet110 $n = 3$	5.6M (-66%)	94.712	85.200
PHResNet110 $n = 4$	4.2M (-75%)	**94.885**	**85.280**

TABLE VIII

REAL-VALUED RESNETS WITH CONVOLUTIONAL FILTERS REDUCED BY 75%, DENOTED BY (S). FULL MODELS EXCEED REDUCED VERSIONS IN EACH OF THE EXPERIMENT, PROVING THAT A SMALLER NUMBER OF PARAMETERS DO NOT LEAD TO HIGHER GENERALIZATION CAPABILITIES

Model	Params	SVHN	CIFAR10	CIFAR100
ResNet18	10.1M	**93.992**	**89.543**	**62.634**
ResNet18 (s)	2.7M (-75%)	93.842	88.310	59.590
ResNet50	22.5M	**94.546**	**89.630**	**65.514**
ResNet50 (s)	5.7M (-75%)	93.915	89.370	62.450
ResNet152	52.6M	**94.625**	**89.580**	**62.053**
ResNet152 (s)	13.2M (-75%)	94.400	89.001	60.850

Fig. 8 summarizes the number of parameters, metrics scores, and computational time in a radar plot from which it is clear that PHSEDnet $n = 2$ gains the best scores and a large time saving at the cost of more parameters with respect to other versions but the real one. A good tradeoff is brought by the PH model $n = 4$ which further reduces the number of parameters at the cost of slightly worse SED_{score} and ER. Moreover, the real-valued SEDnet is capable of obtaining fair scores while having the largest parameters' amount and high computational time demanding.

VIII. ABLATION STUDIES

A. Less Parameters Do Not Lead to Higher Generalization

In the following, we demonstrate that higher accuracies achieved by our method are not caused by parameter reduction which may lead to more generalization. To this end, we perform multiple experiments. First, we test lighter ResNets that were originally built for the CIFAR10 dataset [47]: ResNet20, ResNet56, and ResNet110. Second, we also consider the smallest VGG network, that is, VGG11 which has 14 M parameters. Finally, we perform experiments on SVHN, CIFAR10,

TABLE IX

SED RESULTS WITH TWO MICROPHONE: MAGNITUDES AND PHASES (16 CHANNELS' INPUT). WE TEST HIGHER ORDER HYPERCOMPLEX DOMAINS UP TO SEDONIONS BY SETTING $n = 16$. ALTHOUGH THE INCREDIBLE REDUCTION OF THE NUMBER OF PARAMETERS WITH RESPECT TO THE REAL-VALUED BASELINE IN TABLE V, PHNN WITH $n = 16$ STILL HAS COMPARABLE PERFORMANCE WITH OTHER MODELS. FURTHERMORE, PHSEDNET WITH $n = 8$ ALSO OUTPERFORMS THE QUATERNION BASELINE WHICH HAS MORE DEGREES OF FREEDOM

Model	Conv Params	F_{score} ↑	ER ↓	SED_{score} ↓	P ↑	R ↑
Quaternion SEDnet	0.4M (-75%)	0.580	0.480	0.450	0.655	0.520
PHSEDnet $n = 4$	0.4M (-75%)	0.585	<u>0.470</u>	<u>0.443</u>	0.653	<u>0.530</u>
PHSEDnet $n = 8$	0.2M (-87%)	**0.607**	**0.466**	**0.430**	<u>0.702</u>	**0.534**
PHSEDnet $n = 16$	0.1M (-94%)	<u>0.588</u>	0.509	0.461	**0.734**	0.491

and CIFAR100 with the larger ResNet18, ResNet50, and ResNet152 reducing the number of filters by 75% so as to have the same number of parameters of quaternion and PHNN with $n = 4$ counterparts.

Table VI reports experiments with ResNet20 where we also test $n = 1$ to replicate the real-valued model, outperforming it. Experiments with VGG11 with modified number of filters to be divided by each value of n are also reported in the same table. Finally, in Table VII, we report experiments on SVHN and CIFAR10 with ResNet56 and ResNet110, the latter with modified number of filters. The PH models gain good performance in each test we conduct while reducing the amount of free parameters. Indeed, PHResNet20s gain almost 94% of accuracy on the SVHN dataset involving just 70k parameters.

Finally, to further remove the hypothesis that a smaller number of neural parameters leads to higher generalization capabilities, we perform experiments with real-valued baselines with a number of parameters reduced by 75%. Table VIII shows that reducing the number of filters downgrades the performance, and thus it is not sufficient to improve the generalization capabilities of a model. We do not include standard deviations for values in the ablation studies as the values are similar to previous examples so we aim at favoring article readability.

B. Push the Hyperparameter n Up to 16

In the following, we perform additional experiments for the SED task. We conduct a test considering two microphones and phase information, so as to have an input with 16 channels. For this purpose, we consider as baseline the quaternion model and PHNNs with $n = 4, 8, 16$ so as to test higher order domains. Quaternion and PHSEDnet with $n = 4$ manage the 16 channels by grouping them into four components, thus assembling them in four channels: one channel containing the magnitudes of the first microphone, one channel the phases of the same microphone, and so on. Therefore, the details coming from the magnitudes, which are the most important for SED, are grouped together without properly exploiting this information. On the contrary, using PHC layers allows the model to process information without roughly grouping channels while instead leveraging every information by easily setting n equal to the number of channels, that is, in this case 16. From Table IX, it is clear that using a four-channel model such as quaternion or PHC with $n = 4$ does not lead to higher performance, despite the higher number of parameters. Indeed, the best scores are obtained with the PHC models involving $n = 8$ and $n = 16$ that are able to grasp information from each channel.

IX. CONCLUSION

In this article, we introduce a PHC layer which grasps the convolution rule directly from data and can operate in any domain from 1-D to nD, regardless the algebra regulations are preset. The proposed approach reduces the convolution parameters to $1/n$ with respect to real-valued counterparts and allows capturing internal latent relationships, thanks to parameter sharing among input dimensions. Using this method, jointly with the one in [37], we devise the family of PHNNs, a set of lightweight and efficient neural models exploiting hypercomplex algebra properties for increased performance and high flexibility. We show that our method is flexible to operate in different fields of application by performing experiments with images and audio signals. We also prove the malleability and robustness of our approach to learn convolution rules in any domain by setting different values for the hyperparameter n from 2 to 16.

CO2 Emission Related to Experiments

Experiments were conducted using a private infrastructure, which has a carbon efficiency of 0.445 $kgCO_2eq/kWh$. A cumulative of 2000 h of computation was performed on hardware of type Tesla V100-SXM2-32GB (TDP of 300W). Total emissions are estimated to be 267 $kgCO_2eq$ of which 0% was directly offset. Estimations were conducted using the MachineLearning Impact calculator presented in [57].

More in detail, considering an experiment for the SED task, according to Table V, the real-valued baseline requires approximately 20 h for training and validation, with the corresponding carbon emissions of 2.71 $kgCO_2eq$. Conversely, the proposed PH model takes approximately 17 h with a reduction of carbon emissions of 16%, being 2.28 $kgCO_2eq$.

In conclusion, we believe that the improved efficiency of our method with respect to standard models may be a little step toward reducing carbon emissions.

REFERENCES

[1] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, "Analyzing and improving the image quality of StyleGAN," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 8110–8119.

[2] S. D'Ascoli, H. Touvron, M. Leavitt, A. Morcos, G. Biroli, and L. Sagun, "ConViT: Improving vision transformers with soft convolutional inductive biases," 2021, *arXiv:2103.10697*.

[3] A. Dosovitskiy et al., "An image is worth 16×16 words: Transformers for image recognition at scale," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2021, pp. 1–22.

[4] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, Jul. 2019, pp. 4780–4789.

[5] J. Navarro-Moreno and J. C. Ruiz-Molina, "Wide-sense Markov signals on the tessarine domain. A study under properness conditions," *Signal Process.*, vol. 183, Jun. 2021, Art. no. 108022.

[6] J. Navarro-Moreno, R. M. Fernández-Alcalá, J. D. Jiménez-López, and J. C. Ruiz-Molina, "Tessarine signal processing under the T-properness condition," *J. Franklin Inst.*, vol. 357, no. 14, pp. 10100–10126, Sep. 2020.

[7] S. Sanei, C. C. Took, and S. Enshaeifar, "Quaternion adaptive line enhancer based on singular spectrum analysis," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Apr. 2018, pp. 2876–2880.

[8] M. Xiang et al., "Simultaneous diagonalisation of the covariance and complementary covariance matrices in quaternion widely linear signal processing," *Signal Process.*, vol. 148, pp. 193–204, Jul. 2018.

[9] M. Kobayashi, "Quaternion projection rule for rotor Hopfield neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 2, pp. 900–908, Feb. 2021.

[10] D. Lin, X. Chen, Z. Li, B. Li, and X. Yang, "On the existence of the exact solution of quaternion-valued neural networks based on a sequence of approximate solutions," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Dec. 2, 2021, doi: 10.1109/TNNLS.2021.3129269.

[11] L. Liu, C. L. P. Chen, and Y. Wang, "Modal regression-based graph representation for noise robust face hallucination," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Sep. 6, 2021, doi: 10.1109/TNNLS.2021.3106773.

[12] M. E. Valle and F. Z. D. Castro, "On the dynamics of Hopfield neural networks on unit quaternions," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 6, pp. 2464–2471, Jun. 2018.

[13] Y. Liu et al., "Stability analysis of quaternion-valued neural networks: Decomposition and direct approaches," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 9, pp. 4201–4211, Sep. 2018.

[14] M. E. Valle and R. A. Lobo, "Quaternion-valued recurrent projection neural networks on unit quaternions," *Theor. Comput. Sci.*, vol. 843, pp. 136–152, Dec. 2020.

[15] F. Z. de Castro and M. E. Valle, "A broad class of discrete-time hypercomplex-valued Hopfield neural networks," *Neural Netw.*, vol. 122, pp. 54–67, Feb. 2020.

[16] T. K. Paul and T. Ogunfunmi, "A kernel adaptive algorithm for quaternion-valued inputs," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 10, pp. 2422–2439, Oct. 2015.

[17] A. Hirose, I. Aizenberg, and D. P. Mandic, "Guest editorial special issue on complex- and hypercomplex-valued neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 9, pp. 1597–1599, Sep. 2014.

[18] A. Muppidi and M. Radfar, "Speech emotion recognition using quaternion convolutional neural networks," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Jun. 2021, pp. 6309–6313.

[19] T. Parcollet et al., "Quaternion recurrent neural networks," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, New Orleans, LA, USA, May 2019, pp. 1–19.

[20] E. Grassucci, E. Cicero, and D. Comminiello, "Quaternion generative adversarial networks," in *Generative Adversarial Learning: Architectures and Applications*, R. Razavi-Far, A. Ruiz-Garcia, V. Palade, and J. Schmidhuber, Eds. Cham, Switzerland: Springer, 2022, pp. 57–86.

[21] Y. Tay et al., "Lightweight and efficient neural natural language processing with quaternion networks," in *Proc. 57th Annu. Meeting Assoc. Comput. Linguistics*, 2019, pp. 1494–1503.

[22] A. Cariow and G. Cariowa, "Fast algorithms for deep octonion networks," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Nov. 5, 2021, doi: 10.1109/TNNLS.2021.3124131.

[23] J. Wu, L. Xu, F. Wu, Y. Kong, L. Senhadji, and H. Shu, "Deep octonion networks," *Neurocomputing*, vol. 397, pp. 179–191, Jul. 2020.

[24] M. E. Valle and R. A. Lobo, "Hypercomplex-valued recurrent correlation neural networks," *Neurocomputing*, vol. 432, pp. 111–123, Apr. 2021.

[25] T. Chen, H. Yin, X. Zhang, Z. Huang, Y. Wang, and M. Wang, "Quaternion factorization machines: A lightweight solution to intricate feature interaction modeling," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Oct. 19, 2021, doi: 10.1109/TNNLS.2021.3118706.

[26] E. Grassucci, D. Comminiello, and A. Uncini, "A quaternion-valued variational autoencoder," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Jun. 2021, pp. 3310–3314.

[27] E. Grassucci, D. Comminiello, and A. Uncini, "An information-theoretic perspective on proper quaternion variational autoencoders," *Entropy*, vol. 23, no. 7, p. 856, Jul. 2021.

[28] S. Gai and X. Huang, "Reduced biquaternion convolutional neural network for color image processing," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 32, no. 3, pp. 1061–1075, Mar. 2022.

[29] G. Vieira and M. Eduardo Valle, "Extreme learning machines on Cayley-dickson algebra applied for color image auto-encoding," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2020, pp. 1–8.

[30] C. C. Took and Y. Xia, "Multichannel quaternion least mean square algorithm," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2019, pp. 8524–8527.

[31] C. J. Gaudet and A. S. Maida, "Removing dimensional restrictions on complex/hyper-complex neural networks," in *Proc. IEEE Int. Conf. Image Process. (ICIP)*, Sep. 2021, pp. 319–323.

[32] J. Hoffmann, S. Schmitt, S. Osindero, K. Simonyan, and E. Elsen, "AlgebraNets," 2020, *arXiv:2006.07360*.

[33] A. Cariow and G. Cariowa, "Fast algorithms for quaternion-valued convolutional neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 1, pp. 457–462, Apr. 2020.

[34] C. Huang, A. Touati, P. Vincent, G. K. Dziugaite, A. Lacoste, and A. C. Courville, "Stochastic neural network with Kronecker flow," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2020, pp. 4184–4194.

[35] Z. Tang et al., "SKFAC: Training neural networks with faster Kronecker-factored approximate curvature," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 13479–13487.

[36] D. Wang et al., "Kronecker CP decomposition with fast multiplication for compressing RNNs," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Sep. 17, 2021, doi: 10.1109/TNNLS.2021.3105961.

[37] A. Zhang et al., "Beyond fully-connected layers with quaternions: Parameterization of hypercomplex multiplications with $1/n$ parameters," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2021, pp. 1-13.

[38] T. Le, M. Bertolini, F. Noé, and D.-A. Clevert, "Parameterized hypercomplex graph neural networks for graph classification," 2021, *arXiv:2103.16584*.

[39] R. Karimi Mahabadi, J. Henderson, and S. Ruder, "Compacter: Efficient low-rank hypercomplex adapter layers," 2021, *arXiv:2106.04647*.

[40] H. Wu et al., "CvT: Introducing convolutions to vision transformers," 2021, *arXiv:2103.15808*.

[41] S. Hershey et al., "CNN architectures for large-scale audio classification," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Mar. 2017, pp. 131–135.

[42] T. Parcollet, M. Morchid, and G. Linarès, "A survey of quaternion neural networks," *Artif. Intell. Rev.*, vol. 53, no. 4, pp. 2957–2982, Aug. 2019.

[43] C. J. Gaudet and A. S. Maida, "Deep quaternion networks," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Rio de Janeiro, Brazil, Jul. 2018, pp. 1–8.

[44] H. V. Jemderson, F. Pukelsheim, and S. R. Searle, "On the history of the Kronecker product," *Linear Multilinear Algebra*, vol. 14, no. 2, pp. 113–120, 1983.

[45] T. Parcollet, M. Morchid, and G. Linarès, "Quaternion convolutional neural networks for heterogeneous image processing," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, Brighton, U.K., May 2019, pp. 8514–8518.

[46] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, San Diego, CA, USA, 2015, pp. 1–14.

[47] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

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

[49] D. Comminiello, M. Lella, S. Scardapane, and A. Uncini, "Quaternion convolutional neural networks for detection and localization of 3D sound events," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Brighton, U.K., May 2019, pp. 8533–8537.

[50] M. R. Celsi, S. Scardapane, and D. Comminiello, "Quaternion neural networks for 3D sound source localization in reverberant environments," in *Proc. IEEE 30th Int. Workshop Mach. Learn. Signal Process. (MLSP)*, Espoo, Finland, Sep. 2020, pp. 1–6.

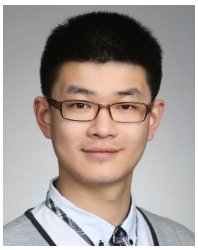
[51] E. Grassucci, G. Mancini, C. Brignone, A. Uncini, and D. Comminiello, "Dual quaternion ambisonics array for six-degree-of-freedom acoustic representation," 2022, *arXiv:2204.01851*.

[52] S. Adavanne, A. Politis, J. Nikunen, and T. Virtanen, "Sound event localization and detection of overlapping sources using convolutional recurrent neural networks," *IEEE J. Sel. Topics Signal Process.*, vol. 13, no. 1, pp. 34–48, Mar. 2019.
[53] R. Wightman, H. Touvron, and H. Jégou, "ResNet strikes back: An improved training procedure in timm," 2021, *arXiv:2110.00476*.
[54] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual explanations from deep networks via gradient-based localization," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 618–626.
[55] A. Mesaros, T. Heittola, T. Virtanen, and M. D. Plumbley, "Sound event detection: A tutorial," *IEEE Signal Process. Mag.*, vol. 38, no. 5, pp. 67–83, Sep. 2021.
[56] E. Guizzo et al., "L3DAS21 challenge: Machine learning for 3D audio signal processing," in *Proc. IEEE 31st Int. Workshop Mach. Learn. Signal Process. (MLSP)*, Oct. 2021, pp. 1–6.
[57] A. Lacoste, A. Luccioni, V. Schmidt, and T. Dandres, "Quantifying the carbon emissions of machine learning," 2019, *arXiv:1910.09700*.

Eleonora Grassucci (Graduate Student Member, IEEE) received the master's degree in data science from the Sapienza University of Rome, Rome, Italy, in 2019, where she is currently pursuing the Ph.D. degree in information and communication technologies.

She is currently a Post-Doctoral at the Department of Information Engineering, Electronics, and Telecommunications (DIET), Sapienza University of Rome. Her research focuses on generative deep learning and hypercomplex neural networks with applications in different fields such as audio, vision, and medical analysis.

Dr. Grassucci was awarded the Best Track Manuscript Recognition by the IEEE Circuits and Systems Society in 2022.



Aston Zhang received the Ph.D. degree in computer science from the University of Illinois at Urbana-Champaign, Champaign, IL, USA, in 2017.

He is currently a Senior Scientist at Amazon Web Services, East Palo Alto, CA, USA. His research interests include deep learning.

Dr. Zhang was a recipient of the ICLR Outstanding Paper Award, the ACM UbiComp Distinguished Paper Award, and the ACM SenSys Best Paper Award Nomination. His book Dive into Deep Learning (D2L.ai) has been adopted for teaching by over 400 universities from 60 countries. He served as an Area Chair for EMNLP.

Danilo Comminiello (Senior Member, IEEE) received the Laurea degree in telecommunication engineering and the Ph.D. degree in information and communication engineering from the Sapienza University of Rome, Rome, Italy, in 2008 and 2012, respectively.

He is currently an Associate Professor with the Department of Information Engineering, Electronics and Telecommunications (DIET), Sapienza University of Rome. His research interests include design and analysis of modern artificial intelligence algorithms, including neural networks and machine learning methods, deep generative models, and adaptive learning systems, with application to several fields, from audio to images, from medical to sensor signals.

Dr. Comminiello is an Elected Member of the IEEE Machine Learning for Signal Processing Technical Committee and of the IEEE Nonlinear Circuits and Systems Technical Committee. He is the General Co-Chair of the 33rd IEEE International Workshop on Machine Learning for Signal Processing (MLSP 2023), Rome. He has served as an Associate Editor for IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS and IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS I: REGULAR PAPERS, ELSEVIER DIGITAL SIGNAL PROCESSING. He is one of the editors of the book *Adaptive Learning Methods for Nonlinear System Modeling* (D. Comminiello and J. C. Principe, eds.), (Elsevier, 2018).

Open Access funding provided by 'Universitá degli Studi di Roma "La Sapienza"' within the CRUI CARE Agreement