Neural Networks 128 (2020) 188–198

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

Fast Haar Transforms for Graph Neural Networks

Ming Li [a,c], Zheng Ma [b], Yu Guang Wang [c,*], Xiaosheng Zhuang [d]

[a] Department of Educational Technology, Zhejiang Normal University, Jinhua, China
[b] Department of Physics, Princeton University, NJ, USA
[c] School of Mathematics and Statistics, The University of New South Wales, Sydney, Australia
[d] Department of Mathematics, City University of Hong Kong, Hong Kong

ARTICLE INFO

Article history:
Received 30 October 2019
Received in revised form 29 February 2020
Accepted 27 April 2020
Available online 4 May 2020

Keywords:
Graph Neural Networks
Haar basis
Graph convolution
Fast Haar Transforms
Geometric deep learning
Graph Laplacian

ABSTRACT

Graph Neural Networks (GNNs) have become a topic of intense research recently due to their powerful capability in high-dimensional classification and regression tasks for graph-structured data. However, as GNNs typically define the graph convolution by the orthonormal basis for the graph Laplacian, they suffer from high computational cost when the graph size is large. This paper introduces a Haar basis, which is a sparse and localized orthonormal system for a coarse-grained chain on the graph. The graph convolution under Haar basis, called Haar convolution, can be defined accordingly for GNNs. The sparsity and locality of the Haar basis allow Fast Haar Transforms (FHTs) on the graph, by which one then achieves a fast evaluation of Haar convolution between graph data and filters. We conduct experiments on GNNs equipped with Haar convolution, which demonstrates state-of-the-art results on graph-based regression and node classification tasks.

1. Introduction

Convolutional neural networks (CNNs) have been very successful machinery in many high-dimensional regression and classification tasks on Euclidean domains. Recently, its generalization to non-Euclidean domains, known as *geometric deep learning*, has attracted growing attention, due to its high potential in pattern recognition and regression for graph-structured data (Bronstein, Bruna, LeCun, Szlam, & Vandergheynst, 2017).

Graph neural networks (GNNs) are a typical model in geometric deep learning, which replaces the partial derivatives in CNNs by the Laplacian operator (Bruna, Zaremba, Szlam, & Lecun, 2014; Henaff, Bruna, & LeCun, 2015). The Laplacian, which carries the structural features of the data, is a second-order isotropic differential operator that admits a natural generalization to graphs and manifolds. In GNNs, input data are convoluted with filters under an orthonormal system for the Laplacian. However, as the algebraic properties of regular Euclidean grids are lost in general manifolds and graphs, FFTs (fast Fourier transforms) for the Laplacian is not available. This issue leads to that the computation of convolution for graph data is not always efficient, especially when the graph dataset is large.

In this paper, we introduce an alternative orthonormal system on the graph, the *Haar basis*. It then defines a new graph convolution for GNNs — *Haar convolution*. Due to the sparsity and locality of the Haar basis, fast Haar transforms (FHTs) can be achieved on graph-structured data. This significantly improves the computational efficiency of GNNs as the Haar convolution guarantees the linear computational complexity. We apply Haar convolution to GNNs and give a novel type of deep convolutional neural networks on graph — HANet. Numerical tests on real graph datasets show that HANet achieves excellent performance and computational efficiency in classification and regression tasks. To the best of our knowledge, our method is the first fast algorithm for spectral graph convolution by appropriately selecting an orthogonal basis on the graph, which is of great importance in the line of building spectral-based GNN models. Overall, we summarize the significant contributions of the paper as three folds.

- The Haar basis is introduced for graphs. Both theoretical analysis and real examples of the sparsity and locality are given. With these properties, the fast algorithms for Haar transforms (FHTs) are developed, and their complexity analysis is studied.
- The Haar convolution under Haar basis is developed. Under FHTs, the computational cost for Haar convolution is proportional to the size of the graph, which is more efficient than Laplacian-based spectral graph convolution. Other technical components, including weight sharing and detaching, chain and pooling, are also presented in detail.
- GNN with Haar convolution (named HANet) is proposed. The experiments illustrate that HANet with high efficiency

* Corresponding author.
E-mail addresses: mingli@zjnu.edu.cn (M. Li), zhengm@princeton.edu (Z. Ma), yuguang.wang@unsw.edu.au (Y.G. Wang), xzhuang7@cityu.edu.hk (X. Zhuang).

https://doi.org/10.1016/j.neunet.2020.04.028
0893-6080/

achieves good performance on a broad range of high-dimensional regression and classification problems on graphs.

We organize the paper as follows. In Section 2, we review recent advances on GNNs. In Section 3, we construct the Haar orthonormal basis using a chain on the graph. The Haar basis will be used to define a new graph convolution, called Haar convolution. In Section 4, we develop fast algorithms for Haar transforms, and the fast Haar transforms allows fast computation of Haar convolution. In Section 5, we use the Haar convolution as the graph convolution in graph neural networks. Section 6 shows the experimental results of GNNs with Haar convolution (HANet) on tasks of graph-based regression and node classification.

2. Related work

Developing deep neural networks for graph-structured data has received extensive attention in recent years (Battaglia et al., 2018; Li, Tarlow, Brockschmidt, & Zemel, 2016; Scarselli, Gori, Tsoi, Hagenbuchner, & Monfardini, 2009; Wang et al., 2019; Wu, Pan, Chen, Long, Zhang, & Yu, 2020; Zhang, Cui, & Zhu, 2020; Zhou et al., 2018). Bruna et al. (2014) first propose graph convolution, which is defined by graph Fourier transforms under the orthogonal basis from the graph Laplacian. The graph convolution uses Laplacian eigendecomposition, which is computationally expensive. Defferrard, Bresson, and Vandergheynst (2016) approximate smooth filters in the spectral domain by Chebyshev polynomials. Kipf and Welling (2017) simplify the convolutional layer by exploiting first-order Chebyshev polynomial for filters. Following this line, several acceleration methods for graph convolutional networks are proposed (Chen, Ma and Xiao, 2018; Chen, Zhu and Song, 2018). Graph wavelet neural networks (GWNN) (Xu, Shen, Cao, Qiu and Cheng, 2019) replace graph Fourier transform by graph wavelet transform in the graph convolution, where Chebyshev polynomials are used to approximate the graph wavelet basis (Hammond, Vandergheynst, & Gribonval, 2011). Although GWNN circumvents the Laplacian eigendecomposition, the matrix inner-product operations are nevertheless not avoidable in wavelet transforms for convolution computation.

Graph convolutional networks with attention mechanisms, e.g., Graph Attention Networks (GATs) proposed by Veličković et al. (2018), can effectively learn the importance between nodes and their neighbors, which is more suitable for node classification task (than graph-based regression). Nevertheless, much computational and memory cost is required to perform the attention mechanism in the convolutional layers. Yang, Wang, Song, Yuan, and Tao (2019) propose the Shortest Path Graph Attention Network (SPAGAN) by using a path-based attention mechanism in node-level aggregation, which leads to superior results than GATs concerning neighbor-based attention.

Some GNN models use multi-scale information and higher-order adjacency matrix to define graph convolution (Abu-El-Haija, Kapoor, Perozzi, & Lee, 2018; Liao, Zhao, Urtasun, & Zemel, 2019; Wu et al., 2019). To increase the scalability of the model for a large-scale graph, Hamilton, Ying, and Leskovec (2017) propose the framework Graph-SAGE with sampling and a neural network-based aggregator over a fixed size node neighbor. Atwood and Towsley (2016) develop diffusion convolutional neural networks by using a diffusion operator for graph convolution. MoNet (Monti et al., 2017) introduces a general methodology to define spatial-based graph convolution by the weighted average of multiple weighting functions in the neighborhood. Gilmer, Schoenholz, Riley, Vinyals, and Dahl (2017) provide a unified framework, the Message Passing Neural Networks (MPNNs), by which some existing GNN models are incorporated. Xu, Hu, Leskovec and Jegelka (2019) present a theoretical analysis for the expressive power of GNNs and propose a simple but powerful variation of GNN, the graph isomorphism network. By generalizing the graph Laplacian to maximal entropy transition matrix derived from a path integral, Ma, Li, and Wang (2019) propose a new framework called PAN that involves every path linking the message sender and receiver with learnable weights depending on the path length.

3. Graph convolution with Haar basis

3.1. Graph Fourier transform

Bruna et al. (2014) first defined the graph convolution based on spectral graph theory (Chung & Graham, 1997) and the graph Laplacian. An un-directed weighted graph $\mathcal{G} = (V, E, w)$ is a triplet with vertices V, edges E and weights $w : E \to \mathbb{R}$. Denote by $N := |V|$ the number of vertices of the graph. Let $l_2(\mathcal{G}) := \{f : V \to \mathbb{R} \mid \sum_{v\in V} |f(v)|^2 < \infty\}$ be the real-valued l_2 space on the graph with inner product $f \cdot g := \sum_{v\in V} f(v)g(v)$. A basis for $l_2(\mathcal{G})$ is a set of vectors $\{u_\ell\}_{\ell=1}^N$ on $\mathcal{G}$ which are linearly independent and orthogonal (i.e. $u_\ell \cdot u_{\ell'} = 0$ if $\ell \neq \ell'$). The (normalized) eigenvectors $\{u_\ell\}_{\ell=1}^{|V|}$ of the graph Laplacian $\mathcal{L}$ forms an orthonormal basis for $l_2(\mathcal{G})$. We call the matrix $U := (u_1, \ldots, u_N)$ the *(graph Fourier)* base matrix, whose columns form the *graph Fourier basis* for $l_2(\mathcal{G})$. The *graph convolution* can then be defined by

$$g \star f = U\big((U^T g) \odot (U^T f)\big), \tag{3.1}$$

where $U^T f$ is regarded as the *adjoint discrete graph Fourier transform* of f, Uc is the *forward discrete graph Fourier transform* of c on $\mathcal{G}$ and $\odot$ is the element-wise Hadamard product.

While graph convolution defined in (3.1) is conceptually essential, it has some limitations in practice. For example, the base matrix U is obtained by using the eigendecomposition of the graph Laplacian in the sense that $\mathcal{L} = U\Lambda U^T$, where Λ is the diagonal matrix of corresponding eigenvalues. The computational complexity is proportional to $\mathcal{O}(N^3)$, which is impractical when the number of vertices of the graph is quite large. Second, the computation of the forward and inverse graph Fourier transforms (i.e., $U^T f$ and Uc) have $\mathcal{O}(N^2)$ computational cost due to the multiplication by (dense) matrices U and U^T. In general, there are no fast algorithms for the graph Fourier transforms as the graph nodes are not regular, and the matrix U is not sparse. Third, filters in the spectral domain cannot guarantee the localization in the spatial (vertex) domain, and $O(Ndm)$ parameters need to be tuned in the convolutional layer with m filters (hidden nodes) and d features for each vertex.

To alleviate the cost of computing the graph Fourier transform, Defferrard et al. (2016) used the Chebyshev polynomials to construct localized polynomial filters for graph convolution, which results in the graph neural network called ChebNet. Kipf and Welling (2017) simplify ChebNet to obtain graph convolutional networks (GCNs). However, such a polynomial-based approximation strategy may lose information in the spectral graph convolutional layer, and matrix multiplication is still not avoidable as FFTs are not available for graph convolution. Thus, the graph convolution in this scenario is also computationally expensive, especially for the dense graph of a large size. Here, we propose an alternative orthonormal basis that allows fast computation for the corresponding graph convolution, which then improves the scalability and efficiency of existing graph models. The basis we use is the Haar basis on a graph. The Haar basis replaces the matrix of eigenvectors U in (3.1) and forms a highly sparse matrix, which reflects the clustering information of the graph. The sparsity of the Haar transform matrix allows fast computation (in nearly linear computational complexity) of the corresponding graph convolution.

3.2. Haar basis

Haar basis rooted in the theory of Haar wavelet basis as first introduced by Haar (1910), is a particular case of Daubechies wavelets (Daubechies, 1992), and later developed onto graph by Belkin, Niyogi, and Sindhwani (2006), see also Chui, Filbir, and Mhaskar (2015). The construction of the Haar basis exploits a chain of the graph. For a graph $\mathcal{G} = (V, E, w)$, a graph $\mathcal{G}^{\mathrm{cg}} := (V^{\mathrm{cg}}, E^{\mathrm{cg}}, w^{\mathrm{cg}})$ is called a *coarse-grained graph* of $\mathcal{G}$ if $|V^{\mathrm{cg}}| \le |V|$ and each vertex of $\mathcal{G}$ associates with exactly one (parent) vertex in $\mathcal{G}^{\mathrm{cg}}$. Each vertex of $\mathcal{G}^{\mathrm{cg}}$ is called a *cluster* of $\mathcal{G}$. Let J_0, J be two integers such that $J > J_0$. A *coarse-grained chain* for $\mathcal{G}$ is a set of graphs $\mathcal{G}_{J\to J_0} := (\mathcal{G}_J, \mathcal{G}_{J-1}, \ldots, \mathcal{G}_{J_0})$ such that $\mathcal{G}_J = \mathcal{G}$, and $\mathcal{G}_j$ is a coarse-grained graph of $\mathcal{G}_{j+1}$ for $j = J_0, J_0+1, \ldots, J-1$. The $\mathcal{G}_{J_0}$ is the *top level* or the *coarsest level* graph while $\mathcal{G}_J$ is the *bottom level* or the *finest level* graph. If the top level $\mathcal{G}_{J_0}$ of the chain has only one node, $\mathcal{G}_{J\to J_0}$ becomes a tree. The chain $\mathcal{G}_{J\to J_0}$ gives a hierarchical partition for the graph $\mathcal{G}$. For details about graphs and chains, see examples in Chui et al. (2015), Chui, Mhaskar, and Zhuang (2018), Chung and Graham (1997), Hammond et al. (2011), Wang and Zhuang (2020) and Wang and Zhuang (2019).

Construction of Haar basis. With a chain of the graph, one can generate a Haar basis for $l_2(\mathcal{G})$ following Chui et al. (2015), see also Gavish, Nadler, and Coifman (2010). We show the construction of Haar basis on $\mathcal{G}$, as follows.

Step 1. Let $\mathcal{G}^{\mathrm{cg}} = (V^{\mathrm{cg}}, E^{\mathrm{cg}}, w^{\mathrm{cg}})$ be a coarse-grained graph of $\mathcal{G} = (V, E, w)$ with $N^{\mathrm{cg}} := |V^{\mathrm{cg}}|$. Each vertex $v^{\mathrm{cg}} \in V^{\mathrm{cg}}$ is a cluster $v^{\mathrm{cg}} = \{v \in V \mid v \text{ has parent } v^{\mathrm{cg}}\}$ of $\mathcal{G}$. Order V^{cg}, e.g., by degrees of vertices or weights of vertices, as $V^{\mathrm{cg}} = \{v_1^{\mathrm{cg}}, \ldots, v_{N^{\mathrm{cg}}}^{\mathrm{cg}}\}$. We define N^{cg} vectors ϕ_ℓ^{cg} on $\mathcal{G}^{\mathrm{cg}}$ by

$$\phi_1^{\mathrm{cg}}(v^{\mathrm{cg}}) := \frac{1}{\sqrt{N^{\mathrm{cg}}}}, \quad v^{\mathrm{cg}} \in V^{\mathrm{cg}}, \tag{3.2}$$

and for $\ell = 2, \ldots, N^{\mathrm{cg}}$,

$$\phi_\ell^{\mathrm{cg}} := \sqrt{\frac{N^{\mathrm{cg}} - \ell + 1}{N^{\mathrm{cg}} - \ell + 2}} \left(\chi_{\ell-1}^{\mathrm{cg}} - \frac{\sum_{j=\ell}^{N^{\mathrm{cg}}} \chi_j^{\mathrm{cg}}}{N^{\mathrm{cg}} - \ell + 1} \right), \tag{3.3}$$

where χ_j^{cg} is the indicator function for the jth vertex $v_j^{\mathrm{cg}} \in V^{\mathrm{cg}}$ on $\mathcal{G}$ given by

$$\chi_j^{\mathrm{cg}}(v^{\mathrm{cg}}) := \begin{cases} 1, & v^{\mathrm{cg}} = v_j^{\mathrm{cg}}, \\ 0, & v^{\mathrm{cg}} \in V^{\mathrm{cg}} \backslash \{v_j^{\mathrm{cg}}\}. \end{cases}$$

Then, the set of functions $\{\phi_\ell^{\mathrm{cg}}\}_{\ell=1}^{N^{\mathrm{cg}}}$ forms an orthonormal basis for $l_2(\mathcal{G}^{\mathrm{cg}})$.

Note that each $v \in V$ belongs to exactly one cluster $v^{\mathrm{cg}} \in V^{\mathrm{cg}}$. In view of this, for each $\ell = 1, \ldots, N^{\mathrm{cg}}$, we extend the vector ϕ_ℓ^{cg} on $\mathcal{G}^{\mathrm{cg}}$ to a vector $\phi_{\ell,1}$ on $\mathcal{G}$ by

$$\phi_{\ell,1}(v) := \frac{\phi_\ell^{\mathrm{cg}}(v^{\mathrm{cg}})}{\sqrt{|v^{\mathrm{cg}}|}}, \quad v \in v^{\mathrm{cg}},$$

here $|v^{\mathrm{cg}}| := k_\ell$ is the size of the cluster v^{cg}, i.e., the number of vertices in $\mathcal{G}$ whose common parent is v^{cg}. We order the cluster v_ℓ^{cg}, e.g., by degrees of vertices, as

$$v_\ell^{\mathrm{cg}} = \{v_{\ell,1}, \ldots, v_{\ell,k_\ell}\} \subseteq V.$$

For $k = 2, \ldots, k_\ell$, similar to (3.3), define

$$\phi_{\ell,k} = \sqrt{\frac{k_\ell - k + 1}{k_\ell - k + 2}} \left(\chi_{\ell,k-1} - \frac{\sum_{j=k}^{k_\ell} \chi_{\ell,j}}{k_\ell - k + 1} \right),$$

where for $j = 1, \ldots, k_\ell$, $\chi_{\ell,j}$ is given by

$$\chi_{\ell,j}(v) := \begin{cases} 1, & v = v_{\ell,j}, \\ 0, & v \in V \backslash \{v_{\ell,j}\}. \end{cases}$$

One can verify that the resulting $\{\phi_{\ell,k} : \ell = 1, \ldots, N^{\mathrm{cg}}, k = 1, \ldots, k_\ell\}$ is an orthonormal basis for $l_2(\mathcal{G})$.

Step 2. Let $\mathcal{G}_{J\to J_0}$ be a coarse-grained chain for the graph $\mathcal{G}$. An orthonormal basis $\{\phi_\ell^{(0)}\}_{\ell=1}^{N_0}$ for $l_2(\mathcal{G}_{J_0})$ is generated using (3.2) and (3.3). We then repeatedly use Step 1: for $j = J_0 + 1, \ldots, J$, we generate an orthonormal basis $\{\phi_\ell^{(j)}\}_{\ell=1}^{N_j}$ for $l_2(\mathcal{G}_j)$ from the orthonormal basis $\{\phi_\ell^{(j-1)}\}_{\ell=1}^{N_{j-1}}$ for the coarse-grained graph $\mathcal{G}_{j-1}$ that was derived in the previous steps. We call the sequence $\{\phi_\ell := \phi_\ell^{(J)}\}_{\ell=1}^{N}$ of vectors at the finest level, the *Haar global orthonormal basis* or simply the *Haar basis* for $\mathcal{G}$ associated with the chain $\mathcal{G}_{J\to J_0}$. The orthonormal basis $\{\phi_\ell^{(j)}\}_{\ell=1}^{N_j}$ for $l_2(\mathcal{G}_j)$, $j = J-1, J-2, \ldots, J_0$ is called the *associated (orthonormal) basis* for the Haar basis $\{\phi_\ell\}_{\ell=1}^{N}$.

Proposition 3.1. *For each level $j = J_0, \ldots, J$, the sequence $\{\phi_\ell^{(j)}\}_{\ell=1}^{N_j}$ is an orthonormal basis for $l_2(\mathcal{G}_j)$, and in particular, $\{\phi_\ell\}_{\ell=1}^{N}$ is an orthonormal basis for $l_2(\mathcal{G})$; each basis $\{\phi_\ell^{(j)}\}_{\ell=1}^{N_j}$ is the Haar basis for the chain $\mathcal{G}_{j\to J_0}$.*

Proposition 3.2. *Let $\mathcal{G}_{J\to J_0}$ be a coarse-grained chain for $\mathcal{G}$. If each parent of level $\mathcal{G}_j$, $j = J-1, J-2, \ldots, J_0$, contains at least two children, the number of different values of the components of a Haar basis vector $\phi_\ell^{(j)}$, $\ell = 1, \ldots, N_j$, is bounded by a constant independent of j.*

The Haar basis depends on the chain for the graph. If the clustering of the chain well reflects the topology of the graph, the Haar basis then contains the crucial geometric information of the graph. For example, by using k-means clustering algorithm or METIS algorithm (Karypis & Kumar, 1998) one can generate a chain that reveals the desired geometric properties of the graph.

Fig. 1b shows a chain $\mathcal{G}_{2\to 0}$ with 3 levels of a graph $\mathcal{G}$. Here, for each level, the vertices are given by

$$\begin{aligned} V^{(2)} &= V = \{v_1, \ldots, v_8\}, \\ V^{(1)} &= \{v_1^{(1)}, v_2^{(1)}, v_3^{(1)}, v_4^{(1)}\} \\ &= \{\{v_1, v_2\}, \{v_3, v_4\}, \{v_5, v_6\}, \{v_7, v_8\}\}, \\ V^{(0)} &= \{v_1^{(0)}, v_2^{(0)}\} = \{\{v_1^{(1)}, v_2^{(1)}\}, \{v_3^{(1)}, v_4^{(1)}\}\}. \end{aligned}$$

Fig. 1a shows the Haar basis for the chain $\mathcal{G}_{2\to 0}$. There are in total 8 vectors of the Haar basis for $\mathcal{G}$. From construction, the Haar basis ϕ_ℓ and the associated basis $\phi_\ell^{(j)}, j = 1, 2$ are closely connected: the ϕ_1, ϕ_2 can be reduced to $\phi_1^{(0)}, \phi_2^{(0)}$ and the $\phi_1, \phi_2, \phi_3, \phi_4$ can be reduced to $\phi_1^{(1)}, \phi_2^{(1)}, \phi_3^{(1)}, \phi_4^{(1)}$. This connection would allow fast algorithms for Haar transforms as given in Algorithms 1 and 2. In Fig. 1, the matrix Φ^T of the 8 Haar basis vectors ϕ_ℓ on $\mathcal{G}$ has good sparsity. With the increase of the graph size, the sparsity of the Haar transform matrix Φ becomes prominent, which we will demonstrate in the experiments in Section 6.3.

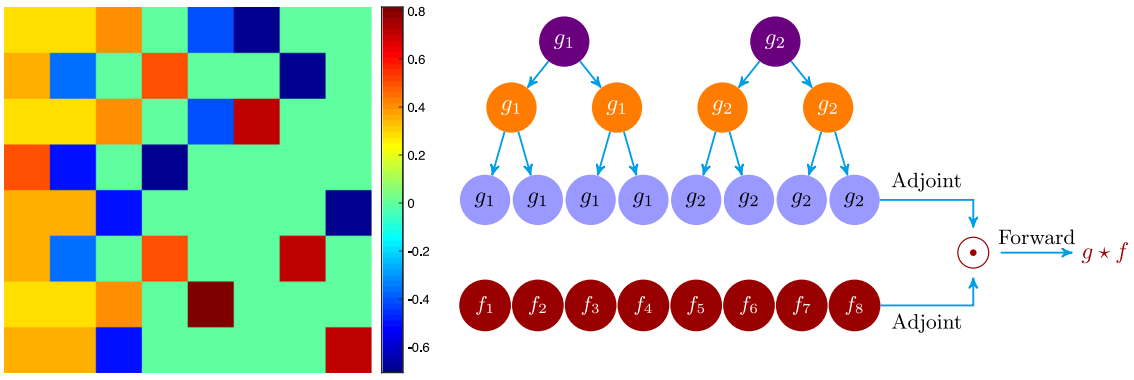
3.3. Haar convolution

With the Haar basis constructed in Section 3.2, we can define Haar convolution as an alternative form of spectral graph convolution in (3.1). Let $\{\phi_\ell\}_{\ell=1}^{N}$ be the Haar basis associated with a chain $\mathcal{G}_{J\to J_0}$ of a graph $\mathcal{G}$. Denoted by $\Phi = (\phi_1, \ldots, \phi_N) \in \mathbb{R}^{N\times N}$ the Haar transform matrix. We define by

$$\Phi^T f = \left(\sum_{v\in V} \phi_1(v) f(v), \ldots, \sum_{v\in V} \phi_N(v) f(v) \right) \in \mathbb{R}^N \tag{3.4}$$

the *adjoint Haar transform* for graph data f on $\mathcal{G}$, and by

$$(\Phi c)(v) = \sum_{\ell=1}^{N} \phi_\ell(v) c_\ell, \quad v \in V, \tag{3.5}$$



(a) Haar transform matrix

(b) Haar Convolution

Fig. 1. (a) The 8×8 matrix Φ of the *Haar Basis* for a graph $\mathcal{G}$ with 8 nodes. The green entries are zero, and the matrix Φ is sparse. The Haar basis is created based on the coarse-grained chain $\mathcal{G}_{2\to 0} := (\mathcal{G}_2, \mathcal{G}_1, \mathcal{G}_0)$, where $\mathcal{G}_2, \mathcal{G}_1, \mathcal{G}_0$ are graphs with $8, 4, 2$ nodes. For $j = 1, 2$, each node of $\mathcal{G}_{j-1}$ is a cluster of nodes in $\mathcal{G}_j$. Each column of Φ is a member of the Haar basis. The first two columns can be compressed as an orthonormal basis of $\mathcal{G}_0$, and the first to fourth columns can be reduced to the orthonormal basis for $\mathcal{G}_1$. (b) *Haar Convolution* $g \star f$ using the Haar basis of (a), where the weight sharing for filter vector g is defined by the chain $\mathcal{G}_{2\to 0}$ and the $g \star f$ is the forward Haar transform of the point-wise product of the adjoint Haar transforms of g and f, where the Haar transforms have a fast algorithmic implementation. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

the *forward Haar transform* for (coefficients) vector $c := (c_1, \ldots, c_N) \in \mathbb{R}^N$. We call the matrix Φ *Haar transform matrix.*

Definition 3.3. The Haar convolution for filter g and graph data f on $\mathcal{G}$ can be defined as

$$g \star f = \Phi((\Phi^T g) \odot (\Phi^T f)). \tag{3.6}$$

Computationally, (3.6) is obtained by performing forward Haar transform of the element-wise Hadamard product between adjoint Haar transform of g and f. Compared with the Laplacian based spectral graph convolution given in (3.1), the Haar convolution has the following features. (i) the Haar transform matrix Φ is sparse and the computation of $\Phi^T f$ or Φc is more efficient than $U^T f$ or Uc; (ii) as the Haar basis is constructed based on the chain of the graph which reflects the clustering property for vertices, the Haar convolution can extract abstract features for input graph data, that is, it provides a learning representation for graph-structured data; (iii) through the sparsity of Haar basis, the adjoint and forward Haar transforms can be implemented by fast algorithms, which have nearly linear computational complexity (to the size of the input graph).

We can presume the filter in the "frequency domain" and skip adjoint Haar transform of filter g (i.e. $\Phi^T g$), and then write Haar convolution as $g \star f = \Phi(g \odot (\Phi^T f))$.

3.4. Fast algorithms for Haar transforms and Haar convolution

The computation of Haar transforms can also be accelerated by using sparse matrix multiplications due to the sparsity of the Haar transform matrix. This acceleration allows the linear computational complexity $O(\epsilon N)$ with sparsity $1 - \epsilon$ of the Haar transform matrix. Moreover, a similar computational strategy to the sparse Fourier transforms (Hassanieh, Indyk, Katabi, & Price, 2012; Indyk, Kapralov, & Price, 2014) can be applied so that the Haar transforms achieve faster implementation with time complexity $O(k \log N)$ for a graph with N nodes and the Haar transform matrix with k non-zero elements. By the sparsity of the Haar transform matrix, the fast Haar transforms (FHTs), which includes *adjoint Haar transform* and *forward Haar transform* can be developed to speed up the implementation of the Haar convolution. Theorems 4.1 and 4.2 in the following section show that the computational cost of the adjoint and forward Haar transforms reaches $\mathcal{O}(N)$ and $\mathcal{O}(N(\log N)^2)$. They are nearly linear computational complexity and are thus called *fast Haar transforms* (FHTs). The Haar convolution in (3.6) consists of two adjoint Haar transforms and a forward Haar transform, and can then be evaluated in $\mathcal{O}(N(\log N)^2)$ steps.

3.5. Weight sharing

We can use weight sharing in Haar convolution to reduce the number of parameters of the filter and capture the common feature of the nodes which belong to the same cluster. As the resulting clusters contain information of the neighborhood, we can use the chain $\mathcal{G}_{J\to J_0}$ for weight sharing: the vertices of the graph which have the same parent at a coarser level share a parameter of the filter. Here, the coarser level is some fixed level $J_1, J_0 \le J_1 < J$. For example, the weight sharing rule for chain $\mathcal{G}_{2\to 0}$ in Fig. 1b is: assign the weight g_i for each node $v_i^{(0)}$, $i = 1, 2$ on the top level, the filter (or the weight vector) at the bottom level is then $g = (g_1, g_1, g_1, g_1, g_2, g_2, g_2, g_2)$. In this way, one has used the filter g with two independent parameters g_1, g_2 to convolute with the input vector with 8 components.

4. Fast algorithms under Haar basis

For the Haar convolution introduced in Definition 3.3, we can develop an efficient computational strategy by the sparsity of the Haar transform matrix. Let $\mathcal{G}_{J\to J_0}$ be a coarse-grained chain of the graph $\mathcal{G}$. For convenience, we label the vertices of the level-j graph $\mathcal{G}_j$ by $V_j := \{v_1^{(j)}, \ldots, v_{N_j}^{(j)}\}$.

4.1. Fast computation for adjoint Haar transform $\Phi^T f$

The adjoint Haar transform in (3.4) can be computed in the following way. For $j = J_0, \ldots, J-1$, let $c_k^{(j)}$ be the number of children of $v_k^{(j)}$, i.e. the number of vertices of $\mathcal{G}_{j+1}$ which belongs to the cluster $v_k^{(j)}$, for $k = 1, \ldots, N_j$. For $j = J$, let $c_k^{(J)} \equiv 1$ for $k = 1, \ldots, N$. For $j = J_0, \ldots, J$ and $k = 1, \ldots, N_j$, we define the weight factor for $v_k^{(j)}$ by

$$w_k^{(j)} := \frac{1}{\sqrt{c_k^{(j)}}}. \tag{4.1}$$

Let $W_{J\to J_0} := \{w_k^{(j)} \,|\, j = J_0, \ldots, J,\ k = 1, \ldots, N_j\}$. Then, the weighted chain $(\mathcal{G}_{J\to J_0}, W_{J\to J_0})$ is a *filtration* if each parent in the

chain $\mathcal{G}_{J\to J_0}$ has at least two children. See e.g. Chui et al. (2015, Definition 2.3).

Let $\{\phi_\ell\}_{\ell=1}^N$ be the Haar basis obtained in Step 2 of Section 3.2, which we also call the Haar basis for the filtration $(\mathcal{G}_{J\to J_0}, W_{J\to J_0})$ of a graph $\mathcal{G}$. We define the weighted sum for $f \in l_2(\mathcal{G})$ by

$$\mathcal{S}^{(J)}\big(f, v_k^{(J)}\big) := f(v_k^{(J)}), \quad v_k^{(J)} \in \mathcal{G}_J, \tag{4.2}$$

and for $j = J_0, \ldots, J-1$ and $v_k^{(j)} \in \mathcal{G}_j$,

$$\mathcal{S}^{(j)}\big(f, v_k^{(j)}\big) := \sum_{v_{k'}^{(j+1)} \in v_k^{(j)}} w_{k'}^{(j+1)} \mathcal{S}^{(j+1)}\big(f, v_{k'}^{(j+1)}\big). \tag{4.3}$$

For each vertex $v_k^{(j)}$ of $\mathcal{G}_j$, the $\mathcal{S}^{(j)}\big(f, v_k^{(j)}\big)$ is the weighted sum of the $\mathcal{S}^{(j+1)}\big(f, v_{k'}^{(j+1)}\big)$ at the level $j+1$ for those vertices $v_{k'}^{(j+1)}$ of $\mathcal{G}_{j+1}$ whose parent is $v_k^{(j)}$.

The adjoint Haar transform can be evaluated by the following theorem.

Theorem 4.1. *Let $\{\phi_\ell\}_{\ell=1}^N$ be the Haar basis for the filtration $(\mathcal{G}_{J\to J_0}, W_{J\to J_0})$ of a graph $\mathcal{G}$. Then, the adjoint Haar transform for the vector f on the graph $\mathcal{G}$ can be computed by, for $\ell = 1, \ldots, N$,*

$$(\Phi^T f)_\ell = \sum_{k=1}^{N_j} \mathcal{S}^{(j)}\big(f, v_k^{(j)}\big) w_k^{(j)} \phi_\ell^{(j)}(v_k^{(j)}), \tag{4.4}$$

where j is the smallest possible number in $\{J_0, \ldots, J\}$ such that $\phi_\ell^{(j)}$ is the ℓth member of the orthonormal basis $\{\phi_\ell^{(j)}\}_{\ell=1}^{N_j}$ for $l_2(\mathcal{G}_j)$ associated with the Haar basis $\{\phi_\ell\}_{\ell=1}^N$ (see Section 3.2), $v_k^{(j)}$ are the vertices of $\mathcal{G}_j$ and weights $w_k^{(j)}$ are given by (4.1).

Proof. By the relation between ϕ_ℓ and $\phi_\ell^{(j)}$,

$$\begin{aligned}
(\Phi^T f)_\ell &= \sum_{k=1}^{N} f(v_k^{(J)}) \phi_\ell(v_k^{(J)}) \\
&= \sum_{k'=1}^{N_{J-1}} \left(\sum_{v_k^{(J)} \in v_{k'}^{(J-1)}} f(v_k^{(J)}) \right) w_{k'}^{(J-1)} \phi_\ell^{(J-1)}(v_{k'}^{(J-1)}) \\
&= \sum_{k'=1}^{N_{J-1}} \mathcal{S}^{(J-1)}(f, v_{k'}^{(J-1)}) w_{k'}^{(J-1)} \phi_\ell^{(J-1)}(v_{k'}^{(J-1)}) \\
&= \sum_{k''=1}^{N_{J-2}} \left(\sum_{v_{k'}^{(J-1)} \in v_{k''}^{(J-2)}} \mathcal{S}^{(J-1)}(f, v_{k'}^{(J-1)}) w_{k'}^{(J-1)} \right) \\
&\quad \times w_{k''}^{(J-2)} \phi_\ell^{(J-2)}(v_{k''}^{(J-2)}) \\
&= \sum_{k''=1}^{N_{J-2}} \mathcal{S}^{(J-2)}(f, v_{k''}^{(J-2)}) w_{k''}^{(J-2)} \phi_\ell^{(J-2)}(v_{k''}^{(J-2)}) \\
&\cdots \\
&= \sum_{k=1}^{N_j} \mathcal{S}^{(j)}(f, v_k^{(j)}) w_k^{(j)} \phi_\ell^{(j)}(v_k^{(j)}),
\end{aligned}$$

where we recursively compute the summation to obtain the last equality, thus completing the proof. □

4.2. Fast computation for forward Haar transform Φc

The forward Haar transform in (3.5) can be computed, as follows.

Theorem 4.2. *Let $\{\phi_\ell\}_{\ell=1}^N$ be the Haar basis for a filtration $(\mathcal{G}_{J\to J_0}, W_{J\to J_0})$ of graph $\mathcal{G}$ and $\{\phi_\ell^{(j)}\}_{\ell=1}^{N_j}, j = J_0, \ldots, J$ be the associated bases at $\mathcal{G}_j$. Then, the forward Haar transform for vector $c = (c_1, \ldots, c_N) \in \mathbb{R}^N$ can be computed by, for $k = 1, \ldots, N$,*

$$(\Phi c)_k = \sum_{j=1}^{J} W_k^{(j)} \left(\sum_{\ell=N_{j-1}+1}^{N_j} c_\ell \phi_\ell^{(j)}(v_{k_j}^{(j)}) \right),$$

where for $k = 1, \ldots, N$, $v_{k_j}^{(j)}$ is the parent (ancestor) of $v_k^{(J)}$ at level j, and $W_k^{(J)} := 1$ and

$$W_k^{(j)} := \prod_{n=2}^{j} w_{k_n}^{(n)} \quad \text{for } j = J_0, \ldots, J-1, \tag{4.5}$$

where the weight factors $w_{k_n}^{(n)}$ for $n = 1, \ldots, J$ are given by (4.1).

Proof. Let $N_j := |V_j|$ for $j = J_0, \ldots, J$ and $N_{J_0-1} := 0$. For $k = 1, \ldots, N_J$, let $v_k^{(J)}$ the kth vertex of $\mathcal{G}_J$. For $i = J_0, \ldots, J-1$, there exists $k_i = 1, \ldots, N_j$ such that $v_{k_i}^{(i)}$ the parent at level i of $v_k^{(J)}$. By the property of the Haar basis, for each vector ϕ_ℓ there exists $j \in \{J_0, \ldots, J\}$ such that $\ell \in \{N_{j-1}+1, \ldots, N_j\}$, ϕ_ℓ is a constant for the vertices of $\mathcal{G}_J = \mathcal{G}$ which have the same parent at level j. Then,

$$\begin{aligned}
\phi_\ell(v_k^{(J)}) &= w_{k_{J-1}}^{(J-1)} \phi_\ell^{(J-1)}(v_{k_{J-1}}^{(J-1)}) \\
&= w_{k_{J-1}}^{(J-1)} w_{k_{J-2}}^{(J-2)} \phi_\ell^{(J-2)}(v_{k_{J-2}}^{(J-2)}) \\
&= \left(\prod_{n=J_0}^{j} w_{k_n}^{(n)} \right) \phi_\ell^{(j)}(v_{k_j}^{(j)}) \\
&= W_k^{(j)} \phi_\ell^{(j)}(v_{k_j}^{(j)}),
\end{aligned} \tag{4.6}$$

where the product of the weights in the third equality only depends upon the level j and the vertex $v_k^{(1)}$, and we have let

$$W_k^{(j)} := \prod_{n=1}^{j} w_{k_n}^{(n)}$$

in the last equality. By (4.6),

$$\begin{aligned}
\Phi(c, v_k^{(J)}) &= \sum_{\ell=1}^{N} c_\ell \phi_\ell(v_k^{(J)}) = \sum_{j=J_0}^{J} \sum_{\ell=N_{j-1}+1}^{N_j} c_\ell \phi_\ell(v_k^{(J)}) \\
&= \sum_{j=J_0}^{J} \sum_{\ell=N_{j-1}+1}^{N_j} c_\ell W_k^{(j)} \phi_\ell^{(j)}(v_{k_j}^{(j)}) \\
&= \sum_{j=J_0}^{J} W_k^{(j)} \left(\sum_{\ell=N_{j-1}+1}^{N_j} c_\ell \phi_\ell^{(j)}(v_{k_j}^{(j)}) \right),
\end{aligned}$$

thus completing the proof. □

4.3. Computational complexity analysis

Algorithm 1 gives the computational steps for evaluating $(\Phi^T f)_\ell$, $\ell = 1, \ldots, N$ in Theorem 4.1. In the first step of Algorithm 1, the total number of summations to compute all elements of Step 1 is no more than $\sum_{i=0}^{j-1} N_{i+1}$; In the second step, the total number of multiplication and summation operations is at most $2\sum_{\ell=1}^{N} C = \mathcal{O}(N)$. Here C is the constant which bounds the number of distinct values of the Haar basis (see Proposition 3.2). Thus, the total computational cost of Algorithm 1 is $\mathcal{O}(N)$.

Algorithm 1: Fast Haar Transforms: Adjoint

Input : A real-valued vector $f = (f_1, \ldots, f_N)$ on the graph $\mathcal{G}$; the Haar basis $\{\phi_\ell\}_{\ell=1}^N$ for $l_2(\mathcal{G})$ with the chain $\mathcal{G}_{J\to J_0}$ and the associated basis $\{\phi_\ell^{(j)}\}_{\ell=1}^{N_j}$ for $l_2(\mathcal{G}_j)$.

Output: The vector $\Phi^T f$ by adjoint Haar transform in (3.4) under the basis $\{\phi_\ell\}_{\ell=1}^N$.

1. Evaluate the following sums for $j = J_0, \ldots, J-1$ in (4.2) and (4.3).

   $\mathcal{S}^{(j)}(f, v_k^{(j)}), \quad v_k^{(j)} \in V_j.$

2. For each ℓ, let j be the integer such that $N_{j-1}+1 \le \ell \le N_j$, where $N_{J_0-1} := 0$. Evaluating $\sum_{k=1}^{N_j} \mathcal{S}^{(j)}(f, v_k^{(j)}) w_k^{(j)} \phi_\ell^{(j)}(v_k^{(j)})$ in (4.4) by the following two steps.

   (a) Compute the product for all $v_k^{(j)} \in V_j$:

   $$T_\ell(f, v_k^{(j)}) = \mathcal{S}^{(j)}(f, v_k^{(j)}) w_k^{(j)} \phi_\ell^{(j)}(v_k^{(j)}).$$

   (b) Evaluate sum $\sum_{k=1}^{N_j} T_\ell(f, v_k^{(j)})$.

Algorithm 2: Fast Haar Transforms: Forward

Input : A real-valued vector $c = (c_1, \ldots, c_N)$ on graph $\mathcal{G}$; the Haar basis $\{\phi_\ell\}_{\ell=1}^N$ for $l_2(\mathcal{G})$ associated with the chain $\mathcal{G}_{J\to J_0}$ and the associated orthonormal basis $\{\phi_\ell^{(j)}\}_{\ell=1}^{N_j}$ for $l_2(\mathcal{G}_j)$.

Output: The vector Φc by forward Haar transform in (3.5) under the basis $\{\phi_\ell\}_{\ell=1}^N$.

1. For each ℓ, let j be the integer such that $N_{j-1}+1 \le \ell \le N_j$, where $N_{J_0-1} := 0$. For all $k = 1, \ldots, N_j$, compute the product

   $$t_\ell(c, v_k^{(j)}) := c_\ell \phi_\ell^{(j)}(v_k^{(j)}).$$

2. For each $j = J_0, \ldots, J$, evaluate the sums

   $$s(c, v_{k_j}^{(j)}) := \sum_{\ell=N_{j-1}+1}^{N_j} t_\ell(c, v_{k_j}^{(j)}).$$

3. Compute the $W_k^{(j)}$ for $k = 1, \ldots, N$ and $j = J_0, \ldots, J-1$ by (4.5).

4. Compute the weighted sum

   $$(\Phi c)_k = \sum_{j=J_0}^{J} W_k^{(j)} s(c, v_{k_j}^{(j)}), \quad k = 1, \ldots, N.$$

Algorithm 3: Fast Haar Convolution

Input : Real-valued vectors $g := (g_1, \ldots, g_N)$ and $f := (f_1, \ldots, f_N)$ on $\mathcal{G}$; chain $\mathcal{G}_{J_0\to J}$ of graph $\mathcal{G}$ where $\mathcal{G}_J := \mathcal{G}$.

Output: Haar convolution $g \star f$ of g and f as given by Definition 3.3.

1. Compute the adjoint Haar transforms $\Phi^T g$ and $\Phi^T f$ by Algorithm 1.
2. Compute the point-wise product of $\Phi^T g$ and $\Phi^T f$.
3. Compute the forward Haar transform of $(\Phi^T g) \odot (\Phi^T f)$ by Algorithm 2.

By Theorem 4.2, the evaluation of the forward Haar transform Φc can be implemented by Algorithm 2. In the first step of Algorithm 2, the number of multiplications is no more than $\sum_{\ell=1}^N C = \mathcal{O}(N)$; in the second step, the number of summations is no more than $\sum_{\ell=1}^N C = \mathcal{O}(N)$; in the third step, the computational steps are $\mathcal{O}(N(\log N)^2)$; in the last step, the total number of summations and multiplications is $\mathcal{O}(N \log N)$. Thus, the total computational cost of Algorithm 2 is $\mathcal{O}(N(\log N)^2)$.

Hence, Algorithms 1 and 2 have linear computational cost (up to a $\log N$ term). We call these two algorithms *fast Haar transforms (FHTs)* under Haar basis on the graph.

Proposition 4.3. *The adjoint and forward Haar Transforms in Algorithms 1 and 2 are invertible in that for any vector f on graph $\mathcal{G}$, $f = \Phi(\Phi^T f)$.*

Proposition 4.3 shows that the forward Haar transform can recover graph data f from the adjoint Haar transform $\Phi^T f$. This means that forward and adjoint Haar transforms have zero-loss in graph data transmission.

Haar convolution, which computational strategy given by Algorithm 3, can be evaluated fast by FHTs in Algorithms 1 and 2. From the above discussion, the total computational cost of Algorithm 3 is $\mathcal{O}(N(\log N)^2)$. That is, using FHTs, we can evaluate Haar convolution in near-linear computational complexity.

5. Graph neural networks with Haar transforms

5.1. Models

The Haar convolution in (3.6) can be applied to any architecture of graph neural network. For graph classification and graph-based regression tasks, we use the model with convolutional layer consisting of m-hidden neutrons and a non-linear activation function σ (e.g. ReLU): for $i = 1, 2 \ldots, m$,

$$f_i^{\text{out}} = \sigma\left(\sum_{j=1}^d \Phi\left(g_{i,j} \odot (\Phi^T f_j^{\text{in}})\right)\right) = \sigma\left(\sum_{j=1}^d \Phi G_{i,j} \Phi^T f_j^{\text{in}}\right), \tag{5.1}$$

for input graph data $F^{\text{in}} = (f_1^{\text{in}}, f_2^{\text{in}}, \ldots, f_d^{\text{in}}) \in R^{N\times d}$ with N nodes and d input features (for each vertex). Here, the feature f_j^{in} of the input graph data is convolved with the learnable filter $g_{i,j} \in \mathbb{R}^N$ by Haar transforms, and then all Haar-transformed features are fused as a new feature f_i^{out}. This gives the output matrix $F^{\text{out}} = (f_1^{\text{out}}, f_2^{\text{out}}, \ldots, f_m^{\text{out}}) \in \mathbb{R}^{N\times m}$. If we write $G_{i,j} \in \mathbb{R}^{N\times N}$ as the diagonal matrix of filter $g_{i,j}$, the convolutional layer has the compact form of the second equality in (5.1). We call the GNN with Haar convolution in (5.1) *HANet*.

Weight detaching. For each layer, $O(Ndm)$ parameters need to be tuned. To reduce the number of parameters, we can replace the filter matrix $G_{i,j}$ by a unified diagonal filter matrix G and a compression matrix $W \in R^{d\times m}$ (which is a detaching approach used in conventional CNN for extracting features). This then leads to a concise form

$$F^{\text{out}} = \sigma\left(\Phi\left(G(\Phi^T F^{\text{in}})\right)W\right). \tag{5.2}$$

Then, it requires $\mathcal{O}(N + dm)$ parameters to train. Recall that constructing the Haar basis uses a chain $\mathcal{G}_{J\to J_0}$ for the graph $\mathcal{G}$, one can implement weight sharing based on the same chain structure. Correctly, one can use k-means clustering algorithm or METIS algorithm (Karypis & Kumar, 1998) to generate a chain, which captures clustering feature of the graph. Suppose a coarser level J_1 ($J_0 \le J_1 < J$) having K clusters, then all vertices in the same cluster share the common filter parameter. The corresponding children vertices in level $J_1 - 1$ share the same filter parameters

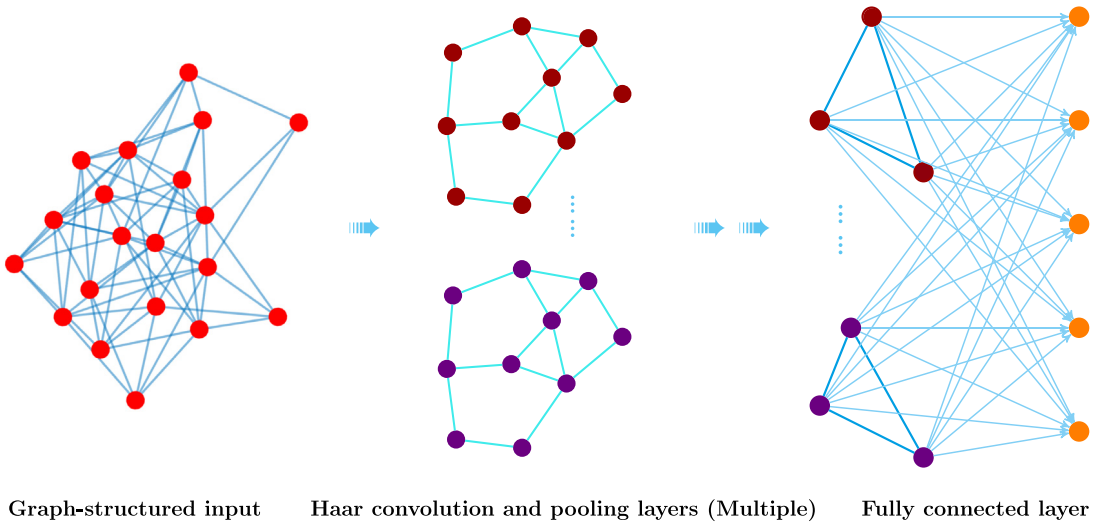


Fig. 2. Network architecture of HANet with multiple Haar convolutional layers and then fully connected by softmax.

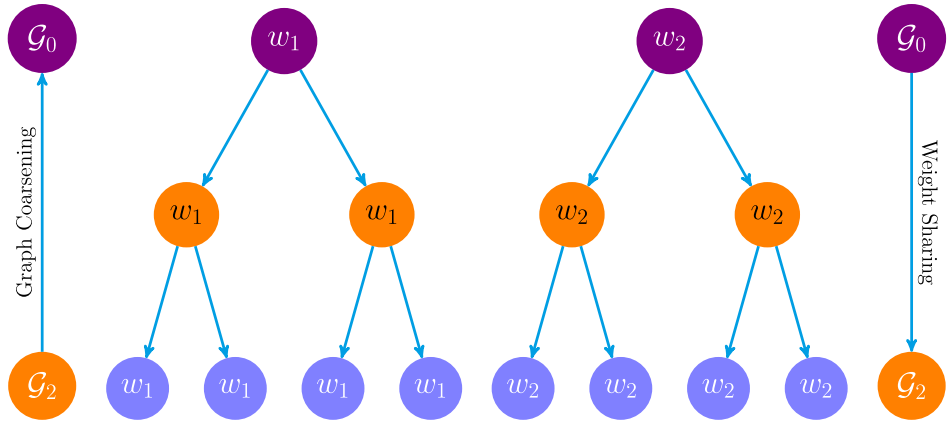


Fig. 3. Weight sharing for Haar convolution and graph coarsening for graph pooling for the chain $\mathcal{G}_{2\to 0}$.

as used in their parent vertices, and the bottom level corresponds to the whole set of vertices of the input graph. Thus, the number of parameters is reduced to $\mathcal{O}(K + dm)$.

The HANet uses d times fast Haar convolutions (consisting of d-times adjoint and forward Haar transforms). The computational cost of Haar convolution in HANet is then $\mathcal{O}(N(\log N)^2 d)$. Deep GNNs with Haar convolution are built by stacking up multiple Haar convolutional layers of (5.2), followed by an output layer (see Fig. 3).

HANet for graph classification and graph-based regression. These two tasks can be formulated as the following supervised learning: For a collection of graph-structured data, $\{f_i\}_{i=1}^n$ with labels $\{y_i\}_{i=1}^n$, the classification task is to find a mapping that can classify or regress labels. The model of HANet uses a similar architecture of the canonical deep convolutional neural network. It has several convolutional layers with Haar convolution and fully connected dense layers. Fig. 2 shows the flowchart for the architecture of HANet with multiple Haar convolutional layers: the chain $\mathcal{G}_{J\to J_0}$ and the Haar basis ϕ_ℓ and the associated basis $\phi_\ell^{(j)}$, $j = J_0, \ldots, J$ are pre-computed; graph-structured input f is Haar-convoluted with filter g which is of length N_J but with N_{J-1} independent parameters, where g is expanded from level $J-1$ to J by weight sharing, and the output f^{out} of the first layer is the ReLU of the Haar convolution of g and f; the graph pooling reduces f^{out} of size N_J to $\widetilde{f}^{\text{in}}$ of size N_{J-1}; and in the second Haar convolutional layer, the input is $\widetilde{f}^{\text{in}}$ and the Haar basis is $\phi_\ell^{(J-1)}$; the following layers continue this process; the final Haar convolutional layer is fully connected by one or multiple dense layers. For classification, an additional dense layer with softmax function is used.

HANet for node classification. In node classification, the whole graph is the only single input data, where a fractional proportion of nodes are labeled. The output is the whole graph with all unknown labels predicted. Here we use the following GNN with two layers.

$$\text{HANet}(f^{\text{in}}) := \text{softmax}\Big(\text{HC}^{(2)}\big(\text{ReLU}\big(\text{HC}^{(1)}(f^{\text{in}})\big)\big)\Big) \tag{5.3}$$

where $\text{HC}^{(1)}$ and $\text{HC}^{(2)}$ are the Haar convolutional layers

$$\text{HC}^{(i)}(f) := \widehat{A}(w_1^{(i)} \star f)w_2^{(i)}, \quad i = 1, 2,$$

where we use the modified Haar convolution $w_1^{(i)} \star f = \Phi\big(w_1^{(i)} \odot (\Phi^T f)\big)$. For a graph with N nodes and M features, in the first Haar convolutional layer, the filter $w_1^{(1)}$ contains $N_0 \times M$ parameters and is extended to a matrix $N \times M$ by weight sharing, where N_0 is the number of nodes at the coarsest level. The $w_2^{(1)}$ plays the role of weight compression and feature extraction. The rectifier activates the first layer, and the second layer is fully connected with softmax. The $\widehat{A}$, as defined in Kipf and Welling (2017), is the square matrix of size N determined by the adjacency matrix of the input graph. This smoothing operation compensates for

the information loss in coarsening by taking a weighted average of features of each vertex and its neighbors. For vertices that are densely connected, it makes their features more similar and significantly improves the ease of node classification task (Li, Han, & Wu, 2018).

5.2. Technical components

Fast computation for HANet. Complexity analysis of FHTs above shows that HANet is more efficient than GNNs with a graph Fourier basis. The graph convolution of the latter incurs $\mathcal{O}(N^3)$ computational cost. Researchers have proposed different methods to improve the computational performance for graph convolution. For example, ChebNet (Defferrard et al., 2016) and GCN (Kipf & Welling, 2017) use a localized polynomial approximation for the spectral filters; GWNN (Xu, Shen et al., 2019) constructs sparse and localized graph wavelet basis matrix for graph convolution. These methods implement the multiplication between a sparse matrix (e.g., the refined adjacency matrix $\hat{A}$ in GCN or the wavelet basis matrix ψ_s in GWNN and input matrix F in the convolutional layer. However, to compute either $\hat{A}F$ or $\psi_s F$, the computational complexity, which is roughly proportional to $\mathcal{O}(\varepsilon N^2 d)$, to a great extent relies on the sparse degree of $\hat{A}$ or ψ_s, where ε, $\varepsilon \in [0, 1]$, represents the percentage of non-zero elements in a square matrix. The $\mathcal{O}(\varepsilon N^2 d)$ may be significantly higher than $\mathcal{O}(N(\log N)^2 d)$ as long as ε is not extremely small, indicating that our FHTs outperform these methods especially when N is quite large and $\varepsilon \approx 1$. Also, the fast computation for sparse matrix multiplication (Golub & Van Loan, 2012) can further speed up the evaluation of Haar convolution. One can develop a strategy of sparse FHTs using the method in Hassanieh et al. (2012) and Indyk et al. (2014).

Chain. In HANet, the chain and the Haar basis can be pre-computed since the graph structure is already known. In particular, the chain is computed by a modified version of the METIS algorithm, which fast generates a chain for the weight matrix of a graph. In many cases, the parents of a chain from METIS have at least two children, which means the weighted chain is a filtration, and thus Proposition 3.2 applies.

Weight sharing for the filter. In the HANet, one can use weight sharing given in Section 3.3 for filters. By doing this, we utilize the local topological property of the graph-structured data to extract the co-feature of neighbor nodes and meanwhile reduce the independent parameters of the filter. One can add weight sharing in each convolutional layer of the HANet. For chain $\mathcal{G}_{J\to J_0}$ with which the Haar basis is associated, weight sharing can act from the coarsest level J_0 to the finest level J or from any level coarser than J to J. For a given filtration, the weight sharing shrinks the number of parameters by at least rate $2^{-(J-J_0)}$, see Fig. 3.

Graph pooling. We use max graph pooling between two convolutional layers of the HANet. Each pooled input is the maximum over children nodes of each node of the current layer of the chain. The pooling applies the same chain as the Haar basis at the same layer. For example, after pooling, the second layer uses the chain $\mathcal{G}_{(J-1)\to J_0}$, as illustrated in Fig. 2. By the construction of Haar basis in Section 3.2, the new Haar basis associated with $\mathcal{G}_{(J-1)\to J_0}$ is exactly the pre-computed basis $\{\phi_\ell^{(J-1)}\}_{\ell=1}^{N_{J-1}}$.

Table 1
Test mean absolute error (MAE) comparison on QM7.

Method	Test MAE
RF (Breiman, 2001)	122.7 ± 4.2
Multitask (Ramsundar et al., 2015)	123.7 ± 15.6
KRR (Cortes & Vapnik, 1995)	110.3 ± 4.7
GC (Altae-Tran, Ramsundar, Pappu, & Pande, 2017)	77.9 ± 2.1
Multitask(CM) (Wu et al., 2018)	10.8 ± 1.3
KRR(CM) (Wu et al., 2018)	10.2 ± 0.3
DTNN (Schütt, Arbabzadah, Chmiela, Müller, & Tkatchenko, 2017)	8.8 ± 3.5
ANI-1 (Smith, Isayev, & Roitberg, 2017)	2.86 ± 0.25
HANet	$\mathbf{9.50 \pm 0.71}$

6. Experiments

In this section, we test the proposed HANet on Quantum Chemistry (graph-based regression) and Citation Networks (node classification). The experiments for graph classification were carried out under the Google Colab environment with Tesla K80 GPU, while for node classification were under the UNIX environment with a 3.3 GHz Intel Core i7 CPU and 16 GB RAM. We implement all the methods in TensorFlow and use SGD+Momentum and Adam optimization methods in the experiments.

6.1. Quantum chemistry for graph-based regression

We test HANet on QM7 (Blum & Reymond, 2009; Rupp, Tkatchenko, Müller, & von Lilienfeld, 2012), which contains 7165 molecules. Each molecule is represented by the Coulomb (energy) matrix and its atomization energy. We treat each molecule as a weighted graph where the nodes are the atoms, and the adjacency matrix is the 23×23-Coulomb matrix of the molecule, where the exact number of atoms may be less than 23. The atomization energy of the molecule is the label. As in most cases, the adjacency matrix is not fully ranked; we take the average of the Coulomb matrices of all molecules as the common adjacency matrix, for which we generate the Haar basis. To avoid exploding gradients in parameter optimization, we take the standard score of each entry over all Coulomb matrices as input (see Table 1).

The network architecture of HANet contains 2 layers of Haar convolution with 8 and 2 filters and then 2 fully connected layers with 400 and 100 neurons. As the graph is not big, we do not use graph pooling or weight sharing. Following Gilmer et al. (2017), we use mean squared error (MSE) plus ℓ_2 regularization as the loss function in training and mean absolute error (MAE) as the test metric. We repeat the experiment over 5 splits with the same proportion of training and test data but with different random seeds. In Table 1, we report the average performance and standard deviation for the HANet compared against other public results (Wu et al., 2018) by methods Random Forest (RF) (Breiman, 2001), Multitask Networks (Multitask) (Ramsundar et al., 2015), Kernel Ridge Regression (KRR) (Cortes & Vapnik, 1995), Graph Convolutional models (GC) (Altae-Tran et al., 2017), Deep Tensor Neural Network (DTNN) (Schütt et al., 2017), ANI-1 (Smith et al., 2017), KRR and Multitask with Coulomb Matrix featurization (KRR(CM)/Multitask(CM)) (Wu et al., 2018). It shows that HANet ranks third in the list with average test MAE 9.50 and average relative MAE 4.31×10^{-6}, which offers a good approximator for QM7 regression.

Table 2
Sparsity of Haar basis and CPU time for basis generating, adjoint FHT (AFHT) and forward FHT (FFHT) on citation networks data set.

Dataset	Basis size	Sparsity	Generating time (s)	AFHT time (s)	FFHT time (s)
Citeseer	3327	99.58%	1.93509	0.05276	0.05450
Cora	2708	98.84%	0.86429	0.06908	0.05515
Pubmed	19717	99.84%	62.67185	1.08775	1.55694

Table 3
Test accuracy comparison on citation networks (%).

Method	Citeseer	Cora	Pubmed
MLP (Kipf & Welling, 2017)	55.1	46.5	71.4
ManiReg (Belkin et al., 2006)	60.1	59.5	70.7
SemiEmb (Weston, Ratle, Mobahi, & Collobert, 2012)	59.6	59.0	71.1
LP (Zhu, Ghahramani, & Lafferty, 2003)	45.3	68.0	63.0
DeepWalk (Perozzi, Al-Rfou, & Skiena, 2014)	43.2	67.2	65.3
ICA (Lu & Getoor, 2003)	69.1	75.1	73.9
Planetoid (Yang, Cohen, & Salakhutdinov, 2016)	64.7	75.7	77.2
ChebNet (Defferrard et al., 2016)	69.8	81.2	74.4
GCN (Kipf & Welling, 2017)	70.3	81.5	79.0
HANet	**70.1**	**81.9**	**79.3**

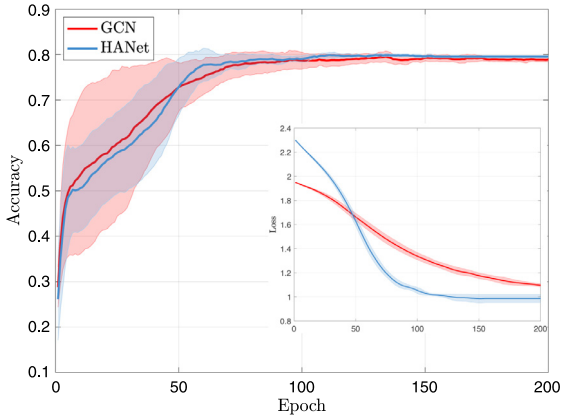


Fig. 4. Main figure: Mean and standard deviation of validation accuracies of HANet and GCN on Cora with epoch $\leq$ 200. Figure in the lower right corner: Validation loss function of HANet and GCN.

6.2. Citation networks for node classification

We test the model in (5.3) on citation networks Citeseer, Cora and Pubmed, following the experimental setup of Kipf and Welling (2017) and Yang et al. (2016). The Citeseer, Cora and Pubmed are 6, 7 and 3 classification problems with nodes 3327, 2708 and 19717, edges 4732, 5429 and 44338, features 3703, 1433 and 500, and label rates 0.036, 0.052 and 0.003, respectively. In Table 3, we compare the performance of the model (5.3) of HANet with methods Multilayer Perceptron (MLP), Manifold Regularization (ManiReg) (Belkin et al., 2006), Semi-supervised Embedding (SemiEmb) (Weston et al., 2012), Traditional Label Propagation (LP) (Zhu et al., 2003), DeepWalk (Perozzi et al., 2014), Link-based Classification (ICA) (Lu & Getoor, 2003), Planetoid (Yang et al., 2016), ChebNet (Defferrard et al., 2016) and GCN (Kipf & Welling, 2017). We repeat the experiment 10 times with different random seeds and report the average test accuracy of HANet. As shown in Table 3, HANet has the top test accuracies on Cora and Pubmed and ranks second on Citeseer.

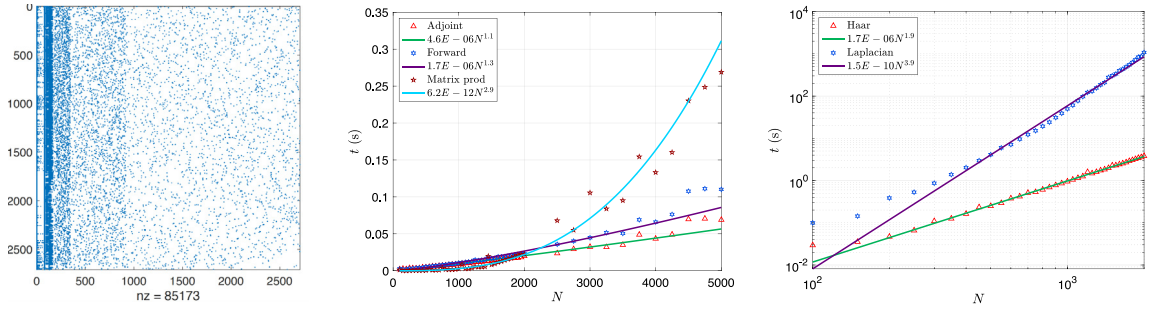
Fig. 4 shows the mean and standard deviation of validation accuracies and the validation loss with up to epoch 200 of HANet and GCN. HANet achieves slightly higher max accuracy as well as smaller standard deviation, and the loss also converges faster than GCN.

6.3. Haar basis and FHTs

In Fig. 5a, we show the matrix of the Haar basis vectors for Cora, which has sparsity (i.e., the proportion of zero entries) 98.84%. The associated chain $\mathcal{G}_{10\to 0}$ has 2708, 928, 352, 172, 83, 41, 20, 10, 5, 2, 1 nodes from level 10 to 0. Fig. 5b shows the comparison of time for FHTs with direct matrix products. It illustrates that FHTs have nearly linear computational cost while the cost of matrix product grows at $\mathcal{O}(N^3)$ for a graph of size N. Fig. 5c shows the comparison of time for generating the Haar basis and the basis for graph Laplacian: Haar basis needs significantly less time than that for graph Laplacian. Table 2 gives the sparsity (i.e., the proportion of zero entries) and the CPU time for generating Haar basis and FHTs on three datasets. All sparsity values for three datasets are very high (around 99%), and the computational cost of FHTs is proportional to N.

7. Conclusion

We introduce a Haar basis, and the Haar transforms on a coarse-grained chain on the graph. From Haar transforms, we define the Haar convolution for GNNs, which has a fast implementation because of the sparsity of the Haar transform matrix. Haar convolution gives a sparse representation of graph data and captures the geometric property of the graph data, and thus provides a useful graph convolution for any architecture of GNN. Overall, as a spectral-based method for graph convolution, our proposed HANet can effectively bypass the inherent drawbacks of the spectral graph convolution based on graph Laplacian, i.e., the vast computational load caused by eigendecomposition of the graph Laplacian to obtain the graph Fourier base matrix, and the



(a) Haar transform matrix Φ (b) CPU time of FFTs (c) CPU time of generating basis

Fig. 5. (a) Haar basis Φ for Cora with a chain of 11 levels (by METIS). Each column is a vector of the Haar basis. The sparsity of the Haar transform matrix is 98.84% (i.e. the proportion of zero entries). (b) Comparison of CPU time for FHTs and Direct Matrix Product for the Haar basis for graphs with nodes $\leq$ 5000. (c) Comparison of CPU time for generating the orthonormal bases for Haar and graph Laplacian on graphs with nodes $\leq$ 2000.

dense matrix multiplication computed in performing the graph Fourier transform.

There are many new problems for future probing. We are making efforts to extend our current framework with careful consideration of local receptive fields (as typically concerned in the traditional CNN), aiming to fill the gap between the spectral-based method (for example, our proposed HANet) and the spatial method. It is also of practical significance to improve the HANet, so it is capable of dealing with large-scale and heterogeneous graphs in various applications. Furthermore, one would expect the theory for the expressive power and generalization capability of the HANet.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors thank the associated editor and reviewers for their constructive comments and suggestions. Ming Li acknowledges support by the National Natural Science Foundation of China under Grant 61802132 and Grant 61877020, and the China Post-Doctoral Science Foundation under Grant 2019T120737. Yu Guang Wang acknowledges support from the Australian Research Council under Discovery Project DP180100506. This work is supported by the National Science Foundation, USA under Grant No. DMS-1439786 while Zheng Ma and Yu Guang Wang were in residence at the Institute for Computational and Experimental Research in Mathematics in Providence, RI, during Collaborate@ICERM 2019. Xiaosheng Zhuang acknowledges support by Research Grants Council of Hong Kong (Project No. CityU 11301419).

References

Abu-El-Haija, S., Kapoor, A., Perozzi, B., & Lee, J. (2018). N-GCN: Multi-scale graph convolution for semi-supervised node classification. ArXiv preprint arXiv:1802.08888.

Altae-Tran, H., Ramsundar, B., Pappu, A. S., & Pande, V. (2017). Low data drug discovery with one-shot learning. *ACS Central Science*, *3*(4), 283–293.

Atwood, J., & Towsley, D. (2016). Diffusion-convolutional neural networks. In *NIPS* (pp. 1993–2001).

Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., et al. (2018). Relational inductive biases, deep learning, and graph networks. ArXiv preprint arXiv:1806.01261.

Belkin, M., Niyogi, P., & Sindhwani, V. (2006). Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research (JMLR)*, *7*(Nov), 2399–2434.

Blum, L. C., & Reymond, J.-L. (2009). 970 million druglike small molecules for virtual screening in the chemical universe database GDB-13. *Journal of the American Chemical Society*, *131*, 8732.

Breiman, L. (2001). Random forests. *Machine Learning*, *45*(1), 5–32.

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

Bruna, J., Zaremba, W., Szlam, A., & Lecun, Y. (2014). Spectral networks and locally connected networks on graphs. In *ICLR*.

Chen, J., Ma, T., & Xiao, C. (2018). FastGCN: fast learning with graph convolutional networks via importance sampling. In *ICLR*.

Chen, J., Zhu, J., & Song, L. (2018). Stochastic training of graph convolutional networks with variance reduction. In *ICML* (pp. 941–949).

Chui, C., Filbir, F., & Mhaskar, H. (2015). Representation of functions on big data: graphs and trees. *Applied and Computational Harmonic Analysis*, *38*(3), 489–509.

Chui, C. K., Mhaskar, H., & Zhuang, X. (2018). Representation of functions on big data associated with directed graphs. *Applied and Computational Harmonic Analysis*, *44*(1), 165–188.

Chung, F. R., & Graham, F. C. (1997). *Spectral graph theory*. American Mathematical Society.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, *20*(3), 273–297.

Daubechies, I. (1992). *Ten lectures on wavelets*. SIAM.

Defferrard, M., Bresson, X., & Vandergheynst, P. (2016). Convolutional neural networks on graphs with fast localized spectral filtering. In *NIPS* (pp. 3844–3852).

Gavish, M., Nadler, B., & Coifman, R. R. (2010). Multiscale wavelets on trees, graphs and high dimensional data: theory and applications to semi supervised learning. In *ICML* (pp. 367–374).

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., & Dahl, G. E. (2017). Neural message passing for quantum chemistry. In *ICML* (pp. 1263–1272).

Golub, G. H., & Van Loan, C. F. (2012). *Matrix computations*. JHU Press.

Haar, A. (1910). Zur theorie der orthogonalen funktionensysteme. *Mathematische Annalen*, *69*(3), 331–371.

Hamilton, W., Ying, Z., & Leskovec, J. (2017). Inductive representation learning on large graphs. In *NIPS* (pp. 1024–1034).

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

Hassanieh, H., Indyk, P., Katabi, D., & Price, E. (2012). Simple and practical algorithm for sparse Fourier transform. In *Proceedings of the 23rd annual ACM-SIAM symposium on discrete algorithms* (pp. 1183–1194).

Henaff, M., Bruna, J., & LeCun, Y. (2015). Deep convolutional networks on graph-structured data. ArXiv preprint arXiv:1506.05163.

Indyk, P., Kapralov, M., & Price, E. (2014). (Nearly) sample-optimal sparse Fourier transform. In *Proceedings of the 25th annual ACM-SIAM symposium on discrete algorithms* (pp. 480–499).

Karypis, G., & Kumar, V. (1998). A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, *20*(1), 359–392.

Kipf, T. N., & Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In *ICLR*.

Li, Q., Han, Z., & Wu, X.-M. (2018). Deeper insights into graph convolutional networks for semi-supervised learning. In *AAAI* (pp. 3538–3545).

Li, Y., Tarlow, D., Brockschmidt, M., & Zemel, R. (2016). Gated graph sequence neural networks. In *ICLR*.

Liao, R., Zhao, Z., Urtasun, R., & Zemel, R. S. (2019). LanczosNet: Multi-scale deep graph convolutional networks. In *ICLR*.

Lu, Q., & Getoor, L. (2003). Link-based classification. In *ICML* (pp. 496–503).

Ma, Z., Li, M., & Wang, Y. G. (2019). PAN: Path integral based convolution for deep graph neural networks. In *ICML workshop on learning and reasoning with graph-structured representations*.

Monti, F., Boscaini, D., Masci, J., Rodola, E., Svoboda, J., & Bronstein, M. M. (2017). Geometric deep learning on graphs and manifolds using mixture model CNNs. In *CVPR* (pp. 5425–5434).

Perozzi, B., Al-Rfou, R., & Skiena, S. (2014). DeepWalk: Online learning of social representations. In *KDD* (pp. 701–710).

Ramsundar, B., Kearnes, S., Riley, P., Webster, D., Konerding, D., & Pande, V. (2015). Massively multitask networks for drug discovery. ArXiv preprint arXiv:1502.02072.

Rupp, M., Tkatchenko, A., Müller, K.-R., & von Lilienfeld, O. A. (2012). Fast and accurate modeling of molecular atomization energies with machine learning. *Physical Review Letters*, *108*, 058301.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., & Monfardini, G. (2009). The graph neural network model. *IEEE Transactions on Neural Networks*, *20*(1), 61–80.

Schütt, K. T., Arbabzadah, F., Chmiela, S., Müller, K. R., & Tkatchenko, A. (2017). Quantum-chemical insights from deep tensor neural networks. *Nature Communications*, *8*, 13890.

Smith, J. S., Isayev, O., & Roitberg, A. E. (2017). ANI-1: An extensible neural network potential with DFT accuracy at force field computational cost. *Chemical Science*, *8*(4), 3192–3203.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., & Bengio, Y. (2018). Graph attention networks. In *ICLR*.

Wang, Y. G., Li, M., Ma, Z., Montufar, G., Zhuang, X., & Fan, Y. (2019). Haar graph pooling. ArXiv preprint arXiv:1909.11580.

Wang, Y. G., & Zhuang, X. (2019). Tight framelets on graphs for multiscale data analysis. In *Proceedings of wavelets and sparsity XVIII, Vol. 11138* (p. 111380B). International Society for Optics and Photonics.

Wang, Y. G., & Zhuang, X. (2020). Tight framelets and fast framelet filter bank transforms on manifolds. *Applied and Computational Harmonic Analysis*, *48*(1), 64–95.

Weston, J., Ratle, F., Mobahi, H., & Collobert, R. (2012). Deep learning via semi-supervised embedding. In *Neural networks: Tricks of the trade* (pp. 639–655).

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., & Yu, P. S. (2020). A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, [ISSN: 2162-2388] 1–21. http://dx.doi.org/10.1109/TNNLS.2020.2978386.

Wu, Z., Ramsundar, B., Feinberg, E. N., Gomes, J., Geniesse, C., Pappu, A. S., et al. (2018). MoleculeNet: A benchmark for molecular machine learning. *Chemical Science*, *9*(2), 513–530.

Wu, F., Zhang, T., Souza Jr, A. H. d., Fifty, C., Yu, T., & Weinberger, K. Q. (2019). Simplifying graph convolutional networks. In *ICML*.

Xu, K., Hu, W., Leskovec, J., & Jegelka, S. (2019). How powerful are graph neural networks?. In *ICLR*

Xu, B., Shen, H., Cao, Q., Qiu, Y., & Cheng, X. (2019). Graph wavelet neural network. In *ICLR*.

Yang, Z., Cohen, W. W., & Salakhutdinov, R. (2016). Revisiting semi-supervised learning with graph embeddings. In *ICML* (pp. 40–48).

Yang, Y., Wang, X., Song, M., Yuan, J., & Tao, D. (2019). SPAGAN: Shortest path graph attention network. In *IJCAI*.

Zhang, Z., Cui, P., & Zhu, W. (2020). Deep learning on graphs: a survey. *IEEE Transactions on Knowledge and Data Engineering*, [ISSN: 1558-2191] http://dx.doi.org/10.1109/TKDE.2020.2981333, 1-1.

Zhou, J., Cui, G., Zhang, Z., Yang, C., Liu, Z., & Sun, M. (2018). Graph neural networks: A review of methods and applications. ArXiv preprint arXiv:1812.08434.

Zhu, X., Ghahramani, Z., & Lafferty, J. D. (2003). Semi-supervised learning using Gaussian fields and harmonic functions. In *ICML* (pp. 912–919).