MagNet: A Neural Network for Directed Graphs

Anonymous Author(s)
Affiliation
Address
email

Abstract

The prevalence of graph-based data has spurred the rapid development of graph neural networks (GNNs) and related machine learning algorithms. Yet, despite the many datasets naturally modeled as directed graphs, including citation, website, and traffic networks, the vast majority of this research focuses on undirected graphs. In this paper, we propose *MagNet*, a spectral GNN for directed graphs based on a complex Hermitian matrix known as the magnetic Laplacian. This matrix encodes undirected geometric structure in the magnitude of its entries and directional information in their phase. A "charge" parameter attunes spectral information to variation among directed cycles. We apply our network to a variety of directed graph node classification and link prediction tasks showing that MagNet performs well on all tasks and that its performance exceeds all other methods on a majority of such tasks. The underlying principles of MagNet are such that it can be adapted to other spectral GNN architectures.

1 Introduction

Endowing a collection of objects with a graph structure allows one to encode pairwise relationships among its elements. These relations often possess a natural notion of direction. For example, the WebKB dataset [32] contains a list of university websites with associated hyperlinks. In this context, one website might link to a second without a reciprocal link to the first. Such datasets are naturally modeled by *directed graphs*. In this paper, we introduce *MagNet*, a graph convolutional neural network for directed graphs based on the magnetic Laplacian.

Most graph neural networks fall into one of two families, *spectral networks* or *spatial networks*. Spatial methods define graph convolution as a localized averaging operation with iteratively learned weights. Spectral networks, on the other hand, define convolution on graphs via the eigendecompositon of the (normalized) graph Laplacian. The eigenvectors of the graph Laplacian assume the role of Fourier modes, and convolution is defined as entrywise multiplication in the Fourier basis. For a comprehensive review of both spatial and spectral networks, we refer the reader to [42] and [40].

Many spatial graph CNNs have natural extensions to directed graphs. However, it is common for these methods to preprocess the data by symmetrizing the adjacency matrix, effectively creating an undirected graph. For example, while [38] explicitly notes that their network is well-defined on directed graphs, their experiments treat all citation networks as undirected for improved performance.

Extending spectral methods to directed graphs is not straightforward since the adjacency matrix is asymmetric and, thus, there is no obvious way to define a symmetric, real-valued Laplacian with a full set of real eigenvalues that uniquely encodes any directed graph. We overcome this challenge by constructing a network based on the magnetic Laplacian $\mathbf{L}^{(q)}$ defined in Section 2. Unlike the directed graph Laplacians used in works such as [26, 30, 36, 37], the magnetic Laplacian is not a real-valued symmetric matrix. Instead, it is a *complex-valued Hermitian* matrix that encodes the fundamentally asymmetric nature of a directed graph via the complex phase of its entries.

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

Since $\mathbf{L}^{(q)}$ is Hermitian, the spectral theorem implies it has an orthonormal basis of complex eigenvectors corresponding to real eigenvalues. Moreover, Theorem 1, stated in Section 5 of the supplement, shows that $\mathbf{L}^{(q)}$ is positive semidefinite, similar to the traditional Laplacian. Setting $q = 0$ is equivalent to symmetrizing the adjacency matrix and no importance is given to directional informatio. When $q = .25$, on the other hand, we have that $\mathbf{L}^{(.25)}(u, v) = -\mathbf{L}^{(.25)}(v, u)$ whenever there is an edge from u to v but not from v to u. Different values of q highlight different graph motifs [16, 17, 19, 29], and therefore the optimal choice of q varies. Learning the appropriate value of q from data allows MagNet to adaptively incorporate directed information. We also note that $\mathbf{L}^{(q)}$ has been applied to graph signal processing [18], community detection [17], and clustering [10, 16, 15].

In Section 3, we show how the networks constructed in [6, 13, 22] can be adapted to directed graphs by incorporating complex Hermitian matrices, such as the magnetic Laplacian. When $q = 0$, we effectively recover the networks constructed in those previous works. Therefore, our work generalizes these networks in a way that is suitable for directed graphs. Our method is very general and is not tied to any particular choice of network architecture. Indeed, the main ideas of this work could be adapted to nearly any spectral network.

In Section 4, we summarize related work on directed graph neural networks as well as other papers studying the magnetic Laplacian and its applications in data science. In Section 5, we apply our network to node classification and link prediction tasks. We compare against several spectral and spatial methods as well as networks designed for directed graphs. We find that MagNet obtains the best or second-best performance on five out of six node-classification tasks and has the best performance on seven out of eight link-prediction tasks tested on real-world data, in addition to providing excellent node-classification performance on difficult synthetic data. We also provide a supplementary document with full implementation details, theoretical results concerning the magnetic Laplacian, extended examples, and further numerical details.

2 The magnetic Laplacian

Spectral graph theory has been remarkably successful in relating geometric characteristics of undirected graphs to properties of eigenvectors and eigenvalues of graph Laplacians and related matrices. For example, the tasks of optimal graph partitioning, sparsification, clustering, and embedding may be approximated by eigenvectors corresponding to small eigenvalues of various Laplacians (see, e.g., [9, 34, 2, 35, 11]). Similarly, the graph signal processing research community leverages the full set of eigenvectors to extend the Fourier transform to these structures [31]. Furthermore, numerous papers [6, 13, 22] have shown that this eigendecomposition can be used to define neural networks on graphs. In this section, we provide the background needed to extend these constructions to directed graphs via complex Hermitian matrices such as the magnetic Laplacian.

We let $G = (V, E)$ be a directed graph where V is a set of N vertices and $E \subseteq V \times V$ is a set of directed edges. If $(u, v) \in E$, then we say there is an edge from u to v. For the sake of simplicity, we will focus on the case where the graph is unweighted and has no self-loops, i.e., $(v, v) \notin E$, but our methods have natural extensions to graphs with self-loops and/or weighted edges. If both $(u, v) \in E$ and $(v, u) \in E$, then one may consider this pair of directed edges as a single undirected edge.

A directed graph can be described by an adjacency matrix $(\mathbf{A}(u, v))_{u,v \in V}$ where $\mathbf{A}(u, v) = 1$ if $(u, v) \in E$ and $\mathbf{A}(u, v) = 0$ otherwise. Unless G is undirected, $\mathbf{A}$ is not symmetric, and, indeed, this is the key technical challenge in extending spectral graph neural networks to directed graphs. In the undirected case, where the adjacency matrix $\mathbf{A}$ is symmetric, the (unnormalized) graph Laplacian can be defined by $\mathbf{L} = \mathbf{D} - \mathbf{A}$, where $\mathbf{D}$ is a diagonal degree matrix. It is well-known that $\mathbf{L}$ is a symmetric, positive-semidefinite matrix and therefore has an orthonormal basis of eigenvectors associated with non-negative eigenvalues. However, when $\mathbf{A}$ is asymmetric, direct attempts to define the Laplacian this way typically yield complex eigenvalues. This impedes the straightforward extension of classical methods of spectral graph theory and graph signal processing to directed graphs.

A key point of this paper is to represent the directed graph through a complex Hermitian matrix $\boldsymbol{\mathcal{L}}$ such that: (1) the magnitude of $\boldsymbol{\mathcal{L}}(u, v)$ indicates the presence of an edge, but not its direction; and (2) the phase of $\boldsymbol{\mathcal{L}}(u, v)$ indicates the direction of the edge, or if the edge is undirected. Such matrices have been explored in the directed graph literature (see Section 4), but not in the context of graph neural networks. They have several advantages over their real-valued matrix counterparts. In particular, a single symmetric real-valued matrix will not uniquely represent a directed graph. Instead, one must use several matrices, as in [37], but this increases the complexity of the resulting network. Alternatively, one can work with an asymmetric, real-valued matrix, such as the adjacency matrix or the random walk matrix. However, the spatial graph filters that result from such matrices are typically limited by the fact that they can only aggregate information from the vertices that can be reached in one hop from a central vertex, but ignore the equally important subset of vertices that can reach the central vertex in one hop. Complex Hermitian matrices, however, lead to filters that aggregate information from both sets of vertices. Finally, one could use a real-valued skew-symmetric matrix but such matrices do not generalize well to graphs with both directed and undirected edges.

The optimal choice of complex Hermitian matrix is an open question. Here, we utilize a parameterized family of magnetic Laplacians, which have proven to be useful in other data-driven contexts [17, 10, 16, 15]. We first define the symmetrized adjacency matrix and corresponding degree matrix by,

$$\mathbf{A}_s(u,v) \coloneqq \frac{1}{2}(\mathbf{A}(u,v)+\mathbf{A}(v,u)), \;\; 1 \le u,v \le N, \quad \mathbf{D}_s(u,u) \coloneqq \sum_{v \in V} \mathbf{A}_s(u,v), \;\; 1 \le u \le N\,,$$

with $\mathbf{D}_s(u,v) = 0$ for $u \neq v$. We capture directional information via a phase matrix,[1] $\mathbf{\Theta}^{(q)}$,

$$\mathbf{\Theta}^{(q)}(u,v) \coloneqq 2\pi q(\mathbf{A}(u,v) - \mathbf{A}(v,u))\,, \quad q \ge 0\,,$$

where $\exp(i\mathbf{\Theta}^{(q)})$ is defined component-wise by $\exp(i\mathbf{\Theta}^{(q)})(u,v) \coloneqq \exp(i\mathbf{\Theta}^{(q)}(u,v))$. Letting $\odot$ denotes componentwise multiplication, we define the complex Hermitian adjacency matrix $\mathbf{H}^{(q)}$ by

$$\mathbf{H}^{(q)} \coloneqq \mathbf{A}_s \odot \exp(i\mathbf{\Theta}^{(q)})\,.$$

Since $\mathbf{\Theta}^{(q)}$ is skew-symmetric, $\mathbf{H}^{(q)}$ is Hermitian. When $q = 0$, we have $\mathbf{\Theta}^{(0)} = \mathbf{0}$ and so $\mathbf{H}^{(0)} = \mathbf{A}_s$. This effectively corresponds to treating the graph as undirected. For $q \neq 0$, the phase of $\mathbf{H}^{(q)}$ encodes edge direction. For example, if there is an edge from u to v but not from v to u we have

$$\mathbf{H}^{(.25)}(u,v) = \frac{i}{2} = -\mathbf{H}^{(.25)}(v,u)\,.$$

Thus, in this setting, an edge from u to v is treated as the opposite of an edge from v to u. On the other hand, if $(u,v),(v,u) \in E$ (which can be interpreted as a single undirected edge), then $\mathbf{H}^{(q)}(u,v) = \mathbf{H}^{(q)}(v,u) = 1$, and we see the phase, $\mathbf{\Theta}^{(q)}(u,v) = 0$, encodes the lack of direction in the edge. For the rest of this paper, we will assume that q lies in between these two extreme values, i.e., $0 \le q \le .25$. We define the normalized and unnormalized and magnetic Laplacians by

$$\mathbf{L}_U^{(q)} \coloneqq \mathbf{D}_s - \mathbf{H}^{(q)} = \mathbf{D}_s - \mathbf{A}_s \odot \exp(i\mathbf{\Theta}^{(q)}), \quad \mathbf{L}_N^{(q)} \coloneqq \mathbf{I} - \left(\mathbf{D}_s^{-1/2}\mathbf{A}_s\mathbf{D}_s^{-1/2}\right) \odot \exp(i\mathbf{\Theta}^{(q)})\,. \quad (1)$$

Note that when G is undirected, $\mathbf{L}_U^{(q)}$ and $\mathbf{L}_N^{(q)}$ reduce to the standard undirected Laplacians.

$\mathbf{L}_U^{(q)}$ and $\mathbf{L}_N^{(q)}$ are Hermitian. Theorem 1 (see Section 5 of the supplement) shows they are positive-semidefinite and thus are diagonalized by an orthonormal basis of complex eigenvectors $\mathbf{u}_1, \ldots, \mathbf{u}_N$ associated to real, nonnegative eigenvalues $\lambda_1, \ldots, \lambda_N$. Similar to the traditional normalized Laplacian, Theorem 2 (see Section 5 of the supplement) shows that the eigenvalues of $\mathbf{L}_N^{q}$ lie in $[0,2]$, and we may factor $\mathbf{L}_N^{(q)} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\dagger$, where $\mathbf{U}$ is the $N \times N$ matrix whose k-th column is $\mathbf{u}_k$, $\mathbf{\Lambda}$ is the diagonal matrix with $\mathbf{\Lambda}(k,k) = \lambda_k$, and $\mathbf{U}^\dagger$ is the conjugate transpose of $\mathbf{U}$ (a similar formula holds for $\mathbf{L}_U^{(q)}$). The magnetic Laplacian encodes in its eigenvectors and eigenvalues. In the directed star graph, for example, directional information is contained in the eigenvectors only, whereas the eigenvalues are invariant directionality. On the other hand, for the directed cycle graph the magnetic Laplacian encodes the directed nature of the graph solely in its spectrum (see Section 6 of the supplement). In general, both the eigenvectors and eigenvalues may contain important information. In Section 3, we will introduce MagNet, a network designed to leverage this spectral information.

[1] Our definition of $\mathbf{\Theta}^{(q)}$ coincides with that used in [18]. However, another definition (differing by a minus sign) also appears in the literature. These resulting magnetic Laplacians have the same eigenvalues and the corresponding eigenvectors are complex conjugates of one another. Therefore, this difference does not affect the performance of our network since our final layer separates the real and imaginary parts before multiplying by a trainable weight matrix (see Section 3 for details on the network structure).

3 MagNet

Most graph neural network architectures can be described as being either *spectral* or *spatial*. Spatial networks such as [38, 20, 1, 14] typically extend convolution to graphs by performing a weighted average of features over neighborhoods $\mathcal{N}(u) = \{v : (u, v) \in E\}$. These neighborhoods are well-defined even when E is not symmetric, so spatial methods typically have natural extensions to directed graphs. However, such simplistic extensions may miss important information in the directed graph. For example, filters defined using $\mathcal{N}(u)$ are not capable of assimilating the equally important information contained in $\{v : (v, u) \in E\}$. Alternatively, these methods may also use the symmetrized adjacency matrix, but they cannot learn to balance directed and undirected approaches.

In this section, we show how to extend spectral methods to directed graphs using the magnetic Laplacian introduced in Section 2. To highlight the flexibility of our approach, we show how three spectral graph neural network architectures can be adapted to incorporate the magnetic Laplacian. Our approach is very general, and so for most of this section, we will perform our analysis for a general complex Hermitian, positive semidefinite matrix. However, we view the magnetic Laplacian as our primary object of interest (and use it in all of our experiments) because of the large body of literature studying its spectral properties and applying it to data science (see Section 4).

3.1 Spectral convolution via the magnetic Laplacian

In this section, we let $\boldsymbol{\mathcal{L}}$ denote a Hermitian, positive semidefinite matrix, such as the normalized or unnormalized magnetic Laplacian introduced in Section 2, on a directed graph $G = (V, E)$, $|V| = N$. We let $\mathbf{u}_1 \ldots, \mathbf{u}_N$ be an orthonormal basis of eigenvectors for $\boldsymbol{\mathcal{L}}$ and let $\mathbf{U}$ be the $N \times N$ matrix whose k-th column is $\mathbf{u}_k$. We define the directed graph Fourier transform for a signal $\mathbf{x} : V \to \mathbb{C}$ by $\widehat{\mathbf{x}} = \mathbf{U}^\dagger\mathbf{x}$, so that $\widehat{\mathbf{x}}(k) = \langle \mathbf{x}, \mathbf{u}_k \rangle$. We regard the eigenvectors $\mathbf{u}_1, \ldots, \mathbf{u}_N$ as the generalizations of discrete Fourier modes to directed graphs. Since $\mathbf{U}$ is unitary, we have the Fourier inversion formula

$$\mathbf{x} = \mathbf{U}\widehat{\mathbf{x}} = \sum_{k=1}^{N} \widehat{\mathbf{x}}(k)\mathbf{u}_k \,. \tag{2}$$

In Euclidean space, convolution corresponds to pointwise multiplication in the Fourier basis. Thus, we define the convolution of $\mathbf{x}$ with a filter $\mathbf{y}$ in the Fourier domain by $\widehat{\mathbf{y} * \mathbf{x}}(k) = \widehat{\mathbf{y}}(k)\widehat{\mathbf{x}}(k)$. By (2), this implies $\mathbf{y} * \mathbf{x} = \mathbf{U}\mathrm{Diag}(\widehat{\mathbf{y}})\widehat{\mathbf{x}} = (\mathbf{U}\mathrm{Diag}(\widehat{\mathbf{y}})\mathbf{U}^\dagger)\mathbf{x}$, and so we say $\mathbf{Y}$ is a convolution matrix if

$$\mathbf{Y} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{U}^\dagger \,, \tag{3}$$

for a diagonal matrix $\boldsymbol{\Sigma}$. This is the natural generalization of the class of convolutions used in [6].

Next, following [13] (see also [21]), we show that a spectral network can be implemented in the spatial domain via polynomials of $\boldsymbol{\mathcal{L}}$ by having $\boldsymbol{\Sigma}$ be a polynomial of $\boldsymbol{\Lambda}$ in (3). This reduces the number of trainable parameters to prevent overfitting, avoids explicit diagonalization of the matrix $\boldsymbol{\mathcal{L}}$, (which is expensive for large graphs), and improves stability to perturbations [24]. As in [13], we define a normalized eigenvalue matrix, with entries in $[-1, 1]$, by $\widetilde{\boldsymbol{\Lambda}} = \frac{2}{\lambda_{\max}}\boldsymbol{\Lambda} - \mathbf{I}$ and assume

$$\boldsymbol{\Sigma} = \sum_{k=0}^{K} \theta_k T_k(\widetilde{\boldsymbol{\Lambda}}) \,,$$

for some real-valued $\theta_1, \ldots, \theta_k$, where for $k \geq 0$, T_k is the Chebyshev polynomial defined by $T_0(x) = 1, T_1(x) = x$, and $T_k(x) = 2xT_{k-1}(x) + T_{k-2}(x)$ for $k \geq 2$. One can use the fact that $(\mathbf{U}\widetilde{\boldsymbol{\Lambda}}\mathbf{U}^\dagger)^k = \mathbf{U}\widetilde{\boldsymbol{\Lambda}}^k\mathbf{U}^\dagger$ to see

$$\mathbf{Y}\mathbf{x} = \mathbf{U}\sum_{k=0}^{K} \theta_k T_k(\widetilde{\boldsymbol{\Lambda}})\mathbf{U}^\dagger\mathbf{x} = \sum_{k=0}^{K} \theta_k T_k(\widetilde{\boldsymbol{\mathcal{L}}})\mathbf{x} \,, \tag{4}$$

where, analogous to $\widetilde{\boldsymbol{\Lambda}}$, we define $\widetilde{\boldsymbol{\mathcal{L}}} := \frac{2}{\lambda_{\max}}\boldsymbol{\mathcal{L}} - \mathbf{I}$. It is important to note that, due to the complex Hermitian structure of $\widetilde{\boldsymbol{\mathcal{L}}}$, the value $\mathbf{Y}\mathbf{x}(u)$ aggregates information both from the values of $\mathbf{x}$ on $\mathcal{N}_k(u)$, the k-hop neighborhood of u, and the values of $\mathbf{x}$ on $\{v : \mathrm{dist}(v, u) \leq k\}$, which consists of those of vertices that can reach u in k-hops. While in an undirected graph these two sets of

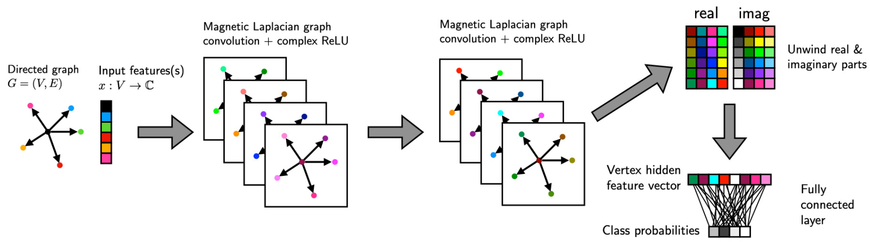


Figure 1: MagNet ($L = 2$) applied to node classification. After two complex convolutional layers, we unwind the real and imaginary parts of our feature matrix and apply a fully connected layer.

vertices are the same, that is not the case for general directed graphs. Furthermore, due to the difference in phase between an edge (u, v) and an edge (v, u), the filter matrix $\mathbf{Y}$ is also capable of aggregating information from these two sets in different ways. This capability is in contrast to any single, symmetric, real-valued matrix, as well as any matrix that encodes just $\mathcal{N}(u)$.

To obtain a network similar to [22], we set $K = 1$, assume that $\boldsymbol{\mathcal{L}} = \mathbf{L}_N^{(q)}$, using $\lambda_{\max} \leq 2$ (see Theorem 2 in Section 5 of the supplement) make the approximation $\lambda_{\max} \approx 2$, and set $\theta_1 = -\theta_0$. With this, we obtain

$$\mathbf{Y}\mathbf{x} = \theta_0(\mathbf{I} + (\mathbf{D}_s^{-1/2}\mathbf{A}_s\mathbf{D}_s^{-1/2}) \odot \exp(i\mathbf{\Theta}^{(q)}))\mathbf{x}\,.$$

As in [22], we substitute $(\mathbf{I} + (\mathbf{D}_s^{-1/2}\mathbf{A}_s\mathbf{D}_s^{-1/2}) \odot \exp(i\mathbf{\Theta}^{(q)}) \to \widetilde{\mathbf{D}}_s^{-1/2}\widetilde{\mathbf{A}}_s\widetilde{\mathbf{D}}_s^{-1/2}\exp(i\mathbf{\Theta}^{(q)})$. This renormalization helps avoid instabilities arising from vanishing/exploding gradients and yields

$$\mathbf{Y}\mathbf{x} = \theta_0\widetilde{\mathbf{D}}_s^{-1/2}\widetilde{\mathbf{A}}_s\widetilde{\mathbf{D}}_s^{-1/2} \odot \exp(i\mathbf{\Theta}^{(q)})\,, \tag{5}$$

where $\widetilde{\mathbf{A}}_s = \mathbf{A}_s + \mathbf{I}$ and $\widetilde{\mathbf{D}}_s(i,i) = \sum_j \widetilde{\mathbf{A}}_s(i,j)$.

3.2 The MagNet architecture

We now define our network. Let L be the number of convolution layers in our network, and let $\mathbf{X}^{(0)}$ be an $N \times F_0$ input feature matrix with columns $\mathbf{x}_1^{(0)}, \ldots \mathbf{x}_{F_0}^{(0)}$. Since our filters are complex, we use a complex version of ReLU defined by $\sigma(z) = z$, if $-\pi/2 \leq \arg(z) < \pi/2$, and $\sigma(z) = 0$ otherwise (where $\arg(z)$ is the complex argument of $z \in \mathbb{C}$). We let F_ℓ be the number of channels in layer ℓ, and for $1 \leq \ell \leq L$, $1 \leq i \leq F_{\ell-1}$, and $1 \leq j \leq F_\ell$, we let $\mathbf{Y}_{ij}^{(\ell)}$ be a convolution matrix defined in the sense of either (3), (4), or (5). To obtain the layer ℓ channels from the layer $\ell - 1$ channels, we define the matrix $\mathbf{X}^{(\ell)}$ with columns $\mathbf{x}_1^{(\ell)}, \ldots \mathbf{x}_{F_\ell}^{(\ell)}$ as:

$$\mathbf{x}_j^{(\ell)} = \sigma\left(\sum_{i=1}^{F_{\ell-1}} \mathbf{Y}_{ij}^{(\ell)}\mathbf{x}_i^{(\ell-1)}\right). \tag{6}$$

In matrix form we write $\mathbf{X}^{(\ell)} = \mathbf{Z}^{(\ell)}\left(\mathbf{X}^{(\ell-1)}\right)$, where $\mathbf{Z}^{(\ell)}$ is a hidden layer of the form (6).

After the convolutional layers, we unwind the complex $N \times F_L$ matrix $\mathbf{X}^{(L)}$ into a real-valued $N \times 2F_L$ matrix, apply a linear layer, consisting of right-multiplication by a $2F_L \times n_c$ weight matrix $\mathbf{W}^{(L+1)}$ (where n_c is the number of classes) and apply softmax. In our experiments, we set $L = 2$ or 3. When $L = 2$, our network applied to node classification, as illustrated in Figure 1, is given by

$$\mathrm{softmax}(\mathrm{unwind}(\mathbf{Z}^{(2)}(\mathbf{Z}^{(1)}(\mathbf{X}^{(0)})))\mathbf{W}^{(3)})\,.$$

For link-prediction, we apply the same method through the unwind layer, but then subtract the rows corresponding to pairs of nodes to obtain the edge features.

4 Related work

In Section 4.1, we describe other graph neural networks designed specifically for directed graphs. Notably, none of these methods encode directionality with complex numbers, instead opting for real-valued, symmetric matrices. In Section 4.2, we review other work studying the magnetic Laplacian which has been studied for several decades and lately has garnered interest in the network science and graph signal processing communities. However, to the best of our knowledge, this is the first work to use it to construct a graph neural network. We also note there are numerous approaches to graph signal processing on directed graphs. Many of these rely on a natural analog of Fourier modes. These Fourier modes are typically defined through either a factorization of a graph shift operator or by solving an optimization problem. For further review, we refer the reader to [27].

4.1 Neural networks for directed graphs

In [26], the authors construct a directed Laplacian, via identities involving the random walk matrix and its stationary distribution $\mathbf{\Pi}$. When G is undirected, one can use the fact that $\mathbf{\Pi}$ is proportional to the degree vector to verify this directed Laplacian reduces to the standard normalized graph Laplacian. However, this method requires G to be strongly connected, unlike MagNet. The authors of [37] use a first-order proximity matrix $\mathbf{A}_F$ (equivalent to $\mathbf{A}_s$ here), as well as two second-order proximity matrices $\mathbf{A}_{S_{\text{in}}}$ and $\mathbf{A}_{S_{\text{out}}}$. $\mathbf{A}_{S_{\text{in}}}$ is defined by $\mathbf{A}_{S_{\text{in}}}(u,v) \neq 0$ if there exists a w such that $(w,u),(w,v) \in E$, and $\mathbf{A}_{S_{\text{out}}}$ is defined analogously. These three matrices collectively describe and distinguish the neighborhood of each vertex and those vertices that can reach a vertex in a single hop. The authors construct three different Laplacians and use a fusion operator to share information across channels. Similarly, inspired by [3], in [30], the authors consider several different symmetric Laplacian matrices corresponding to a number of different graph motifs.

The method of [36] builds upon the ideas of both [26] and [37] and considers a directed Laplacian similar to the one used in [26], but with a PageRank matrix in place of the random-walk matrix. This allows for applications to graphs which are not strongly connected. Similar to [37], they use higher-order receptive fields (analogous to the second-order adjacency matrices discussed above) and an inception module to share information between receptive fields of different orders. We also note [23], which uses an approach based on PageRank in the spatial domain.

4.2 Related work on the magnetic Laplacian and Hermitian adjacency matrices

The magnetic Laplacian has been studied since at least [25]. The name originates from its interpretation as a quantum mechanical Hamiltonian of a particle under magnetic flux. Early works focused on d-regular graphs, where the eigenvectors of the magnetic Laplacian are equivalent to those of the Hermitian adjacency matrix. [19], for example, show that using a complex-valued Hermitian adjacency matrix rather than the symmetrized adjacency matrix reduces the number of small, non-isomorphic cospectral graphs. Topics of current research into Hermitian adjacency matrices include clustering tasks [12] and the role of the parameter q [29].

The magnetic Laplacian is also the subject of ongoing research in graph signal processing [18], community detection [17], and clustering [10, 16, 15]. For example, [16] uses the phase of the eigenvectors to construct eigenmap embeddings analogous to [2]. The role of q is highlighted in the works of [16, 17, 19, 29], which show how particular choices of q may highlight various graph motifs. In our context, this indicates that q should be carefully tuned via cross-validation. Lastly, we note that numerous other directed graph Laplacians have been studied and applied to data science [7, 8, 39]. However, as alluded to in Section 2, these methods typically do not use complex Hermitian matrices.

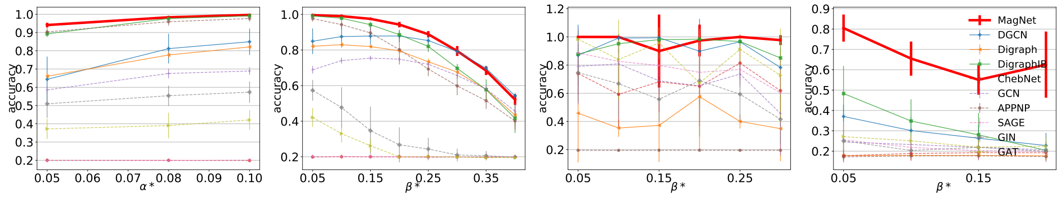
5 Numerical experiments

We test the performance of MagNet for node classification and link prediction on a variety of benchmark datasets as well as a directed stochastic block model.

5.1 Datasets

5.1.1 Directed Stochastic Block Model

We construct a directed stochastic block (DSBM) model as follows. First we divide N vertices into n_c clusters $C_1, \ldots, C_{n_c}$. We, define $\{\alpha_{i,j}\}_{1 \leq i,j \leq n_c}$ to be a collection of probabilities, $0 < \alpha_{i,j} \leq 1$ with $\alpha_{i,j} = \alpha_{j,i}$, and for an unordered pair $u \neq v$ create an undirected edge between u and v with probability $\alpha_{i,j}$ if $u \in C_i, v \in C_j$. To turn this undirected graph into a directed graph, we next define $\{\beta_{i,j}\}_{1 \leq i,j \leq n_c}$ to be a collection of probabilities such that $0 \leq \beta_{i,j} \leq 1$ and $\beta_{i,j} + \beta_{j,i} = 1$. For

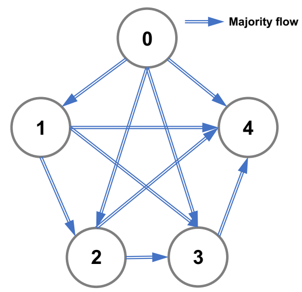


(a) Ordered DSBM with varying edge density. (b) Ordered DSBM with varying net flow. (c) Cyclic DSBM with varying net flow. (d) Noisy Cyclic DSBM with varying net flow.

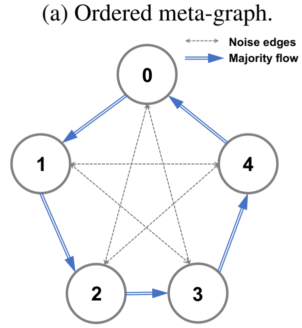
Figure 2: Node classification accuracy. Error bars are one standard error. MagNet is bold red.

each undirected edge $\{u, v\}$, we assign that edge a direction by the rule that the edge points from u to v with probability $\beta_{i,j}$ if $u \in C_i$ and $v \in C_j$, and points from v to u otherwise. We note that if $\alpha_{i,j}$ is constant, then the only way to determine the clusters will be from the directional information.

In Figure 2, we plot the performance of MagNet and other methods on variations of the DSBM. In each of these, we set $n_c = 5$ and the goal is to classify the vertices by cluster. We set $N = 2500$, except in Figure 2a where $N = 500$. In Figure 2a, we plot the performance of our model on the DSBM with $\alpha_{i,j} := \alpha^* = .1, .08$, and $.05$ for $i \neq j$, which varies the density of inter-cluster edges, and set $\alpha_{i,i} = .1$. Here we set $\beta_{i,i} = .5$ and $\beta_{i,j} = .05$ for $i > j$. This corresponds to the ordered meta-graph in Figure 3a. Figure 2b also uses the ordered meta-graph, but here we fix $\alpha_{i,j} = .1$ for all i, j, and set $\beta_{i,j} = \beta^*$, for $i > j$, and allow β^* to vary from .05 to .4, which varies the net flow from one cluster to another. The results in Figure 2c utilize a cyclic meta-graph structure as in Figure 3b (without the gray noise edges). Specifically, we set $\alpha_{i,j} = .1$ if $i = j$ or $i = j \pm 1 \mod 5$ and $\alpha_{i,j} = 0$ otherwise. We define $\beta_{i,j} = \beta^*$, $\beta_{j,i} = 1 - \beta^*$ when $j = (i - 1) \mod 5$, and $\beta_{i,j} = 0$ otherwise. In Figure 2d we add noise to the cyclic structure of our meta-graph by setting $\alpha_{i,j} = .1$ for all i, j and $\beta_{i,j} = .5$ for all (i, j) connected by a gray edge in Figure 3b (keeping $\beta_{i,j}$ the same as in Figure 2c for the blue edges).



(a) Ordered meta-graph.



(b) Cyclic meta-graph.

Figure 3: Meta-graphs for the synthetic data sets.

5.1.2 Real datasets

Texas, *Wisconsin*, and *Cornell* are WebKB datasets modeling links between websites at different universities [32]. We use these datasets for both link prediction and node classification with nodes labeled as student, project, course, staff, and faculty in the latter case. *Telegram* [5] is a pairwise influence network between 245 Telegram channels with 8, 912 links. To the best of our knowledge, this dataset has not previously been studied in the graph neural network literature. Labels are generated from the method discussed in [5], with a total of four classes. The datasets *Chameleon* and *Squirrel* [33] represent links between Wikipedia pages related to chameleons and squirrels. We use these datasets for link prediction. Likewise, *WikiCS* [28] is a collection of Computer Science articles, which we also use for link prediction. *Cora-ML* and *CiteSeer* are popular citation networks with node labels corresponding to scientific subareas. We use the versions of these datasets provided in [4]. Further details are given in the supplementary material.

5.2 Training and implementation details

Node classification is performed in a semi-supervised setting (i.e., access to the test data, but not the test labels, during training). For the datasets *Cornell*, *Texas*, *Wisconsin*, and *Telegram* we use a 60%/20%/20% training/validation/test split, which might be viewed as more akin to supervised learning, because of the small graph size. For *Cora-ML* and *CiteSeer*, we use the same split as [36]. For all of these datasets we use 10 random data splits. For the DSBM datasets, we generated 5 graphs randomly for each type and for each set of parameters, each with 10 different random node splits.

Table 1: Node classification accuracy. The best results are in **bold** and the second best are underlined.

Type	Method	Cornell	Texas	Wisconsin	Cora-ML	CiteSeer	Telegram
Spectral	ChebNet	79.8±5.0	79.2±7.5	81.6±6.3	80.0±1.8	66.7±1.6	70.2 ±6.8
	GCN	59.0±6.4	58.7±3.8	55.9±5.4	82.0±1.1	66.0±1.5	73.4 ±5.8
Spatial	APPNP	58.7±4.0	57.0±4.8	51.8±7.4	**82.6±1.4**	66.9±1.8	67.3 ±3.0
	SAGE	80.0±6.1	**84.3±5.5**	83.1±4.8	82.3±1.2	66.0±1.5	56.6 ±6.0
	GIN	57.9±5.7	65.2±6.5	58.2±5.1	78.1±2.0	63.3±2.5	74.4 ±8.1
	GAT	57.6±4.9	61.1±5.0	54.1±4.2	81.9±1.0	67.3±1.3	72.6 ±7.5
Directed	DGCN	67.3±4.3	71.7±7.4	65.5±4.7	81.3±1.4	66.3±2.0	**90.4 ±5.6**
	Digraph	66.8±6.2	64.9±8.1	59.6±3.8	79.4±1.8	62.6±2.2	82.0 ±3.1
	DiGraphIB	64.4±9.0	64.9±13.7	64.1±7.0	79.3± 1.2	61.1±1.7	64.1±7.0
This paper	MagNet	**84.3±7.0**	83.3±6.1	**85.7±3.2**	79.8±2.5	**67.5±1.8**	87.6 ±2.9
	Best q	0.25	0.15	0.05	0.0	0.0	0.15

We use 20% of the nodes for validation and we vary the proportion of training samples based on the classification difficulty, using 2%, 10%, and 60% of nodes per class for the ordered, cyclic, and noisy cyclic DSBM graphs, respectively, during training, and the rest for testing. Hyperpameters were selected using one of the five generated graphs, and then applied to the other four generated graphs.

There are two types of link prediction tasks conducted for performance evaluation. The first type is to predict the edge direction of pairs of vertices u, v for which either $(u, v) \in E$ or $(v, u) \in E$. The second type is existence prediction. The model is asked to predict if $(u, v) \in E$ by considering ordered pairs of vertices (u, v). For the both types of link prediction, we removed 15% of edges for testing, 5% for validation, and use the rest of the edges for training. The connectivity was maintained during splitting. 10 splits were generated randomly for each graph and the input features are in-degree and out-degree of nodes. Full details are provided in the supplementary material.

In all experiments, we used the normalized magnetic Laplacian and implement MagNet with convolution defined as in (4), meaning that our network may be viewed as the magnetic Laplacian generalization of ChebNet. We compare with multiple baselines in three categories: (i) spectral methods: ChebNet [13], GCN [22]; (ii) spatial methods: APPNP [23], SAGE [20], GIN [41], GAT [38]; and (iii) methods designed for directed graphs: DGCN [37], and two variants of [36], a basic version (DiGraph) and a version with higher order inception blocks (DiGraphIB). All baselines in the experiments have two graph convolutional layers, except for the node classification on the DSBM using the cyclic meta-graphs (Figures 2c, 2d, and 3b) for which we also tested three layers during the grid search. For ChebNet, we use the symmetrized adjacency matrix. For the spatial networks we apply both the symmetrized and asymmmetric adjacency matrix for node classification. The results reported are the better of the two results. Full details are provided in the supplemental material.

5.3 Results

We see that MagNet performs well across all tasks. As indicated in Table 1, our cross-validation procedure selects $q = 0$ for node classification on the citation networks *Cora-ML* and *CiteSeer*. This means we achieved the best performance when regarding directional information as noise, suggesting symmetrization-based methods are appropriate in the context of node classification on citation networks. This matches our intuition. For example, in Cora-ML, the task is to classify research papers by scientific subarea. If the topic of a given paper is "machine learning," then it is likely to both cite and be cited by other machine learning papers. For all other datasets, we find the optimal value of q is nonzero, indicating that directional information is important. Our network exhibits the best performance on three out of six of these datasets and is a close second on *Texas* and *Telegram*. We also achieve an at least four percentage point improvement over both ChebNet and GCN on the four data sets for which $q > 0$. These networks are similar to ours but with the classical graph Laplacian. This isolates the effects of the magnetic Laplacian and shows that it is a valuable tool for encoding directional information. MagNet also compares favorably to non-spectral methods on the WebKB networks (*Cornell*, *Texas*, *Wisconsin*). Indeed, MagNet obtains a $\sim 4\%$ improvement

Table 2: Link prediction accuracy. The best results are in **bold** and the second best are underlined.

	Direction prediction: Cornell	Direction prediction: Wisconsin	Direction prediction: Cora-ML	Direction prediction: CiteSeer	Existence prediction: Cornell	Existence prediction: Wisconsin	Existence prediction: Cora-ML	Existence prediction: CiteSeer
ChebNet	71.0±5.5	67.5±4.5	72.7±1.5	68.0±1.6	80.1±2.3	82.5±1.9	80.0±0.6	77.4±0.4
GCN	56.2±8.7	71.0±4.0	79.8±1.1	68.9±2.8	75.1±1.4	75.1±1.9	81.6±0.5	76.9±0.5
APPNP	69.5±9.0	75.1±3.5	83.7±0.7	77.9±1.6	74.9±1.5	75.7±2.2	82.5±0.6	78.6±0.7
SAGE	75.2±11.0	72.0±3.5	68.2±0.8	68.7±1.5	79.8±2.4	77.3±2.9	75.0±0.0	74.1±1.0
GIN	69.3±6.0	74.8±3.7	83.2±0.9	76.3±1.4	74.5±2.1	76.2±1.9	82.5±0.7	77.9±0.7
GAT	67.9±11.1	53.2±2.6	50.0±0.1	50.6±0.5	77.9±3.2	74.6±0.0	75.0±0.0	75.0±0.0
DGCN	**80.7±6.3**	74.5±7.2	79.6±1.5	78.5±2.3	80.0±3.9	82.8±2.0	82.1±0.5	81.2±0.4
DiGraph	79.3±1.9	82.3±4.9	80.8±1.1	81.0±1.1	**80.6±2.5**	82.8±2.6	81.8±0.5	**82.2±0.6**
DiGraphIB	79.8±4.8	82.0±4.9	83.4±1.1	82.5±1.3	80.5±3.6	82.4±2.2	82.2±0.5	81.0±0.5
MagNet	**80.7±2.7**	**83.6±2.8**	**86.1±0.9**	**85.1±0.8**	**80.6±3.8**	**82.9±2.6**	**82.8±0.7**	79.9±0.5
Best q	0.10	0.05	0.05	0.15	0.25	0.25	0.05	0.05

on *Cornell* and a $\sim 2.5\%$ improvement on *Wisconsin*, while on *Texas* it has the second best accuracy, close behind SAGE. We also see the other directed methods have relatively poor performance on the WebKB networks, perhaps since these graphs are fairly small and have very few training samples.

On the DSBM, as illustrated in Figure 2, we see that MagNet generally performs quite well and is the best performing network in the vast majority cases (for full details, see Section 7 of the supplement). The networks DGCN and DiGraphIB rely on second order proximity matrices. As demonstrated in Figure 2c, these methods are well suited for networks with a cyclic meta-graph structure since nodes in the same cluster are likely to have common neighbors. MagNet, on the other hand, does not use second-order proximity, but can still learn the clusters by stacking multiple layers together. This improves MagNet's ability to adapt to directed networks with different underlying topologies. This is illustrated in Figure 2d where the network has an approximately cyclic meta-graph structure. In this setting, MagNet continues to perform well, but the performance of DGCN and DiGraphIB deteriorate significantly. Interestingly, MagNet performs well on the DSBM cyclic meta-graph (Figure 2c) with $q \approx .1$, whereas $q \geq .2$ is preferred for the other three DSBM tests; we leave a more in-depth investigation for future work. Further details are available in Section 8 of the supplement.

For link prediction, we achieve the best performance on seven out of eight tests as shown in Table 2. We also note that Table 2 reports optimal non-zero q values for each task. This indicates that incorporating directional information is important for link prediction, even on citation networks such as *Cora* and *CiteSeer*. This matches our intuition, since there is a clear difference between a paper with many citations and one with many references.

6 Conclusion

We have introduced MagNet, a neural network for directed graphs based on the magnetic Laplacian. This network can be viewed as the natural extension of spectral graph convolutional networks to the directed graph setting. We demonstrate the effectiveness of our network, and the importance of incorporating directional information via a complex Hermitian matrix, for link prediction and node classification on both real and synthetic datasets. Interesting avenues of future research would be using multiple q's along different channels, exploring the role of different normalizations of the magnetic Laplacian, and to incorporating the magnetic Laplacian into other network architectures.

Limitations and Ethical Considerations: Our method has natural extensions to weighted, directed graphs when all edges are directed. However, it not clear what is the best way to extend it to weighted mixed graphs (with both directed and undirected edges). Our network does not incorporate an attention mechanism and, similar to many other networks, is not scalable to large graphs in its current form (although this may be addressed in future work). All of our data is publicly available for research purposes and does not contain personally identifiable information or offensive content. The method presented here has no greater or lesser impact on society than other graph neural network algorithms.

References

[1] James Atwood and Don Towsley. Diffusion-convolutional neural networks. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, Advances in Neural Information Processing Systems, volume 29, pages 1993–2001. Curran Associates, Inc., 2016.

[2] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. Neural computation, 15(6):1373–1396, 2003.

[3] Austin R Benson, David F Gleich, and Jure Leskovec. Higher-order organization of complex networks. Science, 353(6295):163–166, 2016.

[4] Aleksandar Bojchevski and Stephan Günnemann. Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. In ICLR Workshop on Representation Learning on Graphs and Manifolds, 2017.

[5] Alexandre Bovet and Peter Grindrod. The activity of the far right on telegram. https://www.researchgate.net/publication/346968575_The_Activity_of_the_Far_Right_on_Telegram_v21, 2020.

[6] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and deep locally connected networks on graphs. In International Conference on Learning Representations (ICLR), 2014.

[7] Fan Chung. Laplacians and the cheeger inequality for directed graphs. Annals of Combinatorics, 9(1):1–19, 2005.

[8] Fan Chung and Mark Kempton. A local clustering algorithm for connection graphs. In International Workshop on Algorithms and Models for the Web-Graph, pages 26–43. Springer, 2013.

[9] Fan RK Chung and Fan Chung Graham. Spectral graph theory. Number 92. American Mathematical Soc., 1997.

[10] Alexander Cloninger. A note on markov normalized magnetic eigenmaps. Applied and Computational Harmonic Analysis, 43(2):370 – 380, 2017.

[11] Ronald R Coifman and Stéphane Lafon. Diffusion maps. Applied and computational harmonic analysis, 21(1):5–30, 2006.

[12] Mihai Cucuringu, Huan Li, He Sun, and Luca Zanetti. Hermitian matrices for clustering directed graphs: insights and applications. In International Conference on Artificial Intelligence and Statistics, pages 983–992. PMLR, 2020.

[13] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In Advances in Neural Information Processing Systems 29, pages 3844–3852, 2016.

[14] David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alan Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, Advances in Neural Information Processing Systems, volume 28, pages 2224–2232. Curran Associates, Inc., 2015.

[15] Bruno Messias F. de Resende and Luciano da F. Costa. Characterization and comparison of large directed networks through the spectra of the magnetic laplacian. Chaos: An Interdisciplinary Journal of Nonlinear Science, 30(7):073141, 2020.

[16] Michaël Fanuel, Carlos M. Alaíz, Ángela Fernández, and Johan A.K. Suykens. Magnetic eigenmaps for the visualization of directed networks. Applied and Computational Harmonic Analysis, 44:189–199, 2018.

[17] Michaël Fanuel, Carlos M Alaiz, and Johan AK Suykens. Magnetic eigenmaps for community detection in directed networks. Physical Review E, 95(2):022302, 2017.

[18] Satoshi Furutani, Toshiki Shibahara, Mitsuaki Akiyama, Kunio Hato, and Masaki Aida. Graph signal processing for directed graphs based on the hermitian laplacian. In Machine Learning and Knowledge Discovery in Databases, pages 447–463, 2020.

[19] Krystal Guo and Bojan Mohar. Hermitian adjacency matrix of digraphs and mixed graphs. Journal of Graph Theory, 85(1):217–248, 2017.

[20] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS'17, page 1025–1035, Red Hook, NY, USA, 2017. Curran Associates Inc.

[21] David K Hammond, Pierre Vandergheynst, and Rémi Gribonval. Wavelets on graphs via spectral graph theory. Applied and Computational Harmonic Analysis, 30(2):129–150, 2011.

[22] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In International Conference on Learning Representations (ICLR), 2017.

[23] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. In ICLR, 2019.

[24] Ron Levie, Wei Huang, Lorenzo Bucci, Michael M Bronstein, and Gitta Kutyniok. Transferability of spectral graph convolutional neural networks. arXiv preprint arXiv:1907.12972, 2019.

[25] Elliott H Lieb and Michael Loss. Fluxes, laplacians, and kasteleyn's theorem. In Statistical Mechanics, pages 457–483. Springer, 1993.

[26] Yi Ma, Jianye Hao, Yaodong Yang, Han Li, Junqi Jin, and Guangyong Chen. Spectral-based graph convolutional network for directed graphs. arXiv:1907.08990, 2019.

[27] Antonio G Marques, Santiago Segarra, and Gonzalo Mateos. Signal processing on directed graphs: The role of edge directionality when processing and learning from network data. IEEE Signal Processing Magazine, 37(6):99–116, 2020.

[28] Péter Mernyei and Cătălina Cangea. Wiki-cs: A wikipedia-based benchmark for graph neural networks. arXiv preprint arXiv:2007.02901, 2020.

[29] Bojan Mohar. A new kind of hermitian matrices for digraphs. Linear Algebra and its Applications, 584:343–352, 2020.

[30] Federico Monti, Karl Otness, and Michael M. Bronstein. Motifnet: A motif-based graph convolutional network for directed graphs. In 2018 IEEE Data Science Workshop, pages 225–228, 2018.

[31] Antonio Ortega, Pascal Frossard, Jelena Kovačević, José MF Moura, and Pierre Vandergheynst. Graph signal processing: Overview, challenges, and applications. Proceedings of the IEEE, 106(5):808–828, 2018.

[32] Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. Geom-gcn: Geometric graph convolutional networks. arXiv preprint arXiv:2002.05287, 2020.

[33] Benedek Rozemberczki, Carl Allen, and Rik Sarkar. Multi-scale attributed node embedding. arXiv preprint arXiv:1909.13021, 2019.

[34] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. In Proceedings of IEEE computer society conference on computer vision and pattern recognition, pages 731–737. IEEE, 1997.

[35] Daniel A Spielman and Shang-Hua Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In Proceedings of the thirty-sixth annual ACM symposium on Theory of computing, pages 81–90, 2004.

[36] Z. Tong, Yuxuan Liang, Changsheng Sun, Xinke Li, David S. Rosenblum, and A. Lim. Digraph inception convolutional networks. In NeurIPS, 2020.

[37] Zekun Tong, Yuxuan Liang, Changsheng Sun, David S. Rosenblum, and Andrew Lim. Directed graph convolutional network. arXiv:2004.13970, 2020.

[38] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. International Conference on Learning Representations, 2018.

[39] Palmer W.R. and Zheng T. Spectral clustering for directed networks. Studies in Computational Intelligence, 943, 2021.

[40] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A comprehensive survey on graph neural networks. IEEE Transactions on Neural Networks and Learning Systems, 32(1):4–24, 2020.

[41] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? arXiv preprint arXiv:1810.00826, 2018.

[42] Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. arXiv preprint arXiv:1812.08434, 2018.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] Please see Section 6.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] Please see Section 6.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] Please see Section 5 of the supplement.
   (b) Did you include complete proofs of all theoretical results? [Yes] Please see Section 5 of the supplement.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] Please see Section 1 of the supplemental material.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] Please see Section 5 of the main text and Section 3 of the supplemental material.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] Please see Section 5 of the main text and Section 7 of the supplemental material.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] Please see Section 3 of the supplemental material.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] Please see Section 5 of the main text, the supplemental material, and the anonymous code for this paper that lists all used packages.
   (b) Did you mention the license of the assets? [Yes] Please see our anonymous code for this paper.

(c) Did you include any new assets either in the supplemental material or as a URL? [Yes] Please see our anonymous code for this paper.

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] All data is publicly available and permits use for research.

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] All data, to the best of our knowledge, is de-identified and contains no personal information, nor does it contain offensive content.

5. If you used crowdsourcing or conducted research with human subjects...

   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]