Hypercomplex Generative Adversarial Networks for Lightweight Semantic Labeling

Giorgos Sfikas[1,3,4](✉), George Retsinas[2,4], Basilis Gatos[1],
and Christophoros Nikou[4]

[1] Computational Intelligence Laboratory, Institute of Informatics
and Telecommunications, National Center for Scientific Research "Demokritos",
Athens, Greece
bgat@iit.demokritos.gr

[2] School of Electrical and Computer Engineering, National Technical University
of Athens, Athens, Greece
gretsinas@central.ntua.gr

[3] Department of Surveying and Geoinformatics Engineering,
University of West Attica, Athens, Greece
gsfikas@uniwa.gr

[4] Department of Computer Science and Engineering,
University of Ioannina, Ioannina, Greece
cnikou@cs.uoi.gr

Abstract. Following recent advances on parameterized hypercomplex multiplication [21], we explore the usefulness of hypercomplex convolutions and deconvolutions in a document labeling task. We show that the proposed Hypercomplex Generative Adversarial Networks achieve excellent results while requiring significantly less independent parameters than real-valued models.

Keywords: Quaternions · Parameterized hypercomplex multiplication · Semantic labeling · Multiple labels · Document image processing · Generative Adversarial Networks

1 Introduction and Related Work

Quaternions are numbers that are part of the family that is known as hypercomplex numbers, and perhaps the better known member of this family. Regarding their applications, again the most well-known example concerns quaternions and their use in expressing rotations in 3D (and 4D) space [9,20]. Other applications in computer science are more or less unknown to the majority of researchers in the community, and at best seem exotic or of marginal use. One line of research involves quaternions, and to some extend other hypercomplex numbers, as a tool to generalize standard methods in signal and image processing. Notable examples are the Quaternion Fourier Transform [4], and Quaternion Singular Value Decomposition [10], or extensions of standard image filters like Sobel or the

M. El Yacoubi et al. (Eds.): ICPRAI 2022, LNCS 13363, pp. 251–262, 2022.
https://doi.org/10.1007/978-3-031-09037-0_21

Laplacian to $\mathbb{R}^3$ and colour image processing. In keypoint detection, a quaternionic variant of the Harris-Stephens detector has been recently described [17]. The advantage of a hypercomplex framework is, in a nutshell, that any methodology that assumes scalar values, signals or tensors can be extended to account for multiple values per number, as each hypercomplex number encapsulates in itself multiple (real) values in a holistic manner. On the downside, quaternions and other hypercomplex number systems involve various difficulties; quaternions define a non-commutative multiplication rule, which subsequently leads to problems in many facets of their use. For example, an immediate implication is that the problem $Ax = \lambda x$ is different than $Ax = x\lambda$, which means that there exist two different types of eigenvalues and eigenvectors [17,22]. A determinant is impossible to define for quaternion matrices [3] (at least, without dropping important defining properties of the classic determinant), and so on and so forth.

Another, more recent line of applications of hypercomplex numbers involves their use with neural networks. Networks with values, inputs, outputs, layers, parameters that are hypercomplex-valued have been proposed and successfully used in a variety of signal processing, vision, and pattern recognition tasks in general. In general, perhaps the most successful feature of these networks is that they lead to implementations that are not-as-demanding as real valued models in terms of storage space requirements, while achieving good results for most tasks where they were employed. The key to their being less resource demanding, lies with the definition of hypercomplex multiplication, which in the context of a neural network layer definition, leads naturally to extended parameter sharing. Quaternions have been the paradigm most prominently used in this respect [11–13,16]. Extensions to other high-dimensional number domains have been recently shown not only to be possible, but also very beneficial [21].

Parameterized Hypercomplex Multiplication is a technique that has recently been proposed as a means to extend the parameter sharing/sparsness benefits of quaternion neural networks to arbitrary, n-dimensional number systems [7,21]. Under this framework, quaternion networks are considered a special case for $n = 4$, and the multiplication is learnable.

The main points of contribution of this work are as follows: First, we explore the use of parameterized hypercomplex multiplication and Hamilton (quaternion) product to extend convolution and deconvolution layers. Previous work has addressed convolutions only, and in the context of very standard feed-forward models (VGG, ResNet [7]). The hypercomplex layer integration addressed in the current work is done in the context of a Generative Adversarial Network, with fully convolutional generator and discriminator components. Second, we apply the novel hypercomplex model in a document layout labeling task. Specifically, the task is to label each pixel of a document image according to the semantics of the pixel and its neighbourhood. The task is different than standard semantic segmentation, in that multiple labels are allowed per pixel. We show that a hypercomplex architecture can lead to a useful, lightweight model in terms of size, corroborating the results of recent related work [21].

The paper is structured as follows. In Sect. 2 we present the proposed model, after discussing theoretical preliminaries: we review elements of quaternion algebra, hypercomplex numbers, parameterized hypercomplex operations and describe the proposed model architecture. In Sect. 3 we apply the model on a document labeling task. In Sect. 4 we discuss our conclusions and envisage future work.

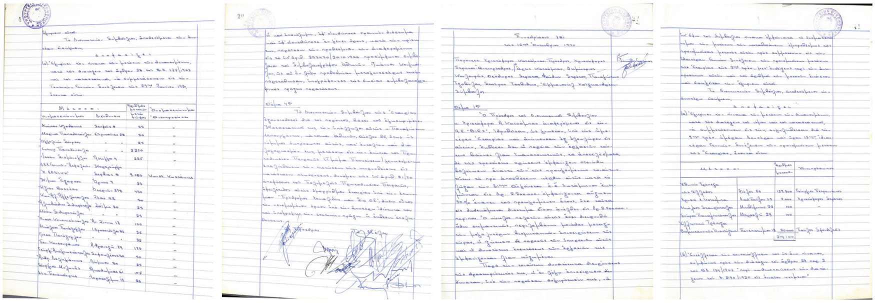


Fig. 1. Sample scanned pages from the PIOP-B dataset, used in our labeling experiments with the proposed Hypercomplex GANs.

2 Proposed Model

2.1 Hypercomplex Layers

Quaternions and Elements of Quaternionic Algebra. Quaternions have been introduced in the 19^{th} century by Hamilton. Historically, Hamilton was initially trying to create a 3-dimensional algebra that would employ one real part and two imaginary components, all orthogonal to one another. He eventually found the undertaking to be impossible, which gave birth to the algebra of quaternions. The latter instead prescribes one real part and three imaginary components. This result has been subsequently codified and formalized in a theorem by Frobenius, which states that the only associative division algebras (up to an isomorphism) that are possible are the real numbers $\mathbb{R}$, the complex numbers $\mathbb{C}$, and quaternionic numbers $\mathbb{H}$ [5].

A quaternion can be formally defined as a number $q \in \mathbb{H}$:

$$q = a + b\boldsymbol{i} + c\boldsymbol{j} + d\boldsymbol{k}, \tag{1}$$

where $\boldsymbol{i}$, $\boldsymbol{j}$, $\boldsymbol{k}$ are independent unit "imaginary" components, and $a, b, c, d \in \mathbb{R}$ are respective coefficients to the real and the imaginary parts. From Eq. 1 it is straightforward that $\mathbb{H}$ is isomorphic to $\mathbb{R}^4$. Hence, quaternions can be understood as a four-dimensional generalization of complex numbers, with two extra imaginary components.

Analogous to what holds with numbers in $\mathbb{C}$, the square of the imaginary unit is -1, i.e. $\boldsymbol{i}^2 = \boldsymbol{j}^2 = \boldsymbol{k}^2 = -1$, where of course the relation holds for any of the three imaginary units. This leads to the interesting consequence that

$\mu^2 = -1$ has infinite solutions for $\mu \in \mathbb{H}$. For quaternion multiplication, we have the following rule:

$$pq = S(p)S(q) - V(p) \cdot V(q) + S(p)V(q) + S(q)V(p) + V(p) \times V(q), \tag{2}$$

where $S(q)$ is the scalar (real) component of q and $V(q)$ is the "vector" component of q, $V(q) = [b \;\; c \;\; d]^T$ (hence $q = S(q) + V(q)$). Operands $\cdot$ and $\times$ denote the dot and cross product respectively. The rule Eq. 2 is occasionally referred to as a Hamilton product in the literature [11]. Another equivalent way to write the same rule is:

$$\begin{aligned} pq = & (a_p a_q - b_p b_q - c_p c_q - d_p d_q) + \qquad (3)\\ & (a_p b_q + b_p a_q + c_p d_q - d_p c_q)\boldsymbol{i} + \qquad (4)\\ & (a_p c_q - b_p d_q + c_p a_q + d_p b_q)\boldsymbol{j} + \qquad (5)\\ & (a_p d_q + b_p c_q - c_p b_q + d_p a_q)\boldsymbol{k}, \qquad (6) \end{aligned}$$

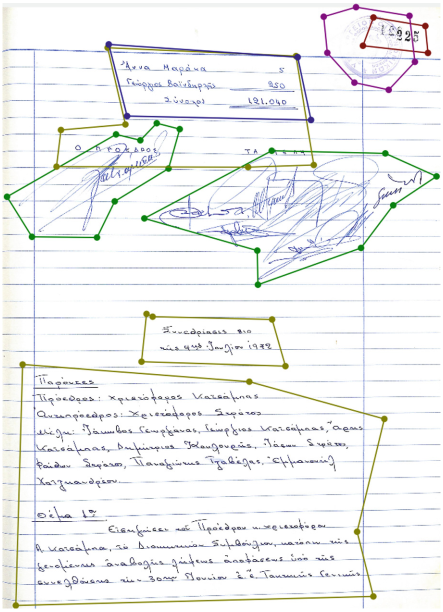
where $p = a_p + b_p\boldsymbol{i} + c_p\boldsymbol{j} + d_p\boldsymbol{k}$ and $q = a_q + b_q\boldsymbol{i} + c_q\boldsymbol{j} + d_q\boldsymbol{k}$. Finally, we can also write the product rule using a matrix-vector notation:

$$\begin{pmatrix} a_{pq} \\ b_{pq} \\ c_{pq} \\ d_{pq} \end{pmatrix} = \begin{pmatrix} a_p & -b_p & -c_p & -d_p \\ b_p & a_p & -d_p & c_p \\ c_p & d_p & a_p & -b_p \\ d_p & -c_p & b_p & a_p \end{pmatrix} \begin{pmatrix} a_q \\ b_q \\ c_q \\ d_q \end{pmatrix}, \tag{7}$$

where we write the product result as $pq = a_{pq} + b_{pq}\boldsymbol{i} + c_{pq}\boldsymbol{j} + d_{pq}\boldsymbol{k}$. We shall see in Sect. 2 that this way of writing the Hamilton product is useful in underpinning the relation of parameterized hypercomplex multiplication to quaternionic operations, also discussed in the same section.

A quaternion conjugate is defined analogously to the complex conjugate, as $\overline{q} = a - b\boldsymbol{i} - c\boldsymbol{j} - d\boldsymbol{k}$, and likewise a quaternion magnitude, as $|q| = \sqrt{q\overline{q}} = \sqrt{\overline{q}q} = \sqrt{a^2 + b^2 + c^2 + d^2}$. Generalizing the formula for $e^{\boldsymbol{i}\theta}$, for any $\mu \in \mathbb{H}$ for which $\mu^2 = -1$ and $\theta \in \mathbb{R}$ it holds $e^{\mu\theta} = cos\theta + \mu \sin\theta$. Using the latter formula, a polar representation of quaternions is possible, as: $q = |q|e^{\mu\theta}$, where $\mu \in \mathbb{H}$ and $\theta \in \mathbb{R}$ are called eigenaxis and eigenangle [4].

Hypercomplex Numbers Beyond Quaternions. Historically, quickly after quaternions were introduced, it was understood that numbers of higher intrinsic dimensionality than that of quaternions (i.e., $d = 4$), were possible (where again, "possibility" should be understood as feasibility of construction of a field or field-like algebraic structure, based on those numbers). Eight-dimensional numbers, called *octonions*, and sixteen-dimensional numbers, called *sedenions*, were in introduced in this manner, as well as a multitude of other constructions [5,15]. These number systems came to be known collectively as hypercomplex numbers. In this work, numbers with dimensionality $d \geq 4$ are of interest as they are related to parameterized hypercomplex variations of "traditional" (i.e. real-valued) neural network operations [21]. Octonions and sedenions also define operations in an analogous manner to that described for quaternions in the previous paragraphs. In a nutshell, "movement" to a higher dimensionality comes at a cost of losing important algebraic structure properties. For example, octonion multiplication is non-commutative, like quaternions, however unlike the latter, it is furthermore non-associative.



12225

Άννα Μαράκα 5
Γεώργιος Βαϊνδηρλής 250
Σύνολον 121.040

Ο ΠΡΟΕΔΡΟΣ ΤΑ ΜΕΛΗ

Συνεδρίασις 210
της 4ης Ιουλίου 1972

Παρόντες
Πρόεδρος: Χριστόφορος Κατσάμπας
Αντιπρόεδρος: Χριστόφορος Στράτος
Μέλη: Ιάκωβος Γεωργάνας, Γεώργιος Κατσάμπας, Άρης Κατσάμπας, Δημήτριος Κουζουρής, Ιάσων Στράτος, Φαίδων Στράτος, Παναγιώτης Τζαβέλας, Εμμανουήλ Χατζημανδρέου.

Θέμα 1ον
Εισηγήσει του Προέδρου κ. Χριστοφόρου Κατσάμπα, το Διοικητικόν Συμβούλιον, κατόπιν της γενομένης αναλόγου γνώμης αποφάσεως υπό της συνελθούσης την 30ην Ιουνίου ε.έ. Τακτικής Γενικής

Fig. 2. Example of manual annotation. Polygon colours correspond to the following semantic labels: 1) Dark green: Handwritten text. 2) Blue: Stamp. 3) Olive green: Matrix. 4) Violet: Signature. 5) Brown: Page number. A single pixel may correspond to a number of labels ranging from none, up to multiple labels. (Color figure online)

Parameterized Hypercomplex Convolutional/Deconvolutional Layer. As a building block in our architecture, we use convolutional and deconvolutional layers. The GAN generator uses a cascade of convolutions and deconvolutions; the GAN discriminator uses a cascade of convolutions (The specific architecture will be described in more detail in the following subsection). Models that make use of parameterized hypercomplex convolutions have been explored recently [7]. In our model, we also experiment with parameterized hypercomplex versions of deconvolutions. Generally, we can write a convolution or deconvolution, parameterized by its kernel H as:

$$y = H * x, \tag{8}$$

where x, y are the input and output tensors respectively, of dimensionality χ and ψ respectively. The operation kernel H is assumed to be rectangular with side equal to k. These operations are well-known to be linear, so we can equivalently write $y = C_H x$, where C_H is the respective Toeplitz or Circulant matrix, depending on the specifics of the operation [8]. Kernel H is in $\mathbb{R}^{\chi \times \psi \times k \times k}$, and the total number of *independent* parameters (or "degrees of freedom") involved is $\chi \times \psi \times k \times k$.

Following the parameterized hypercomplex construction introduced with [21], we can write the operation

$$y = \sum_{i=1}^{n} A_i \otimes F_i * x, \tag{9}$$

where the operation kernel is determined by the sum of kronecker products of matrices A_i and F_i, $i \in [1, n]$, and n is a operation hyperparameter that determines the characteristics of the operation. Matrices A_i are in $\mathbb{R}^{n \times n}$, independent of the input and output tensor dimensions. Matrices F_i are in $\mathbb{R}^{\frac{\chi}{n} \times \frac{\psi}{n} \times k \times k}$. The resulting kronecker products are in $\mathbb{R}^{\frac{\chi}{n} \times \frac{\psi}{n} \times k \times k}$, hence the sum $\sum_{i=1}^{n} A_i \otimes F_i$ of Eq. 9 is also of the same dimensionality. However, the important difference to standard convolution and deconvolution is here that the corresponding parameters are *not independent*; in fact, in this formulation the operation parameters form n-sized groups of shared parameters.

Interestingly, quaternion convolution and deconvolution [16] can be written in the form set in Eq. 9, with n equal to 4 and fixed matrices A_1, A_2, A_3, A_4. Specifically, for quaternion operations these matrices are:

$$A_1 = I_4, A_2 = \begin{pmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 \end{pmatrix}, A_3 = \begin{pmatrix} 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \end{pmatrix}, A_4 = \begin{pmatrix} 0 & 0 & 0 & -1 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \tag{10}$$

Essentially, the matrices of Eq. 10 are determined by the Hamilton product rule (Eq. 2), and more specifically its matrix-vector version (Eq. 7).

For either quaternion or parameterized hypercomplex versions of the convolution and deconvolution operations, their degrees of freedom is much lower than the dimensionality of the tensor space where they are defined. In particular, a parametrized hypercomplex kernel has $\frac{\chi}{n} \times \frac{\psi}{n} \times k^2$ degrees of freedom for each matrix F_i, for a total of $\frac{\chi}{n} \times \psi \times k^2$ for all n matrices. The matrices A_i have $n \times n$ degrees of freedom each, for a total n^3 for all n matrices. Hence, a parameterized hypercomplex convolution or deconvolution has a total of $n^3 + \frac{\chi}{n} \times \psi \times k^2$ independent parameters. For the quaternionic case, we have to set $n = 4$ and disregard the A_i matrices since these are fixed, thus we have $\frac{\chi}{4} \times \psi \times k^2$ independent parameters. In most practical cases, these numbers should be understood as dominated by the parameters of the F_i matrices, or $\frac{\chi}{n} \times \psi \times \psi \times k^2 \gg n^3$. The degrees of freedom associated with each operation kernel version are summarized in Table 1.

Table 1. Degrees of freedom for each convolution/deconvolution kernel operation. Real, quaternion and parameterized hypercomplex variants are compared. n is a "user-defined" operation hyperparameter, k is the size of the kernel, χ and ψ are the input and output feature map dimensionalities.

Operation variant	Degrees of freedom
Standard (real-valued) variant	$\chi \times \psi \times k^2$
Quaternion variant	$\frac{\chi}{4} \times \psi \times k^2$
Parameterized Hypercomplex variant	$n^3 + \frac{\chi}{n} \times \psi \times k^2$

2.2 Model Architecture

A Generative Adversarial Network is composed of two networks: The generator and the discriminator network, denoted as functions $G(\cdot)$ and $D(\cdot)$ respectively. In this work, both networks are fully convolutional. The generator is structured as a U-Net [14], with skip connections used to bridge the corresponding "mirror" layers in the encoder and the decoder part. Both models use hypercomplex convolutions and deconvolutions in all cases, unless stated explicitly otherwise.

The generator is structured as follows: The input is first padded with as many zero channels as required, in order to match the PHM hyperparameter n. We set this padding to 16 channels, to match the maximum value for n that is considered in this work. This action is equivalent to zero-ing the imaginary components of the input. The padded input is passed to a series of convolutional layers, followed by deconvolutional layers, as is the standard structure of U-Net. Four convolutional layers, topped by batch normalization and split-activation leaky ReLU, are followed by an equal number of deconvolutional layers. The number of channels are: 64, 128, 128, 256, 128, 128, 64, 16 (these are real-valued channels; when hypercomplex layers are considered, channels are grouped in n-tuples, hence these numbers should be divided by n for a PHM/n convolution or deconvolution. For quaternion layers, $n = 4$). The first two deconvolutions are also followed by Dropout layers. The last convolutional output is processed by a real-valued 1×1 convolution in order to drop the number of channels to the number of classes $K = 5$. This K-depth map is essentially a stack of class-specific activation maps, with each one corresponding to a different class. This map is followed naturally by a sigmoid activation. Note that, a sigmoid activation is used as each pixel may correspond to zero, one, or more classes, i.e. classes are not mutually exclusive. Hence, the generator is structured so as to take a document image as input, and produce a K-level activation map.

The discriminator is structured as follows: A total of six convolutional blocks comprise the discriminator, each having a convolutional layer and batch normalization. Tensor (real-valued) depths for each block are 16, 64, 128, 256, 256, 16. Then follow a sum over channel values, topped by a sigmoid activation. The input of the discriminator is the document image with the (real or fake) annotation activation map, and the result is an estimate of the model whether the document with the provided annotation seem like an "authentic" pair or not.

Regarding situations where training and evaluation classes are not balanced, several strategies have been proposed, as proceeding with standard loss and/or no resampling or reweighting is non-optimal. Briefly, one major strategy is using resampling in order to prioritize classes that are initially under-represented, by sampling more augmented samples from low-volume classes. The other major strategy is to tweak the loss function, so that under-represented classes are artificially assigned a larger loss. In this manner, training can be implicity manipulated towards an optimum that classifies small classes as well the larger ones. In this work, we have used a reweighting strategy, and balanced each per-class loss term inverse proportionally to the number of "positives" of each class.

After taking into account the aforementioned considerations, the required penalties are combined to a single, multi-task term as $L_{\text{total}} = L_{\text{adversarial}} + \lambda L_{bce}$, where we have $L_{adversarial} = E_x[logD(y)] + E_x[log(1 - D(G(x)))]]$, and $L_{bce} = E_{x,y}BCE(G(x), y)$, where λ_{bce} is a hyperparameter that controls the relative importance of the BCE to the adversarial term on the generator. The BCE term considers inconsistency of the estimate map $G(x)$ to ground truth y, by accounting divergence w.r.t. all class separately.

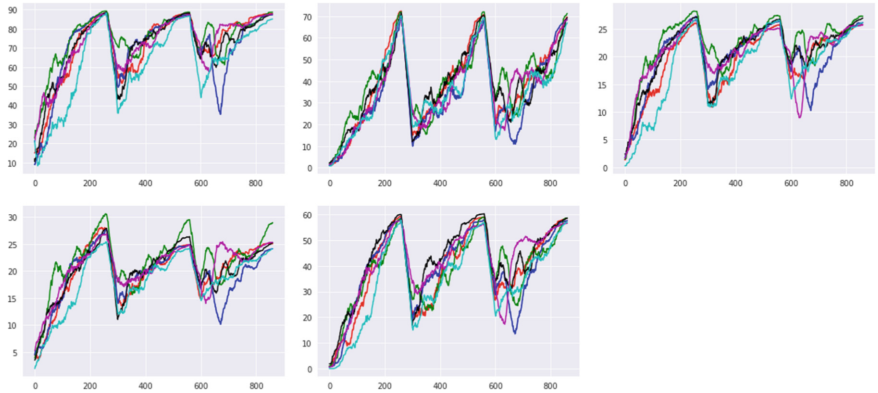


Fig. 3. Plots for resulting Intersection over Union (IoU) values. From top row to bottom row, in "reading order", plots correspond to classes: Handwritten text, Stamp, Matrix, Signature, Page number. Colours correspond to type of employed GAN: Vanilla/real-valued GAN (red), Quaternion GAN (blue), PHM/n = 2 (green), PHM/n = 4 (black), PHM/n = 8 (magenta), PHM/n = 16 (cyan). (Color figure online)

3 Experiments

The PIOP-B dataset consists of 203 manuscripts, scanned from the archives of the Greek *Peiraiki-Patraiki* bank. The dataset has been partitioned into a training and a test set with 162 and 41 document images respectively. All pages have been manually annotated with respect to 5 semantic classes (plus background). These classes are: 1. Handwritten text, 2. Stamp, 3. Matrix, 4. Signature, 5. Page

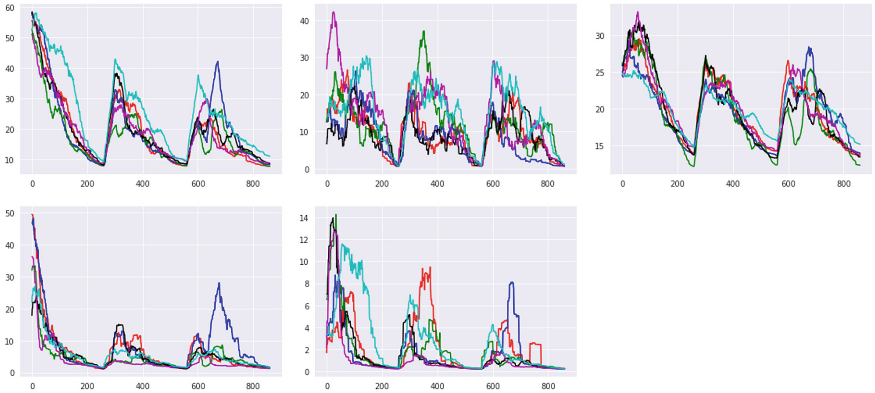


Fig. 4. Plots for resulting test binary cross entropy values. Lower values are better. From top row to bottom row, in "reading order", plots correspond to classes: Handwritten text, Stamp, Matrix, Signature, Page number. Colours correspond to type of employed GAN: Vanilla/real-valued GAN (red), Quaternion GAN (blue), PHM/n = 2 (green), PHM/n = 4 (black), PHM/n = 8 (magenta), PHM/n = 16 (cyan). (Color figure online)

number. Sample pages can be viewed in Fig. 1. An example manually annotated page can be examined in Fig. 2. Note that a single pixel may correspond to multiple labels.

The choices of hyperparameters for our model are as follows. We used a batch size equal to 14, multi-task hyperparameter equal $\lambda = 10$, learning rates for the generator and discriminator equal to 0.01 and 0.001 respectively. All convolutions and deconvolutions use a kernel size equal to 4×4 and stride equal to 2. Leaky ReLU split-activation uses parameter equal to 0.2, and Dropout uses probability parameter equal to 0.5. The Adam optimizer was used to train the models, with cosine annealing and restarts at every 300 epochs, with training for a total of 900 epochs for each case. Activation functions on the hypercomplex networks were "split-type", i.e. all activations treated their inputs as if they were sets of real numbers. Input document images have been sub-sampled to a fixed size, equal to 512×704. This resolution has been chosen with the consideration of preserving aspect ratio as much as possible, while easing computational burden and enabling batch processing during training. Furthermore, as the hypercomplex models treat channels as n-sized groups, care has been taken so as the input and intermediate processing tensors in the generator and discriminator are sized as multiples of the considered values for n, i.e. 2, 4, 8, 16. This was necessary in order to ensure a fair comparison between compared real-valued and hypercomplex models.

Numerical results comparing the vanilla (standard/real-valued) GAN model using the described architecture, against hypercomplex variants, are reported in detail in Table 2. The compared architectures are: vanilla GAN, i.e. a standard GAN with real-valued parameters; this is used as the baseline model. Quaternion GAN and the PHM/n models use hypercomplex layer, input values, intermediate

Table 2. Comparison of GAN models. All models bear the same architectural structure in both generator and discriminator, with the difference being in the type of convolution and deconvolution layers. The "vanilla" variant is a standard, real-valued GAN. The "quaternion" variant uses quaternionic convolutions and deconvolutions, while the PHM variants use parameterized hypercomplex versions of the same layers, for different values of the hyperparameter n. "Economy" denotes how smaller the model is, compared to the vanilla version (rounded up to a single decimal).

Model type	IoU	Test BCE	Generator size	Discriminator size	Economy
Vanilla	53.3	4.74	2, 280, 117	3, 886, 688	0%
PHM (n = 2)	55.5	4.42	1, 141, 493	1, 945, 232	−49.9%
Quaternion	53.0	4.92	572, 085	974, 432	−74.9%
PHM (n = 4)	53.9	4.86	572, 597	974, 816	−74.9%
PHM (n = 8)	52.5	4.96	291, 509	492, 128	−87.3%
PHM (n = 16)	52.3	5.82	177, 845	270, 944	−92.7%

network values, and split-type activations as described in Sect. 2. The difference between PHM/n models is in the choice of the hyperparameter n. Concerning PHM/n = 4 and Quaternion GAN, network values are grouped in quaternions in both cases, but in PHM/n = 4 the $\{A_i\}_{i=1}^4$ matrices are learnable, unlike the Quaternion GAN where they fixed w.r.t. the Hamilton product definition. Intersection over Union figures were computed after binarizing the sigmoid-activated map using threshold = 0.5. Additional plots are in Fig. 3 and Fig. 4, showing the progression of Intersection over Union (higher is better) and test Binary Cross Entropy (lower is better) per semantic class.

Perhaps the most straightforward conclusion is that all models attain evaluation measures that are comparable to each other. Concerning the attained IoU value of ∼55%, it is hampered chiefly by the results in the Matrix and Signature classes, which are perhaps objectively the hardest to detect correctly. The Matrix class for example, differs to Handwritten text only in terms of the way the content is structured, and table horizontal and vertical lines are not necessarily always present. The other classes – Handwritten text, Stamp, Page Number attain higher values. In particular, the final values attained in terms of IoU, per class, for the PHM/n = 2 model are: 88.7, 72.4, 27.2, 30.8, 58.4 for Handwritten text, Stamp, Matrix, Signature, Page number respectively.

The great advantage of the proposed PHM models is that they manage to attain results that are in the ballpark of the standard, real-valued model, by using only a fraction of the model's computational footprint, measured in terms of model size. All versions, including the most light one, PHM/n = 16, which uses only 7.3% of the vanilla model size, give similar, competitive end-results. Also, we note the slight improvement over the vanilla model attained IoU/loss values for the PHM/n = 2 model; while such a difference could be statistically insignificant, it is not inconsistent to similar comparisons in the literature [7,16, 21]. In this respect, we believe that we must also take into account the size of each model with respect to it being a defining factor during model training. Indeed, in model training essentially we look for an optimal value in a search space; if this search space is significantly smaller than a baseline space, then attaining a good value should (all other factors being equal) be significantly easier for the smaller model. Apparently, PHM nets share this benefit while (crucially!) not leading to significant loss w.r.t. the model expressiveness.

4 Conclusion and Future Work

In this work, we have validated the usefulness of parameterized hypercomplex multiplications used in the context of a GAN architecture, comprised of convolutional and deconvolutional layers. Tested in a document labeling scenario, the proposed Hypercomplex GANs attain accuracy that is comparable to that of the GAN baseline, but using only a small fraction of the baseline cost, measured in terms of model size. As future work, we plan to extend the labeling model by considering more modern reweighting/class-balancing methods [1], exploring the use of Ising-like smoothness penalties [2,18,19], or experiment with using keyword spotting or shape [6] as a semantic content cue. In terms of hypercomplex components, we also plan to experiment with activation functions that are also fully hypercomplex (non "split-type"). Also, an interesting point would be whether imposing an orthogonality constraint on the PHM components (A_i matrices) could improve or accelerate learning.

Acknowledgments. This research has been partially co-financed by the EU and Greek national funds through the Operational Program Competitiveness, Entrepreneurship and Innovation, under the calls "OPEN INNOVATION IN CULTURE" (project *Bessarion* - T6YB*Π*-00214) and "RESEARCH - CREATE - INNOVATE" (project *Culdile* - T1E*Δ*K-03785).

References

1. Cui, Y., Jia, M., Lin, T.Y., Song, Y., Belongie, S.: Class-balanced loss based on effective number of samples. In: Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR), pp. 9268–9277 (2019)
2. Dimitrakopoulos, P., Sfikas, G., Nikou, C.: ISING-GAN: annotated data augmentation with a spatially constrained generative adversarial network. In: 2020 IEEE 17th International Symposium on Biomedical Imaging (ISBI), pp. 1600–1603. IEEE (2020)
3. Dyson, F.J.: Quaternion determinants. Helv. Phys. Acta **45**(2), 289 (1972)
4. Ell, T.A., Sangwine, S.J.: Hypercomplex Fourier transforms of color images. IEEE Trans. Image Process. **16**(1), 22–35 (2007)
5. Fraleigh, J.B.: A First Course in Abstract Algebra, 7th edn. (2002)
6. Giotis, A.P., Sfikas, G., Nikou, C., Gatos, B.: Shape-based word spotting in handwritten document images. In: 13th International Conference on Document Analysis and Recognition (ICDAR), pp. 561–565. IEEE (2015)
7. Grassucci, E., Zhang, A., Comminiello, D.: Lightweight convolutional neural networks by hypercomplex parameterization. arXiv preprint arXiv:2110.04176 (2021)

8. Jain, A.K.: Fundamentals of Digital Image Processing. Prentice-Hall Inc., Hoboken (1989)
9. Kuipers, J.B.: Quaternions and Rotation Sequences: A Primer with Application to Orbits, Aerospace and Virtual Reality. Princeton University Press, Princeton (1999)
10. Le Bihan, N., Mars, J.: Singular value decomposition of quaternion matrices: a new tool for vector-sensor signal processing. Signal Process. **84**(7), 1177–1199 (2004)
11. Parcollet, T., Morchid, M., Linarès, G.: 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)
12. Parcollet, T., Morchid, M., Linarès, G.: A survey of quaternion neural networks. Artif. Intell. Rev. **53**(4), 2957–2982 (2019). https://doi.org/10.1007/s10462-019-09752-1
13. Parcollet, T., et al.: Quaternion convolutional neural networks for end-to-end automatic speech recognition. arXiv preprint arXiv:1806.07789 (2018)
14. Ronneberger, O., Fischer, P., Brox, T.: U-Net: convolutional networks for biomedical image segmentation. In: Navab, N., Hornegger, J., Wells, W.M., Frangi, A.F. (eds.) MICCAI 2015. LNCS, vol. 9351, pp. 234–241. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24574-4_28
15. Sangwine, S.J.: Biquaternion (complexified quaternion) roots of- 1. Adv. Appl. Clifford Algebras **16**(1), 63–68 (2006). https://doi.org/10.1007/s00006-006-0005-8
16. Sfikas, G., Giotis, A.P., Retsinas, G., Nikou, C.: Quaternion generative adversarial networks for inscription detection in byzantine monuments. In: Del Bimbo, A., et al. (eds.) ICPR 2021. LNCS, vol. 12667, pp. 171–184. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-68787-8_12
17. Sfikas, G., Ioannidis, D., Tzovaras, D.: Quaternion Harris for multispectral keypoint detection. In: 2020 IEEE International Conference on Image Processing (ICIP), pp. 11–15. IEEE (2020)
18. Sfikas, G., Nikou, C., Galatsanos, N., Heinrich, C.: MR brain tissue classification using an edge-preserving spatially variant Bayesian mixture model. In: Metaxas, D., Axel, L., Fichtinger, G., Székely, G. (eds.) MICCAI 2008. LNCS, vol. 5241, pp. 43–50. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-85988-8_6
19. Sfikas, G., Nikou, C., Galatsanos, N., Heinrich, C.: Majorization-minimization mixture model determination in image segmentation. In: CVPR 2011, pp. 2169–2176. IEEE (2011)
20. Vince, J.: Quaternions for Computer Graphics. Springer, Cham (2011). https://doi.org/10.1007/978-1-4471-7509-4
21. Zhang, A., et al.: Beyond fully-connected layers with quaternions: parameterization of hypercomplex multiplications with $1/n$ parameters. arXiv preprint arXiv:2102.08597 (2021)
22. Zhang, F.: Quaternions and matrices of quaternions. Linear Algebra Appl. **251**, 21–57 (1997)