Keyword Spotting with Quaternionic ResNet: Application to Spotting in Greek Manuscripts

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

[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 Computer Science and Engineering, University of Ioannina, Ioannina, Greece
{agiotis,cnikou}@cse.uoi.gr

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

Abstract. Quaternionized versions of standard (real-valued) neural network layers have shown to lead to networks that are sparse and as effective as their real-valued counterparts. In this work, we explore their usefulness in the context of the Keyword Spotting task. Tests on a collection of manuscripts written in modern Greek show that the proposed quaternionic ResNet achieves excellent performance using only a small fraction of the memory footprint of its real-valued counterpart. Code is available at https://github.com/sfikas/quaternion-resnet-kws.

Keywords: Quaternions · Keyword spotting · Document image processing · Modern greek

1 Introduction

Keyword spotting (KWS) in document images is the go-to application when a search for a specific word or words and their instances inside a digitized document is required, but full recognition of the document may not be the optimal option. Handwritten Text Recognition (HTR) is in general not a trivial task, especially when the collection includes many different writing styles, is of bad digitization quality or is heavily degraded. Furthermore, while learning-based HTR has had many spectacular successes owing to the use of deep learning, accurate recognition relies on a well-trained system, which in turn requires a large and diverse annotated training set. Transferability of a model, i.e. training

S. Uchida et al. (Eds.): DAS 2022, LNCS 13237, pp. 382–396, 2022.
https://doi.org/10.1007/978-3-031-06555-2_26

for a specific style or styles of handwriting and applying to a different target is usually not to be considered as granted. For these reasons, KWS is still a very much viable alternative. Modern KWS methods also rely heavily on deep learning models [21], however in comparison to HTR methods and the training data requirements, they tend to be less resource-hungry as the task is objectively simpler.

Since the advent of convolutional neural networks for KWS [20,23], the literature on KWS methods largely follows and builds on the developments in deep learning for signal or vision applications in general. A recent trend is the use of layers that challenge the status of the two-dimensional canonical grid of input images and feature maps as an immutable parameter; graph convolutional networks operate on word graph inputs, and [35] use a neural network to map word image graph representations to Pyramidal Histogram of Character (PHOC) descriptions. With respect to a wider application scope, a recent trend in machine learning is the use of self-attention layers, popularized in the transformer model [32] initially proposed for a natural language processing setting. The mechanism of self-attention aspires to completely replace two other "pillars" of neural networks, namely fully-connected layers and convolution layers, and indeed has found success in a diverse array of application contexts. In computer vision it has quickly been employed in many different applications [9].

Creating sparse, light-weight neural networks is another important trend [19]. Many different techniques have been proposed in this regard, in an effort to create networks that are as efficient as networks that are larger and/or slower. One such technique is the use of quaternion network layers, which replace standard (real-valued) inputs, parameters and outputs with quaternion-valued components [1,38]. Quaternions form an algebra of intrinsically four-dimensional objects, equipped with its special version of the multiplication operation. Except a very well-known application in representing rotations in space, quaternion algebra has found uses in digital image processing and vision [3,10,26]. Starting in the 90s, a parallel line of research has given formulations that described neural networks with non-real values for inputs, features and layers [1,11,13,14]. Complex and hypercomplex neural networks have been rediscovered recently, with applications that further extended the scope of the first formulations. Following the deep learning "revolution" of the latter half of the 2010s,s, deep complex neural networks have been proposed in 2017 [31], and they were very soon to be followed by papers on quaternion neural networks [6,16,38], published almost simultaneously. These works covered quaternion versions for dense, convolution and recurrent layers. Later works [15] have explored quaternion versions of other layers, models and diverse applications (e.g. Generative Adversarial Networks [25] or CNN-RNN networks for speech recognition [16]). Also, theoretical extensions of the quaternionic layer were considered, which included exploring higher hypercomplex dimensions [2] or parameterizing a generic hypercomplex operation [30,36].

In the current work, we propose employing quaternion layers combined with a ResNet architecture, in order to create a light-weight KWS system. In a nutshell, compared to related prior work, our contribution with this paper is:

– the introduction of a Quaternionic version of the ResNet block, and the Quaternionic ResNet as a network that is comprised of this type of blocks.
– testing the proposed model for the problem of KWS, and showing that it can be as effective as its real-valued counterpart. The important advantage here (present also in quaternion networks in general) is that the network requires only one quarter of the parameters of a real-valued model that uses the same architecture (same number of layers, and Quaternionic layer blocks in place of standard layer blocks).

The remainder of this paper is structured as follows. In Sect. 2 we present elementary notions concerning the algebra of quaternions, and discuss how it is applied to create quaternionic variants of neural networks. We explain why quaternionic layers inherently lead to lightweight/"less-costly" networks in Sect. 3. In Sect. 4 we review the proposed quaternionic network for keyword spotting. Section 5 numerically confirms our approach compared to non-quaternionic architectures. We close the paper with Sect. 6 where we draw our conclusions and discuss future work.

2 Quaternions in Neural Networks

2.1 Elementary Notions

Quaternions have been introduced in the 19^{th} century by Hamilton [34]. It was presented as a special kind of algebra over quadruples of real numbers, after having being realized that a triplet-based algebraic structure would necessarily be inherently constrained in terms of its properties. The definition of a quaternion is:

$$q = \alpha + \beta \boldsymbol{i} + \gamma \boldsymbol{j} + \delta \boldsymbol{k}, \tag{1}$$

where $\alpha, \beta, \gamma, \delta$ are real numbers and $\boldsymbol{i}, \boldsymbol{j}, \boldsymbol{k}$ are imaginary units. This definition is reminiscent of the definition of complex numbers, and indeed one may interpret quaternions $\mathbb{H}$ as a generalization of $\mathbb{C}$. The important difference is that while we have a single "real" part (α) in both $\mathbb{C}$ and $\mathbb{H}$, in quaternions we have three *independent* imaginary parts (β, γ, δ) and an equal number of imaginary units/axes ($\boldsymbol{i}, \boldsymbol{j}, \boldsymbol{k}$). This relation is made more evident with the Cayley-Dickson form of quaternions:

$$q = \chi + \psi \boldsymbol{j}, \tag{2}$$

where now χ and ψ are *complex* coefficients, $\chi = \alpha + \beta \boldsymbol{i}, \psi = \gamma + \delta \boldsymbol{i}$. The form Eq. 2 is equivalent to Eq. 1. One may arrive from one to the other after considering the following multiplication rule for imaginary units:

$$\boldsymbol{i}^2 = \boldsymbol{j}^2 = \boldsymbol{k}^2 = \boldsymbol{ijk} = -1,$$

$$\boldsymbol{ij} = -\boldsymbol{ji} = \boldsymbol{k}, \boldsymbol{jk} = -\boldsymbol{kj} = \boldsymbol{i}, \boldsymbol{ki} = -\boldsymbol{ik} = \boldsymbol{j}. \tag{3}$$

As illustrated in Eq. 3, multiplication for quaternions is not commutative, and for imaginary units in particular, changing the order of multiplications leads

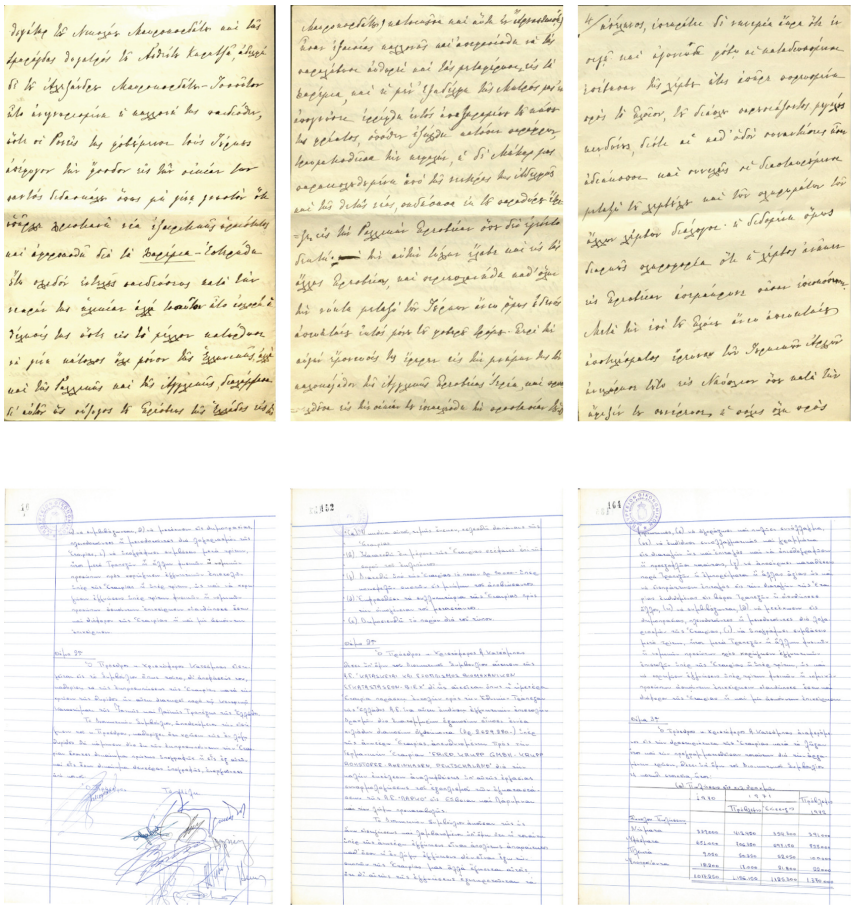

Fig. 1. Sample pages from the two datasets used in this work, "Memoirs" (top row) and "PIOP-DAS" (bottom row).

to the opposite of the initial result. In general, we can write that $pq \neq qp$ for $p, q \in \mathbb{H}$.

Addition is defined simply by adding up respective coefficients, and retains the "usual" properties with no big surprises (commutative law, associative law, and the distributive property combined with the aforementioned multiplication rule). These considerations lead to the following multiplication rule:

$$\begin{aligned} pq = &(\alpha_p\alpha_q - \beta_p\beta_q - \gamma_p\gamma_q - \delta_p\delta_q) + \\ &(\alpha_p\beta_q + \beta_p\alpha_q + \gamma_p\delta_q - \delta_p\gamma_q)\boldsymbol{i} + \\ &(\alpha_p\gamma_q - \beta_p\delta_q + \gamma_p\alpha_q + \delta_p\beta_q)\boldsymbol{j} + \\ &(\alpha_p\delta_q + \beta_p\gamma_q - \gamma_p\beta_q + \delta_p\alpha_q)\boldsymbol{k}, \end{aligned} \tag{4}$$

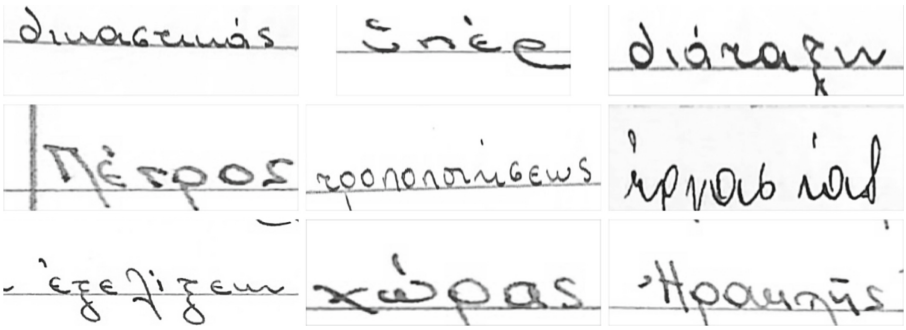

Fig. 2. Word image samples from the "PIOP-DAS" dataset.

where $p = \alpha_p + \beta_p \boldsymbol{i} + \gamma_p \boldsymbol{j} + \delta_p \boldsymbol{k}$ and $q = \alpha_q + \beta_q \boldsymbol{i} + \gamma_q \boldsymbol{j} + \delta_q \boldsymbol{k}$. This rule can also be shorthanded as:

$$pq = S(p)S(q) - V(p) \cdot V(q) + S(p)V(q) + S(q)V(p) + V(p) \times V(q), \tag{5}$$

where $\cdot$ and $\times$ denote the dot and cross product respectively, $S(\cdot)$ is the "scalar" part of the quaternion and $V(\cdot)$ is the "vector" part of the quaternion (i.e. $S(p) = \alpha \in \mathbb{R}$ and $V(p) = [\beta\ \gamma\ \delta]^T \in \mathbb{R}^3$).

Other useful relations include the definition of a quaternion conjugate, and a quaternion magnitude. These are defined as follows:

$$\bar{q} = \alpha - \beta \boldsymbol{i} - \gamma \boldsymbol{j} - \delta \boldsymbol{k}, \tag{6}$$

and

$$|q| = \sqrt{q\bar{q}} = \sqrt{\bar{q}q} = \sqrt{\alpha^2 + \beta^2 + \gamma^2 + \delta^2}. \tag{7}$$

Again, analogies to complex numbers can be straightforwardly drawn, considering $\gamma = 0, \delta = 0$ to obtain the well-known relations for complex numbers. Finally, we note that matrix calculus can be extended to matrices with quaternionic elements, $\mathbb{H}^{m \times n}$ [37].

2.2 Quaternionized Versions of Standard NN Layers

Quaternion Neural Networks (QNNs) are defined as neural networks that have quaternion-valued inputs, outputs and parameters. In order to deal with providing quaternion-valued inputs where real-valued scalars and matrices are available, the respective structures are padded with zero channel values when the dimensionality of the input is $d < 4$. In the present use-case, grayscale images are inherently one-dimensional -valued in each pixel, and colour images would be three-dimensional -valued, accounting for the Red, Green and Blue channels (using a different colorspace would lead to an analogous consideration). Hence in particular, a scalar value a is mapped to quaternion $a + 0\boldsymbol{i} + 0\boldsymbol{j} + \boldsymbol{k}$, and a colour value $[r, g, b]^T$ is mapped to quaternion $0 + r\boldsymbol{i} + g\boldsymbol{j} + b\boldsymbol{k}$ by convention.

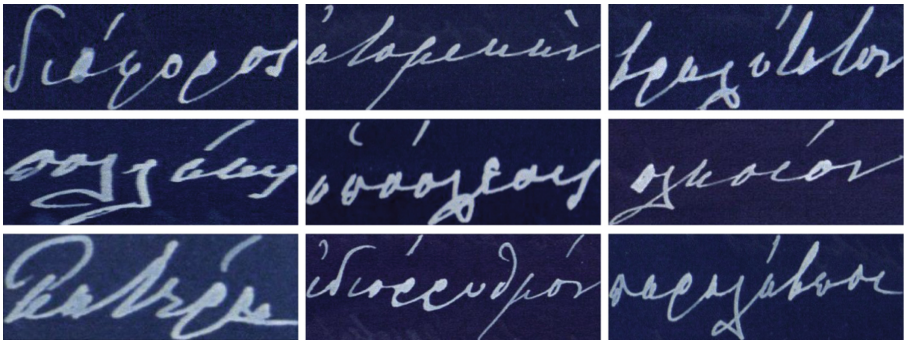

Fig. 3. Word image samples from the "Memoirs" dataset.

Dense Layer. A QNN comprises a cascade of layers as is the case with real-valued neural networks. The dense or fully-connected layer for quaternions can be written as

$$f_{dense}(x; W, h) = Wx + h$$

where $x \in \mathbb{H}^N$ is a quaternion-valued input vector, $W^{M \times N}$ is a quaternion-valued matrix of weights and $h \in \mathbb{H}^N$ denotes a quaternion-valued bias term. We assume that the input is of dimensionality equal to N, and the output dimensionality is equal to M.

Convolution Layer. Regarding convolution, we must note that a number of different options here are possible regarding the exact form of the convolution operation. In particular for two-dimensional convolution, which is of interest in an image processing network, there is a left-sided convolution, a right-sided convolution, and also a two-sided convolution (or "bi-convolution") [4], with the difference being in whether the convolution kernel multiplies the signal from the left or the right. The two-sided option corresponds to the case where an horizontal kernel multiplies the signal from the left, and a vertical kernel multiplies the signal from the right. In this work, we choose to use the left-multiplying convention, so formally we have:

$$f_{conv}(m; K) = K * m,$$

where $m \in \mathbb{H}^{M \times N}$ is the input feature map, and $K \in \mathbb{H}^{d \times d}$ is the convolution kernel.

Activation Functions. We use "split - activation" functions to handle non-linearities in the quaternion domain, which means that each quaternion channel is treated by the activation function separately, as if it were part of a tensor valued in $\mathbb{R}^4$. For our architecture, we use split-activation versions of standard real-valued functions (Rectified Linear Unit, the leaky Rectified Linear Unit, sigmoid). The same rationale is followed for other network components: the dropout layer and the batch normalization layer are applied as if a quaternionic tensor $H \times W \times D$ were its real isomorphic image $H \times W \times 4D$.

3 Why Are Quaternionic Layers Less Costly?

The main motivation behind using a quaternionic network for a given task is that quaternionic layers are inherently less costly than corresponding standard (real-valued) layers (And more importantly, without significant sacrifices in terms of performance). "Cost" here is to be understood in terms of total required *independent* parameters. The reason for this useful trait is that the definition of quaternionic layers comes with extensive parameter sharing. This is due to i) the definition of quaternionic ("Hamilton") multiplication itself, and ii) that every four real-valued tensor channels are grouped together and mapped to quaternion real/imaginary components. For example, an input color image, comprising of 3 channels plus 1 zero-padded channel would be mapped as a single-channel quaternionic 2D signal.

Let us illustrate this with a simple example. Assume a real-valued linear layer, without bias or activation, that transforms an input comprised of 4 neurons to an output of 4 neurons. In other words, an input $x \in \mathbb{R}^4$ is mapped to an output $y \in \mathbb{R}^4$, and the operation can be written as the matrix-vector multiplication:

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} \alpha & \beta & \gamma & \delta \\ \epsilon & \zeta & \eta & \theta \\ \iota & \kappa & \lambda & \mu \\ \nu & \xi & o & \pi \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}, \tag{8}$$

where greek letters $\alpha, \beta, \cdots, \pi$ denote the $4 \times 4 = 16$ operation parameters. Assuming that we want to define a quaternionic linear layer on the same input and a same-sized output, the input and output vectors would be mapped to a single quaternion each; we shall again denote these quaternions using the notation x_1, x_2, x_3, x_4 and y_1, y_2, y_3, y_4, where components 1–4 correspond to the real and the the three imaginary quaternion components. As our input and output, under these terms, is a single quaternion and a single quaternion respectively, a quaternionic linear layer is composed of a single multiplication operation. This operation can be written as:

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} \alpha & -\beta & -\gamma & -\delta \\ \beta & \alpha & -\delta & \gamma \\ \gamma & \delta & \alpha & -\beta \\ \delta & -\gamma & \beta & \alpha \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}, \tag{9}$$

where we have re-written Eq. 4 as a matrix-vector multiplication and changed notation accordingly. The parameters of the operation are again 16 as in Eq. 8, but they are grouped into 4 groups of 4, or in other words we have 4 *independent* parameters.

This paradigm is extended to cover any-size inputs and outputs, as long as they are multiples of 4. If they are not multiples of 4, they can easily be padded using zero-valued channels to the next closest multiple of 4 (The network will easily learn to ignore the paddings, hence there is no real overhead involved). Alternatively, an operation such as 1×1 convolution can be used to map a

real-valued input to the desired channel multiple [30]. In general, an operation that is written as $y = Wx$, where $y \in \mathbb{R}^{4K}$, $x \in \mathbb{R}^{4L}$ and $W \in \mathbb{R}^{4K \times 4L}$ is thus mapped to a quaternionic operation $\hat{y} = \hat{W}\hat{x}$, where $\hat{y} \in \mathbb{H}^K$, $\hat{x} \in \mathbb{H}^L$ and $\hat{W} \in \mathbb{H}^{K \times L}$. Parameter vector $\hat{W}$ only contains $4 \times K \times L$ parameters compared to $4 \times 4 \times K \times L$ of W, hence we have a $4\times$ saving.

The above considerations hold for any operation, as long as it can be written as a linear transformation, or a composition that includes linear transformations. Hence, not only linear layers can be quaternionized, but convolutions [8,38] and deconvolutions [30], or resnet blocks as in the current work.

4 Proposed Model

The proposed model is structured as a feed-forward convolutional network, accepting a batch of word images as input, and processing them into a batch of fixed-size PHOC descriptor targets [24]. The input is set to the real part of the quaternion input map, and imaginary components are set to zero.

In our architecture, we group QNN processing layers in ResNet blocks. Each block groups a cascade of quaternion convolutions, structured as follows: We assume that a feature map x is the input of the block. A quaternion convolution of stride equal to 1 and kernel size equal to 3×3 is followed by a batch normalization layer and a ReLU activation. The result of this step is convolved by a second quaternion convolution with the same operation characteristics, followed by a second batch normalization layer. The output $\phi(x)$ is connected with the input x via a skip connection and followed by a final ReLU activation, so the block output can be written formally as $ReLU(\phi(x) + x)$.

On a high level, the network is composed as a sequence of a convolutional backbone, followed by a pooling and flattening layer, which in turn is topped by a fully-connected head. The backbone is made up of 7 ResNet blocks. These are parametrized, in terms of the number of input and output channels respectively, as follows: $(1, 16)$, $(16, 32)$, $(32, 64)$, $(64, 64)$, $(64, 64)$, $(64, 128)$, $(128, 32)$, where the n^{th} tuple is equal to (#input channels, #output channels). Pooling is performed using pyramidal spatial pooling with 3 layers and an output size of $2,688$. Then, two quaternionic dense layers follow, mapping first to 256 neurons before passing to a dropout layer and the output logits. The logits are transformed into per-attribute (unigrams, bigrams) probabilities with a split-activation sigmoid function. The quantity to be optimized by the network is a binary cross-entropy (BCE) loss function, measuring divergence of the attribute estimate probabilities against the true PHOC values.

5 Experiments

5.1 Datasets

For our experiments, we have used two different collections of handwritten pages. The manuscripts are written in the modern greek language. The very few non-greek words which exist in the set have been manually removed for the tests,

Table 1. Model size comparison, in terms of total number of trainable parameters. For the quaternionic variants, the equivalent number of real-valued parameters is reported, to ease comparisons.

Model/Network type	"Small" size	"Standard" size
Quaternionic Resnet	1,317,428	5,946,164
Real-valued Resnet	5,233,840	23,730,864

as training would be too difficult for them due to their small sample size. Both datasets have been manually segmented into a number of word images, which we use as queries and retrieval targets for our Query-by-Example keyword spotting trials. Also, both datasets use the polytonic Greek script [24]. However, for the PIOP-DAS dataset we have only *monotonic* annotations, meaning that only the "acute" accent and the "diairesis" are used, and the actually existing text diacritics are either represented by an acute accent (this is the case for the grave accent and the circumflex) or not mapped at all (this is the case for the smooth and rough breatings, subscript). Samples of the pages contained in the set are shown in Fig. 1.

PIOP-DAS Dataset. The PIOP-DAS dataset consists of 22 manuscripts, that were scanned from the archives of the Greek "Πειραϊκή-Πατραϊκή " (*"Peiraiki-Patraiki"*) bank. The documents record 4 sessions of the bank governing committee, held between December 1971–April 1972. The manuscript digitizations have been segmented into a total of 12,362 words. 80% of the dataset was assigned as the training set, and the rest is used as the test set. In absolute figures, 9,341 words and 3,021 words are assigned to the training and test set respectively. We use as queries all words in the test set, except those that appear only once. Examples of segmented words from this set are shown in Fig. 2. The dataset is publicly available at https://github.com/sfikas/piop-das-dataset.

Memoirs Dataset. The Memoirs dataset [5,7] comprises 46 manuscripts, written in the late 19^{th} century in the form of a personal diary. The text is written by a single author, Sophia Trikoupi, who was the sister of the Greek prime minister Charilaos Trikoupis. A total of 4,941 is available in this dataset. We use the training and test partitions defined in [24], which comprise 2,000 words each. There are 941 remaining words that correspond to the validation set, which we do not use in this work. To select queries, we "initially" use the words selected in [5]. [5] uses a learning-free KWS baseline, so this query list is trimmed down subsequently here as some words do not exist in the test set. Examples of segmented words from this set are shown in Fig. 3. The dataset is publicly available at https://github.com/sfikas/sophia-trikoupi-handwritten-dataset/.

5.2 Hyperparameters and Other Training Considerations

Concerning the encoding of the Greek script into unigram descriptor bins, we choose to use separate bins for unaccented and accented characters, following [24] (for example, ω and $\acute{\omega}$ are treated as different letters. A shared bin is used for upper-case and lower-case versions of the same letter. We set the learning rate to 10^{-3}, and the batch size to 40. All images are resized to 32×128 pixels and cast as grayscale images before entering the network, either for training or inference. We use the Adam optimizer with weight decay set to 5×10^{-5}, and a cosine annealing scheduler with restarts every 300 epochs, for a maximum of 900 epochs of training. Data augmentation is employed during training, applying a small affine transform on each drawn training sample [22].

5.3 Results

We have run trials testing the proposed Quaternionic model against models that differ in terms of the domain of their structural components (layers, parameters, inputs and outputs) and in terms of their size. With regard to the size of the models, we name the two sizes "standard" and "small". The "standard" model corresponds exactly to the architecture presented in Sect. 4. The "small" model comprises only 3 resnet blocks, instead of 7 blocks for the "standard" model. Experiments were run over both datasets PIOP-DAS and Memoirs. Model training results may be examined in Figs. 4, 5 and Table 2. We report loss and accuracy in terms of the progression of model training, as well as the best accuracy figures by the end of training and best attained overall. We show training and test loss, both computed as a binary cross-entropy average over data on the training and test sets respectively. Over the same plots, model accuracy is measured in terms of mean average precision (MAP).

We can see that in all cases the quaternion-valued and the real-valued models, regardless of size fare equally well, with almost excellent results. In one case –in particular, Memoirs/"small" network – the quaternionic network achieves better accuracy figures that its real counterpart. We believe however that the most important trait of the quaternionic models is related to the size of the respective architectures, as reported in Table 1. Therein, we see that quaternionic

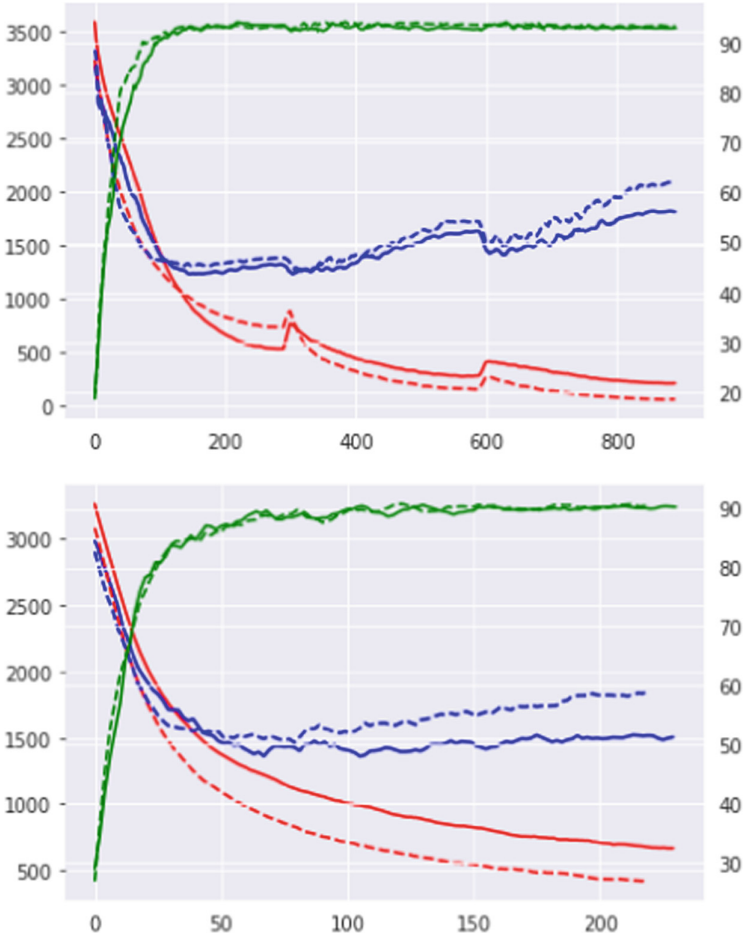

Fig. 4. Results for the "PIOP-DAS" dataset: The top (bottom) plot shows results for the "standard" ("small") network. Training set BCE loss, test set BCE loss and MAP are shown using red, blue and green colors respectively. Solid lines correspond to the proposed Quaternionic models. Dashed lines correspond to real-valued networks with the same architecture as the proposed networks, but using standard real-valued components in place of the quaternionic ones. The horizontal axis corresponds to the number of training epochs. The vertical axis corresponds to loss value (shown on the left, lower is better) and map accuracy percentage (shown on the right, higher is better). The plots were smoothed with a uniform kernel of size 10 to ease visualization.

models enjoy a much smaller size in terms of number of trainable parameters (see discussion in Sect. 3 for a theoretical justification of this result). The eventual gain is an operation of significantly lower complexity.

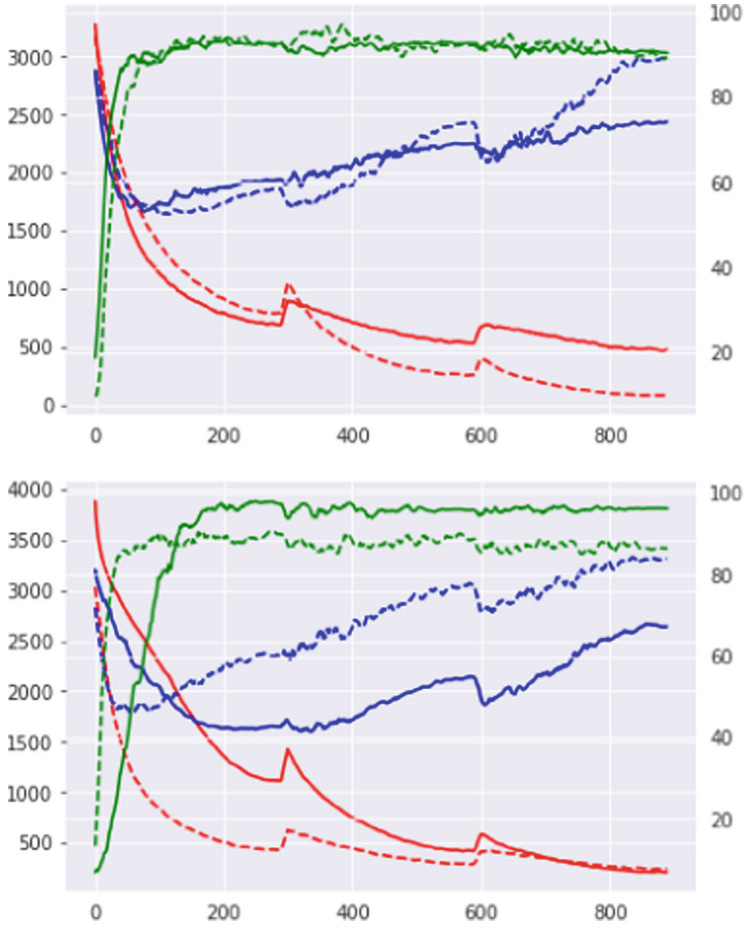

Fig. 5. Results for the "Memoirs" dataset: The top (bottom) plot shows results for the "large" ("small") network. Training set BCE loss, test set BCE loss and MAP are shown using red, blue and green colors respectively. Solid lines correspond to the proposed Quaternionic models. Dashed lines correspond to real-valued networks with the same architecture as the proposed networks, but using standard real-valued components in place of the quaternionic ones. The horizontal axis corresponds to the number of training epochs. The vertical axis corresponds to loss value (shown on the left, lower is better) and map accuracy percentage (shown on the right, higher is better). The plots were smoothed with a uniform kernel of size 10 to ease visualization (Color figure online).

6 Conclusion and Future Work

With this work we have introduced a novel Quaternionic ResNet block and validated the value of using Quaternionized versions of standard (i.e., real-valued) layers in the context of the document keyword spotting task. We have used a

Table 2. Model accuracy in terms of mean average precision (higher is better). Results correspond to the end of training and best figure reported over all epochs.

MAP%	Memoirs		PIOP-DAS	
	Last epoch	Best epoch	Last epoch	Best epoch
Quaternion/Standard	96.1%	98.6%	92.9%	94.3%
Real/Standard	89.3%	98.6%	93.3%	94.6%
Quaternion/Small	90.2%	94.6%	90.3%	92.8%
Real/Small	86.3	94.5%	90.0%	92.2%

PHOC-based architecture as our baseline, and extended it to a Quaternion Neural Network. Our results show that the proposed QNN has achieved excellent retrieval performance while being much less resource-demanding compared to non-quaternionic networks, in terms of model size. Concerning the new PIOP dataset, we plan to publish a much extended version of the dataset, along with richer annotation meta-data, in the near future. Regarding future work, we plan to continue with more extensive experiments, which may include the newer developments in the field of applications of hypercomplex layers in neural networks [36]. Other considerations include working with more complex network architectures [17,33,35] or combining with a probabilistic, Bayesian paradigm using a more "classic" [18,27–29] or a more modern formulation [12].

With respect to new research directions, perhaps an interesting question would be *why* are quaternion networks (and in general, hypercomplex networks) so effective. We believe that a factor here is the ease of "navigation" on a much more compact parameter space during learning; there may however exist other, more important factors that are related to this (unreasonable?) effectiveness.

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 "RESEARCH - CREATE - INNOVATE" (project *Culdile* - code T1EΔK-03785, project *Impala* - code T1EΔK-04517) and "OPEN INNOVATION IN CULTURE" (project *Bessarion* - T6YBΠ-00214).

References

1. Arena, P., Fortuna, L., Occhipinti, L., Xibilia, M.G.: Neural networks for quaternion-valued function approximation. In: Proceedings of IEEE International Symposium on Circuits and Systems-ISCAS 1994, vol. 6, pp. 307–310. IEEE (1994)
2. Bojesomo, A., Liatsis, P., Marzouqi, H.A.: Traffic flow prediction using deep sedenion networks. arXiv preprint arXiv:2012.03874 (2020)
3. Ell, T.A., Le Bihan, N., Sangwine, S.J.: Quaternion Fourier Transforms for Signal and Image Processing. Wiley, Hoboken (2014)
4. Ell, T.A., Sangwine, S.J.: Hypercomplex Fourier transforms of color images. IEEE Trans. Image Process. **16**(1), 22–35 (2007)

5. Gatos, B., et al.: GRPOLY-DB: An old Greek polytonic document image database. In: Proceedings of the International Conference on Document Analysis and Recognition (ICDAR), pp. 646–650. IEEE (2015)
6. Gaudet, C.J., Maida, A.S.: Deep quaternion networks. In: 2018 International Joint Conference on Neural Networks (IJCNN), pp. 1–8. IEEE (2018)
7. 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)
8. Grassucci, E., Zhang, A., Comminiello, D.: Lightweight convolutional neural networks by hypercomplex parameterization. arXiv preprint arXiv:2110.04176 (2021)
9. Han, K., et al.: A survey on visual transformer. CoRR abs/2012.12556 (2020). https://arxiv.org/abs/2012.12556
10. Hui, W., Xiao-Hui, W., Yue, Z., Jie, Y.: Color texture segmentation using quaternion-Gabor filters. In: 2006 International Conference on Image Processing, pp. 745–748. IEEE (2006)
11. Isokawa, T., Kusakabe, T., Matsui, N., Peper, F.: Quaternion neural network and its application. In: Palade, V., Howlett, R.J., Jain, L. (eds.) KES 2003. LNCS (LNAI), vol. 2774, pp. 318–324. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-45226-3_44
12. Kobyzev, I., Prince, S.J., Brubaker, M.A.: Normalizing flows: an introduction and review of current methods. IEEE Trans. Pattern Anal. Mach. Intell. **43**(11), 3964–3979 (2020)
13. Leung, H., Haykin, S.: The complex backpropagation algorithm. IEEE Trans. Signal Process. **39**(9), 2101–2104 (1991)
14. Nitta, T.: A quaternary version of the back-propagation algorithm. In: Proceedings of ICNN'95-International Conference on Neural Networks. vol. 5, pp. 2753–2756. IEEE (1995)
15. 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
16. Parcollet, T., et al.: Quaternion convolutional neural networks for end-to-end automatic speech recognition. arXiv preprint arXiv:1806.07789 (2018)
17. Prieto, J.R., Vidal, E.: Improved graph methods for table layout understanding. In: Lladós, J., Lopresti, D., Uchida, S. (eds.) ICDAR 2021. LNCS, vol. 12822, pp. 507–522. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-86331-9_33
18. Prince, S.J.: Computer Vision: Models, Learning, and Inference. Cambridge University Press, Cambridge (2012)
19. Retsinas, G., Elafrou, A., Goumas, G., Maragos, P.: Weight pruning via adaptive sparsity loss. arXiv preprint arXiv:2006.02768 (2020)
20. Retsinas, G., Louloudis, G., Stamatopoulos, N., Gatos, B.: Efficient learning-free keyword spotting. IEEE Trans. Pattern Anal. Mach. Intell. **41**(7), 1587–1600 (2018)
21. Retsinas, G., Sfikas, G., Nikou, C., Maragos, P.: From Seq2Seq recognition to handwritten word embeddings. In: Proceedings of the British Machine Vision Conference (BMVC) (2021)
22. Retsinas, G., Sfikas, G., Stamatopoulos, N., Louloudis, G., Gatos, B.: Exploring critical aspects of CNN-based keyword spotting. a phocnet study. In: Proceedings of the International Workshop on Document Analysis Systems (DAS), pp. 13–18. IEEE (2018)
23. Rusakov, E., Sudholt, S., Wolf, F., Fink, G.A.: Exploring architectures for CNN-based word spotting. arXiv preprint arXiv:1806.10866 (2018)

24. Sfikas, G., Giotis, A.P., Louloudis, G., Gatos, B.: Using attributes for word spotting and recognition in polytonic greek documents. In: Proceedings of the International Conference on Document Analysis and Recognition (ICDAR), pp. 686–690. IEEE (2015)
25. Sfikas, G., Giotis, A.P., Retsinas, G., Nikou, C.: Quaternion generative adversarial networks for inscription detection in byzantine monuments. In: Del Bimbo, A., Cucchiara, R., Sclaroff, S., Farinella, G.M., Mei, T., Bertini, M., Escalante, H.J., Vezzani, R. (eds.) ICPR 2021. LNCS, vol. 12667, pp. 171–184. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-68787-8_12
26. Sfikas, G., Ioannidis, D., Tzovaras, D.: Quaternion Harris for multispectral keypoint detection. In: Proceedings of the International Conference on Image Processing (ICIP), pp. 11–15. IEEE (2020)
27. 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
28. Sfikas, G., Nikou, C., Galatsanos, N., Heinrich, C.: Majorization-minimization mixture model determination in image segmentation. In: Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2169–2176. IEEE (2011)
29. Sfikas, G., Retsinas, G., Gatos, B.: A PHOC decoder for lexicon-free handwritten word recognition. In: Proceedings of the International Conference on Document Analysis and Recognition (ICDAR), vol. 1, pp. 513–518. IEEE (2017)
30. Sfikas, G., Retsinas, G., Gatos, B.: Hypercomplex generative adversarial networks for lightweight semantic labeling. In: International Conference on Pattern Recognition and Artificial Intelligence (2022)
31. Trabelsi, C., et al.: Deep complex networks. arXiv preprint arXiv:1705.09792 (2017)
32. Vaswani, A., et al.: Attention is all you need. In: Advances in neural information processing systems (NIPS), pp. 5998–6008 (2017)
33. Vidal, E., Toselli, A.H.: Probabilistic indexing and search for hyphenated words. In: Lladós, J., Lopresti, D., Uchida, S. (eds.) ICDAR 2021. LNCS, vol. 12822, pp. 426–442. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-86331-9_28
34. Vince, J.: Quaternions for Computer Graphics. Springer, London (2021). https://doi.org/10.1007/978-1-4471-7509-4
35. Wolf, F., Fischer, A., Fink, G.A.: Graph convolutional neural networks for learning attribute representations for word spotting. In: Lladós, J., Lopresti, D., Uchida, S. (eds.) ICDAR 2021. LNCS, vol. 12821, pp. 50–64. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-86549-8_4
36. Zhang, A., et al.: Beyond fully-connected layers with quaternions: parameterization of hypercomplex multiplications with $1/n$ parameters. In: Proceedings of the International Conference on Learning Representations (ICLR) (2021)
37. Zhang, F.: Quaternions and matrices of quaternions. Linear Algebra Appl. **251**, 21–57 (1997)
38. Zhu, X., Xu, Y., Xu, H., Chen, C.: Quaternion convolutional neural networks. In: Proceedings of the European Conference on Computer Vision (ECCV), pp. 631–647 (2018)