Haar-Laplacian for directed graphs*

Theodor-Adrian Badea, Bogdan Dumitrescu

Department of Automatic Control and Computers
National University of Science and Technology Politehnica Bucharest, Romania
Emails: theodor.badea@upb.ro, bogdan.dumitrescu@upb.ro.

Abstract

This paper introduces a novel Laplacian matrix aiming to enable the construction of spectral convolutional networks and to extend the signal processing applications for directed graphs. Our proposal is inspired by a Haar-like transformation and produces a Hermitian matrix which is not only in one-to-one relation with the adjacency matrix, preserving both direction and weight information, but also enjoys desirable additional properties like scaling robustness, sensitivity, continuity, and directionality. We take a theoretical standpoint and support the conformity of our approach with the spectral graph theory. Then, we address two use-cases: graph learning (by introducing HaarNet, a spectral graph convolutional network built with our Haar-Laplacian) and graph signal processing. We show that our approach gives better results in applications like weight prediction and denoising on directed graphs.

Keywords: directed graphs, Laplacian

1. Introduction

Let $\mathcal{G} = (\mathcal{V},\ \mathcal{E})$, with $|\mathcal{V}| = N$, be a graph; $\mathcal{V}$ and $\mathcal{E}$ are the sets of vertices and edges, respectively. We assume that the graph is simple: there is at most an edge (u, v), for any vertices $u, v \in \mathcal{V}$. If the graph is undirected, then the edges (u, v) and (v, u) are identical. An edge $(u, v) \in \mathcal{E}$ has a weight $w_{uv} \in \mathbb{R}$, $w_{uv} \neq 0$; the weights can have negative values. The adjacency

*This work was supported by a grant of the Ministry of Research, Innovation and Digitization, CNCS - UEFISCDI, project number PN-III-P4-PCE-2021-0154, within PNCDI III.

matrix A of $\mathcal{G}$ is defined by

$$a_{uv} = \begin{cases} w_{uv}, & \text{if } (u,v) \in \mathcal{E} \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

We assume that the graph has no self loops, hence $a_{uu} = 0$, $\forall u \in \mathcal{V}$. We also assume that the graph is connected; there are no isolated vertices.

Let D be the diagonal matrix

$$D = \operatorname{diag}(|A| \cdot \mathbf{1}), \tag{2}$$

where $|A|$ is the matrix whose elements are $|a_{uv}|$ and $\mathbf{1}$ is a vector with all elements equal to 1. When all weights are equal to one, this is the degree matrix.

The Laplacian associated to the graph is

$$L = D - A \tag{3}$$

and the symmetrically normalized Laplacian is

$$\bar{L} = D^{-1/2} L D^{-1/2} = I_N - D^{-1/2} A D^{-1/2}. \tag{4}$$

If the graph is undirected with nonnegative weights, the Laplacian enjoys nice spectral properties, which proved useful in graph signal processing [1, 2, 3]. In particular, both Laplacians are symmetric and positive semidefinite; the eigenvalues of the normalized Laplacian (4) lie in the interval $[0, 2]$. The case of negative weights in undirected graphs is covered in [4].

Contribution. We propose here a new construction, called Haar-Laplacian, based on the symmetrized and skew-symmetrized adjacency matrices associated with a directed graph. Like other Laplacians for directed graphs [5, 6, 7], the Haar-Laplacian is a Hermitian matrix and thus is has real eigenvalues and orthogonal eigenvectors. We show that the Haar-Laplacian enjoys some properties that former Laplacians do not share. Moreover, we use the Haar-Laplacian in two types of applications. For graph learning, we introduce a spectral graph convolutional network called HaarNet and show its behavior for several instances of the link prediction problem in directed graphs; the best results are obtained for weight prediction, where the one-to-one relation of the Haar-Laplacian with the adjacency matrix gives it an edge over the existing approaches that do not have this property. We also illustrate the benefits of the Haar-Laplacian in graph signal processing tasks like denoising. A related work is [8], where we have used the symmetrized

and skew-symmetrized adjacency matrices for building separate blocks in an autoencoder network dedicated to anomaly detection; no connection with the Laplacian was made; no results, theoretical or experimental, are replicated in this paper.

Contents. In Sec. 2, we review the state-of-the-art in the topic of Laplacians for directed graphs, from two perspectives, those of graph learning and graph signal processing. In Sec. 3, we introduce the Haar-Laplacian and discuss its properties, showing its advantages over existing Laplacians. We also present its properties for simple graphs. Section 4 is dedicated to the description of the graph learning problems that we tackle and to the presentation of the solution based on the Haar-Laplacian, the HaarNet network. Section 5 contains the numerical results obtained for learning (three types of link prediction problems) and signal processing (denoising). Section 6 concludes the paper.

2. Context

The core of our discussion is a key concept in spectral graph theory, namely the Graph Fourier Transform (GFT). The transformation itself leverages the eigenvectors of the Laplacian to perform signal processing on graphs. Essentially, the GFT enables the analysis and manipulation of signals defined over graph nodes, transforming them from the vertex domain to a frequency domain that reflects the underlying graph structure. This notion is employed both in the learning and signal processing matters that we will further discuss throughout this paper.

Considering a graph signal $x \in \mathbb{R}^N$ on the graph $\mathcal{G}$, the Fourier transform and its inverse are defined as

$$\hat{x} = U^T x, \quad x = U\hat{x}. \tag{5}$$

where $U = [u_0, ..., u_{N-1}]$ is the matrix of eigenvectors (Fourier basis) corresponding to the diagonal matrix $\Lambda = \text{diag}([\lambda_0, ..., \lambda_{N-1}])$ of eigenvalues (frequencies) of the Laplacian. As mentioned in the previous section, if $\mathcal{G}$ is undirected with nonnegative weights then it enjoys nice spectral properties. It has a full set of real and nonnegative eigenvalues with corresponding orthonormal eigenvectors, thus highlighting the meaning of Λ and U in the GFT definition.

However, when the graph is directed, the spectrum of the Laplacian is complex and it is hard to associate it with meaningful spectral graph operations. Several Laplacians have been proposed, starting with that of

the symmetrized adjacency matrix

$$A_s = \frac{1}{2}\left(A + A^T\right). \tag{6}$$

Using it in (3) or (4) gives symmetric positive semidefinite Laplacians. However, such a Laplacian catches only the average weight between two nodes; the differences are lost. It is in fact a brutal replacement of a directed graph with an undirected one, all information about orientation being lost. Such a Laplacian was used in [9], in the context of a spectral graph convolutional network (GCN), with good results in classification. When only connectivity matters, this Laplacian can be successfully used.

2.1. Learning perspective

We start by discussing the viewpoint taken in learning applications related to directed graphs.

Perhaps the most prominent association of the Laplacian within the frames of learning applications is with spectral graph convolutional networks (GCNs), which extend the convolutional neural networks (CNNs) to graph-structured data. Graphs lack an appropriate translation operation, therefore the convolution in the vertex domain is unfeasible. However, the convolution operation on graphs is enabled in the spectral domain by means of the graph Fourier transform (GFT) [10]. Then, in a manner similar to the classical signal processing, the convolution of two graph signals x and y in the spectral domain turns into a multiplication:

$$x *_{\mathcal{G}} y = U\left(\left(U^T x\right) \odot \left(U^T y\right)\right), \tag{7}$$

where U is the GFT matrix and $\odot$ is the element-wise (Hadamard) product.

This approach allows GCNs to extend the concept of spatial locality to the unique and irregular structure of graphs. In plain terms, a layer of a GCN is a convolution between a graph signal and a learnable filter g_θ. A rudimentary g_θ could be a non-parametric filter, but such filter would not be localized [11]. Moreover, the convolution itself is a computationally intensive matrix multiplication and dealing with large graphs makes the Laplacian eigendecomposition a prohibitive operation. Hence, to address these issues, it is suggested to make use of a recursively-formulated polynomial evaluated in the Laplacian as a parametrization for the filter. The recurrence of Chebyshev polynomials can be exploited [12] to approximate g_θ as a truncated expansion, which further allows writing the filtering procedure

as

$$g_\theta *_\mathcal{G} x = g_\theta\left(\bar{L}\right) x = \sum_{k=0}^{K-1} \theta_k T_k\left(\frac{2}{\lambda_{max}}\bar{L} - I_N\right) x, \tag{8}$$

where θ_k are learnable filter coefficients and λ_{max} is the largest eigenvalue of the normalized Laplacian $\bar{L}$. In [9], the simplification to a model that performs first-order local smoothing is proposed, which is computationally efficient and sufficient for many tasks. Additionally, $\lambda_{max} = 2$ is employed as an approximation, this way eliminating the need to actually compute the largest eigenvalue.

To address the asymmetry issues of directed graphs, several approaches to extend the spectral methods have been proposed. The most interesting build a Laplacian by using Hermitian matrices, which have real eigenvalues. The magnetic Laplacian [5, 6] is such a solution. (An early proposal [13] has several computational disadvantages and its normalized version can be dense.) Denoting

$$D_s = \mathrm{diag}(|A_s| \cdot \mathbf{1}), \tag{9}$$

the magnetic Laplacian is

$$L_m = D_s - H, \tag{10}$$

with

$$H = A_s \odot \exp(i\mathbf{\Theta}), \tag{11}$$

where $\odot$ is the element-wise (Hadamard) product,

$$\theta_{uv} = 2\pi q(a_{uv} - a_{vu}), \tag{12}$$

and $q \geq 0$ is a parameter. So, $\mathbf{\Theta}$ acts as a phase matrix, being zero when the graph is undirected; the direction of the edges is visible in $\mathbf{\Theta}$; if $(u, v) \in \mathcal{E}$, but $(v, u) \notin \mathcal{E}$, then $\mathrm{Im}(h_{uv}) > 0$; in the opposite case, where $(v, u) \in \mathcal{E}$, but $(u, v) \notin \mathcal{E}$, then $\mathrm{Im}(h_{uv}) < 0$. Taking $q = 0.25$, if $|a_{uv}| \leq 1$, it follows that $\theta_{uv} \in [-\pi, \pi]$, which ensures the unicity of the phase. The magnetic Laplacian (10) is positive semidefinite. The same property applies to its normalized version

$$\bar{L}_m = I_N - D_s^{-1/2} H D_s^{-1/2}. \tag{13}$$

A drawback of the magnetic Laplacian is its lack of robustness to scaling. When multiplying A with a constant c, the real part of (10) scales correctly; however, the phase is multiplied with c, so it may go outside the interval $[-\pi, \pi]$ and thus change its meaning. For example, take $a_{uv} = 1$, $a_{vu} = 0$ and so $h_{uv} = 0.5(\cos(\pi/2) + i\sin(\pi/2)) = 0.5i$. Then, for the matrix $4A$ we would obtain $h_{uv} = 2(\cos(2\pi) + i\sin(2\pi)) = 2$; this value is the same as

that for an adjacency matrix with $a_{uv} = a_{vu} = 2$, which has two antiparallel edges and thus could be associated with symmetry. Similarly, for the matrix $3A$, it results that $h_{uv} = 1.5(\cos(3\pi/2) + i\sin(3\pi/2)) = -1.5i$; hence, one may say that the edge is reversed (or the sign of the weight is flipped).

So, working with the magnetic Laplacian implies special care in choosing the value of the parameter q. For example, in [14], the value $q = 1/2\max(a_{uv} - a_{vu})$ is taken, ensuring non-ambiguity of phase. In [15], the value of q is chosen by appealing to Johnson's algorithm for detecting directed cycles.

A more general cure was proposed in [7], where the sign-magnetic Laplacian was introduced. It replaces (11) with

$$H = A_s \odot \left(\mathbf{1} \cdot \mathbf{1}^T - \operatorname{sgn}(|A - A^T|) + i \cdot \operatorname{sgn}(|A| - |A^T|)\right), \qquad (14)$$

where sgn is the sign function. Indeed, in this case scaling works as it should: magnitude is modified, but phase is not. In all the above examples with $a_{uv} = 1$, $a_{vu} = 0$, it results that h_{uv} is purely imaginary, with positive imaginary part. Inserting (14) in (10) and (13) gives positive definite matrices.

However, while the sign-magnetic Laplacian captures perfectly the orientation of the edges, it is not devoid of criticism. Essentially, we would like that different graphs have different Laplacian, a fact that is true for undirected graphs; however, this is not true for the Laplacians in this section. In particular, the sign-magnetic Laplacian fails when both edges (u, v) and (v, u) exist. It is easy to see in (14) that the same value h_{uv} is obtained by taking, for example, $a_{uv} = 10000$ and $a_{vu} = 1$ in one case and $a_{uv} = 5001$ and $a_{vu} = 5000$ in the other; in both cases, it results that $h_{uv} = 5000.5i$; so, there is no sensitivity to variations that may be huge. Also, the sign-magnetic Laplacian is not continuous; if $a_{uv} = 1$ and $a_{vu} = 1$, then $h_{uv} = 1$; if $a_{uv} = 1.01$ and $a_{vu} = 0.99$, then $h_{uv} = i$; there is a palpable discontinuity even though the change of weights is minor. Finally, the sign-magnetic Laplacian is zero when A is skew-symmetric, since $A_s = 0$.

We thus appreciate that the sign-magnetic Laplacian is not necessarily the best option for graphs with pairs of antiparallel edges, like those modeling money transfers (it matters how imbalanced are the transfers between two accounts, not only the average transferred amount) or trust networks; if trust is evaluated on a scale from -10 to 10, having $a_{uv} = 10$ and $a_{vu} = -10$ means that u totally trusts v, but v has no confidence at all in u; this can be a stab-in-the-back situation; while $a_{uv} = 1$ and $a_{vu} = -1$ might mean that u and v barely know each other; however, both cases are coded with the

same value, 0, in the sign-magnetic Laplacian (as well as in the magnetic Laplacian).

Other Laplacians in the same vein [14, 16, 17] share some of the above weaknesses. In [17], the phase is coded such that all edges orientations and weight signs have distinct values in the Laplacian; however, like in the sign-magnetic Laplacian, there is no sensitivity to weight disparity in antiparallel edges. Although not used in the spectral context, the Hermitian adjacency matrix [18, 19], which may be seen as a simplified version of (11), has been recently used in learning [20] and could be used for building a Laplacian.

In [21], the antiparallel edges (digons) are finally addressed and a solution is proposed by the use of quaternions, as a generalization of both the classical and the sign-magnetic Laplacians. The Laplacian is no longer a Hermitian complex matrix, but a Hermitian quaternion. Different graphs have different Laplacians; however, the mapping is not surjective: there are Hermitian quaternions that do not correspond to graphs. So, there is finally a distinction between all situations, but it is obtained with an excess of means. Also, the mapping from the adjacency matrix to the quaternion Laplacian is not continuous; the situation is similar to that of the sign-magnetic Laplacian.

Another track of research, that we do not follow here, is based on Laplacians that are symmetric although the graph is directed. An example, based on a personalized PageRank approach, is [22].

2.2. Signal processing perspective

We present now the viewpoint taken in graph signal processing (GSP) applications, where the main theoretical notions associated to a Laplacian are those of Graph Fourier Transform (GFT) and graph frequencies [23]. Since the topic is very generous and a wealth of information can be found in overview papers [1, 2, 3] (the latter two emphasizing the case of directed graphs), we only mention the main approaches.

Random walks on directed graphs lead to a Laplacian construction [24]; the same work also defines the notion of degree of asymmetry, which measures the skewness of the Laplacian (and adjacency matrix). A more detailed analysis is presented in [25]. The GFT may be obtained by diagonalization, which is not always possible. Like the GFT via the Jordan decomposition [26], computation can be affected by numerical instability. Also, the GFT is not orthogonal, which is always desirable for a transform.

So, again, Hermitian Laplacians come into play, with their convenient spectral properties. For example, the magnetic Laplacian was proposed for GSP in [27].

In [28], the singular value decomposition (SVD) of (3) is used as basis for a GFT. The GFT is real, but its size is double, since two orthogonal matrices are involved. The singular values of the Laplacian are the frequencies. The left and right singular vectors get the meaning of frequency components. A somewhat related method, in the sense that SVD has a central role, uses the polar decomposition [29].

Optimization based methods try to build an orthogonal transformation that minimizes the directed total variation [30] or the spectral dispersion [31] of the graph. A more general view of the GFT is given in [32].

A comparison of these Laplacians would be quite ample, since theoretically they all have some useful properties and the range of applications is large. So, we will proceed by introducing our proposed Laplacian.

3. The Haar-Laplacian

The shortcomings discussed in Section 2.1 are eliminated by our proposal. Let

$$A_a = \frac{1}{2}\left(A - A^T\right) \tag{15}$$

be the skew-symmetric matrix naturally generated from A.

Remark 1. Building the symmetrized (6) and skew-symmetrized (15) matrices amounts to a Haar-like transformation (spatial instead of temporal) of the weights. Together, these matrices contain the same information as A (the "inverse transform" is $A = A_s + A_a$); the symmetric part contains the averages ("low frequency" information) and the skew-symmetric part contains the differences ("high frequency"). So, the situation is similar with that of a Haar filter bank, although no explicit downsampling is involved; the number of independent elements is the same in A as in A_s and A_a together, and the transformation is linear. ■

Definition 2. Consider the Hermitian matrix

$$H_h = A_s + iA_a \tag{16}$$

and the diagonal matrix

$$D_h = \mathrm{diag}(|H| \cdot \mathbf{1}). \tag{17}$$

We define the *Haar-Laplacian*

$$L_h = D_h - H_h \tag{18}$$

and the *normalized Haar-Laplacian*

$$\bar{L}_h = I_N - D_h^{-1/2} H_h D_h^{-1/2}. \tag{19}$$

Remark 3. The elements of the Hermitian matrix (16) obey to

$$|h_{uv}| = \sqrt{\frac{1}{2}(a_{uv}^2 + a_{vu}^2)}. \tag{20}$$

Proposition 4. The Haar-Laplacians (18) and (19) are positive semidefinite and have real eigenvalues. The eigenvalues of the normalized Haar-Laplacian (19) lie in the interval $[0, 2]$.

Proof. Both Haar-Laplacians are Hermitian matrices and hence have real eigenvalues. The matrix

$$\bar{H_h} = D_h^{-1/2} H_h D_h^{-1/2}$$

is similar with

$$D_h^{-1/2}(D_h^{-1/2} H_h D_h^{-1/2}) D_h^{1/2} = D_h^{-1} H_h.$$

The sum of the absolute values of the (non-diagonal) elements on each row is

$$\sum_{v=1, v\neq u} \frac{|h_{uv}|}{d_{uu}} \leq 1,$$

due to (17). Since the diagonal elements of the matrix are zero, Gershgorin's circle theorem [33] says that its eigenvalues lie inside the unit disk. It results immediately that the eigenvalues of (19) are in $[0, 2]$ and hence the matrix (19) is positive semidefinite. Since

$$L_h = D_h^{1/2} \bar{L}_h D_h^{1/2},$$

the matrix (18) is also positive semidefinite. ■

Remark 5. The properties discussed critically in Section 2.1 for other Laplacians hold trivially for the Haar-Laplacian (18). Some of the properties follow from the fact that no information is lost from A to H_h; the map $A \longrightarrow H_h$ is one-to-one; none of the previous Laplacians has this property.

Scaling. If A is replaced by cA, then H_h is replaced by cH_h. The ratio between the imaginary and real parts stays the same, so the phase is unchanged.

Sensitivity. A change in A will lead to a change in H_h.

Continuity. The map $A \longrightarrow H_h$ is continuous.

Orientation. The Haar-Laplacians clearly show the prevalent orientation of the edges. If $a_{uv} > a_{vu}$ (one of the weights may be zero, if the edge does not exist), then the signs of the real and imaginary parts of h_{uv} show if the flow goes from u to v or viceversa and if the average flow is positive or negative. One can see also if only one edge exists: the real and imaginary parts are equal in absolute value. (This wasn't possible for the magnetic and sign-magnetic Laplacians.) ■

Remark 6. The idea of the proof of Prop. 4, based on Gershgorin circles, can be used for most Laplacians mentioned in this paper; the proof is much shorter than existing ones. ■

Remark 7. The Haar-Laplacian is Hermitian, hence it has real eigenvalues. However, the eigenvectors are complex, in general. They form a unitary matrix.

When the matrix A is symmetric, $H_h = A_s$ is real and symmetric and its eigenvectors are real. The Haar-Laplacian is identical with the standard Laplacian (3) (and so are the magnetic Laplacians).

When the matrix A is skew-symmetric and so $A_s = 0$, then $H_h = iA_a$ and its eigenvectors have real and imaginary parts with equal norm for all nonzero eigenvalues. This property is certainly known, but we did not find a proof. So, we give it below. Note that for skew-symmetric matrices, both magnetic and sign-magnetic Laplacians are zero. ■

Proposition 8. Let A_a be a real skew-symmetric matrix, $i\lambda$, with $\lambda \in \mathbb{R}$, one of its eigenvalues (a skew-symmetric matrix has purely imaginary eigenvalues), and $x + iy$ the corresponding eigenvector. If $\lambda \neq 0$, then $\|x\| = \|y\|$. If $\lambda = 0$, then $y = 0$.

Proof. Since $A_a(x + iy) = i\lambda(x + iy)$, it results that $A_a x = -\lambda y$ and $A_a y = \lambda x$. From the first equality, we can write $|\lambda|\|y\| = \|A_a x\| \leq \|A_a\|\|x\|$. If $\lambda \neq 0$, taking into account that $|\lambda| \leq \|A_a\|$, it results that $\|y\| \geq \|x\|$. Similarly, from the second inequality we obtain $\|x\| \geq \|y\|$.

If $\lambda = 0$, then $Ax = Ay = 0$, hence one can take $y = 0$ (the eigenvector is naturally real). ■

Let

$$L_h = U\Lambda U^H \tag{21}$$

be the eigenvalue decomposition of the Haar-Laplacian (18), where $\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_n)$, $\lambda_1 \leq \ldots \leq \lambda_n$, and $U \in \mathbb{C}^{n\times n}$ is unitary and contains

on columns the eigenvectors $u_1, \ldots, u_n$. In GSP context, the matrix U is called GFT and the eigenvalues $\lambda_j \geq 0$ are named graph frequencies. As discussed in [23] and other works, smaller eigenvalues are associated with lower frequencies.

Example 9. A testbed for the GFT is its form for the directed cycle graph, whose adjacency matrix is (here illustrated for $n = 4$)

$$S = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix} \tag{22}$$

The corresponding Haar-Laplacian is

$$L_h = \sqrt{2}I - \frac{1}{2}(S + S^T) - i\frac{1}{2}(S - S^T) = \sqrt{2}I - \frac{1+i}{2}S - \frac{1-i}{2}S^T.$$

Like (22), it is also a circulant matrix, hence the GFT U_h from (21) is the discrete Fourier transform matrix W. So, the Fourier transformation of this graph is the same as in classical signal processing, like for the random walk Laplacian [25]. However, the frequencies are different. Since S is also diagonalized by W and its eigenvalues are $\exp(2\pi ij/n)$, $j = 0 : n-1$, the eigenvalues of L_h are

$$\sqrt{2} - \cos(2\pi j/n) - \sin(2\pi j/n), \ \ j = 0 : n-1. \tag{23}$$

They take values in the interval $[0, \sqrt{2}]$. Due to the symmetries of the function $\cos\xi + \sin\xi$ with respect to $\pi/4$ and $5\pi/4$, the n values (23) are distinct when n is not a multiple of 4; otherwise, excepting the smallest and the largest, they appear in pairs.

For the adjacency matrix of a directed cycle graph, as well as for matrices that have a single value of 1 on each row, one can see from (10)-(12) and (18) that, if $q = 1/8$, the Haar and magnetic Laplacians are related through $L_h = \sqrt{2}L_m$. So, the two Laplacians produce essentially identical frequencies. However, for other values of $q \leq 1/2$ they are different.

It is also immediate that for $q = 1/4$, the magnetic and sign-magnetic Laplacians are identical for the directed cycle graph. For other values of $q \leq 1/2$, they are different.

Figure 1 shows the sorted frequencies of the Haar-Laplacian as well as those of the SVD [28], magnetic [6] and sign-magnetic [7] Laplacians for $n \in \{24, 26\}$. The distribution of frequencies is similar, with a slight advantage for the Haar-Laplacian. ■

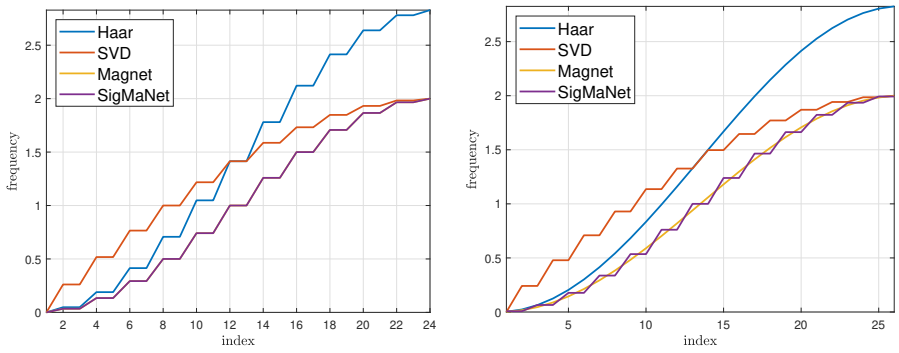

Figure 1: Left: sorted frequencies for Haar, SVD, magnetic ($q = 1/4$) and sign-magnetic Laplacians for $n = 24$ (the latter two curves are identical). Right: same, with $n = 26$, $q = 1/8$.

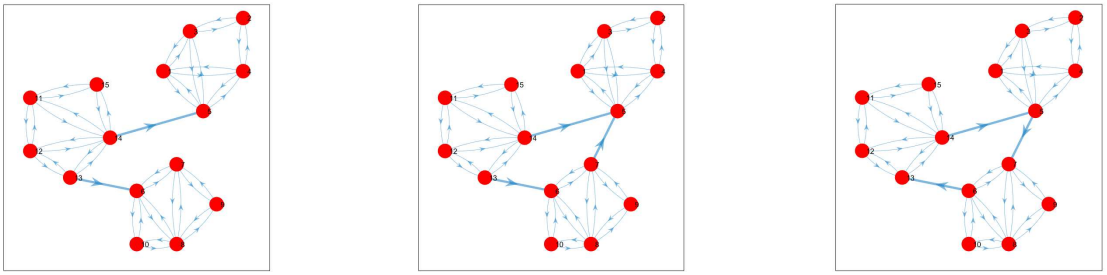

Figure 2: Simple graphs with 15 nodes proposed in [30], named here G15a (left), G15b (middle), G15c (right).

Example 10. The graphs shown in Figure 2 were proposed in [30] for illustrating the properties of the eigenvectors of the GFT in a relatively intuitive way. Some of them were used also in [31, 28]. Each graph has three clusters made of five nodes; the edges in a cluster are all digons; clusters are connected with directed edges represented with thick arrows; the first graph (G15a) has two such directed edges and the second (G15b) has three; the third graph (G15c) has also three and they are part of a cycle.

The sorted frequencies associated with the GFTs of the Haar, SVD, magnetic (with $q = 1/8$) and sign-magnetic Laplacians are shown in Figure 3. There are insignificant differences between the three sets of frequencies; in particular, the magnetic and sign-magnetic Laplacians have the same eigenvalues for G15a. (Note that for $q = 1/4$ the magnetic and sign-magnetic Laplacians are again identical.) The frequencies are nicely spread, so we can consider that all four Laplacians pass this sanity check.

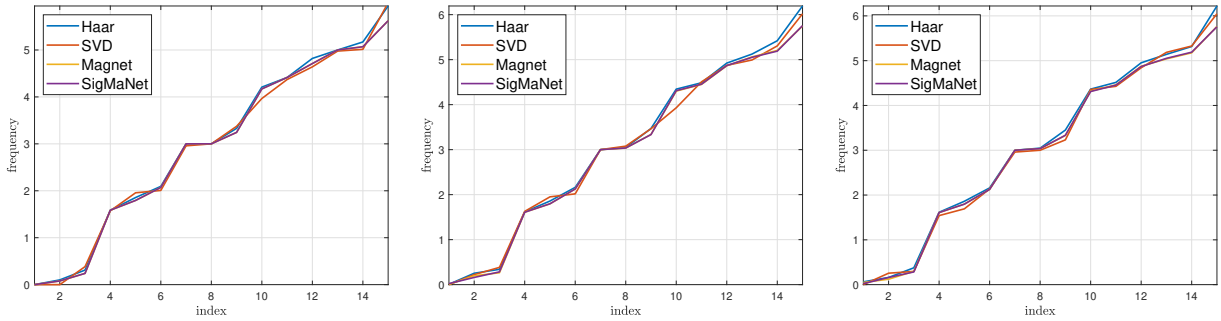

Figure 3: Frequencies associated with the graphs from Fig. 2.

Finally, Figures 4–6 show the values of the $n = 15$ eigenvectors of the Haar-Laplacian GFT. In figure k (counting from the top, on rows), the coefficients of eigenvector u_k (sorted in increasing order of frequencies) are represented with colors; since the coefficients are complex, we chose to represent the value $\operatorname{sign}(\operatorname{Re} u_{\ell k}) \cdot |u_{\ell k}|$ associated with node ℓ.

We note that the eigenvectors appear to follow the intuition on what can represent low and high frequencies in such graphs. The first eigenvector has nearly constant values. The next few ones clearly respect the cluster structure, especially in the case of G15a (Fig. 4), which has the least connectivity between clusters. The eigenvectors corresponding to high frequencies show significant larger variation in the elements, as they should. The nodes connected with other clusters are especially characterized by large coefficients, more so in the graph containing a cycle, G15c (Fig. 6); there, the last two eigenvectors show significant variations of the values corresponding to nodes belonging to the cycle. ■

4. Learning with the Haar-Laplacian

Throughout this section, we first set the context of the problem that we address by means of learning using the proposed Haar-Laplacian. Then, we describe the architecture that we adopted for the neural network model.

4.1. Link prediction problems

In order to assess the capabilities of the proposed Laplacian on learning tasks, we address the *link prediction* problem, a fundamental task in machine learning on graphs, where the goal is to predict the existence of edges (or relationships) between pairs of nodes. This is highly relevant in a variety

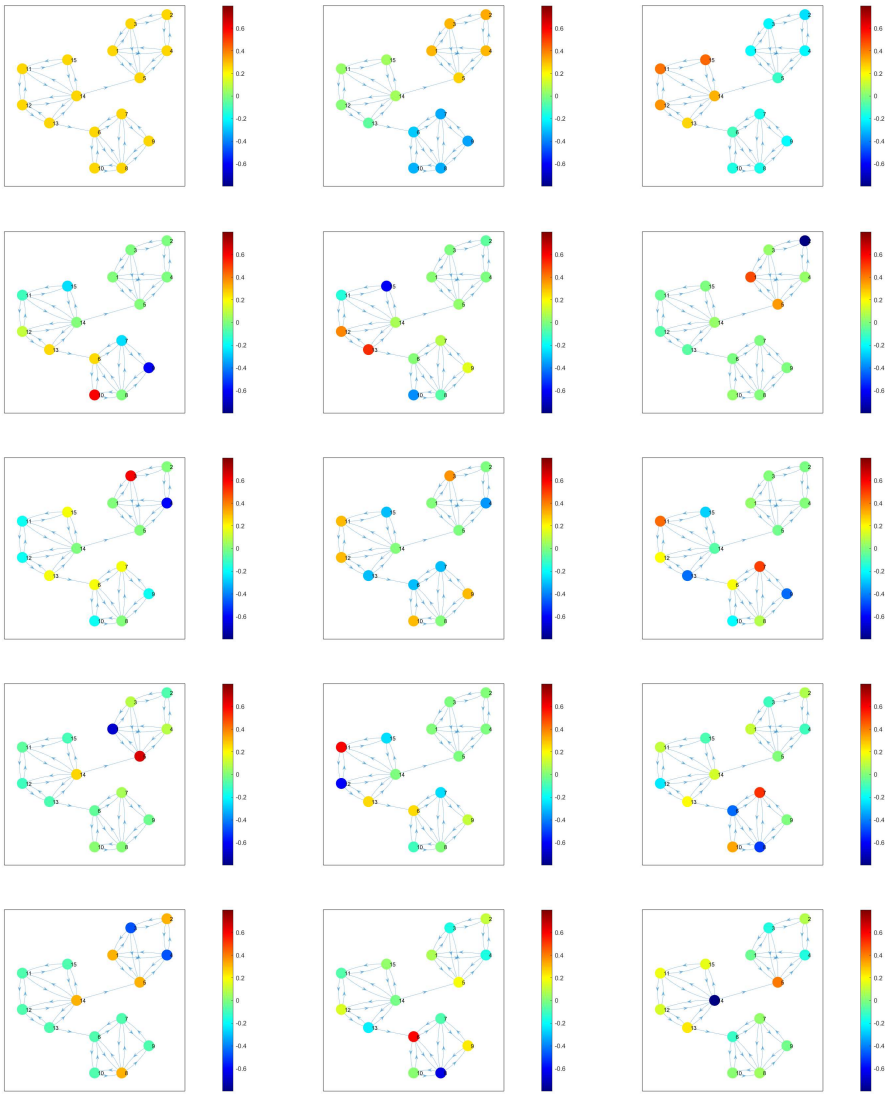

Figure 4: Eigenvectors of graph G15a.

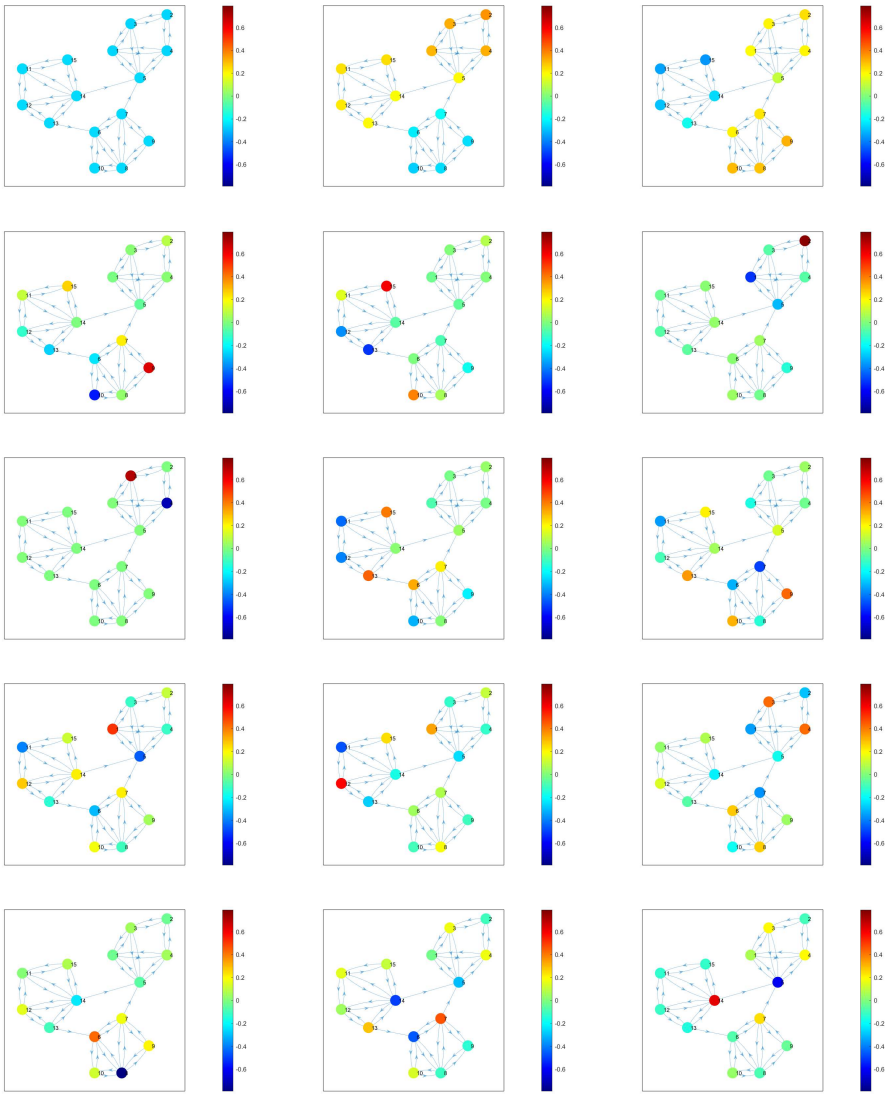

Figure 5: Eigenvectors of graph G15b.

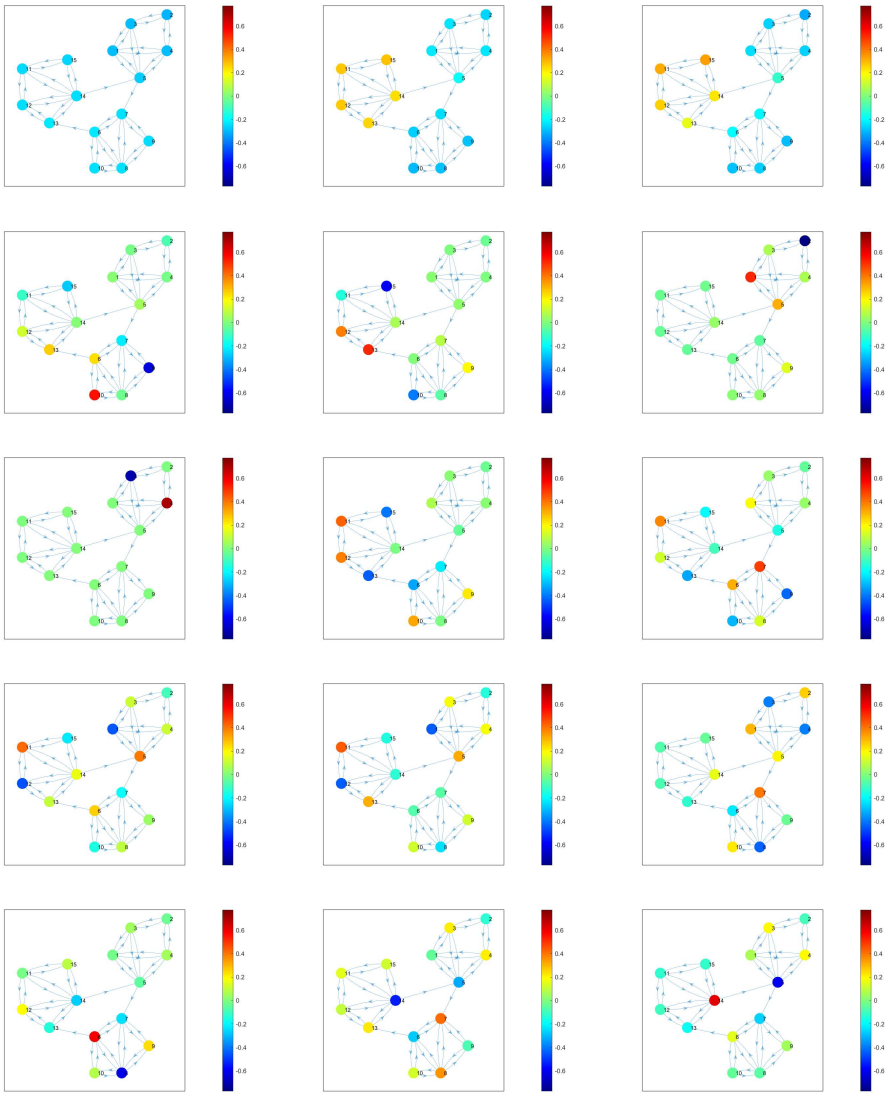

Figure 6: Eigenvectors of graph G15c.

of domains such as social networks, biological networks, and recommender systems, where inferring new or missing connections can provide valuable insights.

Before the advent of GNNs, traditional methods for link prediction relied on graph-based heuristics and feature engineering. Some of these methods include approaches based on graph metrics or indices (e.g. common neighbors, Jaccard coefficient, Salton index, Adamic-Adar index, Katz score, etc.).

Traditional heuristics have been complemented or surpassed by modern GNN-based methods, which aim to infer missing links, discover potential new links, or forecast future connections between vertices in an end-to-end manner by learning complex patterns from graph structure and node features.

GNNs excel at capturing the underlying graph topology and feature relationships, thus they prove highly effective for link prediction tasks. Their advantages include: capturing graph structure by aggregating node and neighbor information (including both local and global relationships in the graph), incorporating node features, and learning node representation which summarize properties of vertices and their neighbors.

We are concerned with three flavors of the link prediction task, which we address in a supervised setting:

- *existence* problem - for an ordered pair of vertices, predict whether the directed edge exists or not;
- *three-class* problem - for an ordered pair of vertices, predict the existence of the directed edge or of the edge in reversed direction;
- *weight prediction* problem - for an ordered pair of vertices, predict the weight of the directed edge (or 0 in case the edge does not exist).

The problem has been addressed in literature by means of both approaches. For example, in [34, 35] the common neighbor metric is used to predict the existence of links in undirected graphs, but without considering any information about the weights. A so-called *reliable-route method* is presented in [36], which extends unweighted local similarity indices to weighted ones, therefore not only embedding weights in the technique, but also predicting both the existence of links and their weights. Link weight prediction in directed graphs with arbitrary weights is studied in [37], where authors introduce the $F \times G$ score, based on two novel measures to characterize a vertex: *fairness* and *goodness*. A more complex approach is shown in [38],

where decision tree ensembles constructed using such heuristics paired with random forest, gradient boosting decision trees, eXtreme gradient boosting, and light gradient boosting machine are proposed. In [39] it is highlighted that each heuristic has a strong assumption on when two vertices are likely to link and their effectiveness on networks where these assumptions fail might be limited. Consequently, it is suggested that a more reasonable approach would be to learn a suitable heuristic from a given network instead of using predefined ones. So, a heuristic learning paradigm for link prediction has been proposed, aiming to learn a function mapping the subgraph patterns to link existence, thus automatically learning a heuristic suitable for the graph. Other approaches are based on inference from neighbor sets [40], matrix factorization [41, 42], feature extraction and learning methods [43, 44].

Graph neural networks have also been employed to address the three flavors of the link prediction problem. Several works tackling the existence and three-class tasks in the context of alternative Laplacians for directed graphs include the magnetic approaches ([6], [7]) and the quaternionic Laplacian [21]. Two rather complicated, costly, and limited to undirected graphs techniques are shown in [45] and [46]. These methods require multiple elaborate steps. The one from [45] relies on subgraph extraction, subgraph node ordering through a variation of the Weisfeiler-Lehman algorithm, and – finally – training a neural network model with a classical convolutional layer and several fully-connected hidden layers. The approach from [46] shares the first two steps, then a line graph transformation is additionally required. Another difference is that [46] employs a GCN model. Finally, some recent works are based on graph attention networks [47], [48]. We note that some of the above approaches are designed for undirected graphs only.

4.2. Network architecture

We tackle the three sub-tasks of the link prediction problem mentioned above by adopting a model similar to the ones used in [6] and [7]. Spectral graph convolutional layers with the proposed Haar-Laplacian are used to learn node representations based on node features and graph structure. Once the node embeddings are learned, a downstream link prediction is performed using concatenation and a dense layer.

The core of the employed neural network model is a stack of convolutional layers, with equal dimensions; each of them is followed by a ReLU [49] nonlinearity. These layers are constructed as a spectral convolutional filter – as described in (8). To write in a formal manner, let us start by considering a matrix of input graph signals $X \in \mathbb{C}^{N \times C}$, where C denotes the number

of input channels, and $\Theta \in \mathbb{C}^{C\times d}$ a matrix of learnable filter parameters of dimension d. The propagation through a convolutional layer of the network can be written as

$$Y = \sigma(\tilde{D}_h^{-\frac{1}{2}}\tilde{H}_h\tilde{D}_h^{-\frac{1}{2}}X\Theta), \tag{24}$$

where, following [9], $\tilde{D}_h$ and $\tilde{H}_h$ (see Definition 2) are computed using $\tilde{A}_s = A_s + I_N$, and σ denotes the nonlinearity. We apply the nonlinearity in a custom manner in order to accomodate the complex nature of the argument; thus, ReLU is applied individually on the real and imaginary parts of the matrix.

The output $Y \in \mathbb{C}^{N\times d}$ of the convolutions is a matrix of complex node embeddings. Since the link prediction problem is focused on edges, we apply an unwind operation, similarly to [6] and [7]. We concatenate the real and imaginary parts of the learned node embeddings from Y corresponding to source and destination vertices of each edge in the dataset:

$$Y_{unw} = \mathit{Unwind}(Y), \tag{25}$$

with $Y_{unw} \in \mathbb{R}^{e\times 4d}$, where e is the number of edges of the training dataset. In this manner, we learn a pairwise measure which characterizes the edge between the two vertices. Then, a dropout layer and a dense (linear) layer are added. The linear layer transforms the output to a dimension s matching the sub-task format (1 for the weight prediction, 2 for the existence, and 3 for the three-class). The result becomes

$$Z = \mathit{Linear}(\mathit{Dropout}(Y_{unw})), \tag{26}$$

where $Z \in \mathbb{R}^{e\times s}$ is the output of the network.

When addressing the weight prediction sub-task, the set of labels has a size of $e \times 1$, therefore Z is used directly with mean-squared error loss function for training and assessment. For the existence and three-class flavors of the problem, we also apply logarithmic softmax on the network output Z, which, as mention earlier, has a size of $e \times 2$ and $e \times 3$, respectively:

$$Z_{class} = \mathit{LogSoftmax}(Z), \tag{27}$$

where column indices of the maximal values in each row of Z_{class} give the edge prediction label according to the tackled sub-task. Moreover, the logarithmic softmax further allows the direct usage of the negative logarithmic likelihood loss.

5. Numerical results

5.1. Learning applications results

To evaluate[1] our Haar Laplacian on learning tasks, we compare the results with several other alternatives from the literature: magnetic Laplacian (MagNet) [6] and sign-magnetic Laplacian (SigMaNet) [7]; in the respective works, they are shown to perform better than other methods for learning on directed graphs. Given the architectures described throughtout section 4.2, we train models with various parameter configurations:

- number of layers: 2, 4, 8;
- layer dimension d: 16, 32, 64;
- learning rate: 0.001, 0.005, 0.01, 0.05.

So, the resulting number of models is 36. We choose a probability of 0.5 for the dropout layer and a weight decay of $5 \cdot 10^{-4}$ is set to the Adam optimizer. The learning approach is supervised, thus we split the datasets for training (80%), validation (5%), and testing (15%), as suggested in [6]. Experiments are run with k-cross validation, where $k = 10$, and graph connectivity is preserved when building each training set by guaranteeing that the graph used for training in each fold contain a spanning tree. Negative sampling is also used; the number of inexistent edges is taken equal to that of actual edges. As for the training process, we set the maximum number of epochs to 1000, and enforce an early stopping condition of 200 iterations without the validation error decreasing.

Our assessement relies on three real datasets: Telegram [50]; Bitcoin Alpha and Bitcoin OTC [37]. The first dataset represents a pairwise-influence network of Telegram channels, used to analyze interactions concerning the spread of political ideologies. It includes 245 nodes, grouped into four classes, and contains 8912 edges with positive integer weights up to 7934. The Bitcoin datasets originate from cryptocurrency exchanges and are structured as trust networks. In both the Alpha and OTC datasets, users rate each other on a scale from -10 to $+10$ (excluding 0), where -10 denotes a scammer and $+10$ indicates a fully legitimate user. The Bitcoin Alpha dataset comprises 3783 nodes, with 22650 positive and 1536 negative edges. In contrast, the slightly larger Bitcoin OTC dataset has 5881 nodes, with 32029 positive and 3563 negative edges.

[1] `https://github.com/theodorbadea/Haar-Laplacian`

In order to offer an exhaustive perspective, we test on the Bitcoin datasets both with and without the negative weights. The removal of negative edges is done as in [7] and we further denote the resulting graphs with positive edge weights by Bitcoin Alpha+ and Bitcoin OTC+. As a remark, in the original formulation, MagNet does not support negative weights because the diagonal degree matrix in the original approach [6] is constructed by summating the colums of A_s, which may lead to negative entries. However, by using absolute values like in (2), the problem disappears; note that otherwise the relations (10)–(12) stand valid for negative weights.

Normalizing the data would, in many scenarios, support model convergence, numerical stability, bias prevention, and interpretability. For MagNet, additionally, a properly chosen normalization is highly beneficial, since it alleviates the phase scaling issues described in Sec. 2.1. We choose to run experiments with a normalization applied to the datasets. This way, we can fairly compare the results in a setup which does not impair MagNet. However, given the two different distributions encountered in the weight values, we cannot adopt the same normalization method for all the graphs. When it comes to the Telegram dataset, a linear scaling operation is not enough because the weights range between 1 and 7934, most of them noticeably closer to the inferior limit. As in [44], we scale via $w_{new} = e^{-\frac{1}{w_{old}}}$; the new weights are in the interval $(0, 1)$. For the Alpha and OTC graphs, a linear operation which scales the weights to the $[-1, 1]$ interval is adequate, as it preserves the original distribution and is not affected by, neither it affects the signs.

For the first two flavors of the task (existence and 3-class), we report the prediction accuracy, which is the ratio of correct predictions (in all predictions). The weight prediction is a regression problem, therefore we report the results in terms of root mean square error (RMSE) and, additionally, R^2 metric. Tables 1-4 show the values obtained for these metrics.

As already mentioned, our experiments emulate an optimal setup for the magnetic Laplacian, as weights are normalized and $q = 0.25$. This ensures a well-behaved phase matrix, with values between $-\pi$ and π. The accuracy values point towards MagNet as the best performing when it comes to existence and 3-class prediction tasks. These two tasks rely mostly on the directionality of the edges, without a strong dependency on the actual weight assigned to edges. MagNet clearly captures the directional information and obtains the best results on all the datasets for the 3-class problem and is evidently outperformed only on the Telegram by our HaarNet for the existence task. HaarNet wins also on the Bitcoin Alpha graph, but only

Table 1: Accuracy (%) for the existence problem

Model	Telegram	Alpha	OTC	Alpha+	OTC+
Haar	**87.04**±0.43	**87.42**±0.41	88.20±0.26	87.48±0.61	88.78±0.31
MagNet	86.56±0.60	87.41±0.36	**88.98**±0.32	**87.60**±0.30	**89.13**±0.22
SigMaNet	86.52±0.72	86.66±0.21	87.62±0.30	87.08±0.41	88.67±0.33

Table 2: Accuracy (%) for the 3-class problem

Model	Telegram	Alpha	OTC	Alpha+	OTC+
Haar	83.17±0.32	84.65±0.79	84.77±0.35	85.05±0.50	85.28±0.44
MagNet	**83.28**±0.43	**84.76**±0.62	**85.11**±0.54	**85.40**±0.77	**85.53**±0.63
SigMaNet	82.72±0.47	84.44±0.46	83.90±0.53	85.12±0.41	85.08±0.66

Table 3: Root Mean Square Error (RMSE) for the weight prediction problem

Model	Telegram	Alpha	OTC	Alpha+	OTC+
Haar	**0.2526** ± 0.0028	**0.2017** ± 0.0040	**0.2241** ± 0.0025	**0.1380** ± 0.0035	**0.1343** ± 0.0014
MagNet	0.2544 ± 0.0036	0.2034 ± 0.0041	0.2256 ± 0.0028	0.1380 ± 0.0034	0.1344 ± 0.0016
SigMaNet	0.2566 ± 0.0030	0.2060 ± 0.0036	0.2317 ± 0.0031	0.1387 ± 0.0033	0.1351 ± 0.0013

Table 4: Coefficient of determination (R^2 score) for the weight prediction problem

Model	Telegram	Alpha	OTC	Alpha+	OTC+
Haar	**0.5335** ± 0.0107	**0.2092** ± 0.0161	**0.2743** ± 0.0110	**0.3108** ± 0.0131	**0.3050** ± 0.0063
MagNet	0.5268 ± 0.0140	0.1957 ± 0.0169	0.2649 ± 0.0119	0.3101 ± 0.0119	0.3037 ± 0.0099
SigMaNet	0.5184 ± 0.119	0.1751 ± 0.0170	0.2245 ± 0.0137	0.3030 ± 0.0116	0.2959 ± 0.0084

by a narrow margin. On one hand, the limited range of weights and their variability in the Bitcoin trust networks seem to favor the ability of MagNet to learn the structure of the graph. On the other hand, not only a large proportion of nodes in the Bitcoin graphs have antiparallel edges (digons), most are also with identical weights. Thus, a significant number of entries in the magnetic Laplacian perfectly overlap with A_s. A similar reasoning can be applied to the 3-class task, the same structure supporting the network in learning the "directed edge exists" predictions.

However, when we look at the results obtained testing with the Telegram dataset, which contains an insignificant number of antiparallel edges and has a much larger and varied spectrum of weights, the existence experiments yield close results for SigMaNet and MagNet and noticeably better results for HaarNet, outperforming both of them. If for MagNet the phase matrix partly encodes some information about the weights, when it comes to SigMaNet, the phase matrix is entirely dedicated to encoding the directionality,

but not complete enough for handling digons. Although the sign-magnetic Laplacian seems more suitable to sign prediction tasks, nevertheless both our HaarNet and MagNet perform better in terms of existence problems.

The last series of experiments, addressing the weight prediction flavor of the problem, proves that our Haar Laplacian is indeed able to encode all graph information: structure, directionality, and weights. The task itself closely resembles the existence one, however the difficulty arises from replacing the binary prediction with edge weight values and so shifting the perspective towards the regression family of machine learning problems. As it can be seen in Tables 3 and 4, HaarNet outperforms both MagNet and SigMaNet. For Alpha+ and OTC+, although the RMSE obtained with MagNet is equal or very close, the corresponding R^2 is better for HaarNet. We motivate the superior results by fully preserving the information in the continous mapping $A \longrightarrow H_h$ and not focusing either on weight or direction, or making a compromise for better encoding one to the expense of the other.

5.2. Signal processing results

We give here some results in denoising on graphs, a standard graph signal processing problem. We build a random geometric graph as follows. The n nodes are given uniformly random coordinates in the square $[0,1] \times [0,1]$. Each node u is connected with the nodes v that are at distance at most r from it; the direction of the edge is random, with equal probability of (u,v) and (v,u). With probability p, there are edges in both directions: $(u,v),(v,u) \in \mathcal{E}$. The weights of the edges are uniformly random in the interval $[w_{\min}, w_{\max}]$. Figure 7 shows an example of such graph, with $n = 500$, $r = 2/\sqrt{n}$, and $p = 0.5$.

We consider a graph signal defined by $z(u) = 10 + 10x(u) + 5y(u)$, where $x(u)$ and $y(u)$ are the coordinates of node u; the choice of a plane was made to ensure that the signal is smooth; we have obtained similar results with other simple signals. To the signal, we add Gaussian noise with standard deviation σ; let $\tilde{z}$ be the noisy signal. The weights of the graph are generated in the interval defined by $w_{\min} = 0.8$, $w_{\max} = 1.2$. The setup of the experiment is similar to that from [28], but on a larger graph.

Denoising is performed with the GFT in the standard way. The first m coefficients of the "frequency" domain signal $\mathrm{GFT}(\tilde{z}) = U^H \tilde{z}$, where U is the matrix of eigenvectors, see (21), are preserved and the remaining coefficients are set to zero. Then, the inverse GFT is applied, thus obtaining the denoised signal $\hat{z}$. The SNR is

$$\mathrm{SNR} = -20 \log_{10} \frac{\|\hat{z} - z\|}{\|z\|}.$$

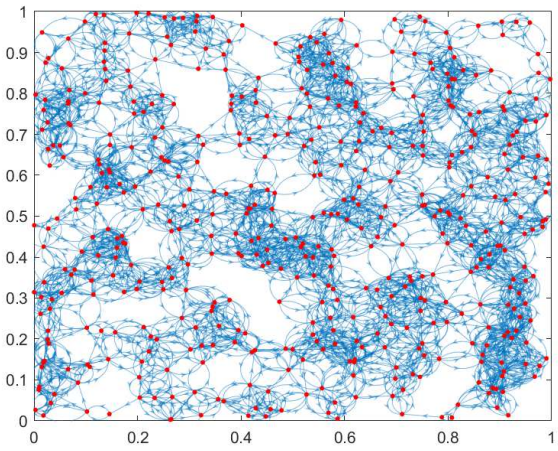

Figure 7: Directed graph based on geometric distance, with $n = 500$.

Figure 8 shows the denoising results for four GFTs: our Haar-Laplacian, the SVD based one [28], and the magnetic (with $q = 1/4$) [6] and sign magnetic [7] Laplacians. The SNRs are averaged over 10 different graphs and 100 runs for each graph (with different noise realizations). One can see that for a small number of preserved coefficients, the SVD is better. However, growing the number of coefficients is beneficial to the Haar-Laplacian, which is able to give the overall best result. The magnetic Laplacian gives rather poor results, but is able to improve performance with growing m. The sign-magnetic Laplacian, which is an artificial construction meant mostly for coding the properties of the graph rather than for offering a direct interpretation, is obviously useless for denoising (but useful for learning, as we will see later).

Figure 9 displays the results for $p = 1$, the other parameters being the same as above. This means that all nodes that are connected have edges with both orientations between them. Now the Haar-Laplacian is the best for all considered values of m. From both figures it is also visible that, as σ grows, the best SNR is obtained for smaller values of m.

6. Conclusions

In this paper, we have proposed a novel Laplacian matrix, called Haar-Laplacian, designed to enable spectral techniques for directed graphs. Our approach enjoys several key properties that distinguish it from existing

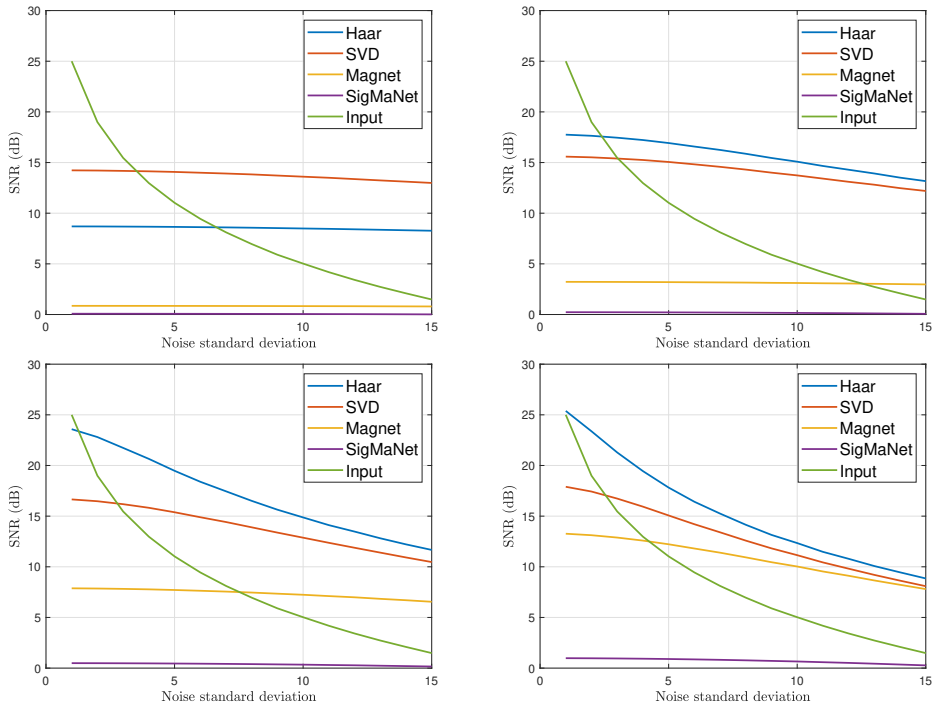

Figure 8: SNRs obtained by GFT truncation to $m = 10$ (up, left), $m = 25$ (up, right), $m = 50$ (down, left), and $m = 100$ (down, right), as a function of the noise standard deviation. Graph parameters: $n = 500$, $p = 0.5$, $w_{\min} = 0.8$, $w_{\max} = 1.2$.

methods, including continuity, scaling robustness, sensitivity, and the inherent consideration of directionality in the structure of the graph. The theoretical foundations of our method have been rigorously established, ensuring that the proposed Laplacian is in conformity with the spectral graph theory.

Then, we have addressed both learning and signal processing tasks and we have shown that it is a viable and promising tool for a wide range of graph-based applications.

References

[1] A. Sandryhaila, J. Moura, Discrete signal processing on graphs, IEEE Trans. Signal Proc. 61 (2013) 1644–1656.

[2] A. Marques, S. Segarra, G. Mateos, Signal processing on directed

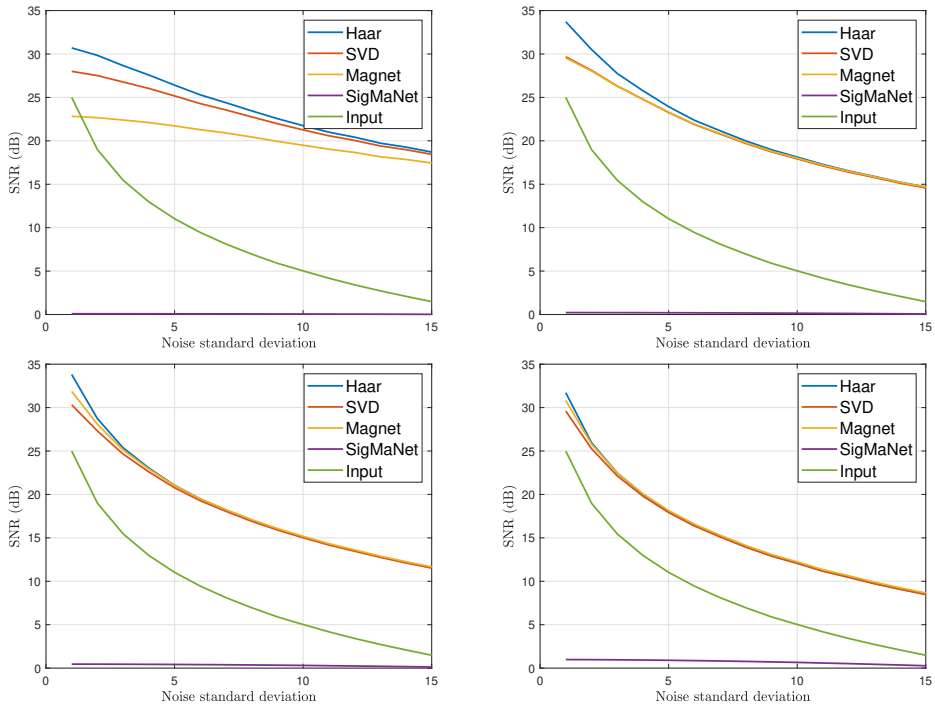

Figure 9: SNRs obtained by GFT truncation to $m = 10$ (up, left), $m = 25$ (up, right), $m = 50$ (down, left), and $m = 100$ (down, right), as a function of the noise standard deviation. Graph parameters: $n = 500$, $p = 1$, $w_{\min} = 0.8$, $w_{\max} = 1.2$.

graphs: The role of edge directionality when processing and learning from network data, IEEE Signal Proc. Mag. 37 (2020) 99–116.

[3] E. Isufi, F. Gama, D. Shuman, S. Segarra, Graph filters for signal processing and machine learning on graphs, IEEE Trans. Signal Proc. (2024).

[4] J. Kunegis, S. Schmidt, A. Lommatzsch, J. Lerner, E. De Luca, S. Albayrak, Spectral analysis of signed graphs for clustering, prediction and visualization, in: Proceedings of the 2010 SIAM Int. Conf. Data Mining, SIAM, 2010, pp. 559–570.

[5] M. Shubin, Discrete magnetic Laplacian, Communications in mathematical physics 164 (1994) 259–275.

[6] X. Zhang, Y. He, N. Brugnone, M. Perlmutter, M. Hirn, MagNet: A

neural network for directed graphs, Advances in neural information processing systems 34 (2021) 27003–27015.

[7] S. Fiorini, S. Coniglio, M. Ciavotta, E. Messina, SigMaNet: One Laplacian to rule them all, in: Proc. AAAI Conference on Artificial Intelligence, volume 37, 2023, pp. 7568–7576.

[8] T. Badea, B. Dumitrescu, Dualgcn: A convolutional network for anomaly detection in directed graphs based on symmetrized and skew-symmetrized laplacians, in: IEEE International Workshop on Machine Learning for Signal Processing (MLSP), IEEE, 2024, pp. 1–6.

[9] T. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, arXiv preprint arXiv:1609.02907 (2016).

[10] J. Bruna, W. Zaremba, A. Szlam, Y. LeCun, Spectral networks and locally connected networks on graphs, in: International Conference on Learning Representations (ICLR2014), CBLS, April 2014, 2014.

[11] M. Defferrard, X. Bresson, P. Vandergheynst, Convolutional neural networks on graphs with fast localized spectral filtering, in: Proceedings of the 30th International Conference on Neural Information Processing Systems, 2016, p. 3844–3852.

[12] D. Hammond, P. Vandergheynst, R. Gribonval, Wavelets on graphs via spectral graph theory, Applied and Computational Harmonic Analysis 30 (2011) 129–150.

[13] F. Chung, Laplacians and the Cheeger inequality for directed graphs, Annals of Combinatorics 9 (2005) 1–19.

[14] Y. He, M. Perlmutter, G. Reinert, M. Cucuringu, MSGNN: A spectral graph neural network based on a novel magnetic signed Laplacian, in: Learning on Graphs Conference, PMLR, 2022, pp. 40–1.

[15] J. Zhang, B. Hui, P. Harn, M. Sun, W. Ku, MGC: A complex-valued graph convolutional network for directed graphs, arXiv preprint arXiv:2110.07570 (2021).

[16] R. Singh, Y. Chen, Signed graph neural networks: A frequency perspective, arXiv preprint arXiv:2208.07323 (2022).

[17] T. Ko, Y. Choi, C. Kim, A spectral graph convolution for signed directed graphs via magnetic laplacian, Neural Networks 164 (2023) 562–574.

[18] J. Liu, X. Li, Hermitian-adjacency matrices and hermitian energies of mixed graphs, Linear Algebra and its Applications 466 (2015) 182–207.

[19] K. Guo, B. Mohar, Hermitian adjacency matrix of digraphs and mixed graphs, Journal of Graph Theory 85 (2017) 217–248.

[20] Z. Ke, H. Yu, J. Li, H. Zhang, DUPLEX: Dual GAT for Complex Embedding of Directed Graphs, arXiv preprint arXiv:2406.05391 (2024).

[21] S. Fiorini, S. Coniglio, M. Ciavotta, E. Messina, Graph Learning in 4D: A Quaternion-Valued Laplacian to Enhance Spectral GCNs, in: Proc. AAAI Conference on Artificial Intelligence, volume 38, 2024, pp. 12006–12015.

[22] Z. Tong, Y. Liang, C. Sun, X. Li, D. Rosenblum, A. Lim, Digraph inception convolutional networks, Advances in neural information processing systems 33 (2020) 17907–17918.

[23] A. Sandryhaila, J. Moura, Discrete signal processing on graphs: Frequency analysis, IEEE Trans. Signal Proc. 62 (2014) 3042–3054.

[24] Y. Li, Z. Zhang, Digraph Laplacian and the degree of asymmetry, Internet Mathematics 8 (2012) 381–401.

[25] H. Sevi, G. Rilling, P. Borgnat, Harmonic analysis on directed graphs and applications: From Fourier analysis to wavelets, Applied and Computational Harmonic Analysis 62 (2023) 390–440.

[26] R. Singh, A. Chakraborty, B. Manoj, Graph Fourier transform based on directed Laplacian, in: 2016 Int. Conf. Signal Processing and Communications (SPCOM), IEEE, 2016, pp. 1–5.

[27] S. Furutani, T. Shibahara, M. Akiyama, K. Hato, M. Aida, Graph signal processing for directed graphs based on the Hermitian Laplacian, in: European Conference on Machine Learning and Knowledge Discovery in Databases, Würzburg, Germany, September 16–20, 2019, Proceedings, Part I, Springer, 2020, pp. 447–463.

[28] Y. Chen, C. Cheng, Q. Sun, Graph Fourier transform based on singular value decomposition of the directed Laplacian, Sampling Theory, Signal Processing, and Data Analysis 21 (2023) 24.

[29] S. Kwak, L. Shimabukuro, A. Ortega, Frequency analysis and filter design for directed graphs with polar decomposition, in: IEEE Int.

Conf. Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2024, pp. 9661–9665.

[30] S. Sardellitti, S. Barbarossa, P. Di Lorenzo, On the graph Fourier transform for directed graphs, IEEE Journal of Selected Topics in Signal Processing 11 (2017) 796–811.

[31] R. Shafipour, A. Khodabakhsh, G. Mateos, E. Nikolova, A directed graph fourier transform with spread frequency components, IEEE Trans. Signal Proc. 67 (2018) 946–960.

[32] B. Girault, A. Ortega, S. Narayanan, Irregularity-aware graph Fourier transforms, IEEE Trans. Signal Proc. 66 (2018) 5746–5761.

[33] S. Gerschgorin, Uber die abgrenzung der eigenwerte einer matrix, Izvestija Akademii Nauk SSSR, Serija Matematika 7 (1931) 749–754.

[34] I. Ahmad, M. U. Akhtar, S. Noor, A. Shahnaz, Missing link prediction using common neighbor and centrality based parameterized algorithm, Scientific reports 10 (2020) 364.

[35] L. Yao, L. Wang, L. Pan, K. Yao, Link prediction based on common-neighbors for dynamic social network, Procedia Computer Science 83 (2016) 82–89.

[36] J. Zhao, L. Miao, J. Yang, H. Fang, Q. Zhang, M. Nie, P. Holme, T. Zhou, Prediction of links and weights in networks by reliable routes, Scientific reports 5 (2015) 12261.

[37] S. Kumar, F. Spezzano, V. Subrahmanian, C. Faloutsos, Edge weight prediction in weighted signed networks, in: IEEE 16th international conference on data mining (ICDM), IEEE, 2016, pp. 221–230.

[38] T. Qiu, M. Zhang, X. Liu, J. Liu, C. Chen, W. Zhao, A directed edge weight prediction model using decision tree ensembles in industrial internet of things, IEEE Transactions on Industrial Informatics 17 (2020) 2160–2168.

[39] M. Zhang, Y. Chen, Link prediction based on graph neural networks, in: Advances in Neural Information Processing Systems, volume 31, Curran Associates, Inc., 2018.

[40] B. Zhu, Y. Xia, X. Zhang, Weight prediction in complex networks based on neighbor set, Scientific reports 6 (2016) 38080.

[41] G. Chen, C. Xu, J. Wang, J. Feng, J. Feng, Graph regularization weighted nonnegative matrix factorization for link prediction in weighted complex network, Neurocomputing 369 (2019) 50–60.

[42] Z. Cao, Y. Zhang, J. Guan, S. Zhou, G. Chen, Link weight prediction using weight perturbation and latent factor, IEEE Transactions on Cybernetics 52 (2022) 1785–1797.

[43] C. Fu, M. Zhao, L. Fan, X. Chen, J. Chen, Z. Wu, Y. Xia, Q. Xuan, Link weight prediction using supervised learning methods and its application to yelp layered network, IEEE Transactions on Knowledge and Data Engineering 30 (2018) 1507–1518.

[44] Z. Liu, H. Li, C. Wang, NEW: A generic learning model for tie strength prediction in networks, Neurocomputing 406 (2020) 282–292.

[45] U. Zulaika, S.-C. R., A. A., D. López-de Ipiña, LWP-WL: Link weight prediction based on CNNs and the Weisfeiler–Lehman algorithm, Applied Soft Computing 120 (2022) 108657.

[46] J. Liang, C. Pu, Line graph neural networks for link weight prediction, arXiv preprint arXiv:2309.15728 (2023).

[47] Z. Liu, W. Zuo, D. Zhang, C. Zhou, Self-attention enhanced auto-encoder for link weight prediction with graph compression, IEEE Transactions on Network Science and Engineering 11 (2024) 89–99.

[48] M. Grassia, G. Mangioni, wsGAT: Weighted and Signed Graph Attention Networks for Link Prediction, in: Complex Networks & Their Applications X, Springer International Publishing, 2022, pp. 369–375.

[49] K. Fukushima, Visual feature extraction by a multilayered network of analog threshold elements, IEEE Transactions on Systems Science and Cybernetics 5 (1969) 322–333.

[50] A. Bovet, P. Grindrod, The activity of the far right on telegram v2.11, 2020.