Graph Learning in 4D: A Quaternion-Valued Laplacian to Enhance Spectral GCNs

Stefano Fiorini[1], **Stefano Coniglio**[2], **Michele Ciavotta**[3], **Enza Messina**[3]
[1] Italian Institute of Technology, Genoa, Italy
[2] University of Bergamo, Bergamo, Italy
[3] University of Milano-Bicocca, Milan, Italy
stefano.fiorini@iit.it, stefano.coniglio@unibg.it, michele.ciavotta@unimib.it, enza.messina@unimib.it

Abstract

We introduce *QuaterGCN*, a spectral Graph Convolutional Network (GCN) with quaternion-valued weights at whose core lies the *Quaternionic Laplacian*, a quaternion-valued Laplacian matrix by whose proposal we generalize two widely-used Laplacian matrices: the *classical Laplacian* (defined for undirected graphs) and the complex-valued *Sign-Magnetic Laplacian* (proposed to handle digraphs with weights of arbitrary sign). In addition to its generality, our Quaternionic Laplacian is the only Laplacian to completely preserve the topology of a digraph, as it can handle graphs and digraphs containing antiparallel pairs of edges (digons) of different weights without reducing them to a single (directed or undirected) edge as done with other Laplacians. Experimental results show the superior performance of QuaterGCN compared to other state-of-the-art GCNs, particularly in scenarios where the information the digons carry is crucial to successfully address the task at hand.

Introduction

Deep Learning (DL) has recently achieved a striking success, contributing to the advancement of several research areas such as natural language processing (Vaswani et al. 2017), business intelligence (Khan et al. 2020), and cybersecurity (Dixit and Silakari 2021), only to name a few. While many of the most popular DL architectures are designed to process data arranged in grid-like structures (such as RGB pixels in 2D images and video streams), many real-world phenomena are ruled by more general relationships that are better represented by a graph (Bronstein et al. 2017). For example, e-commerce relies on graphs to model the user-product interaction to make recommendations, drug discovery models molecule bioactivity via interaction graphs, and information discovery uses multi-relational knowledge graphs to extract information between the entities (Ye et al. 2022).

Graph Convolutional Networks (GCNs) model the interaction between the entities of a complex system via a graph-based *convolution operator*. They are the primary tool for machine learning tasks on data that enjoy a graph-like structure. In spatial GCNs, the convolution operator is defined as a localized aggregation operator (Xu et al. 2018). There are various implementations and extensions of this operator. E.g., in attention-based GCNs the convolutional operator is used to dynamically learn attention scores on the graph edges (Veličković et al. 2018), while, in recurrence-based GCNs, the operator leverages gated recurrent units to capture edge-specific information and temporal dependencies during message passing (Li et al. 2016).

While spatial GCNs rely on different heuristics for the definition of their convolution operator, *spectral* GCNs are solidly grounded in signal and algebraic graph theory. They rely on a convolution operator based on a Fourier transform applied to the eigenspace of the Laplacian matrix of the graph (Kipf and Welling 2017; He et al. 2022c; Wu et al. 2022) to capture the global structure of the graph. Spectral GCNs have been shown to achieve superior performance to their spatial-based counterpart in a number of papers, see, e.g., (Zhang et al. 2021b; Fiorini et al. 2023). Our focus in the paper will be on extending the applicability of spectral GCNs. Spatial GCNs will be considered only for comparison purposes within computational experiments.

Despite the growing interest in academia and industry alike for neural-network methods suitable for progressively more general classes of graphs, the literature on spectral GCNs has only recently begun to include extensions beyond the basic case of unweighted, undirected graphs. As a result, the adoption of spectral CGNs in learning tasks involving graphs such as those featuring directed edges, digons, and edges with negative weights (He et al. 2022c; Wu et al. 2022) is still limited. To (partially) address such limitations, alternative notions of the Laplacian matrix have been put forward, among which the *Magnetic Laplacian* (Lieb and Loss 1993), which was originally proposed in physics and first used within a spectral GCN in (Zhang et al. 2021b), albeit with the limitation of only handling digraphs non-negative edge weights. Spectral-based GCNs suitable for digraphs with weights of unrestricted sign have been proposed only recently. Three such networks are SigMaNet (Fiorini et al. 2023) (based on the therein proposed *Sign-Magnetic Laplacian*), MSGNN (He et al. 2022a) (based on an extension of the *Magnetic Laplacian*), and the network proposed in (Ko 2022).

Graphs with *digons* (pairs of antiparallel directed edges) occur in many important applications including, e.g., traffic and networking and, so far, have eluded every spectral-based GCN proposed in the literature. Indeed, to the best of our knowledge none of the known spectral-based GCNs can handle graphs with digons with arbitrary weights and signs

Copyright © 2024, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

without collapsing them to single edges (either directed or undirected), which often results in destroying any topological information such digons may carry.

In this work, we extend the theory of spectral GCNs to digraphs with unrestricted edge weights also including digons. This is achieved by introducing the *Quaternionic Laplacian* $L^{ϙ}$, the first, to our knowledge, graph Laplacian matrix with quaternion-valued entries.[1] We prove that $L^{ϙ}$ generalizes different previously proposed Laplacians (both real- and complex-valued), that it satisfies all the properties that are needed to build a convolutional operator around it, and that it can naturally represent digons in such a way that the graph can be fully reconstructed from it without any loss of topological information. Around the Quaternionic Laplacian, we design QuaterGCN, a spectral GCN which relies on both $L^{ϙ}$ and quaternion-valued network weights.

While some GCNs employing quaternion weights have been recently proposed (Nguyen, Phung et al. 2021; Wang et al. 2022; Le, Tran, and Le 2023), only QGNN, proposed by Nguyen, Phung et al. (2021), is spectral based. As QGNN relies on the classical convolution operator of Kipf and Welling (2017) bases on the classical (real-valued) graph Laplacian, it fails to fully capture the whole graph topology unless the graph has non-negative edge weights and is undirected (which implies that it contains no digons).

We summarize the differences between the Quaternionic Laplacian $L^{ϙ}$ we introduce in this work and previous proposals in Table 1. For each Laplacian, the table reports the paper in which it was proposed or used within a spectral GCN and the corresponding symbol (which is often reused with some overload). It also indicates the type of number system associated with each Laplacian and its capability to handle edges with negative weights and digons.

Main Contributions of the Work

- We propose the quaternion-valued Quaternionic Laplacian matrix $L^{ϙ}$, which naturally captures the presence of digons of different weight (*asymmetric*) without reducing them to a single edge as done in many previously-proposed Laplacians.
- We prove that $L^{ϙ}$ generalizes both the standard Laplacian and the Sign-Magnetic Laplacian, as it coincides with the former when G is undirected and with the latter when G features directed edges with weights of arbitrary sign and all its digons have the same weight (*symmetric*).
- We incorporate $L^{ϙ}$ into QuaterGCN, a spectral-based GCN that includes quaternion-valued convolutional layers and quaternion-valued network weights.
- Our experiments demonstrate that QuaterGCN consistently outperforms state-of-the-art spatial and spectral GCNs, particularly on tasks and datasets where the information carried by the digons is critical.

Preliminaries and Previous Works

Let $G = (V, E)$ be an undirected graph with $n = |V|$ vertices without weights nor signs associated with its edges and let $A \in \{0,1\}^{n\times n}$ be its adjacency matrix. The *classical Laplacian matrix* $L \in \mathbb{Z}_+^{n\times n}$ of G is defined as $L := D - A$, where $D := \operatorname{diag}(A\,e)$ is the degree matrix of G, e is the all-one vector of appropriate size, and the operator diag builds a diagonal matrix with the argument on the main diagonal. The normalized version of L is defined as $L_{\text{norm}} := D^{-\frac{1}{2}}(D - A)D^{-\frac{1}{2}} = I - D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$.

For a spectral convolution operator to be well-defined, the graph Laplacian must fulfill three properties: **P.1)** it must be diagonalizable, i.e., it must admit an eigenvalue decomposition; **P.2)** it must be positive semidefinite; **P.3)** its spectrum must be upper-bounded by 2 (Kipf and Welling 2017).

While L and L_{norm} always satisfy such properties if G is undirected (i.e., A is symmetric) and has nonnegative edge weights (i.e., A is component-wise nonnegative), this is not always the case for more general graphs. When G is a digraph, L is sometimes defined as a function of $A_s := \frac{1}{2}(A^\top + A)$ and $D_s := \operatorname{Diag}(A_s e)$, rather than of A and D. This is, e.g., the case of QGNN (Nguyen, Phung et al. 2021). While such a choice preserves the mathematical properties of L, it significantly alters the topology of the graph. Indeed, when general graphs are considered, the following issues may arise:

- **Issue 1.** If G is a digraph, defining L as a function of A_s is equivalent to transforming the original graph into an undirected version of it, losing any directional information the original graph contained.
- **Issue 2.** If G features negative-weighted edges, neither A nor A_s belong, in general, to $\mathbb{Z}_+^{n\times n}$. This can lead to $D_{uu} < 0$ for some $u \in V$, in which case L is not well defined in $\mathbb{Z}_+^{n\times n}$ due to $D^{-\frac{1}{2}}$ being irrational.
- **Issue 3.** If G contains asymmetric digons, the weight asymmetry is completely lost in A_s, since the latter only features the average of the two weights. For example, in an author-citation graph, a digon representing two authors, the first one citing the second one 50 times while being cited by them only 2 times, would be identical to the two authors symmetrically citing each other precisely 26 times.

Issue 1 was first addressed by Zhang et al. (2021b,a) by introducing Magnet, a spectral GCN relying on the *Magnetic Laplacian* $L^{(q)}$. Such a Laplacian is a complex-valued extension of L to unweighted directed graphs, and was originally proposed within an electro-magnetic charge model by Lieb and Loss (1993). It is defined as follows:

$$L^{(q)} := D_s - H^{(q)}, \quad \text{with}$$
$$H^{(q)} := A_s \odot \exp\left(\mathbf{i}\,\Theta^{(q)}\right), \qquad \Theta^{(q)} := 2\pi q\left(A - A^\top\right),$$

where $\odot$ denotes the Hadamard product, $\mathbf{i} = \sqrt{-1}$, Θ is a phase matrix that encodes the edge directions, and $\exp\left(\mathbf{i}\,\Theta^{(q)}\right) := \cos(\Theta^{(q)}) + \mathbf{i}\sin(\Theta^{(q)})$, where cos and sin are applied component-wise. The parameter $q \in \mathbb{R}_0^+$ is usually chosen in $\left[0, \frac{1}{4}\right]$ or $\left[0, \frac{1}{2}\right]$ (Zhang et al. 2021b; Fanuel, Alaíz, and Suykens 2017). Choosing $q = 0$ implies $\Theta^{(q)} = 0$ and thus $L^{(q)}$ boils down to the Laplacian matrix L defined on A_s (in which case the directionality of G is lost).

[1]The letter ϙ in $L^{ϙ}$ is an archaic Greek letter often replaced in modern text by the Latin letter "q" as in "quaternion".

Laplacian Symbol	Proposed in	Number System	Allows Negative Edge Weights?	Allows Digons? if $w_1 = w_2$	if $w_1 = -w_2$	if $w_1 \neq w_2$	if $w_1 \neq -w_2$
L	Kipf and Welling (2017)	$\mathbb{R}$	✗	✓	✗	✗	✗
$L^{(q)}$	Zhang et al. (2021b)	$\mathbb{C}$	✗	✓	✗	✗	✗
L^{σ}	Fiorini et al. (2023)	$\mathbb{C}$	✓	✓	✗	✗	✗
$L^{(q)}$	He et al. (2022a)	$\mathbb{C}$	✓	✓	✗	✗	✗
$L^{(q)}$	Ko (2022)	$\mathbb{C}$	✓	✓	✓	✗	✗
$\boldsymbol{L^{♀}}$	**This paper**	$\mathbb{H}$	✓	✓	✓	✓	✓

Table 1: Main differences between Laplacian matrices used in spectral GCN literature. For a graph with edge set E, w_1 and w_2 denote the weights of the digons $(u, v), (v, u) \in E$.

Issue 2 was first addressed by Fiorini et al. (2023) via the introduction of the spectral GCN SigMaNet and the *Sign-Magnetic Laplacian* L^{σ}. Unlike $L^{(q)}$, L^{σ} is well defined for graphs with negative edge weights and enjoys some robustness properties to weight scaling (which in $L^{(q)}$ could artificially alter the sign pattern of Θ and thus the directionality of the edges). L^{σ} is defined as follows:

$$L^{\sigma} := \bar{D}_s - H^{\sigma}, \quad \text{with}$$
$$H^{\sigma} := A_s \odot \left(ee^{\top} - \operatorname{sgn}(|A - A^{\top}|) + \mathbf{i}\operatorname{sgn}\left(|A| - |A^{\top}|\right)\right),$$

where $\bar{D}_s := \operatorname{Diag}(|A_s|\, e)$ and $\operatorname{sgn} : \mathbb{R} \to \{-1, 0, 1\}$ is the *signum* function applied component-wise. After Fiorini et al. (2023), He et al. (2022a) extended the Magnetic Laplacian as a function of $\bar{D}_s := \operatorname{Diag}(\frac{|A|+|A|^{\top}}{2} e)$ rather than D_s. This Laplacian is well-defined for graphs with negative edge weights, but it still suffers from the scaling issue pointed out by Fiorini et al. (2023) which L^{σ} avoids.

Issue 3, to the best of our knowledge, has not yet been addressed. We set ourselves out to do so in this paper.

A Quaternion-Valued Laplacian

In this section, we introduce the *Quaternionic Laplacian* matrix, a positive semidefinite quaternion-valued Hermitian matrix which fully captures the directional and weight information of a digraph, even in the presence of digons, without restrictions on the sign or magnitude of the edge weights.

The Quaternion Number System

Quaternions are an extension of complex numbers to three imaginary components (Hamilton 1866). They are often used in quantum mechanics, where they lead to elegant expressions of the Lorentz transformation, which forms the basis of modern relativity theory (Jia 2008). In computer graphics, quaternions are commonly used to represent and manipulate 3D objects for rotation estimation and pose graph optimization (Carlone et al. 2015).

Formally, a quaternion $q \in \mathbb{H}$ takes the form $q = q_0 + \mathbf{i}q_1 + \mathbf{j}q_2 + \mathbf{k}q_3$, where q_0, q_1, q_2 and q_3 are real numbers and $\mathbf{i}, \mathbf{j}$, and $\mathbf{k}$ are three imaginary units satisfying $\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -1$. The four basis elements are 1, $\mathbf{i}$, $\mathbf{j}$, and $\mathbf{k}$. The conjugate of q is $\bar{q} \equiv q^* = q_0 - \mathbf{i}q_1 - \mathbf{j}q_2 - \mathbf{k}q_3$. q is called imaginary if its real part q_0 is zero. The multiplication of quaternions satisfies the distribution law but is not commutative. A quaternion-valued matrix $Q = (q_{uv}) \in \mathbb{H}^{m\times n}$ reads $Q = Q_0 + \mathbf{i}Q_1 + \mathbf{j}Q_2 + \mathbf{k}Q_3$, with $Q_0, Q_1, Q_2, Q_3 \in \mathbb{R}^{m\times n}$. $Q^{\top} = (q_{vu})$ is the transpose of Q. $\bar{Q} = (\bar{q}_{uv})$ is the conjugate of Q. $Q^* = (\bar{q}_{vu}) = \bar{Q}^{\top}$ is the conjugate transpose of Q. A square quaternion matrix $Q \in \mathbb{H}^{n\times n}$ is called Hermitian if $Q^* = Q$ and skew-Hermitian if $Q^* = -Q$. We denote the real part and the three imaginary parts of a quaternion $q \in \mathbb{H}$ by $\Re(q)$, $\Im_1(q)$, $\Im_2(q)$, and $\Im_3(q)$.

The Quaternionic Laplacian

Let us now introduce the Quaternionic Laplacian, which we denote by $L^{♀}$. First, we introduce the following matrices:

- Let $T := \operatorname{sgn}(|A|) \in \{0, 1\}^{n\times n}$ be a binary matrix that encodes the graph's topology, with $T_{uv} = 1$ if G contains an edge from node u to node v and $T_{uv} = 0$ otherwise.
- Let $O := T \odot T^{\top} \in \{0, 1\}^{n\times n}$ be the binary matrix that encodes the topology of the subgraph of G that only contains digons (by definition, O is symmetric and $O_{uv} = O_{vu} = 1$ iff $T_{uv} = T_{vu} = 1$).
- Let $N := \operatorname{sgn}(|A - A^{\top}|) \in \{0, 1\}^{n\times n}$ be a binary matrix that encodes the topology of the subgraph of G obtained by dropping any symmetric digons (by definition, $N_{uv} = 0$ if $A_{uv} = A_{vu}$ and $N_{uv} = T_{uv}$ otherwise).
- Let $R := \operatorname{sgn}(|A| - |A^{\top}|) \in \{-1, 0, 1\}^{n\times n}$ be a signed binary matrix in which every asymmetric digon $(u, v), (v, u)$ is reduced to a single edge in the direction of the largest absolute weight ($R_{uv} = T_{uv}$ if $A_{uv} > A_{vu}$, $R_{uv} = -T_{uv}$ if $A_{uv} < A_{vu}$, and $R_{uv} = 0$ otherwise).

With these definitions, we introduce the four matrices H^0, H^1, H^2, H^3, whose elements are valued in {-1, 0, 1}:

$$H^0 := ee^{\top} - N \qquad H^1 := R \odot \left(ee^{\top} - O\right)$$
$$H^2 := O \odot N \odot \left(U(T) - L(T^{\top})\right) \quad H^3 := -H^2,$$

where U and L are the unary operators that construct an upper- or lower-triangular matrix from the upper or lower triangle of the matrix given to them as input.

H^0 only encodes G's symmetric digons, H^1 encodes all of G's edges excluding digons, and H^2 and H^3 encode G's asymmetric digons in a skew-symmetric way. We remark that the three matrices H^1, H^2, H^3 are skew-symmetric by construction (i.e., $H^1_{uv} = -H^1_{vu}$, $H^2_{uv} = -H^2_{vu}$, $H^3_{uv} = -H^3_{vu}$ for all $u, v \in V$), whereas H^0 is symmetric.

Based on the four matrices H^0, H^1, H^2, H^3, we now define the Quaternionic Laplacian as follows:

$$L^{♀} := \bar{D}_s - H^{♀}, \quad \text{with}$$
$$H^{♀} = A_s^1 \odot (H^0 + \mathbf{i}H^1) + \mathbf{j}A_s^2 \odot H^2 + \mathbf{k}A_s^3 \odot H^3, \quad (1)$$

with $A_s^1 := \frac{A+A^\top}{2}$, $A_s^2 := \frac{U(A)+L(A^\top)}{2}$, $A_s^3 := \frac{L(A)+U(A^\top)}{2}$, and $\bar{D}_s := \mathrm{Diag}(|A_s^1|\,e)$. Its normalized version reads:

$$L^{♀}_{\text{norm}} := \bar{D}_s^{-\frac{1}{2}} L^{♀} \bar{D}_s^{-\frac{1}{2}} = I - \bar{D}_s^{-\frac{1}{2}} H^{♀} \bar{D}_s^{-\frac{1}{2}}. \quad (2)$$

On the Nature of the Elements of $H^{♀}$

Let us illustrate how the topology of G and its weights are mapped into the matrix $H^{♀}$.

1. For every edge $(u, v) \in E$ with $(v, u) \notin E$ (i.e., not contained in a digon) $H^{♀}_{uv} = -H^{♀}_{vu} = 0 + \mathbf{i}\frac{1}{2}A_{uv} + \mathbf{j}0 + \mathbf{k}0$ holds. Thus, the edge is purely mapped in the first imaginary component ($\mathbf{i}$) of $H^{♀}$.
2. For every symmetric digon $(u, v), (v, u) \in E$ with $A_{uv} = A_{vu}$, $H^{♀}_{uv} = H^{♀}_{vu} = \frac{1}{2}(A_{uv} + A_{vu}) + \mathbf{i}0 + \mathbf{j}0 + \mathbf{k}0$ holds. Thus, such digons are encoded purely in the real part of $H^{♀}$ (as if they coincided with an undirected edge).
3. For every asymmetric digon $(u, v), (v, u) \in E$ with $A_{uv} \neq A_{vu}$, $H^{♀}_{uv} = -H^{♀}_{vu} = 0 + \mathbf{i}0 + \mathbf{j}\frac{1}{2}A_{uv} - \mathbf{k}\frac{1}{2}A_{vu}$ (if $u < v$) and $H^{♀}_{uv} = -H^{♀}_{vu} = 0 + \mathbf{i}0 - \mathbf{j}\frac{1}{2}A_{uv} + \mathbf{k}\frac{1}{2}A_{vu}$ (if $u > v$) hold. Such digons are thus encoded purely in the second and third ($\mathbf{j}$ and $\mathbf{k}$) imaginary components.

$L^{♀}$ realizes the same mapping as L^σ in cases 1 and 2, but not in case 3. In such a case, while in L^σ every asymmetric digon is mapped into a single directed edge in the direction of largest magnitude (thus losing parts of it topology), in $L^{♀}$ the topology of such a digon is completely preserved.

For each $u, v \in V$, the three imaginary parts of $L^{♀}$ are orthogonal: if u and v share only one edge, this edge is reported in the imaginary component $\mathbf{i}$; if they share two edges of different weights, the two edges are reported in the imaginary components $\mathbf{j}$ and $\mathbf{k}$; if they share two edges of the same weight, the edges are reported as a single undirected edge in the real component of $L^{♀}$; if the two nodes share no edges, nothing is reported.

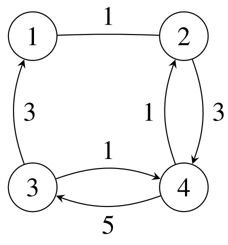
From a Graph to Its Quaternionic Laplacian

As an illustrative example, consider the graph depicted in Figure 1, together with its weighted adjacency matrix. The graph has the following characteristics:

1. A single undirected edge $(1, 2)$ (which is equivalent to a symmetric digon).
2. A directed edge $(3, 1)$.
3. Two asymmetric digons, $(2, 4) - (4, 2)$ and $(3, 4) - (4, 3)$, with different weights that share the same sign.

For this graph, $L^{♀}$ reads:

$$L^{♀} := \bar{D}_s - H^{♀}, \quad \text{with}$$
$$H^{♀} := A_s^1 \odot (H^0 + \mathbf{i}H^1) + \mathbf{j}A_s^2 \odot H^2 + \mathbf{k}A_s^3 \odot H^3,$$



$$\longleftrightarrow \quad A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 3 \\ 3 & 0 & 0 & 1 \\ 0 & 1 & 5 & 0 \end{bmatrix}$$

Figure 1: Graph and Adjacency Matrix.

with $A_s^1 = \begin{bmatrix} 0 & 1 & 1.5 & 0 \\ 1 & 0 & 0 & 2 \\ 1.5 & 0 & 0 & 3 \\ 0 & 2 & 3 & 0 \end{bmatrix}$, $A_s^2 = \begin{bmatrix} 0 & 0.5 & 0 & 0 \\ 0.5 & 0 & 0 & 1.5 \\ 0 & 0 & 0 & 0.5 \\ 0 & 1.5 & 0.5 & 0 \end{bmatrix}$,

$A_s^3 = \begin{bmatrix} 0 & 0.5 & 1.5 & 0 \\ 0.5 & 0 & 0 & 0.5 \\ 1.5 & 0 & 0 & 2.5 \\ 0 & 0.5 & 2.5 & 0 \end{bmatrix}$, and

$$H^0 = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}, \qquad H^1 = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix},$$
$$H^2 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & -1 & -1 & 0 \end{bmatrix}, \qquad H^3 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & -1 \\ 0 & 1 & 1 & 0 \end{bmatrix}.$$

We note that H^0 contains a 1 for each edge that does not belong to a digon with different weights or to a single directed edge without an antiparallel one. It features an all-1 diagonal, whose values would become zeros after the multiplication by A_s^1. H^1 contains a ± 1 for each undirected edge with a positive sign for the actual edge direction and a negative one for the inverse direction. H^2 and H^3 contain a 1 for each pair of edges belonging to a digon with different edge weights. By construction, H^3 is the opposite of H^2.

$L^{♀}$ thus the reads:

$$L^{♀} = \begin{bmatrix} 2.5 & -1 & \mathbf{i}\,1.5 & 0 \\ -1 & 3 & 0 & -\mathbf{j}\,1.5 + \mathbf{k}\,0.5 \\ -\mathbf{i}\,1.5 & 0 & 4.5 & -\mathbf{j}\,0.5 + \mathbf{k}\,2.5 \\ 0 & \mathbf{j}\,1.5 - \mathbf{k}\,0.5 & \mathbf{j}\,0.5 - \mathbf{k}\,2.5 & 5 \end{bmatrix}.$$

By inspecting $L^{♀}$, one can observe that it encodes the elements of the graph in the following way:

1. The undirected edge is encoded via the real component $L^{♀}_{12} = L^{♀}_{21} = -1$;
2. The directed edge is encoded via the $\mathbf{i}$ component, $L^{♀}_{13} = L^{♀}_{31} = \mathbf{i}\,1.5$;
3. The two digons with different weights that share the same sign are encoded via the $\mathbf{j}$ and $\mathbf{k}$ components:
   (a) $L^{♀}_{24} = -L^{♀}_{42} = -\mathbf{j}\,1.5 + \mathbf{k}\,0.5$;
   (b) $L^{♀}_{34} = -L^{♀}_{43} = -\mathbf{j}\,0.5 + \mathbf{k}\,2.5$.

Relationship Between $L^{♀}$ and Other Laplacians

The Quaternionic Laplacian is designed in such a way that it satisfies several desirable properties which we now illustrate.

For undirected graphs with positive edge weights, $L^{♀}$ generalizes the classical Laplacian L:

Theorem 1. *$L^{♀} = L$ for every graph with A symmetric and nonnegative and $D_{vv} > 0$ for all $v \in V$.*

For digraphs with arbitrary edge weight featuring, if any, symmetric digons, $L^{♀}$ generalizes the Sign-Magnetic Laplacian L^{σ}:

Theorem 2. *$L^{♀} = L^{\sigma}$ if, for all $u, v \in V$, either $A_{uv} = 0$ or $A_{uv} = A_{vu}$.*

As L^{σ} coincides with $L^{(q)}$ with $q = \frac{1}{4}$ (Fiorini et al. 2023), the following holds:

Corollary 1. *$L^{♀} = L^{(q)}$ with $q = \frac{1}{4}$ for every digraph with $A \in \{0,1\}^{n\times n}$ containing, if any, symmetric digons.*

If the graph does not feature any symmetric digons, the matrix $H^{♀}$ from which $L^{♀}$ is defined coincides with a linear combination of L^{σ} with a Hermitian matrix encoding the asymmetric digons:

Theorem 3. *Consider a weighted digraph without symmetric digons and let $H^m := \left(A_s^2 \odot H_1^2 + A_s^3 \odot H_1^3\right)$, where $H_1^2 = N \odot \left(U(T) - L(T^\top)\right)$ and $H_1^3 = -H_1^2$. We have $H^0 = 0$ and $H^{♀} = H^{\sigma} \odot \left(ee^\top - O\right) + O \odot H^m$. Thus, each component of $H^{♀}_{uv}$ is a linear combination of the component H^{σ}_{uv} of the Sign-Magnetic Laplacian H^{σ} and the component H^m_{uv} of the quaternionic Hermitian matrix H^m.*

Spectral Properties of $L^{♀}$

As $H^{♀}$ is Hermitian by construction, $L^{♀}$ and $L^{♀}_{\text{norm}}$ are Hermitian as well. As any Hermitian quaternion matrix Q is diagonalizable via the (right) eigenvalue decomposition $Q = U^*\Lambda U$ (see Qi and Luo (2021) for a proof), $L^{♀}$ and $L^{♀}_{\text{norm}}$ satisfy the property P.1. Both matrices admit exactly n right eigenvalue-eigenvector pairs, all of which are real:

Theorem 4. *$L^{♀}$ and $L^{♀}_{norm}$ are positive semidefinite.*

The right eigenvalues of the normalized version of $L^{♀}$ are upper bounded by 2:

Theorem 5. *$\lambda_{max}(L^{♀}_{norm}) \leq 2$, where λ_{max} denotes the (real) right eigenvalue of largest value.*

These theorems show that $L^{♀}$ and $L^{♀}_{\text{norm}}$ enjoy the two remaining properties P.2 and P.3. Thus, $L^{♀}_{\text{norm}}$ can be employed for the definition of a convolution operator, as shown in the following section.

QuaterNet: A Spectral GCN Based on $L^{♀}$

Following Hammond, Vandergheynst, and Gribonval (2011) and Kipf and Welling (2017), we define the convolution operation of a signal $x \in \mathbb{R}^n$ and a filter $y \in \mathbb{R}^n$ as

$$y * x = Yx = (\theta_0 I - \theta_0(L^{♀}_{\text{norm}} - I))x = \theta_0(2I - L^{♀}_{\text{norm}})x. \quad (3)$$

The equation is obtained by approximating the Fourier transform on the graph Laplacian with Chebyshev's polynomial (of the first kind) of order 1.

Due to Theorems 4 and 5, $L^{♀}_{\text{norm}}$ enjoys properties P.1, P.2, and P.3. Thus, Y is a well-defined convolution operator and, by definition of $L^{♀}_{\text{norm}}$, we have:

$$\begin{aligned} y * x = Yx &= \theta_0(2I - (I - \bar{D}_s^{-\frac{1}{2}} H^{♀} \bar{D}_s^{-\frac{1}{2}})x \\ &= \theta_0(I + \bar{D}_s^{-\frac{1}{2}} H^{♀} \bar{D}_s^{-\frac{1}{2}})x. \end{aligned} \quad (4)$$

Following Kipf and Welling (2017) to avoid numerical instabilities, we apply Eq. (4) with $\tilde{D}_s^{-\frac{1}{2}} \tilde{H}^{♀} \tilde{D}_s^{-\frac{1}{2}}$ rather than $I + \bar{D}_s^{-\frac{1}{2}} H^{♀} \bar{D}_s^{-\frac{1}{2}}$, where $\tilde{H}^{♀}$ and $\tilde{D}_s$ are defined as a function of $\tilde{A} := A + I$ rather than A.

We generalize the feature vector signal $x \in \mathbb{H}^{n\times 1}$ to a feature matrix signal $X \in \mathbb{H}^{n\times c}$ with c input channels (i.e., a c-dimensional feature vector for every node of the graph). Let $\Theta \in \mathbb{H}^{c\times f}$ be a matrix representing the parameters of an f-dimensional filter and let ϕ be an activation function (such as the *ReLU*) applied component-wise to the input matrix.

In QuaterGCN, we define the convolutional layer as the following mapping from X to $Z^{\sigma}(X) \in \mathbb{H}^{n\times f}$:

$$Z^{♀}(X) = \phi(\tilde{D}_s^{-\frac{1}{2}} \tilde{H}^{♀} \tilde{D}_s^{-\frac{1}{2}} X\Theta) = \phi(X^{♀}\Theta).$$

Since $X^{♀}\Theta$ is a quaternion-valued matrix and traditional activation functions require a real argument, we follow the approach of Parcollet et al. (2019) and apply the activation function ϕ to each element of its quaternionic input as

$$\phi : (a + \mathbf{i}b + \mathbf{j}c + \mathbf{k}d) \mapsto \phi(a) + \mathbf{i}\phi(b) + \mathbf{j}\phi(c) + \mathbf{k}\phi(d).$$

Thanks to this, the output $Z^{♀}(X)$ of the convolutional layer is quaternion-valued. As usually done in spectral GCNs, we lastly adopt an *unwind* layer by which we transform the matrix $Z^{♀}(X) \in \mathbb{H}^{n\times f}$ into the 4-times larger real-valued matrix, i.e. matrix of dimension $n \times 4f$,

$$\Big(\Re(Z^{♀}(X)) \mid \Im_1(Z^{♀}(X)) \mid \Im_2(Z^{♀}(X)) \mid \Im_3(Z^{♀}(X))\Big).$$

As stated by Parcollet, Morchid, and Linarès (2020) and Nguyen, Phung et al. (2021), the adoption of quaternion algebra in the product $X^{♀}\Theta$ leads to an extensive interaction among the components of $X^{♀}$ which is likely to lead to more expressive vector representations than those achieved with real- and complex-valued networks.

Based on the task at hand (see next section), after the convolution layer we described before QuaterGCN features either a linear layer with weights W or a $1D$ convolution. Considering, for example, a node-classification task of predicting which of a set of unknown classes a graph vertex belongs to, QuaterGCN implements the function

$$\text{softmax}\left(\text{unwind}\left(Z^{♀(2)}\left(Z^{♀(1)}\left(X^{(0)}\right)\right)\right)W\right),$$

where $X^{(0)} \in \mathbb{H}^{n\times c}$ is the input feature matrix, $Z^{♀(1)} \in \mathbb{H}^{n\times f_1}$ and $Z^{♀(2)} \in \mathbb{H}^{n\times f_2}$ are two convolutional layers, $W \in \mathbb{R}^{4f_2\times d}$ are the weights of the linear layer (with d being the number of classes), and softmax : $\mathbb{R}^{n\times d} \to [0,1]^{n\times d}$ is the normalized exponential activation function typically used to recover the node classes.

Complexity of QuaterGCN

For a graph with n nodes with a c-dimensional feature vector each, the complexity of QuaterGCN, with two convolutional layers with f_1 and f_2 filters each, is $O\left(nc(n+f_1)+nf_1(n+f_2)+m^{\text{train}}f_2d\right)$ for an edge-classification task and $O\left(nc(n+f_1)+nf_1(n+f_2)+nf_2d\right)$ for a node-classification one, where d is the number of classes (see the next section for more details on these tasks). Such a complexity is quadratic in the number of nodes n and it coincides with MagNet's, MSGNN's, and SigMaNet's. As the scalar-scalar product between two quaternions requires 8 multiplications and 7 additions between real numbers rather than 4 multiplications and 3 additions for the complex case and a single multiplication for the real case, the least-efficient operation carried out by QuaterGCN can be up to 64 times slower than it would be if the network featured real-only numbers. While apparently large, such a quantity is constant w.r.t. the size of the graph, which implies that QuaterGCN scales comparably to previous proposals in the literature.

Numerical Experiments

We compare QuaterGCN with state-of-the-art GCNs across four tasks: *node classification* (NC), *three-class edge prediction* (3CEP), *four-class edge prediction* (4CEP), and *five-class edge prediction* (5CEP). Throughout this section, the tables report the best results in **boldface** and the second-best are underlined. The datasets and the code we used are publicly available at https://github.com/Stefa1994/QuaterGCN.

We experiment on the six widely-used real-world directed graphs `Bitcoin-OTC`, `Bitcoin Alpha`, `WikiRfa`, `Telegram`, `Slashdot`, and `Epinions` (see Kumar et al. (2016); Bovet and Grindrod (2020); West et al. (2014); Leskovec, Huttenlocher, and Kleinberg (2010)). The first three feature edge weights of unrestricted sign and magnitude; the fourth contains graphs with positive edge weights, while the last two have graphs with weights satisfying $A \in \{-1, 0, +1\}^{n\times n}$.

To study the relationship between performance and graph density, we also employ `DBSM` graphs: synthetic digraphs with positive random weights already used by Fiorini et al. (2023). They are generated via a direct stochastic block model (DSBM) with edge weights greater than 1 and a different number of nodes per cluster (N) and number of clusters (C). Inter- and intra-cluster edges are created with, respectively, probability α_{uv} and α_{uu}, and a connected pair of nodes $\{u, v\}$ with $u < v$ shares the edge (u, v) with probability β_{uv} and (v, u) with probability $1 - \beta_{uv}$.

As the `DBSM` graphs are digons-free, we introduce a second class of synthetic digraphs with a variable percentage of digons $\delta \in (0, 1)$ with positive random weights between 2 and 4: `Di150` (150 nodes) and `Di500` (500 nodes). `Di150` features graphs with $N = 150$, $C = 5$, $\alpha_{uu} = 0.1$, $\beta_{uv} = 0.2$, and $\alpha_{uv} = 0.6$. `Di500` contains graphs with $N = 500$, $C = 5$, $\alpha_{uu} = 0.1$, $\beta_{uv} = 0.2$, and $\alpha_{uv} = 0.1$. Notice that `Di500` is sparser than `Di250`.

Node Classification Task (NC)

The task is to predict the class of each node. We consider the `Telegram` dataset and the 9 aforementioned synthetic datasets, i.e., every dataset except for those that lack a predetermined node class.

We compare QuaterGCN with: (i) the three spectral GCNs designed for undirected graphs: ChebNet (Defferrard, Bresson, and Vandergheynst 2016) and GCN (Kipf and Welling 2017) and the spectral GCN with quaternionic weights QGNN designed for undirected graphs (Nguyen, Phung et al. 2021); (ii) the four spectral GCNs designed for directed graphs: DGCN (Tong et al. 2020b), DiGraph (Tong et al. 2020a), DiGCL (Tong et al. 2021), MagNet (Zhang et al. 2021b), and SigMaNet (Fiorini et al. 2023); and (iii) the five spatial GCNs: APPNP (Klicpera, Bojchevski, and Günnemann 2019), SAGE (Hamilton, Ying, and Leskovec 2017), GIN (Xu et al. 2018), GAT (Veličković et al. 2018), and SSSNET (He et al. 2022b). The experiments are run with 10-fold cross-validation with a 60%/20%/20% split for training, validation, and testing.

Table 2 and 3 show that QuaterGCN achieves a remarkable performance across all datasets, being the best method in 9 cases out of 10. The percentage difference between QuaterGCN and the second-best performer ranges from 0.2% (for `DBSM` with $\alpha_{uv} = 0.1$) to 242.81% (for `Di500` with $\delta = 0.7$). The average performance improvement of QuaterGCN across all datasets is 68.27%. QuaterGCN consistently outperforms the state of the art on, in particular, `Di500` and `Di150`, where it achieves an average improvement w.r.t. the second-best performer of 28.19% on the `Di150` dataset and of 175% on the `Di500` one. The larger improvement of QuaterGCN and, in particular, the overall weaker performance of every other method on the `Di500` dataset seems to be correlated with the dataset being sparser than the smaller `Di150`, which suggests that the learning task is harder. The difference between QuaterGCN and QGNN (the only available GCN with quaternion-valued weights which, though, employs the classical real-valued Laplacian), is substantial, as QuaterGCN outperforms QGNN by 309% on average. As the two networks share a similar architecture, we attribute such a large difference to QuaterGCN's convolution being done via our proposed Quaternionic Laplacian $L^{♀}$.

Three-Class Edge Prediction Task (3CEP)

The task is to predict whether $(u, v) \in E$, $(v, u) \in E$, or $(u, v) \notin E \wedge (v, u) \notin E$. In order to maximize the number of spectral methods we can compare to, for this task our analysis focuses on the datasets with positive weights, i.e., `Telegram`, `DSBM`, `Di150`, and `D500`. Following Fiorini et al. (2023), we also consider `Bitcoin Alpha`* and `Bitcoin OTC`*, obtained by removing any negative-weight edge from `Bitcoin Alpha` and `Bitcoin OTC`.

We compare QuaterGCN to the same methods we considered for the NC task. We run the experiments with 10-fold cross-validation with an 80%/15%/5% split for training, testing, and validation, preserving graph connectivity.

The results are reported in Table 4. Those obtained on the `Di500` dataset are omitted as on it every method achieves the same performance of about 30% (equal to a uniform random predictor). The table shows that QuaterGCN outperforms the other methods on 7 datasets out of 9. Compared with the second-best model, QuaterGCN achieves an average performance improvement of 0.98%, with a maximum of 3.60% and a minimum of 0.02%.

	Node classification						
		DBSM			Di500		
	Telegram	α_{uv} = 0.05	α_{uv} = 0.08	α_{uv} = 0.1	δ = 0.2	δ = 0.5	δ = 0.7
ChebNet	61.73±4.25	20.06±00.18	20.50±00.77	19.98±00.06	19.90±0.24	20.00±0.00	19.94±0.13
GCN	60.77±3.67	20.06±00.18	20.02±00.06	20.01±00.01	20.04±0.12	20.10±0.30	20.08±0.30
QGNN	51.35±9.10	20.03±00.12	20.01±00.02	19.99±00.07	20.23±0.21	19.94±0.18	20.00±0.00
APPNP	55.19±6.26	33.46±07.43	34.72±14.98	36.16±14.92	20.64±1.32	20.16±0.37	20.10±0.30
SAGE	65.38±5.15	67.64±09.81	68.28±10.92	82.96±10.98	23.68±3.83	20.44±0.95	20.02±0.14
GIN	72.69±4.62	28.46±08.01	20.12±00.20	20.98±08.28	20.14±0.42	19.88±0.36	20.00±0.00
GAT	72.31±3.01	22.34±03.13	21.90±02.89	21.58±01.80	19.90±0.20	20.04±0.28	20.16±0.42
SSSNET	24.04±9.29	**91.04±03.60**	94.94±01.01	96.77±00.80	31.41±5.91	22.34±1.31	21.13±1.03
DGCN	71.15±6.32	30.02±06.57	30.22±11.94	28.40±08.62	20.10±0.30	20.00±0.00	20.00±0.00
DiGraph	71.16±5.57	53.84±14.28	38.50±12.20	34.78±09.94	32.82±2.14	24.44±2.33	20.76±1.34
DiGCL	64.62±4.50	19.51±01.21	20.24±00.84	19.98±00.45	20.00±0.00	20.00±0.00	20.00±0.00
MagNet	55.96±3.59	78.64±01.29	87.52±01.30	91.58±01.04	31.46±2.20	22.74±1.12	20.88±1.62
SigMaNet	74.23±5.24	87.44±00.99	96.14±00.64	98.60±00.31	31.26±2.08	22.32±1.69	19.94±1.07
GraQuaterGCN	**75.58±3.85**	87.46±00.73	**96.44±00.12**	**98.80±00.20**	**64.28±1.04**	**70.60±1.62**	**71.58±1.52**

Table 2: Accuracy (%) on the node classification task.

	Node classification		
	Di150		
	δ = 0.2	δ = 0.5	δ = 0.7
ChebNet	19.93±00.20	20.00±00.00	20.00±00.00
GCN	20.00±00.00	20.00±00.00	20.07±00.20
QGNN	19.87±00.40	19.93±00.20	20.00±00.00
APPNP	22.87±08.60	21.07±03.20	20.00±00.00
SAGE	78.40±14.35	33.20±14.10	20.33±01.69
GIN	24.67±06.76	28.13±08.87	23.67±06.24
GAT	51.20±10.18	29.33±10.05	21.40±03.78
SSSNET	92.71±12.48	86.18±17.77	61.78±24.44
DGCN	26.87±08.43	21.87±05.60	20.00±00.00
DiGraph	97.40±01.01	88.27±03.14	58.93±08.47
DiGCL	20.00±00.00	20.00±00.00	20.00±00.00
MagNet	97.87±01.90	74.53±10.43	31.13±06.65
SigMaNet	74.67±04.15	49.60±03.07	24.67±04.40
QuaterGCN	**99.73±00.33**	**99.85±00.13**	**99.93±00.03**

Table 3: Accuracy (%) on the node classification task.

Differently from the NC task, in the 3CEP task the difference in performance between the methods is smaller, as already observed by Zhang et al. (2021b) and Fiorini et al. (2023). Nevertheless, the results indicate that the advantages provided by the Quaternionic Laplacian are still event, albeit being of smaller magnitude.

Focusing on QGNN, QuaterGCN outperforms it on 9 out of 10 datasets by an average of 25.58%. This further reinforces that the better performance of QuaterGCN is largely due to it relying on our proposed Laplacian matrix $L^{♀}$ rather than on the mere adoption of quaternionic weights as done in QGNN with the classical Laplacian.

Table 4 suggests that simpler methods designed for undirected graphs perform increasingly better when δ increases on the Di150 dataset. This is likely due the fact that these graphs feature a small difference in edge weight. If $A_{uv} \simeq A_{vu}$, not much is lost if the (almost symmetric) digon $(u, v), (v, u)$ is reduced to a single undirected edge of weight $\frac{1}{2}(A_{uv} + A_{vu})$, as done when, e.g., using the classical real-valued Laplacian matrix, as this would only lead to a small loss of information, if any. What is more, the larger the number of digons, the more the graph becomes close to being undirected if the difference in weight is small, which explains why simpler (and arguably easier to train) methods designed for undirected graphs achieve an increasingly better performance as the percentage of digons δ increases.

Four/Five-Class Edge Prediction Task (4/5CEP)

The 4CEP task is to predict whether $(u, v) \in E^+$, $(u, v) \in E^-$, $(v, u) \in E^+$, and $(v, u) \in E^-$ (with E^+ and E^- being the positive- and negative-weight edges), while the 5CEP task also considers the class where $(u, v) \notin E \wedge (v, u) \notin E$. Due to their nature, for both tasks we focus on every dataset featuring both positive and negative weights, i.e., on all the real-world datasets except for Telegram. We run the experiments with 5-fold cross-validation with an 80%/20% split for training and testing, preserving graph connectivity.

Due to the nature of the tasks, we compare QuaterGCN against the only methods that can handle the sign of the edge weights, i.e.: *i)* the signed graph neural networks SGCN (Derr, Ma, and Tang 2018), SiGAT (Huang et al. 2019), SNEA (Li et al. 2020), SDGNN (Huang et al. 2021), and SSSNET (He et al. 2022b); and *ii)* the spectral GCNs that are well-defined for negative edge weights, i.e., SigMaNet (Fiorini et al. 2023) and MSGNN (He et al. 2022a).

Table 5 and 6 show that, when compared to the other approaches, QuaterGCN achieves superior performance in 8 out of 10 cases, while being the second-best model in the other 2. In comparison with the second-best model, QuaterGCN achieves an average performance improvement of 1.14%, with a maximum of 3.61% and a minimum of 0.13%.

While not as large as for the NC task, the better performance that QuaterGCN achieves on the 4CEP and 5CEP tasks confirms the superior performance of the model we proposed.

	Three-Class Edge prediction								
				Di150			DBSM		
	Telegram	Bit Alpha*	Bitcoin OTC*	$\delta = 0.2$	$\delta = 0.5$	$\delta = 0.7$	$\alpha_{uv} = 0.05$	$\alpha_{uv} = 0.08$	$\alpha_{uv} = 0.1$
ChebNet	63.65±4.65	82.82±0.82	83.01±1.09	40.58±0.07	48.81±0.53	52.10±0.88	33.34±0.01	33.36±0.07	33.33±0.02
GCN	53.86±1.60	82.61±0.67	82.49±0.99	40.60±0.07	49.00±0.08	**53.51±0.10**	33.33±0.02	33.32±0.03	33.34±0.01
QGNN	52.33±1.50	80.93±0.63	79.97±0.80	40.60±0.07	49.00±0.08	52.51±0.09	33.38±0.09	33.43±0.26	33.37±0.04
APPNP	50.82±6.31	82.14±0.89	81.77±0.63	40.55±0.08	48.98±0.09	52.50±0.10	37.67±4.04	37.84±5.70	37.52±5.33
SAGE	69.28±7.24	55.82±1.60	85.19±0.64	40.62±0.14	49.00±0.08	52.52±0.09	39.50±3.74	38.51±3.55	42.69±3.87
GIN	58.41±1.26	77.93±0.86	76.35±0.77	40.56±0.08	48.98±0.10	52.47±0.10	34.65±2.62	33.34±0.01	33.52±0.37
GAT	67.34±2.50	84.93±1.20	85.02±0.74	40.64±1.79	48.44±1.63	52.51±0.09	33.70±0.79	33.35±0.07	33.91±1.63
DGCN	75.01±3.60	85.01±0.95	85.03±0.64	40.57±0.06	48.82±0.50	52.51±0.08	34.12±2.17	34.78±2.11	35.24±2.36
DiGraph	74.27±1.02	83.66±0.72	84.14±0.82	41.38±0.92	48.90±0.11	52.40±0.09	41.30±1.41	42.57±1.62	53.57±1.73
DiGCL	66.03±0.84	77.68±0.74	76.35±0.77	29.70±0.04	25.50±0.04	23.74±0.04	38.30±0.15	38.17±0.07	37.58±0.12
MagNet	**82.28±0.84**	85.72±0.67	85.66±0.78	45.47±1.70	48.78±0.35	52.19±0.43	43.62±1.11	46.76±1.13	47.76±1.12
SigMaNet	80.13±0.87	85.52±0.61	84.61±0.79	45.50±1.41	47.02±0.91	51.81±0.80	43.65±0.36	**47.26±0.17**	48.60±0.17
QuaterGCN	81.17±0.74	**86.17±0.57**	**86.06±0.60**	**47.14±0.21**	**49.01±0.16**	52.07±0.22	**44.10±0.58**	**47.26±0.56**	**48.68±0.26**

Table 4: Accuracy (%) on the three-class edge prediction task.

	Four-Class Edge prediction				
	Bitcoin Alpha	Bitcoin OTC	WikiRfa	Slashdot	Epinions
SGCN	48.05±0.29	52.52±0.71	68.37±0.51	64.01±0.24	67.99±0.56
SiGAT	50.12±1.80	50.86±1.45	57.68±0.63	54.82±0.32	60.21±0.26
SDGNN	48.05±0.29	54.77±0.67	62.35±1.09	62.82±4.16	69.48±0.13
SNEA	47.61±1.26	49.25±0.86	59.30±1.32	57.66±0.24	60.35±0.45
SSSNET	49.53±1.13	52.75±1.71	65.84±0.77	64.53±1.98	69.89±2.26
SigMaNet	59.59±1.68	60.79±0.82	74.09±0.14	78.54±0.17	79.12±0.22
MSGNN	58.91±1.17	63.12±0.86	75.07±0.41	**79.46±0.25**	80.96±0.32
QuaterGCN	**61.74±0.94**	**65.36±0.84**	**75.19±0.47**	79.21±0.13	**81.10±0.18**

Table 5: Accuracy (%) on the four-class edge prediction tasks.

	Five-Class Edge prediction				
	Bitcoin Alpha	Bitcoin OTC	WikiRfa	Slashdot	Epinions
SGCN	78.43±0.36	77.54±0.56	67.74±0.29	64.74±0.16	74.07±0.32
SiGAT	76.68±0.47	74.37±1.18	58.49±1.51	48.01±0.95	57.58±1.34
SDGNN	77.75±0.82	77.28±0.58	62.83±1.90	60.53±4.88	73.27±0.09
SNEA	79.25±0.38	77.36±0.27	62.61±0.44	62.21±0.16	70.70±0.31
SSSNET	77.89±0.41	75.06±0.55	63.74±2.58	67.15±0.44	73.40±1.16
SigMaNet	81.68±0.37	80.92±0.36	74.22±0.12	78.31±0.06	82.85±0.08
MSGNN	81.95±0.47	82.02±0.13	**76.63±0.24**	78.45±0.35	83.54±0.23
QuaterGCN	**82.56±0.46**	**82.13±0.21**	76.33±0.16	**78.55±0.35**	**84.03±0.09**

Table 6: Accuracy (%) on the five-class edge prediction tasks.

Conclusions

We have proposed the *Quaternionic Laplacian* $L^{\mathbb{Q}}$, a quaternion-valued graph Laplacian which generalizes different previously-proposed Laplacian matrices while allowing for the seamless representation of graphs and digraphs of any weight and sign featuring any number of digons without suffering from losses of topological information. We have then proposed QuaterGCN, a spectral GCN with quaternionic network weights that employs a quaternion-valued convolution operator built on top of $L^{\mathbb{Q}}$. Our extensive experimental campaign has highlighted the advantages of employing our quaternion-valued graph Laplacian matrix to leverage the full topology of input graphs featuring digons. Future works include extending the Quaternionic Laplacian to multi-relational graphs with multiple directional edges and to temporal (time-extended) graphs.

Acknowledgements

This work was partially funded by the PRIN 2020 project *ULTRA OPTYMAL - Urban Logistics and sustainable TRAnsportation: OPtimization under uncertainty and MAchine Learning* (grant number 20207C8T9M) and the PRIN-PNRR 2022 project *HEXAGON: Highly-specialized EXact Algorithms for Grid Operations at the National level* (grant number P20227CYT3), both funded by the Italian University and Research Ministry.

References

Bovet, A.; and Grindrod, P. 2020. The activity of the far right on Telegram. *ResearchGate preprint*, DOI: 10.13140/RG.2.2.16700.05764: 1–19.

Bronstein, M. M.; Bruna, J.; LeCun, Y.; Szlam, A.; and Vandergheynst, P. 2017. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4): 18–42.

Carlone, L.; Tron, R.; Daniilidis, K.; and Dellaert, F. 2015. Initialization techniques for 3D SLAM: A survey on rotation estimation and its use in pose graph optimization. In *2015 IEEE international conference on robotics and automation (ICRA)*, 4597–4604. IEEE.

Defferrard, M.; Bresson, X.; and Vandergheynst, P. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29.

Derr, T.; Ma, Y.; and Tang, J. 2018. Signed graph convolutional networks. In *2018 IEEE International Conference on Data Mining (ICDM)*, 929–934. IEEE.

Dixit, P.; and Silakari, S. 2021. Deep learning algorithms for cybersecurity applications: A technological and status review. *Computer Science Review*, 39: 100317.

Fanuel, M.; Alaíz, C. M.; and Suykens, J. A. K. 2017. Magnetic eigenmaps for community detection in directed networks. *Physical Review E*, 95(2).

Fiorini, S.; Coniglio, S.; Ciavotta, M.; and Messina, E. 2023. Sigmanet: One laplacian to rule them all. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, 7568–7576.

Hamilton, W.; Ying, Z.; and Leskovec, J. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30: 1–11.

Hamilton, W. R. 1866. *Elements of quaternions*. Longmans, Green, & Company.

Hammond, D. K.; Vandergheynst, P.; and Gribonval, R. 2011. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 30(2): 129–150.

He, Y.; Perlmutter, M.; Reinert, G.; and Cucuringu, M. 2022a. Msgnn: A spectral graph neural network based on a novel magnetic signed laplacian. In *Learning on Graphs Conference*, 40–1. PMLR.

He, Y.; Reinert, G.; Wang, S.; and Cucuringu, M. 2022b. SSSNET: Semi-Supervised Signed Network Clustering. In *Proceedings of the 2022 SIAM International Conference on Data Mining (SDM)*, 244–252. SIAM.

He, Y.; Zhang, X.; Huang, J.; Rozemberczki, B.; Cucuringu, M.; and Reinert, G. 2022c. PyTorch Geometric Signed Directed: A Software Package on Graph Neural Networks for Signed and Directed Graphs. *arXiv preprint arXiv:2202.10793*.

Huang, J.; Shen, H.; Hou, L.; and Cheng, X. 2019. Signed graph attention networks. In *International Conference on Artificial Neural Networks*, 566–577. Springer.

Huang, J.; Shen, H.; Hou, L.; and Cheng, X. 2021. SDGNN: Learning Node Representation for Signed Directed Networks. 35: 196–203.

Jia, Y.-B. 2008. Quaternions and rotations. *Com S*, 477(577): 15.

Khan, W. A.; Chung, S. H.; Awan, M. U.; and Wen, X. 2020. Machine learning facilitated business intelligence (Part I) Neural networks learning algorithms and applications. *Industrial Management & Data Systems*, 120(1): 164–195.

Kipf, T. N.; and Welling, M. 2017. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*.

Klicpera, J.; Bojchevski, A.; and Günnemann, S. 2019. Predict then propagate: Graph neural networks meet personalized pagerank. In *Proceedings of the 7th International Conference on Learning Representations*, 1–15.

Ko, T. 2022. A Graph Convolution for Signed Directed Graphs. *arXiv preprint arXiv:2208.11511*.

Kumar, S.; Spezzano, F.; Subrahmanian, V. S.; and Faloutsos, C. 2016. Edge Weight Prediction in Weighted Signed Networks. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, 221–230.

Le, T.; Tran, H.; and Le, B. 2023. Knowledge graph embedding with the special orthogonal group in quaternion space for link prediction. *Knowledge-Based Systems*, 110400.

Leskovec, J.; Huttenlocher, D.; and Kleinberg, J. 2010. Signed networks in social media. In *Proceedings of the SIGCHI conference on human factors in computing systems*, 1361–1370.

Li, Y.; Tian, Y.; Zhang, J.; and Chang, Y. 2020. Learning signed network embedding via graph attention. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, 4772–4779.

Li, Y.; Zemel, R.; Brockschmidt, M.; and Tarlow, D. 2016. Gated Graph Sequence Neural Networks. In *Proceedings of ICLR'16*.

Lieb, E. H.; and Loss, M. 1993. Fluxes, Laplacians, and Kasteleyn's theorem. In *Statistical Mechanics*, 457–483. Springer.

Nguyen, T. D.; Phung, D.; et al. 2021. Quaternion graph neural networks. In *Asian conference on machine learning*, 236–251. PMLR.

Parcollet, T.; Morchid, M.; and Linarès, G. 2020. A survey of quaternion neural networks. *Artificial Intelligence Review*, 53: 2957–2982.

Parcollet, T.; Ravanelli, M.; Morchid, M.; Linarès, G.; Trabelsi, C.; Mori, R. D.; and Bengio, Y. 2019. Quaternion Recurrent Neural Networks. In *International Conference on Learning Representations*.

Qi, L.; and Luo, Z. 2021. A Note on Quaternion Skew-Symmetric Matrices. *arXiv preprint arXiv:2110.09282*.

Tong, Z.; Liang, Y.; Ding, H.; Dai, Y.; Li, X.; and Wang, C. 2021. Directed graph contrastive learning. *Advances in Neural Information Processing Systems*, 34: 19580–19593.

Tong, Z.; Liang, Y.; Sun, C.; Li, X.; Rosenblum, D. S.; and Lim, A. 2020a. Digraph inception convolutional networks. *Advances in Neural Information Processing Systems*, 2020-December(NeurIPS): 1–12.

Tong, Z.; Liang, Y.; Sun, C.; Rosenblum, D. S.; and Lim, A. 2020b. Directed Graph Convolutional Network. arXiv:2004.13970.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L. u.; and Polosukhin, I. 2017. Attention is All you Need. In Guyon, I.; Luxburg, U. V.; Bengio, S.; Wallach, H.; Fergus, R.; Vishwanathan, S.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2018. Graph Attention Networks. In *International Conference on Learning Representations*.

Wang, C.; Li, L.; Zhang, H.; and Li, D. 2022. Quaternion-based knowledge graph neural network for social recommendation. *Knowledge-Based Systems*, 257: 109940.

West, R.; Paskov, H. S.; Leskovec, J.; and Potts, C. 2014. Exploiting social network structure for person-to-person sentiment analysis. *Transactions of the Association for Computational Linguistics*, 2: 297–310.

Wu, L.; Wang, D.; Feng, S.; Zhou, X.; Zhang, Y.; and Yu, G. 2022. Graph Collaborative Filtering for Recommendation in Complex and Quaternion Spaces. In *Proc. of Web Information Systems Engineering (WISE 2022): 23rd International Conference*, 579–594. Springer.

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2018. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*.

Ye, Z.; Kumar, Y. J.; Sing, G. O.; Song, F.; and Wang, J. 2022. A Comprehensive Survey of Graph Neural Networks for Knowledge Graphs. *IEEE Access*, 10: 75729–75741.

Zhang, J.; Hui, B.; Harn, P.-W.; Sun, M.-T.; and Ku, W.-S. 2021a. sMGC: A Complex-Valued Graph Convolutional Network via Magnetic Laplacian for Directed Graphs. arXiv:2110.07570.

Zhang, X.; He, Y.; Brugnone, N.; Perlmutter, M.; and Hirn, M. 2021b. Magnet: A neural network for directed graphs. *Advances in neural information processing systems*, 34: 27003–27015.