How Framelets Enhance Graph Neural Networks

Xuebin Zheng [*1] **Bingxin Zhou** [*1] **Junbin Gao** [1] **Yu Guang Wang** [2 3 4]
Pietro Liò [5] **Ming Li** [6] **Guido Montúfar** [7 2]

Abstract

This paper presents a new approach for assembling graph neural networks based on framelet transforms. The latter provides a multi-scale representation for graph-structured data. We decompose an input graph into low-pass and high-pass frequencies coefficients for network training, which then defines a framelet-based graph convolution. The framelet decomposition naturally induces a graph pooling strategy by aggregating the graph feature into low-pass and high-pass spectra, which considers both the feature values and geometry of the graph data and conserves the total information. The graph neural networks with the proposed framelet convolution and pooling achieve state-of-the-art performance in many node and graph prediction tasks. Moreover, we propose shrinkage as a new activation for the framelet convolution, which thresholds high-frequency information at different scales. Compared to ReLU, shrinkage activation improves model performance on denoising and signal compression: noises in both node and structure can be significantly reduced by accurately cutting off the high-pass coefficients from framelet decomposition, and the signal can be compressed to less than half its original size with well-preserved prediction performance.

[*]Equal contribution [1]The University of Sydney Business School, The University of Sydney, Camperdown, NSW 2006, Australia. [2]Max Planck Institute for Mathematics in the Sciences, Leipzig, Germany. [3]Institute of Natural Sciences and School of Mathematical Sciences, Shanghai Jiao Tong University, China. [4]School of Mathematics and Statistics, The University of New South Wales, Sydney, Australia. [5]Department of Computer Science and Technology, University of Cambridge, Cambridge, United Kingdom. [6]Key Laboratory of Intelligent Education Technology and Application of Zhejiang Province, Zhejiang Normal University, Jinhua, China. [7]Department of Mathematics and Department of Statistics, University of California, Los Angeles, United States. Correspondence to: Bingxin Zhou <bzho3923@uni.sydney.edu.au>, Yu Guang Wang <yuguang.wang@mis.mpg.de>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

1. Introduction

Graph neural networks (GNNs) are a powerful deep learning method for prediction tasks on graph-structured data (Wu et al., 2021). Most existing GNN models are spatial-based, such as GCN (Kipf & Welling, 2017), GAT (Veličković et al., 2018) and GIN (Xu et al., 2019b). These methods compute graph convolution on vertices and edges in the form of message passing (Gilmer et al., 2017), but leave the signal frequency of graph data unexplored. In this paper, we seek to exploit signal processing for GNNs. Graph framelets (Dong, 2017; Zheng et al., 2021), akin to traditional wavelets, provide a multiresolution analysis (MRA) for graph signals. The fully tensorized *framelet transforms* guarantee an efficient design of graph convolution that combines low-pass and high-pass information, where the transforms only require graph Laplacian, Chebyshev polynomial approximation, and filter banks. We propose *framelet convolution* that exploits the decomposition and reconstruction procedures of framelet transforms, and the network learns in the frequency domain.

The wavelet-like graph data analysis allows us to exploit traditional tools from signal processing. An effective practice is the *shrinkage* that thresholds high-pass coefficients in the framelet representation. The multi-scale property of framelet convolution introduces diffusion for GNNs. The associated shrinkage defines a new type of activation that adapts to the diffusion scales in nonlinear feature transformation and extraction. Shrinkage in framelet convolution provides a mechanism for effective noise reduction of input graph data, where noise may appear in node and/or edge features. This ability is inherited from the traditional wavelet denoising model. Moreover, the shrinkage activation provides a way to compress graph signals. The shrinkage trims coefficients and significantly compresses the graph data representation while at the same time the underlying GNN reaches state-of-the-art performance in multiple tasks. The framelet MRA and shrinkage threshold provide GNNs with multi-scale and compression characteristics, which distinguishes our model from existing graph convolution methods.

The framelet transform naturally induces a graph pooling strategy by aggregating the different scales of framelet features. The consequent framelet pooling conserves the

total information due to energy conservation of framelet spectra, and offers an efficient graph dimensionality reduction. Framelet-based GNNs outperform existing spatial or spectral-based methods on benchmark node and graph prediction tasks. Moreover, the framelet convolution with ReLU or shrinkage can achieve excellent performance in denoising both node features and graph structure. This behavior suggests that the framelets play an important role in bridging signal processing and graph neural networks.

2. Related Works

Graph Framelets and Transforms The construction of wavelet analysis on graphs was first explored by Crovella & Kolaczyk (2003). Maggioni & Mhaskar (2008) used polynomials of a differential operator to build multi-scale transforms. The spectral graph wavelet transforms (Hammond et al., 2011) define the graph spectrum from a graph's Laplacian matrix, where the scaling function is approached by the Chebyshev approximation. Behjat et al. (2016) encoded energy spectral density to design tight frames on graphs with both graph topology and signal features. Dong (2017) approximated piece-wise smooth functions with undecimated tight wavelet frames. Fast decomposition and reconstruction become possible with the filtered Chebyshev polynomial approximation and proper design of filter banks.

The other regime of signal processing on graph data is backboned by the multiresolution analysis (Mallat, 1989) that establishes a tree-based wavelet system with localization properties. Crovella & Kolaczyk (2003) defined the 'h-hop' neighborhoods on binary graphs and Gavish et al. (2010) constructed Haar-like bases. The Haar-like orthonormal wavelet system (Chui et al., 2015) has been applied to deep learning tasks on undirected graphs (Wang et al., 2020b; Li et al., 2020b; Zheng et al., 2020). Meanwhile, fast tight framelet filter bank transforms on quadrature-based framelets are explored on graph domain (Wang & Zhuang, 2019; Zheng et al., 2021) and manifold space (Wang & Zhuang, 2018) with a low redundancy rate.

Graph Convolution and Graph Pooling The theory of graph convolution (Bruna et al., 2014) facilitated the later development of advanced deep learning methods. For example, spectral-based GNNs (Defferrard et al., 2016a; Xu et al., 2019a; Li et al., 2020b; Balcilar et al., 2021) transform graph signals to the spectral domain and process them with filter operations. Alternatively, spatial-based graph convolution performs node property prediction via aggregating feature information over neighborhood nodes (Kipf & Welling, 2017; Gilmer et al., 2017; Wang et al., 2020a; Vignac et al., 2020; Chen et al., 2020). For graph property prediction, one pursues topology-aware graph embedding via graph pooling. Some global pooling strategies refine vertex features in one-shot (Zhang et al., 2018b; Lee et al., 2019), while others process graph information hierarchically (Cangea et al., 2018; Gao & Ji, 2019; Knyazev et al., 2019; Wang et al., 2020b; Ma et al., 2020).

Signal Compression and Denoising Signal compression is critical for high-speed signal transmission. Wavelets play an important role in compressing signal and have contributed to the prevalent JPEG 2000 (ISO, 2019). Our shrinkage framelet convolution provides an algorithm for compressing graph signals. Denoising problems have long been studied in image processing. Many models have been proposed for image restoration (Milanfar, 2013). In particular, wavelets provide a sparse and multi-scale representation for images and have proved an impressive regularizer for reducing Gaussian white noise for signals in 2D (Figueiredo & Nowak, 2003; Cai et al., 2012; Dong & Shen, 2013; Shen, 2010). Graph spectral theory and graph wavelets have been widely used for image processing (Cheung et al., 2018). Our convolution uses graph framelets and provides a solution to the denoising model for structured data using GNN training.

3. Multiresolution Analysis of Graph Framelets

Our convolution uses the undecimated graph framelets and their transforms, which were introduced by Dong (2017, Section 3); Zheng et al. (2021, Section 4.1). Framelets on a specific graph $\mathcal{G} = (V, E, \omega) \in L_2(\mathcal{G})$ are defined by a *filter bank* and the spectrum of its graph Laplacian $\mathcal{L}$. The filter bank $\boldsymbol{\eta} := \{a; b^{(1)}, \ldots, b^{(n)}\} \in l_0(\mathbb{Z})$ for a framelet system is a set of compactly supported sequences, where n denotes the number of high pass filters. The *low-pass* and *high-pass filters* of the framelet transforms, a and $b^{(r)}$, distill and represent the approximation and detail information of the graph signal. The scaling functions $\Psi = \{\alpha; \beta^{(1)}, \ldots, \beta^{(n)}\}$ with respect to the filter bank $\boldsymbol{\eta}$ are used to generate the framelets. The α, $\beta^{(r)}$ and their Fourier transforms are in $L_2(\mathbb{R})$. For $\xi \in \mathbb{R}$, the filters satisfy the classic refining equations

$$\widehat{\alpha}(2\xi) = \widehat{a}(\xi)\widehat{\alpha}(\xi),\ \widehat{\beta^{(r)}}(2\xi) = \widehat{b^{(r)}}(\xi)\widehat{\alpha}(\xi),\ r = 1, \ldots, n.$$

Graph Framelets Suppose $\{(\lambda_\ell, \boldsymbol{u}_\ell)\}_{j=1}^N$ are the eigenvalue and eigenvector pairs for $\mathcal{L}$ of graph $\mathcal{G}$ with N nodes. The (undecimated) framelets at *scale level* $j = 1, \ldots, J$ for graph $\mathcal{G}$ with the above scaling functions are defined, for $n = 1, \ldots, r$, by

$$\begin{aligned} \boldsymbol{\varphi}_{j,p}(v) &= \sum_{\ell=1}^{N} \widehat{\alpha}\left(\frac{\lambda_\ell}{2^j}\right) \overline{\boldsymbol{u}_\ell(p)} \boldsymbol{u}_\ell(v) \\ \boldsymbol{\psi}^n_{j,p}(v) &= \sum_{\ell=1}^{N} \widehat{\beta^{(n)}}\left(\frac{\lambda_\ell}{2^j}\right) \overline{\boldsymbol{u}_\ell(p)} \boldsymbol{u}_\ell(v), \end{aligned} \tag{1}$$

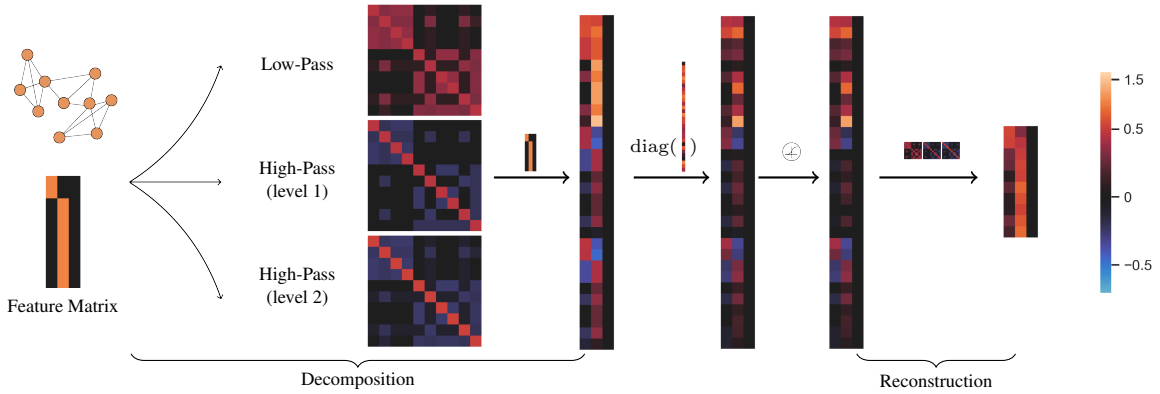

Figure 1. Computational flow of the proposed undecimated framelet graph convolution (UFGConv). This is an illustration with shrinkage activation, which will be discussed in detail in Section 4. Given a graph with structure (adjacency matrix) and feature information, the target is to properly embed the graph by graph convolution. The demonstrative sample is a graph with 10 nodes and 3 features extracted from **PROTEINS** in **TUDataset**. The framelet dilation and scale level are both set to default value 2. The UFGCONV applies tensor-based framelet transform, and constructs one low-pass and two high-pass *framelet transform matrices* $\mathcal{W}_{r,j}$, which are then multiplied by the input feature matrix to produce the framelet coefficients. Moreover, these coefficients are processed by the trainable network filter and compressed by the shrinkage. Finally, the activated coefficients are reconstructed and sent back to the spatial domain as the convolution output by using the framelet transform matrices again with transposed alignment.

where $\boldsymbol{\varphi}_{j,p}$ or $\boldsymbol{\psi}^r_{j,p}$ is the low-pass or high-pass framelet translated at node p. The low-pass and high-pass *framelet coefficients* for a signal f on graph $\mathcal{G}$ are $v_{j,p}$ and $w^r_{j,p}$, which are the projections $\langle \boldsymbol{\varphi}_{j,p}, f \rangle$ and $\langle \boldsymbol{\psi}^r_{j,p}, f \rangle$ of the graph signal onto framelets at scale j and node p. Here we use Haar-type filters for framelets (Dong, 2017). The dilation factor is 2^j with the *dilation* (base) 2.

The above framelet system is *tight* if it provides an exact representation for any function in $L_2(\mathcal{G})$. The tightness is determined by filter banks (See Theorem 1 in Appendix B). This condition guarantees a unique representation of a graph signal with framelet coefficients. It also helps manipulate the graph data in the framelet (frequency) domain.

The graph convolution developed in this work with a tight graph framelet system is effective for several reasons. First, the undecimated framelet transforms have a simplified formula that uses only graph Laplacian and filtered Chebyshev polynomial approximation, as will be further explained below. The resulting framelet transforms can be written as a sparse tensor and the corresponding framelet coefficients are evaluated fast in tensor computation. See Figure 1 for the computational flow of framelet convolution. Moreover, the tight framelets on the graph have a low redundancy rate, which is analogous to the framelets on a manifold as considered by Wang & Zhuang (2018; 2019). We give more discussion about framelets in the Appendix.

Tensor-based $\mathcal{G}$-framelet Transforms Framelet transforms map between a graph signal f and its representative framelet coefficients. We point out an approximate formula for (1), which exploits the Chebyshev polynomial approximation to the filters $\widehat{\alpha}$ and $\widehat{\beta^{(r)}}$, and thus gives a simplified version and fast evaluation for framelet transforms. By the framelet transform theorem (in Appendix), the corresponding framelet transforms are implemented by $\mathcal{W}$ and $\mathcal{V}$, the decomposition and reconstruction operators. For graph signal f, we define $\mathcal{W} = \{\mathcal{W}_{r,j} | r = 1, \ldots, n; j = 1, \ldots, J\} \cup \{\mathcal{W}_{0,J}\}$ entry-wisely, where $\mathcal{W}_{r,1} f = \mathcal{T}^k_r(2^{-J}\mathcal{L}) f$, and

$$\mathcal{W}_{r,j} f = \mathcal{T}^k_r(2^{K+j-1}\mathcal{L})\mathcal{T}^k_0(2^{K+j-2}\mathcal{L}) \cdots \mathcal{T}^k_0(2^{-K}\mathcal{L}) f$$

for $j = 2, \ldots, J$. Here $\mathcal{T}^k_r$ is the r-degree Chebyshev polynomial, $\mathcal{L}$ is the graph Laplacian, and K is the constant determined by the maximum eigenvalue of $\mathcal{L}$ that satisfies $\lambda_{\max} \leq 2^K \pi$. The $\mathcal{W}_{0,J} f = \{v_{J,p}\}_{p \in V}$ are the *low-pass coefficients* and $\mathcal{W}_{r,j} f = \{w^r_{j,p}\}_{p \in V}$ are *high-pass coefficients* of f. The j indicates the scale level, and $r = 1, \ldots, n$ with n the number of high-passes. The reconstruction operator $\mathcal{V}$ is the realignment of the framelet transform matrices of the decomposition operator $\mathcal{W}$.

Figure 1 gives the fast $\mathcal{G}$-framelet transform algorithm based on tensorized $\mathcal{W}$ and $\mathcal{V}$ with scale level 2 and shrinkage threshold $\sigma = 1$ (the meaning of σ will be discussed in detail in Section 5). In practice, we turn the computation into merely sparse matrix multiplication by properly aligning the low-pass and high-pass elements of $\mathcal{W}$ and $\mathcal{V}$. The tensor-based framelet transforms have time complexity $\mathcal{O}\left(N^2(nJ+1)Kd\right)$ and space complexity $\mathcal{O}\left(N^2(nJ+1)d\right)$ for an N-node graph and d features. Here, the n, J and K are constants independent of graph data. See Appendix for an empirical study of the complexity of framelet transforms on benchmarks.

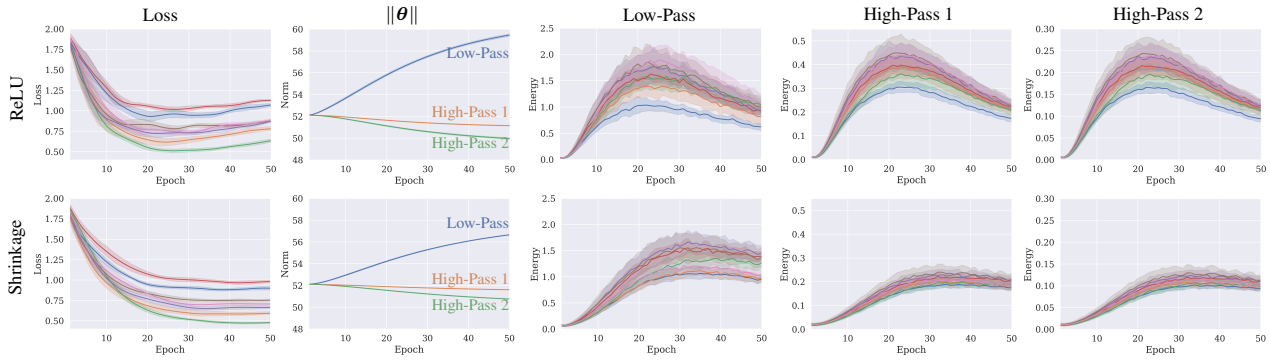

Figure 2. Learning behavior of the final framelet convolutional layer of GNN with two UFGCONV for **Cora**. Top is for UFGCONV with ReLU activation in (2). Bottom is for UFGCONV with Shrinkage activation in (3). From left to right we show some key network learning properties: training loss, l_2 norm of network filter $\boldsymbol{\theta}$, power spectrum of framelet coefficients for low-pass and high-passes at scale levels 1 and 2. The curves of the quantities for each of 7 label classes are shown. Framelets provide a good feature representation and shrinkage makes training more stable, where central information energy is conserved via thresholding high-pass coefficients.

4. Framelet Convolution

With the above $\mathcal{G}$-framelet transforms, we can define a *framelet (graph) convolution* similar to the spectral graph convolution. For *network filter* $\boldsymbol{\theta}$ and input *graph feature* $X \in \mathbb{R}^{N\times d}$ of the graph $\mathcal{G}$ with N nodes, we define

$$\boldsymbol{\theta} \star X = \mathrm{ReLU}\left(\mathcal{V}\left(\mathrm{diag}(\boldsymbol{\theta})(\mathcal{W}X')\right)\right),\ X' = XW. \quad (2)$$

As mentioned, $\mathcal{W}X'$ is the framelet coefficient matrix for the transformed X', where $\mathcal{W}$ is a sequence of $nJ+1$ transform matrices (each of size $N\times N$) for low-pass and high-passes. The size of the vector $\boldsymbol{\theta}$ is $(nJ+1)N$ which matches the total number of the framelet coefficients for each feature. The network filter $\boldsymbol{\theta}$ lies in the frequency domain, each component of which is multiplied to the corresponding row of $\mathcal{W}X'$. The matrix W in (2) is a trainable weight matrix with dimension $d\times d'$.

We can replace the ReLU activation in (2) with a wavelet shrinkage threshold function, or *shrinkage*. As the framelet coefficients lie in multiple scales, the "one size fits all" criterion of ReLU could potentially damage the multi-scale property of the framelet representation. In contrast, the shrinkage threshold adapts to the varying scales of the coefficients and provides a more precise cutoff. Thus we can use fewer coefficients while maintaining a comparable performance in framelet representation, see Section 8.1. The framelet convolution with *shrinkage activation* is

$$\mathcal{V}\left(\mathrm{Shrinkage}\big(\mathrm{diag}(\boldsymbol{\theta})(\mathcal{W}X')\big)\right),\ X' = XW. \quad (3)$$

Different from ReLU which works in the spatial domain, in (3), the shrinkage activation is carried on before the framelet reconstruction $\mathcal{V}$ as the shrinkage thresholds for the high-pass coefficients in the framelet domain. Figure 1 demonstrates a computational flow of the framelet convolution that learns the graph embedding of a graph with feature matrix $X^{\mathrm{in}} \in \mathbb{R}^{N\times d}$ by a framelet system of 2 scale levels ($j = 1, 2$) and 1 high-pass filter ($r = 0, 1$). Here, we omit the feature transformation step for a simple illustration. The decomposition $\mathcal{W} = \{\mathcal{W}_{0,2}; \mathcal{W}_{1,1}, \mathcal{W}_{1,2}\}$ transforms the input feature matrix with one low-pass and two high-pass operators. The operators can be rearranged to $\mathcal{W}^\natural := [\mathcal{W}_{0,2}, \mathcal{W}_{1,1}, \mathcal{W}_{1,2}]^\top$ of three $N\times N$ sparse matrices. The coefficients $\mathcal{W}^\natural X^{\mathrm{in}} \in \mathbb{R}^{3N\times d}$ can then be obtained by matrix multiplication. The network learning propagates in the frequency domain, where one applies the network filter $\boldsymbol{\theta}$ to the framelet coefficients $\mathcal{W}^\natural X^{\mathrm{in}}$. For the framelet convolution with shrinkage, the filtered coefficients would be activated before applying $\mathcal{V}^\natural = (\mathcal{W}^\natural)^\top$. Otherwise the ReLU activation will act on the reconstructed signal in the spatial domain.

5. Shrinkage and Signal Compression

Wavelet shrinkage is intimately linked to multiresolution properties of the wavelet transform in the classic wavelet theory. The shrinkage only applies to finer scales, i.e., detail coefficients (Donoho et al., 1995), so that the wavelet scalogram (a paradigm of the time-frequency energy localization of a signal) experiences a minimal change. This property promises a meaningful estimator for signal compression and an explainable graph convolution with framelets.

Sparsity and Compression The high-pass coefficients in the frequency domain can be cut off by shrinkage thresholding. For example, the *soft-thresholding* (Donoho & Johnstone, 1994; Donoho et al., 1995; Tibshirani, 1996) defines

$$\mathrm{Shrinkage}(x) = \mathrm{sgn}(x)(|x| - \lambda)_+ \quad \forall x \in \mathbb{R},$$

where λ is the threshold value. Any x with its absolute value less than λ shall return to zero. Applying the above soft-thresholding to the shrinkage activation in (3) only influences small high-pass framelet coefficients. We also consider the scale-dependent selection threshold with demarcation point (Donoho, 1995): $\lambda = \sigma\sqrt{2\log(N)}/\sqrt{N}$

for N coefficients. The hyperparameter σ is an analogue to the noise level of the wavelet denoising model. We let σ be associated with the magnitude order of the coefficients so it reflects the scale of the framelet representation.

The shrinkage benefits framelet graph convolution by reducing noise in framelet coefficients and compressing the signal in the frequency domain simultaneously. The traditional wavelet denoising for 1D functions suggests that using shrinkage at high-pass coefficients can effectively filter out the Gaussian white noise in the mean square error sense. This is also true for our case when we embed shrinkage in graph convolution. In Figure 3, the framelet convolution with shrinkage activation (UFGCONV-S) outperforms the ReLU case (UFGCONV-R) for reducing node and structure noises. Both methods surpass the classic spatial convolutions GCN (Kipf & Welling, 2017) and GAT (Veličković et al., 2018). Apart from denoising, graph convolution with shrinkage diminishes the proportion of non-zero framelet coefficients while maintains a comparable learning performance. We define the *compression ratio* for a shrinkage framelet convolutional layer as the ratio of the number of non-zero coefficients after and before shrinkage. Tables 1 and 2 show that UFGCONV-S compresses up to 70% non-zero coefficients with top performance for various node classification tasks.

Framelet Spectrum, Training Loss and Network Capacity The coefficients after shrinkage activation are proportional to the framelet power spectrum at the coefficient scale. We thus let the threshold level σ proportionate upon the framelet energy $\|\mathcal{W}_{r,j}X\|^2$ $(r > 0)$ for high-passes. For example, the framelet spectrum curves in training for the UFGCONV of Figure 2 show a higher magnitude order of the low-pass (column 3) than those of high-passes (columns 4-5). This is because coefficients in high-passes reflect more detailed characteristics than in low-pass. Compared with the ReLU case (row 1), the shrinkage activation (row 2) filters out some high-pass coefficients in graph convolution, which results in much smaller framelet spectra for high-passes. In contrast, the low-pass shrinkage involves no cutoff, and the energy is less distinguishable from the ReLU case.

The training loss curve of each output feature (column 1) indicates that shrinkage allows for more stable training, with a monotonically decreasing loss. The splitting in low-pass and high-passes for loss suggests a more flexible and precise control of the training. It also opens the possibility of designing a weighted new loss taking account of framelet spectrum. Moreover, the l_2 norm of the network filter $\boldsymbol{\theta}$ (column 2) has an increasing trend for the low-pass part and a decreasing trend for the high-pass parts during training. This observation is identical to the spectral bias (Rahaman et al., 2019), whereby the fitting capacity of the framelet convolutional GNN with either ReLU or shrinkage activation comes from the low-pass channel.

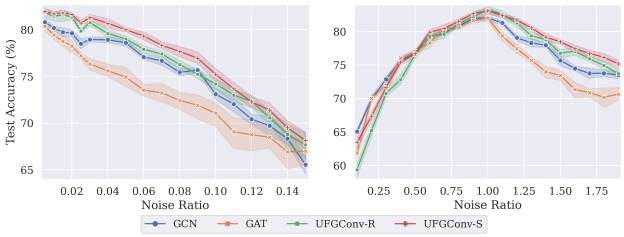

Figure 3. Node (left) and structure (right) perturbation analysis on **Cora**. Results from the other two datasets are in Appendix. The framelet dilation and scale level are both set to default value 2, and the optimal threshold σ for shrinkage is searched from $\{0.05, 0.1, 0.15\}$. Framelet convolution with shrinkage performs the best under both node and structure perturbations.

6. Robustness of Framelet Convolution under Feature and Structure Noises

Real-world data are usually noisy. For instance, graph data are sometimes polluted due to adversarial attacks as GNNs exchange node information (Xu et al., 2020), where node feature and graph structure could both be perturbed.

Our shrinkage framelet convolution has a motivation from the image deconvolution (or image restoration) model. Given the original and observed (degraded) image features x and y, one defines $y = Hx + \epsilon$, where H represents the convolution matrix of x. The noise ϵ is assumed multivariate Gaussian. In statistical formulation, the model solves

$$\widehat{x} = \arg\min_x \big\{\log(\Pr(y|x)) - \mathrm{Pen}(x)\big\}$$

with a penalty function Pen. On the real axis, wavelets are critical in restoring x from noisy y (Figueiredo & Nowak, 2003; Cai et al., 2012; Dong & Shen, 2013; Shen, 2010). With wavelet transform Φ, the maximum penalized likelihood estimator (MPLE) takes the form

$$\widehat{x} = \Phi^\top D\Phi y, \tag{4}$$

where D acts as a denoising operation of Φy.

The above wavelet-based convolution can be generalized to graph data restoration with x and y replaced by clean and distorted features on the graph. That is,

$$y = Px + \epsilon, \tag{5}$$

where ϵ is the entry-wise noise. The linear transform P is a permutation of node features or a change of the adjacency matrix. Similar models using graph Fourier spectrum were considered in image restoration, see Cheung et al. (2018, Sect V.B) and Milanfar (2013).

Similar to (4), our graph convolution replaces Φ with the graph framelet transform $\mathcal{W}$, and D with the trainable network filter $\boldsymbol{\theta}$. Our shrinkage activation is partly motivated by

LASSO (Tibshirani, 1996). The latter uses shrinkage thresholding to denoise the signal in (5) in high dimensions. Based on this connection, our proposed framelet convolution in Section 4 has a good potential against perturbation from (5). We test UFGCONV with perturbed nodes and edges in citation network datasets (**Cora**, **Citeseer** and **Pubmed**). The noise ϵ follows Bernoulli distribution to match binary node features of the tasks. That is, we randomly change node feature or edge weight from 0/1 to its opposite 1/0. Figure 3 reports the test accuracy of UFGCONV, GCN and GAT. The x-axis indicates the proportion of the distorted nodes or edges (which is equivalent to the signal-to-noise ratio (SNR)). As the SNR increases, UFGCONV behaves well ahead of GCN and GAT while with higher test accuracy and smaller variance. The slightly lower performance of UFGCONV only occurs when more than 50% edges are removed, where misinformation dominates the graph structure. Moreover, the shrinkage thresholding in all situations manages to filter out more noise and thus achieves higher performance. This example illustrates the effectiveness of framelet convolution in predicting node property with node features or structure of graphs that are distorted.

7. Framelet Pooling

Graph pooling is a critical ingredient of GNNs when the model is predicting graph-level properties with a constant feature dimension but varying graph size and structure. We propose *framelet pooling* for GNNs using framelet transforms. As an illustrative example, we use 2 scale levels for framelet decomposition. Similar to the graph convolution in Section 4, given a graph with feature matrix $X \in \mathbb{R}^{N\times d}$, we can obtain a set of framelet coefficients $\mathcal{W}_{r,j}f$ including one low pass $\mathcal{W}_{0,2}f$ at level 2 and two high passes $\mathcal{W}_{1,1}f$ and $\mathcal{W}_{1,2}f$ at levels 1 and 2, respectively. Each scale-wise framelet coefficient is an $N \times d$ real-valued matrix, and its ith feature column $(\mathcal{W}_{r,j}X)_i$ for $i = 1, \ldots, d$ would be aggregated by the sum, or the sum of squares of the elements. The two aggregation methods correspond to two framelet pooling strategies (see below). The calculation compresses the $N \times d$ coefficients to a d-dimensional vector, and the pooled output from the three framelet coefficients results in 3 d-dimensional vectors. Figure 4 visualizes the computational flowchart for our pooling model.

The framelet pooling benefits the network training by employing the information from multi-scales, as all scales in the framelet representation of the graph signal is taken into account. Depending on how we aggregate the framelet coefficients, we distinguish the two strategies by UFGPOOL-SUM or UFGPOOL-SPECTRUM. The latter aggregates the nodes by the wavelet (power) spectrum (that is, the sum of absolute squares of framelet coefficients over nodes, $\sum_{p\in V}|v_{j,p}|^2$ and $\sum_{p\in V}|w^r_{j,p}|^2$). In this way, the total information of the graph signal X^{in} is well-conserved after the pooling. The sum of wavelet power spectrum is equal to the total energy of the signal, that is, $\|X^{\text{pooled}}\| = \|X^{\text{in}}\|$ (see Theorem 1(iii) in Appendix). We present empirical evidence of the precedence of the proposed framelet pooling over existing graph pooling methods in Section 8.2.

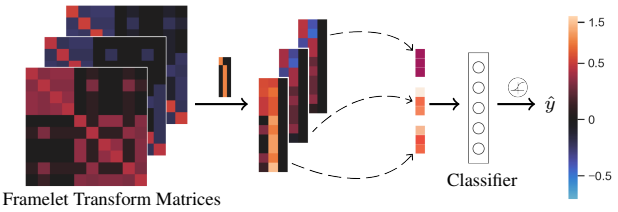

Figure 4. Framelet pooling for graph property prediction. The three framelet transform matrices are retrieved from Figure 1 with the same protein sample and parameter setting. The scale-wise framelet coefficients are aggregated to three vectors by sum or sum of squares (framelet spectrum). The (1 low pass and 2 high passes) vectors are then concatenated as the readout for the classifier.

8. Experiments

In this section, we show a variety of numerical tests for our framelet convolution and pooling. Section 8.1 tests the performance of framelet convolution (UFGCONV) on node classification benchmarks. Section 8.2 studies the ablation for the proposed framelet pooling (UFGPOOL). Section 8.3 provides a sensitivity analysis for UFGCONV in terms of dilation and scale. All experiments run in PyTorch on NVIDIA® Tesla V100 GPU with 5,120 CUDA cores and 16GB HBM2 mounted on an HPC cluster.

8.1. Framelet Convolution for Node Classification

We test UFGCONV with both ReLU and shrinkage activations on four node classification datasets. We denote the two variants by UFGCONV-R and UFGCONV-S.

Dataset The first experiment of node classification tasks is conducted on **Cora, Citeseer** and **Pubmed**, which are three benchmark citation networks. Moreover, we employ **ogbn-arxiv** from open graph benchmark **OGB** (Hu et al., 2020) to illustrate the power of our framelet convolution on large-scale graph-structured data.

Setup We design our UFGCONV model with two convolution layers for learning the graph embedding, the output of which is proceeded by a softmax activation for final prediction. Most hyperparameters are set to default, except for learning rate, weight decay, hidden units and dropout ratio in training. A grid search is conducted for fine tuning on these hyperparameters from the search space detailed in Appendix. Both methods are trained with the ADAM optimizer. The maximum number of epochs is 200 for citation

Table 1. Test accuracy (in percentage) for citation networks with standard deviation after ±. Compression ratio in **(green)** is the ratio of numbers of nonzero coefficients after and before shrinkage, and is with threshold level $\sigma = 1$.

Method	Cora	Citeseer	Pubmed
MLP	55.1	46.5	71.4
DEEPWALK	67.2	43.2	65.3
SPECTRAL	73.3	58.9	73.9
CHEBYSHEV	81.2	69.8	74.4
GCN	81.5	70.3	79.0
GWNN	82.8	**71.7**	**79.1**
GAT	**83.0**±0.7	**72.5**±0.7	79.0±0.3
MPNN	78.0±1.1	64.0±1.9	75.6±1.0
GRAPHSAGE	74.5±0.8	67.2±1.0	76.8±0.6
LANCZOSNET	79.5±1.8	66.2±1.9	78.3±0.3
DCNN	79.7±0.8	69.4±1.3	76.8±0.8
UFGCONV-S	**83.0**±0.5	71.0±0.6	**79.4**±0.4
(Compression)	(**47.7**)	(**39.0**)	(**27.7**)
UFGCONV-R	**83.6**±0.6	**72.7**±0.6	**79.9**±0.1

† The top three are highlighted by **First**, **Second**, **Third**.

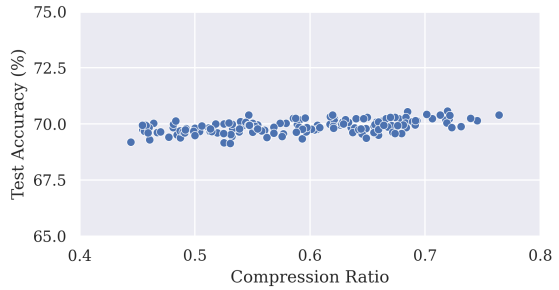

Figure 5. Trade-off between compression ratio and test accuracy in UFGCONV-S with **ogbn-arxiv**.

networks and 500 for **ogbn-arxiv**. All the datasets follow the standard public split and processing rules. The average test accuracy and its standard deviation come from 10 runs.

Baseline The UFGCONV-R and UFGCONV-S are compared against other methods for node classification tasks. We consider multiple baseline models that are applicable to the tasks. For citation networks, the reported accuracy are retrieved from public results: MLP, DEEPWALK (Perozzi et al., 2014), CHEBYSHEV (Defferrard et al., 2016a) and GCN (Kipf & Welling, 2017) from Kipf & Welling (2017); SPECTRAL CNN (Bruna et al., 2014) and GWNN (Xu et al., 2019a); MPNN (Gilmer et al., 2017), GRAPHSAGE (Hamilton et al., 2017), LANCZOSNET (Liao et al., 2019) and DCNN (Singer et al., 2009) from Liao et al. (2019); and GAT (Veličković et al., 2018) from their authors. For **ogbn-arxiv**, we also compared with NODE2VEC (Grover & Leskovec, 2016), GRAPHZOOM (Deng et al., 2020), P&L+C&S (Huang et al., 2021), DEEPERGCN (Li et al., 2020a), SIGN (Rossi et al., 2020) and GAAN (Zhang et al., 2018a) from the **OGB** leaderboard.

Table 2. Test accuracy (in percentage) for **ogbn-arxiv** with standard deviation after ±. The compression ratio for UFGCONV-S with shrinkage threshold level $\sigma = 1$ is **64.2%**.

Method	Test Acc.	Val. Acc.	#Params
MLP	55.50±0.23	57.65±0.12	110,120
NODE2VEC	70.07±0.13	71.29±0.13	21,818,792
GRAPHZOOM	71.18±0.18	72.20±0.07	8,963,624
P&L + C&S	71.26±0.01	73.00±0.01	5,160
GRAPHSAGE	71.49±0.27	72.77±0.17	218,664
GCN	71.74±0.29	73.00±0.17	142,888
DEEPERGCN	71.92±0.17	72.62±0.14	491,176
SIGN	**71.95**±0.11	73.23±0.06	3,566,128
GAAN	**71.97**±0.18	–	1,471,506
UFGCONV-S	70.04±0.22	71.04±0.11	1,633,183
UFGCONV-R	**71.97**±0.12	73.21±0.05	1,633,183

† The top three are highlighted by **First**, **Second**, **Third**.

Results We report the accuracy score in percentage with the top-3 highlighted in Tables 1 and 2. For UFGCONV-S, we also report the compression ratio for shrinkage (in green). In Table 1, the UFGCONV-R method achieves the highest prediction accuracy among all baseline models. The learned UFGCONV-S with threshold level $\sigma = 1$ trims up to 50% information but still obtains the top-3 rank on two tasks. A similar outstanding performance is reported in Table 2 for the **ogbn-arxiv** dataset, where the UFGCONV-R again ranks first with a moderate number of parameters, and the UFGCONV-S with threshold $\sigma = 1$ achieves a comparable accuracy at 70% using 64.2% information.

We test UFGCONV-S with different compression ratios. Ideally, a high test accuracy is preferable to pair with a low compression ratio, and the change in accuracy should be minimal due to the insensitivity of our model to the hyperparameters. However, as shown in Figure 5, an increasing compression ratio generally results in a slightly higher prediction accuracy as more coefficients for the framelet representation is used by the convolution.

8.2. Framelet Pooling for Graph Property Prediction

The second experiment evaluates two framelet pooling methods, UFGPOOL-SUM and UFGPOOL-SPECTRUM, by ablation studies on graph classification and regression tasks.

Dataset We select six benchmarks to test the proposed pooling strategies, including four graph classification tasks with moderate sample sizes, one regression task, and one large-scale classification task. First five tasks use **TU-Dataset benchmarks** (Morris et al., 2020), including **D&D** (Dobson & Doig, 2003; Shervashidze et al., 2011), **PROTEINS** (Dobson & Doig, 2003; Borgwardt et al., 2005) to categorize proteins into enzyme and non-enzyme structures; **NCI1** (Wale et al., 2008) to identify chemical compounds that block lung cancer cells; **Mutagenicity** (Kazius et al.,

Table 3. Performance comparison for graph property prediction. **QM7** is a regression task in MSE; **ogbg-molhiv** is a classification task in ROC-AUC in percentage; others are for classification in test accuracy in percentage. The value after ± is standard deviation.

Datasets	PROTEINS	Mutagenicity	D&D	NCI1	ogbg-molhiv	QM7
TOPKPOOL	73.48±3.57	79.84±2.46	74.87±4.12	75.11±3.45	78.14±0.62	175.41±3.16
ATTENTION	73.93±5.37	80.25±2.22	77.48±2.65	74.04±1.27	74.44±2.12	177.99±2.22
SAGPOOL	**75.89**±2.91	79.86±2.36	74.96±3.60	76.30±1.53	75.26±2.29	**41.93**±1.14
SUM	74.91±4.08	**80.69**±3.26	**78.91**±3.37	**76.96**±1.70	77.41±1.16	42.09±0.91
MAX	73.57±3.94	78.83±1.70	75.80±4.11	75.96±1.82	78.16±1.33	177.48±4.70
MEAN	73.13±3.18	80.37±2.44	76.89±2.23	73.70±2.55	**78.21**±0.90	177.49±4.69
UFGPOOL-SUM	**77.77**±2.60	**81.59**±1.40	**80.92**±1.68	**77.88**±1.24	**78.80**±0.56	**41.74**±0.84
UFGPOOL-SPECTRUM	**77.23**±2.40	**82.05**±1.28	**79.83**±1.88	**77.54**±2.24	**78.36**±0.77	**41.67**±0.95

† The top three are highlighted by **First**, **Second**, **Third**.

2005; Riesen & Bunke, 2008) to recognize mutagenic molecular compounds for potentially marketable drug; and **QM7** (Blum & Reymond, 2009; Rupp et al., 2012) to predict atomization energy value of molecules. The dataset **ogbg-molhiv** (Hu et al., 2020) is for large-scale molecule classification.

Setup The network architecture for all baseline models is fixed to two convolutional layers followed by one pooling layer. The graph convolution for the five **TUDataset**s uses the GCN model, and for **ogbg-molhiv** uses GIN with virtual nodes (Ishiguro et al., 2019). Given graph representations, the prediction is made by a two-layer MLP, where the hidden unit is identical to that of the convolutional layer. The hyperparameters (learning rate, weight decay, number of hidden units in each convolutional layer, and dropout ratio in the readout layer) are fine-tuned with grid search.

Each dataset is split into training, validation and test sets by 80%, 10% and 10%. The training stops when the validation loss stops improving for 20 consecutive epochs or reaching maximum 200 epochs. All results are averaged over 10 repetitions. The classification tasks report mean test accuracy for **TUDataset** and ROC-AUC score for **ogbg-molhiv**. The regression task on **QM7** reports mean square error (MSE).

Baseline We compare our framelet pooling (UFGPOOL-SUM and UFGPOOL-SPECTRUM) with six baseline methods that are capable for global pooling to verify the effectiveness of the learned graph representation. The baselines include TOPKPOOL (Gao & Ji, 2019; Cangea et al., 2018), ATTENTIONPOOL (Li et al., 2016), SAGPOOL (Lee et al., 2019), as well as the classic SUM, MEAN and MAX pooling.

Results Table 3 summarizes the performance comparison. Our UFGPOOL methods outperform other methods on all datasets. Specifically, UFGPOOL-SUM achieves the top accuracy in four out of six datasets, and the second best accuracy in the other two, where the top performance is achieved by UFGPOOL-SPECTRUM. We also observe that UFGPOOL-SPECTRUM performs better on small molecules prediction: **Mutagenicity**, **QM7** and **ogbg-molhiv**. This precedence might come from encoding the multi-scale signal energy to the network where the framelet spectra capture the practically significant features of molecular data.

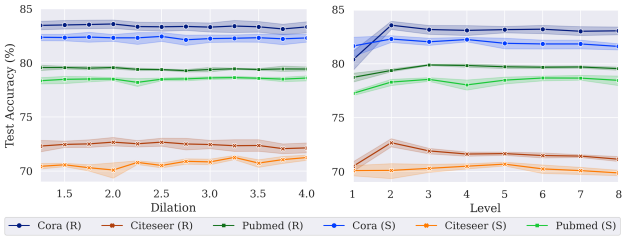

Figure 6. Sensitivity analysis for dilation (left) and scale level (right) with UFGCONV-R and UFGCONV-S on citation networks.

8.3. Sensitivity Analysis

This section analyses the sensitivity of UFGCONV-R and UFGCONV-S on the hyperparameters dilation and scale level in the framelet system. The experiment is conducted on **Cora**, **Citeseer** and **Pubmed**. The dilation analysis selects values from 1.25 to 4 with step 0.25, and the scale levels in those three datasets are fixed to 2, 2 and 3 respectively. For the scale level analysis, the values are set from 1 to 8 with step 1, and the dilation stays at default 2. All other hyperparameters are tuned in the same way as Section 8.1. We use shrinkage threshold $\sigma = 1$ for UFGCONV-S.

From Figure 6, we can observe that changing dilation or scale level does not drastically impact on the accuracy for either method. In particular, the mean test accuracy is stable over all dilation values and reaches the peak with a small scale level (2 for **Cora** & **Citeseer**; 3 for **Pubmed**). For scale level 1, the decreased accuracy is due to the insufficiency of scale and then not salient multiresolution. Thus, we can use dilation 2 and scale level 2 in practice, in which the GNN uses multi-scale framelet analysis to achieve supreme performance with a low computational cost.

9. Conclusion

We explore the adaptation of graph framelets for graph neural networks in this paper. As a multi-scale graph representa-

tion method, framelet transforms link graph neural networks and signal processing. In many node-level or graph-level tasks, framelet convolutions can reduce both feature and structure noises. We also introduce shrinkage activation that thresholds high-pass coefficients in framelet convolution, which strengthens the network denoising capability and simultaneously compresses graph signal at a remarkable rate. Moreover, we design graph pooling using framelet spectra at low and high passes. The proposed framelet convolutions with both ReLU and shrinkage surpass typical spatial-based and spectral-based graph convolutions on most benchmarks, and the framelet pooling outperforms baselines on a variety of graph property prediction tasks.

Acknowledgements

YW and GM have been supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant n^{o} 757983). ML acknowledges supports from the National Natural Science Foundation of China (No. 61802132) and the "Qianjiang Talent Program D" of Zhejiang Province. This project was undertaken with the assistance of resources and services from the National Computational Infrastructure (NCI), which is supported by the Australian Government. This research was also undertaken with the assistance of resources and services from the GPU cluster Cobra of Max Planck Society. The authors would like to acknowledge support from the UNSW Resource Allocation Scheme managed by Research Technology Services at UNSW Sydney. The authors also acknowledge the technical assistance provided by the Sydney Informatics Hub, a Core Research Facility of the University of Sydney.

References

Balcilar, M., Renton, G., Héroux, P., Gaüzère, B., Adam, S., and Honeine, P. Analyzing the expressive power of graph neural networks in a spectral perspective. In *ICLR*, 2021.

Behjat, H., Richter, U., Van De Ville, D., and Sörnmo, L. Signal-adapted tight frames on graphs. *IEEE Transactions on Signal Processing*, 64(22):6017–6029, 2016.

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

Borgwardt, K. M., Ong, C. S., Schönauer, S., Vishwanathan, S. V. N., Smola, A. J., and Kriegel, H.-P. Protein function prediction via graph kernels. *Bioinformatics*, 21(suppl_1): i47–i56, 2005.

Brauchart, J., Dick, J., Saff, E., Sloan, I., Wang, Y., and Womersley, R. Covering of spheres by spherical caps and worst-case error for equal weight cubature in sobolev spaces. *Journal of Mathematical Analysis and Applications*, 431(2):782 – 811, 2015. ISSN 0022-247X.

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

Cai, J.-F., Dong, B., Osher, S., and Shen, Z. Image restoration: total variation, wavelet frames, and beyond. *Journal of the American Mathematical Society*, 25(4):1033–1089, 2012.

Cangea, C., Veličković, P., Jovanović, N., Kipf, T., and Liò, P. Towards sparse hierarchical graph classifiers. In *NeurIPS Workshop on Relational Representation Learning*, 2018.

Chen, M., Wei, Z., Huang, Z., Ding, B., and Li, Y. Simple and deep graph convolutional networks. In *ICML*, pp. 1725–1735. PMLR, 2020.

Cheung, G., Magli, E., Tanaka, Y., and Ng, M. K. Graph spectral image processing. *Proceedings of the IEEE*, 106 (5):907–930, 2018.

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

Cohen, A., Daubechies, I., Jawerth, B., and Vial, P. Multiresolution analysis, wavelets and fast algorithms on an interval. *Comptes rendus de l'Académie des sciences. Série 1, Mathématique*, 316(5):417–421, 1993.

Crovella, M. and Kolaczyk, E. Graph wavelets for spatial traffic analysis. In *IEEE INFOCOM 2003*, volume 3, pp. 1848–1857. IEEE, 2003.

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

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

Deng, C., Zhao, Z., Wang, Y., Zhang, Z., and Feng, Z. Graphzoom: A multi-level spectral approach for accurate and scalable graph embedding. In *ICLR*, 2020.

Dobson, P. D. and Doig, A. J. Distinguishing enzyme structures from non-enzymes without alignments. *Journal of Molecular Biology*, 330(4):771–783, 2003.

Dong, B. Sparse representation on graphs by tight wavelet frames and applications. *Applied and Computational Harmonic Analysis*, 42(3):452–479, 2017.

Dong, B. and Shen, Z. *MRA-Based Wavelet Frames and Applications*. IAS Lecture Note Series, 06 2013.

Donoho, D. L. Wavelet shrinkage and wvd: a 10-minute tour. In *Presented on the International Conference on Wavelets and Applications, Tolouse, France*, 1992.

Donoho, D. L. De-noising by soft-thresholding. *IEEE Transactions on Information Theory*, 41(3):613–627, 1995.

Donoho, D. L. and Johnstone, J. M. Ideal spatial adaptation by wavelet shrinkage. *Biometrika*, 81(3):425–455, 1994.

Donoho, D. L., Johnstone, I. M., Kerkyacharian, G., and Picard, D. Wavelet shrinkage: Asymptopia? *Journal of the Royal Statistical Society: Series B (Methodological)*, 57(2):301–337, 1995.

Efron, B. and Morris, C. Data analysis using Stein's estimator and its generalizations. *Journal of the American Statistical Association*, 70(350):311–319, 1975.

Figueiredo, M. A. and Nowak, R. D. An EM algorithm for wavelet-based image restoration. *IEEE Transactions on Image Processing*, 12(8):906–916, 2003.

Gao, H. and Ji, S. Graph U-nets. In *ICML*, 2019.

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

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

Glorot, X. and Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, pp. 249–256, 2010.

Grover, A. and Leskovec, J. node2vec: Scalable feature learning for networks. In *ACM SIGKDD*, pp. 855–864, 2016.

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

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

Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M., and Leskovec, J. Open graph benchmark: Datasets for machine learning on graphs. In *NeurIPS*, 2020.

Huang, Q., He, H., Singh, A., Lim, S.-N., and Benson, A. R. Combining label propagation and simple models out-performs graph neural networks. *ICLR*, 2021.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, pp. 448–456, 2015.

Ishiguro, K., Maeda, S.-i., and Koyama, M. Graph warp module: an auxiliary module for boosting the power of graph neural networks. *arXiv:1902.01020*, 2019.

ISO. ISO/IEC 15444-1:2019 Information technology - JPEG 2000 image coding system - Part 1: Core coding system. 2019.

Kazius, J., McGuire, R., and Bursi, R. Derivation and validation of toxicophores for mutagenicity prediction. *Journal of Medicinal Chemistry*, 48(1):312–320, 2005.

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

Knyazev, B., Taylor, G. W., and Amer, M. R. Understanding attention and generalization in graph neural networks. In *NeurIPS*, 2019.

Lee, J., Lee, I., and Kang, J. Self-attention graph pooling. In *ICML*, 2019.

Li, G., Xiong, C., Thabet, A., and Ghanem, B. DeeperGCN: All you need to train deeper GCNs. *arXiv:2006.07739*, 2020a.

Li, M., Ma, Z., Wang, Y. G., and Zhuang, X. Fast haar transforms for graph neural networks. *Neural Networks*, pp. 188–198, 2020b.

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

Liao, R., Zhao, Z., Urtasun, R., and Zemel, R. Lanczosnet: Multi-scale deep graph convolutional networks. In *ICLR*, 2019.

Ma, Z., Xuan, J., Wang, Y. G., Li, M., and Liò, P. Path integral based convolution and pooling for graph neural networks. In *NeurIPS*, volume 33, pp. 16433–16445, 2020.

Maggioni, M. and Mhaskar, H. H. Diffusion polynomial frames on metric measure spaces. *Applied and Computational Harmonic Analysis*, 24(3):329 – 353, 2008.

Mallat, S. G. A theory for multiresolution signal decomposition: the wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(7):674–693, 1989.

Milanfar, P. A tour of modern image filtering: New insights and methods, both practical and theoretical. *IEEE Signal Processing Magazine*, 30(1):106–128, 2013.

Morris, C., Kriege, N. M., Bause, F., Kersting, K., Mutzel, P., and Neumann, M. TUDataset: A collection of benchmark datasets for learning with graphs. In *ICML Workshop "Graph Representation Learning and Beyond"*, 2020.

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

Rahaman, N., Baratin, A., Arpit, D., Draxler, F., Lin, M., Hamprecht, F., Bengio, Y., and Courville, A. On the spectral bias of neural networks. In *ICML*, pp. 5301–5310. PMLR, 2019.

Riesen, K. and Bunke, H. Iam graph database repository for graph based pattern recognition and machine learning. In *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR)*, pp. 287–297. Springer, 2008.

Rossi, E., Frasca, F., Chamberlain, B., Eynard, D., Bronstein, M., and Monti, F. SIGN: Scalable inception graph neural networks. In *ICML Workshop "Graph Representation Learning and Beyond"*, 2020.

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

Shen, Z. Wavelet frames and image restorations. In *the International Congress of Mathematicians (ICM)*, pp. 2834–2863, 2010.

Shervashidze, N., Schweitzer, P., Van Leeuwen, E. J., Mehlhorn, K., and Borgwardt, K. M. Weisfeiler-Lehman graph kernels. *Journal of Machine Learning Research*, 12(77):2539–2561, 2011.

Singer, A., Shkolnisky, Y., and Nadler, B. Diffusion interpretation of nonlocal neighborhood filters for signal denoising. *SIAM Journal on Imaging Sciences*, 2(1):118–139, 2009.

Tibshirani, R. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

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

Vignac, C., Loukas, A., and Frossard, P. Building powerful and equivariant graph neural networks with structural message-passing. In *NeurIPS*, volume 33, pp. 14154–14166, 2020.

Wale, N., Watson, I. A., and Karypis, G. Comparison of descriptor spaces for chemical compound retrieval and classification. *Knowledge and Information Systems*, 14 (3):347–375, 2008.

Wang, X., Zhu, M., Bo, D., Cui, P., Shi, C., and Pei, J. Am-gcn: Adaptive multi-channel graph convolutional networks. In *ACM SIGKDD*, pp. 1243–1253, 2020a.

Wang, Y. G. and Zhuang, X. Tight framelets and fast framelet filter bank transforms on manifolds. *Applied and Computational Harmonic Analysis*, 2018.

Wang, Y. G. and Zhuang, X. Tight framelets on graphs for multiscale data analysis. In *Wavelets and Sparsity XVIII*, volume 11138, pp. 100 – 111. SPIE, 2019.

Wang, Y. G., Li, M., Ma, Z., Montufar, G., Zhuang, X., and Fan, Y. Haar graph pooling. In *ICML*, pp. 9952–9962. PMLR, 2020b.

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Yu, P. S. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):4–24, 2021.

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

Xu, H., Ma, Y., Liu, H.-C., Deb, D., Liu, H., Tang, J.-L., and Jain, A. K. Adversarial attacks and defenses in images, graphs and text: A review. *International Journal of Automation and Computing*, 17(2):151–178, 2020.

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

Zhang, J., Shi, X., Xie, J., Ma, H., King, I., and Yeung, D.-Y. GaAN: Gated attention networks for learning on large and spatiotemporal graphs. In *UAI*, pp. 339–349, 2018a.

Zhang, M., Cui, Z., Neumann, M., and Chen, Y. An end-to-end deep learning architecture for graph classification. In *AAAI*, 2018b.

Zheng, X., Zhou, B., Li, M., Wang, Y. G., and Gao, J. MathNet: Haar-like wavelet multiresolution-analysis for graph representation and learning. *arXiv:2007.11202*, 2020.

Zheng, X., Zhou, B., Wang, Y. G., and Zhuang, X. Decimated framelet system on graphs and fast G-framelet transforms. *Journal of Machine Learning Research*, 2021.

A. Undecimated Framelets on Graph

An undirected and weighted graph $\mathcal{G}$ is an ordered triple $\mathcal{G} = (V, E, \omega)$ with a finite vertex set V, and edge set $E \subseteq V \times V$, and a non-negative edge weight function $\omega : E \to \mathbb{R}$. We consider the l_2 space on the graph $\mathcal{G}$, $l_2(\mathcal{G})$, with inner product

$$\langle f, g\rangle := \sum_{v\in V} f(v)\overline{g(v)}, \quad f, g \in l_2(\mathcal{G}),$$

where $\overline{g}$ is the complex conjugate to g, and induced norm $\|f\| := \sqrt{\langle f, f\rangle}$. Let N be the number of vertices for $\mathcal{G}$.

The construction of framelets uses the graph spectrum, and filter bank that is a set of *filters*. A filter (or mask) is complex-valued sequence on the graph $h := \{h_k\}_{k\in\mathbb{Z}} \subseteq \mathbb{C}$ satisfying $|h_k| < \infty$. The *Fourier series* of a sequence $\{h_k\}_{k\in\mathbb{Z}}$ is the 1-periodic function $\widehat{h}(\xi) := \sum_{k\in\mathbb{Z}} h_k e^{-2\pi ik\xi}$, $\xi \in \mathbb{R}$. The *scaling functions* $\Psi_j = \{\alpha; \beta^{(1)}, \ldots, \beta^{(n)}\}$ associated with the filter bank $\boldsymbol{\eta} := \{a; b^{(1)}, \ldots, b^{(n)}\}$ are complex-valued functions on the real axis, which satisfy the equations, for $r = 1, \ldots, n$, $\xi \in \mathbb{R}$,

$$\widehat{\alpha}(2\xi) = \widehat{a}(\xi)\widehat{\alpha}(\xi), \quad \widehat{\beta^{(r)}}(2\xi) = \widehat{b^{(r)}}(\xi)\widehat{\alpha}(\xi). \tag{6}$$

Here the a is called *low-pass filter* and $b^{(r)}$, $n = 1, \ldots, r$ are *high-pass filters*. Let $\{(\boldsymbol{u}_\ell, \lambda_\ell)\}_{\ell=1}^N$ the eigen-pair for the graph Laplacian $\mathcal{L}$ on $l_2(\mathcal{G})$. For $j \in \mathbb{Z}$ and $p \in V$, the *undecimated framelets* $\boldsymbol{\varphi}_{j,p}(g)$ and $\boldsymbol{\psi}^r_{j,p}(g)$, $v \in V$ at scale j are *filtered Bessel kernels* (or summability kernels)

$$\begin{aligned}
\boldsymbol{\varphi}_{j,p}(g) &:= \sum_{\ell=1}^N \widehat{\alpha}\left(\frac{\lambda_\ell}{2^j}\right)\overline{\boldsymbol{u}_\ell(p)}\boldsymbol{u}_\ell(v),\\
\boldsymbol{\psi}^r_{j,p}(g) &:= \sum_{\ell=1}^N \widehat{\beta^{(r)}}\left(\frac{\lambda_\ell}{2^j}\right)\overline{\boldsymbol{u}_\ell(p)}\boldsymbol{u}_\ell(v), \quad r = 1, \ldots, n.
\end{aligned} \tag{7}$$

See for example, Brauchart et al. (2015); Maggioni & Mhaskar (2008). Here, j and p in $\boldsymbol{\varphi}_{j,p}(g)$ and $\boldsymbol{\psi}^r_{j,p}(g)$ are the "dilation" at scale j and the "translation" at a vertex $p \in V$, which have their counterpart in the traditional wavelets on the real axis. For two integers J, J_1 satisfying $J > J_1$, we define an *undecimated framelet system* $\mathsf{UFS}(\Psi, \boldsymbol{\eta}; \mathcal{G})$ (starting from a scale J_1) as a non-homogeneous, stationary affine system:

$$\begin{aligned}
\mathsf{UFS}^J_{J_1}(\Psi, \boldsymbol{\eta}) &:= \mathsf{UFS}^J_{J_1}(\Psi, \boldsymbol{\eta}; \mathcal{G})\\
&:= \{\boldsymbol{\varphi}_{J_1,p} : p \in V\} \cup \{\boldsymbol{\psi}^r_{j,p} : p \in V, j = J_1, \ldots, J\}^n_{r=1}.
\end{aligned} \tag{8}$$

The system $\mathsf{UFS}^J_{J_1}(\Psi, \boldsymbol{\eta})$ is then called an *undecimated tight frame* for $l_2(\mathcal{G})$ and the elements in $\mathsf{UFS}^J_{J_1}(\Psi, \boldsymbol{\eta})$ are called *undecimated tight framelets* on $\mathcal{G}$.

There are many types of filters. In the experiments, we use the Haar-type filter with one high pass: for $x \in \mathbb{R}$,

$$\widehat{a}(x) = \cos(x/2) \text{ and } \widehat{b^{(1)}}(x) = \sin(x/2).$$

B. Equivalence Conditions of Tightness of Framelet System

The framelet transforms between time domain and framelet domain can attain zero loss. It is due to the *tightness* of the undecimated framelet system. That is, the (9) holds for all $f \in l_2(\mathcal{G})$. The tightness of a framelet system is met by an appropriate choice of the filter bank. In our case, the classical filter bank that satisfies the partition of unity would guarantee the tightness of the framelet system, which are equivalent conditions (iv) and (v) in Theorem 1 (see below). The theorem was proved in Zheng et al. (2021). For completeness, we give a brief proof here.

Theorem 1 (Equivalence of Framelet Tightness) *Let $\mathcal{G} = (g, E, \omega)$ be a graph and $\{(\boldsymbol{u}_\ell, \lambda_\ell)\}_{\ell=1}^N$ a set of orthonormal eigen-pairs for $l_2(\mathcal{G})$. Let $\Psi = \{\alpha; \beta^{(1)}, \ldots, \beta^{(n)}\}$ be a set of functions in $L_1(\mathbb{R})$ associated with a filter bank $\boldsymbol{\eta} = \{a; b^{(1)}, \ldots, b^{(n)}\}$ satisfying (6). Let integer $J \geq 1$, and $\mathsf{UFS}^J_{J_1}(\Psi, \boldsymbol{\eta}; \mathcal{G})$, $J_1 = 1, \ldots, J$, be an undecimated framelet system given in (8) with framelets $\boldsymbol{\varphi}_{j,p}$ and $\boldsymbol{\psi}^n_{j,p}$ in (7). Then, the following statements are equivalent.*

(i) For each $J_1 = 1, \ldots, J$, the undecimated framelet system $\mathsf{UFS}^J_{J_1}(\Psi, \boldsymbol{\eta}; \mathcal{G})$ is a tight frame for $l_2(\mathcal{G})$, that is, $\forall f \in l_2(\mathcal{G})$,

$$\|f\|^2 = \sum_{p\in V} \left| \langle f, \boldsymbol{\varphi}_{J_1,p} \rangle \right|^2 + \sum_{j=J_1}^{J} \sum_{r=1}^{n} \sum_{p\in V} \left| \langle f, \boldsymbol{\psi}^r_{j,p} \rangle \right|^2. \tag{9}$$

(ii) For all $f \in l_2(\mathcal{G})$ and for $j = 1, \ldots, J-1$, the following identities hold:

$$f = \sum_{p\in V} \langle f, \boldsymbol{\varphi}_{J,p} \rangle \, \boldsymbol{\varphi}_{J,p} + \sum_{r=1}^{n} \sum_{p\in V} \langle f, \boldsymbol{\psi}^r_{J,p} \rangle \, \boldsymbol{\psi}^r_{J,p}, \tag{10}$$

$$\sum_{p\in V} \langle f, \boldsymbol{\varphi}_{j+1,p} \rangle \, \boldsymbol{\varphi}_{j+1,p} = \sum_{p\in V} \langle f, \boldsymbol{\varphi}_{j,p} \rangle \, \boldsymbol{\varphi}_{j,p} + \sum_{r=1}^{n} \sum_{p\in V} \langle f, \boldsymbol{\psi}^r_{j,p} \rangle \, \boldsymbol{\psi}^r_{j,p}. \tag{11}$$

(iii) For all $f \in l_2(\mathcal{G})$ and for $j = 1, \ldots, J-1$, the following identities hold:

$$\|f\|^2 = \sum_{p\in V} \left| \langle f, \boldsymbol{\varphi}_{J,p} \rangle \right|^2 + \sum_{r=1}^{n} \sum_{p\in V} \left| \langle f, \boldsymbol{\psi}^r_{J,p} \rangle \right|^2, \tag{12}$$

$$\sum_{p\in V} \left| \langle f, \boldsymbol{\varphi}_{j+1,p} \rangle \right|^2 = \sum_{p\in V} \left| \langle f, \boldsymbol{\varphi}_{j,p} \rangle \right|^2 + \sum_{r=1}^{n} \sum_{p\in V} \left| \langle f, \boldsymbol{\psi}^r_{j,p} \rangle \right|^2. \tag{13}$$

(iv) The functions in Ψ satisfy

$$1 = \left| \widehat{\alpha}\left(\frac{\lambda_\ell}{2^J}\right) \right|^2 + \sum_{r=1}^{n} \left| \widehat{\beta^{(r)}}\left(\frac{\lambda_\ell}{2^J}\right) \right|^2 \quad \forall \ell = 1, \ldots, N, \tag{14}$$

$$\left| \widehat{\alpha}\left(\frac{\lambda_\ell}{2^{j+1}}\right) \right|^2 = \left| \widehat{\alpha}\left(\frac{\lambda_\ell}{2^j}\right) \right|^2 + \sum_{r=1}^{n} \left| \widehat{\beta^{(r)}}\left(\frac{\lambda_\ell}{2^j}\right) \right|^2 \quad \forall \ \begin{matrix} \ell = 1, \ldots, N, \\ j = 1, \ldots, J-1. \end{matrix} \tag{15}$$

(v) The identities in (14) hold and the filters in the filter bank $\boldsymbol{\eta}$ satisfy

$$\left| \widehat{a}\left(\frac{\lambda_\ell}{2^j}\right) \right|^2 + \sum_{n=1}^{r} \left| \widehat{b^{(r)}}\left(\frac{\lambda_\ell}{2^j}\right) \right|^2 = 1 \quad \forall \ell \in \sigma^{(j)}_\alpha, \ j = 2, \ldots, J, \tag{16}$$

with

$$\sigma^{(j)}_\alpha := \left\{ \ell \in \{1, \ldots, N\} : \widehat{\alpha}\left(\frac{\lambda_\ell}{2^j}\right) \neq 0 \right\}.$$

Proof (i)$\Longleftrightarrow$(ii). Let $\Phi_j := \operatorname{span}\{\boldsymbol{\varphi}_{j,p} : p \in V\}$ and $\Psi^r_j := \operatorname{span}\{\boldsymbol{\psi}^r_{j,p} : p \in V\}$. Define projections $\mathbf{P}_{\Phi_j}, \mathbf{P}_{\Psi^r_j}$, $r = 1, \ldots, n$ by

$$\mathbf{P}_{\Phi_j}(f) := \sum_{p\in V} \langle f, \boldsymbol{\varphi}_{j,p} \rangle \, \boldsymbol{\varphi}_{j,p}, \quad \mathbf{P}_{\Psi^r_j}(f) := \sum_{p\in V} \langle f, \boldsymbol{\psi}^r_{j,p} \rangle \, \boldsymbol{\psi}^r_{j,p}, \quad f \in l_2(\mathcal{G}). \tag{17}$$

Since $\mathsf{UFS}^J_{J_1}(\Psi, \boldsymbol{\eta})$ is a tight frame for $l_2(\mathcal{G})$ for $J_1 = 1, \ldots, J$, we obtain by polarization identity,

$$f = \mathbf{P}_{\Phi_{J_1}}(f) + \sum_{j=J_1}^{J} \sum_{r=1}^{n} \mathbf{P}_{\Psi^r_j}(f) = \mathbf{P}_{\Phi_{J_1+1}}(f) + \sum_{j=J_1+1}^{J} \sum_{r=1}^{n} \mathbf{P}_{\Psi^r_j}(f) \tag{18}$$

for all $f \in l_2(\mathcal{G})$ and for all $J_1 = 1, \ldots, J$. Thus, for $J_1 = 1, \ldots, J-1$,

$$\mathbf{P}_{\Phi_{J_1+1}}(f) = \mathbf{P}_{\Phi_{J_1}}(f) + \sum_{r=1}^{n} \mathbf{P}_{\Psi^r_{J_1}}(f), \tag{19}$$

which is (11). Moreover, when $J_1 = J$, (18) gives (10). Consequently, (i)$\Longrightarrow$(ii). Conversely, recursively using (19) gives

$$\mathbf{P}_{\Phi_{m+1}}(f) = \mathbf{P}_{\Phi_{J_1}}(f) + \sum_{j=J_1}^{m}\sum_{r=1}^{n}\mathbf{P}_{\Psi_j^r}(f) \tag{20}$$

for all $J_1 \le m \le J-1$. Taking $m = J-1$ together with (10), we deduce (18), which is equivalent to (9). Thus, (ii)$\Longrightarrow$(i).

(ii)$\Longleftrightarrow$(iii). The equivalence between (ii) and (iii) simply follows from the polarization identity.

(ii)$\Longleftrightarrow$(iv). By the orthonormality of $\boldsymbol{u}_\ell$,

$$\langle f, \boldsymbol{\varphi}_{j,p}\rangle = \sum_{\ell=1}^{N}\overline{\widehat{\alpha}\left(\frac{\lambda_\ell}{2^j}\right)}\widehat{f}_\ell\,\boldsymbol{u}_\ell(p), \quad \langle f, \boldsymbol{\psi}_{j,p}^r\rangle = \sum_{\ell=1}^{N}\overline{\widehat{\beta^{(r)}}\left(\frac{\lambda_\ell}{2^j}\right)}\widehat{f}_\ell\,\boldsymbol{u}_\ell(p),$$

where $\widehat{f}_\ell = \langle f, \boldsymbol{u}_\ell\rangle$ is the Fourier coefficient of f with respect to $\boldsymbol{u}_\ell$. This together with (17) and (7) gives, for $j \ge 1$ and $r = 1, \dots, n$, the Fourier coefficients for the projections $\mathbf{P}_{\Phi_j}(f)$ and $\mathbf{P}_{\Psi_j^r}(f)$:

$$\left(\widehat{\mathbf{P}_{\Phi_j}(f)}\right)_\ell = \left|\widehat{\alpha}\left(\frac{\lambda_\ell}{2^j}\right)\right|^2\widehat{f}_\ell, \quad \left(\widehat{\mathbf{P}_{\Psi_j^r}(f)}\right)_\ell = \left|\widehat{\beta^{(r)}}\left(\frac{\lambda_\ell}{2^j}\right)\right|^2\widehat{f}_\ell, \quad \forall \ell = 1, \dots, N, \tag{21}$$

which implies that (10) and (11) are equivalent to (14) and (15) respectively. Thus, (ii)$\Longleftrightarrow$(iv).

(iv)$\Longleftrightarrow$(v). By the relation in (6), it can be deduced that for $\ell = 1, \dots, N$ and $j \ge 1$,

$$\left|\widehat{\alpha}\left(\frac{\lambda_\ell}{2^j}\right)\right|^2 + \sum_{r=1}^{n}\left|\widehat{\beta^{(r)}}\left(\frac{\lambda_\ell}{2^j}\right)\right|^2 = \left(\left|\widehat{a}\left(\frac{\lambda_\ell}{2^{j+1}}\right)\right|^2 + \sum_{r=1}^{n}\left|\widehat{b^{(r)}}\left(\frac{\lambda_\ell}{2^{j+1}}\right)\right|^2\right)\left|\widehat{\alpha}\left(\frac{\lambda_\ell}{2^{j+1}}\right)\right|^2.$$

This shows that (15) is equivalent to (16). Therefore, (iv)$\Longleftrightarrow$(v). ■

C. Framelet Transforms

With framelet system $\mathsf{UFS}_{J_1}^J(\Psi, \boldsymbol{\eta}; \mathcal{G}), J_1 = 1, \dots, J$ introduced in Section A, we define the *framelet coefficients* for a function f on the graph $\mathcal{G}$ by

$$\mathbf{v}_0 = \left\{\langle f, \boldsymbol{\varphi}_{0,p}\rangle\right\}_{p\in V} \text{ and } \mathbf{w}_j^r = \left\{\langle f, \boldsymbol{\psi}_{j,p}^r\rangle\right\}_{p\in V}, \quad j = 0, \dots, J, \;\; r = 1, \dots, n, \tag{22}$$

where $\mathbf{v}_0$ and $\mathbf{w}_j^n$ are the *low-pass* and *high-pass* coefficients for f. The *framelet transforms* are the mapping between the graph signal f and its framelet coefficients $\{\mathbf{v}_0; \mathbf{w}_0^n, \dots, \mathbf{w}_J^n\}$. The (22) can be written as the matrix-vector form, as follows. For the eigenpair $\{(\lambda_\ell, \boldsymbol{u}_\ell)\}_{\ell=1}^N$ for the graph Laplacian $\mathcal{L}$, let $U = [\boldsymbol{u}_1, \dots, \boldsymbol{u}_N]$ be the square matrix (of size $N \times N$) of eigenvectors, and $\Lambda = \mathrm{diag}(\lambda_1, \dots, \lambda_N)$ be the diagonal of eigenvalues. We then have

$$\widehat{\alpha}\left(\frac{\Lambda}{2^{j+1}}\right) = \mathrm{diag}\left(\widehat{\alpha}\left(\frac{\lambda_1}{2^{j+1}}\right), \dots, \widehat{\alpha}\left(\frac{\lambda_N}{2^{j+1}}\right)\right), \quad \widehat{\beta^{(r)}}\left(\frac{\Lambda}{2^{j+1}}\right) = \mathrm{diag}\left(\widehat{\beta^{(r)}}\left(\frac{\lambda_1}{2^{j+1}}\right), \dots, \widehat{\beta^{(r)}}\left(\frac{\lambda_N}{2^{j+1}}\right)\right)$$

the filtered diagonal matrices with low-pass and high-pass filters. Then, the framelet coefficients have the following representation.

Proposition 2 *With the notation given above,*

$$\mathbf{v}_0 = U\widehat{\alpha}\left(\frac{\Lambda}{2}\right)U^\top f \;\; \textit{and} \;\; \mathbf{w}_j^r = U\widehat{\beta^{(r)}}\left(\frac{\Lambda}{2^{j+1}}\right)U^\top f \quad \forall j = 0, \dots, J, \; r = 1, \dots, n. \tag{23}$$

C.1. Decomposition and Reconstruction

We call estimating coefficients from f *framelet decomposition*, and its inverse *framelet reconstruction*. In our construction for the framelets and framelet transforms, the framelet decomposition and reconstruction are *invertible*. The decomposition

and reconstruction can be achieved efficiently via a *filter bank* $\boldsymbol{\eta} = \{a; b^{(1)}, \ldots, b^{(n)}\}$. We use the filter bank of Haar-type in Dong (2017), by which the decomposition and reconstruction can be implemented recursively in a fast algorithm. By repeated use of the refinement equation (6) of the filter, we arrive at the following transforms, which pave the way for efficiently computing framelet coefficients in (23). For $r = 1, \ldots, n$ and $j = 1, \ldots, J$, we define operators for $(r, j) \in \{(1,1), \ldots, (1, J), \ldots (n, 1), \ldots, (n, J)\} \cup \{(0, J)\}$ by, for $f \in l_2(\mathcal{G})$,

$$\begin{aligned} \mathcal{W}^{\flat}_{r,1} f &= U\widehat{b^{(r)}}\left(2^{-K}\Lambda\right) U^{\top} f, \\ \mathcal{W}^{\flat}_{r,j} f &= U\widehat{b^{(r)}}\left(2^{K+j-1}\Lambda\right) \widehat{a}\left(2^{K+j-2}\Lambda\right) \ldots \widehat{a}\left(2^{-K}\Lambda\right) U^{\top} f, \quad \forall j \geq 2, \end{aligned} \tag{24}$$

where as mentioned before, K is the real value such that the graph Laplacian's biggest eigenvalue $\lambda_{\max} \leq 2^K \pi$. With the transforms $\mathcal{W}^{\flat}_{r,j}$ in (24), we can write the the decomposition and reconstruction explicitly, as follows.

Theorem 3 (Framelet Decomposition and Reconstruction) *The framelet decomposition can be achieved via filter bank $\boldsymbol{\eta}$ recursively: for $r = 1, \ldots, n$ and $j = 1, \ldots, J$,*

$$\mathbf{v}_0 = \mathcal{W}^{\flat}_{0,J} f \;\; \textit{and} \;\; \mathbf{w}^r_j = \mathcal{W}^{\flat}_{r,j} f. \tag{25}$$

The reconstruction for a set of coefficients $\{\mathbf{v}_0\} \cup \{\mathbf{w}^1_j, \ldots, \mathbf{w}^n_j\}_{j=1}^J$ on $\mathcal{G}$ can be computed by

$$f_J = \mathcal{W}^{\flat,\star}_{0,J}\, \mathbf{v}_0 + \sum_{j=1}^{J} \sum_{r=1}^{n} \mathcal{W}^{\flat,\star}_{r,j}\, \mathbf{w}^r_j, \tag{26}$$

where the $\star$ indicates the conjugate transpose of the associated matrix. The decomposition and reconstruction in (25) and (26) are invertible, that is, $f_J = f$.

Proof We prove the invertibility between (25) and (26). As mentioned above, the framelet coefficients in (25) are equivalent with (23) and then with the original definition in (22), due to that the scaling and filter functions satisfy the refinement equation (6). By the equivalence between Theorem 1(v) and (ii), using the framelet coefficients in (25) for (26), we thus obtain $f_J = f$. ■

Note that we count the level index j in (29) from 1, which is in essence the same as (9) of Theorem 26, and Theorem 1 below.

C.2. Tensor-based $\mathcal{G}$-framelet Transforms

Due to the computational difficulty of eigendecomposition for the large-scale graph Laplacian matrix, the decomposition and reconstruction in Theorem 3 cannot be directly computed in an efficient way. To fast evaluate them, we apply the approximation by Chebyshev polynomials $\mathcal{T}_0, \ldots, \mathcal{T}_n$ of a fixed degree t, for the filter $a \approx \mathcal{T}_0$ and $b^{(r)} \approx \mathcal{T}_r$. Here t is a sufficiently large integer such that the Chebyshev polynomial approximation is of high precision. Then, the (24) can be approximated by

$$\begin{aligned} \mathcal{W}^{\flat}_{r,1} f &= U\widehat{b^{(r)}}\left(2^{-K}\Lambda\right) U^{\top} f \approx U\mathcal{T}_r\left(2^{-K}\Lambda\right) U^{\top} f, \\ \mathcal{W}^{\flat}_{r,j} f &= U\widehat{b^{(r)}}\left(2^{K+j-1}\Lambda\right) \widehat{a}\left(2^{K+j-2}\Lambda\right) \ldots \widehat{a}\left(2^{-K}\Lambda\right) U^{\top} f \\ &\approx U\mathcal{T}_r\left(2^{K+j-1}\Lambda\right) \mathcal{T}_0\left(2^{K+j-2}\Lambda\right) \ldots \mathcal{T}_0\left(2^{-K}\Lambda\right) U^{\top} f, \quad \forall j \geq 2. \end{aligned} \tag{27}$$

By the property of the polynomial of matrix and the eigendecomposition of the graph Laplacian $U\Lambda U^{\top} = \mathcal{L}$, the (27) then becomes

$$\begin{aligned} \mathcal{W}^{\flat}_{r,1} f &\approx \mathcal{T}_r\left(2^{-K}\mathcal{L}\right) f =: \mathcal{W}_{r,1} f, \\ \mathcal{W}^{\flat}_{r,j} f &\approx \mathcal{T}_r\left(2^{K+j-1}\mathcal{L}\right) \mathcal{T}_0\left(2^{K+j-2}\mathcal{L}\right) \ldots \mathcal{T}_0\left(2^{-K}\mathcal{L}\right) f =: \mathcal{W}_{r,j} f, \quad \forall j \geq 2. \end{aligned} \tag{28}$$

With filters given, we can pre-compute the coefficients for the Chebyshev polynomial approximation up to degree t by a quadrature rule. The framelet decomposition and reconstruction can be evaluated by the approximate transforms $\mathcal{W}_{r,j}$ analogously to Theorem 3, as follows.

Theorem 4 (Framelet Transforms by Chebyshev Polynomial Approximation) *The framelet decomposition and reconstruction can be approximated by: for $r = 1, \ldots, n$ and $j = 1, \ldots, J$,*

$$\mathbf{v}_0 = \mathcal{W}_{0,J} f \;\; \textit{and} \;\; \mathbf{w}_j^r = \mathcal{W}_{r,j} f.$$

The reconstruction for a set of coefficients $\{\mathbf{v}_0\} \cup \{\mathbf{w}_j^1, \ldots, \mathbf{w}_j^n\}_{j=1}^J$ on $\mathcal{G}$ can be evaluated by

$$f_J = \mathcal{W}_{0,J}^\star \, \mathbf{v}_0 + \sum_{r=1}^{n} \sum_{j=1}^{J} \mathcal{W}_{r,j}^\star \, \mathbf{w}_j^r, \tag{29}$$

where the $\star$ indicates the conjugate transpose of the associated matrix.

As mentioned, with proper alignment of $\mathcal{W}_{r,j}$, we have a tensor-based evaluation for framelet transforms. Define the matrix (of size $(nJ+1)N \times N$)

$$\mathcal{W}^\natural = \left[\mathcal{W}_{0,J}, \mathcal{W}_{1,J}, \ldots, \mathcal{W}_{n,J}, \ldots, \mathcal{W}_{1,J}, \ldots, \mathcal{W}_{n,J}\right]^\top. \tag{30}$$

Let c be the concatenation of coefficients $\mathbf{v}_0, \mathbf{w}_1^1, \ldots, \mathbf{w}_1^n, \ldots, \mathbf{w}_J^1, \ldots, \mathbf{w}_J^n$ associated with a framelet system with n high passes and graph with N nodes. Then, c is a column vector of length $(nJ+1)N$. The following corollary gives the decomposition and reconstruction for the tensor-based $\mathcal{G}$-framelet transforms, as used in the main part and experiments.

Corollary 5 *The coefficients from framelet decomposition up to scale level J is given by*

$$c = \mathcal{W}^\natural f$$

and the framelet reconstruction is given by

$$f_J = (\mathcal{W}^\natural)^\top c.$$

D. Shrinkage for Wavelet Denoising and LASSO

Wavelets play an important role in denoising. The typical model is to learn a function f on $[0,1]$ from one dimensional noisy data

$$y_i = f(x_i) + \sigma \epsilon_i, \quad i = 1, \ldots, N, \tag{31}$$

for given σ and i.i.d. white noise ϵ_i. Donoho (1995) gave a method of finding an approximate f from (31) by 1D wavelets and soft-thresholding shrinkage. Typical steps include:

1. Apply pyramid wavelet filtering (Cohen et al., 1993) for the scaled input data $y_i/\sqrt{N}$, which then yields noisy wavelet coefficients up to scale level J: $w_{j,k}$, $j = 0, \ldots, J$, $k = 0, \ldots, 2^j - 1$.

2. Use soft-threshold nonlinearity to the high-pass wavelets $w_{j,k}$, with threshold value $\sigma\sqrt{2\log(N)/N}$. This then gives an estimate $w_{j,k}^\sharp$ for the original wavelet coefficients.

3. Reconstruct the signal by using inverse wavelet transforms for the shrinkaged coefficients.

This shrinkage method is also used as an example of statistical multivariate estimation by Efron & Morris (1975).

Moreover, shrinkage plays a pivotal role in LASSO that estimates the coefficients of regression

$$\min_{\alpha,\beta_j} \left\{ \sum_{i=1}^{N} \left(y_i - \alpha - \sum_j \beta_j x_{i,j} \right) \right\} \quad \text{subject to } \sum_j |\beta_j| \le t. \tag{32}$$

The soft-thresholding provides a solution to (32), and thus a regression selection for variables. The unbiased estimate of the risk or mean-squared error for the estimator $\widehat{\beta}$ is equivalent to the risk for the shrinkage wavelet denoising approximator in Donoho & Johnstone (1994).

Connection to Data Compression With the above shrinkage for wavelet denoising problem (31), we can remove the noise and also compress the wavelet representation which considerably reduces the number of coefficients. The inverse wavelet transform for thresholded coefficients compresses the l_2-energy of the signal into a few number of large wavelet coefficients but the Gaussian white noise in the signal or wavelet coefficients does not change their energy level. Thus, the large coefficients that contain main information of the original signal can be separated and distilled from the white noise. See Donoho (1992); Dong & Shen (2013).

E. Comparison of Existing Spectral-based GNNs

We compare the key characteristics of our proposed UFGCONV with four classic spectral convolution models for a better understanding. The summary is provided in Table 4.

Table 4. Brief summary for the existing spectral-based GNNs.

Model	Graph convolution	Network filter	Computational strategy	Multi-scale
SPECTRALCNN (Bruna et al., 2014)	$y = U\boldsymbol{\theta}' U^T X$	$\boldsymbol{\theta}' = \text{Diag}(\boldsymbol{\theta})$	No, U is based on $L = U\Lambda U^T$	$\times$
CHEBYSHEV (Defferrard et al., 2016a)	$y = U\boldsymbol{\theta}' U^T X$	$\boldsymbol{\theta}' = \sum_{k=0}^{K-1} \alpha_k \Lambda^k$	No, $y = \sum_{k=0}^{K-1} \alpha_k L^k$	$\times$
GWNN (Xu et al., 2019a)	$y = \Psi_s \boldsymbol{\theta}' \Psi_s^{-1} X$	$\boldsymbol{\theta}' = \text{Diag}(\boldsymbol{\theta})$	Yes, Ψ_s is graph wavelet transform	$\surd$
HANET (Li et al., 2020b)	$y = \Phi \boldsymbol{\theta}' \Phi^T x$	$\boldsymbol{\theta}' = \text{Diag}(\boldsymbol{\theta})$	Yes, Φ is Haar transforms	$\surd$
UFGCONV-R	$y = \text{ReLU}\left(\mathcal{V}\left(\boldsymbol{\theta}'(\mathcal{W}X)\right)\right)$	$\boldsymbol{\theta}' = \text{Diag}(\boldsymbol{\theta})$	$\mathcal{W}, \mathcal{V}$ are framelet transforms	$\surd$
UFGCONV-S	$y = \mathcal{V}\left(\text{Shrinkage}\left(\boldsymbol{\theta}'(\mathcal{W}X)\right)\right)$	$\boldsymbol{\theta}' = \text{Diag}(\boldsymbol{\theta})$	$\mathcal{W}, \mathcal{V}$ are framelet transforms	$\surd$

† $G_s = \text{diag}\{e^{s\lambda_1}, \ldots, e^{s\lambda_N}\}$ and Ψ_s can be approximated fast via Chebyshev polynomial (Hammond et al., 2011)

F. Experiments

In this section, we provide more details for each experiment conducted in this paper. Also, we attach more experiment results regarding the robustness analysis of our proposed UFGCONV on **Citeseer** and **Pubmed** in Section F.2, and the visualizations of the shrinkage effect on **Cora** in Section F.6. For all the experiments, we initialize parameters of our model UFGCONV according to: *zeros* for the bias term; *xavier uniform* (Glorot & Bengio, 2010) for the weight matrix W; and *uniform distribution* $\mathcal{U}(0.9, 1.1)$ for the network filter $\boldsymbol{\theta}$.

F.1. Node Classification

The first experiment of node classification, which corresponds to the results in Table 1, is performed on the benchmark citation networks. For both UFGCONV-S and UFGCONV-R, we use default values for the scale level $(J = 2)$ and the dilation $(\text{base} = 2)$. The rest of the hyperparameters are tuned by using a grid search with the following search spaces listed in the "First experiment" column in Table 5.

The second node classification task whose results shown in Table 2 is conducted on **ogbn-arxiv** from **OGB** (Hu et al., 2020). It demonstrates the superior performance of our proposed model UFGCONV on large-scale graph-structured data. We select a variety of existing models with their publicly available prediction performances as the baselines. A full list of the leaderboard can be found in the OGB website[1]. For a fair comparison, our model aligns with the model structure of GCN (Kipf & Welling, 2017) and GRAPHSAGE's (Hamilton et al., 2017) that produces the scores on **ogbn-arxiv** leaderboard: three convolutional layers with one dropout layer inserted between every two consecutive convolutional layers. Moreover, a batch normalization (Ioffe & Szegedy, 2015) is applied after each convolutional layer (before the ReLU activation function for UFGCONV-R or after the reconstruction process for UFGCONV-S). Again, a grid search is performed for fine tuning the hyperparameters in the search spaces listed in the "Second experiment" column in Table 5.

The results of the trade-off analysis with **ogbn-arxiv** in Figure 5 correspond to multiple runs of UFGCONV-S with different shrinkage thresholds $\sigma = [1, 3, 5, 7, 9]$. Other hyperparameters are fixed at: 0.001 for learning rate, 0.001 for weight decay, 2.0 for dilation, 0.5 for dropout, and 3 for scale level.

Table 6 documents some descriptive statistics of the datasets used for the node classification tasks.

[1] https://ogb.stanford.edu/docs/leader_nodeprop/#ogbn-arxiv

Table 5. Hyperparameter searching space for node classification.

Hyperparameters	**First experiment**	**Second experiment**
Learning rate	5e-2, 1e-2, 5e-3	5e-3, 1e-3, 5e-4
Weight decay (L_2)	5e-2, 1e-2, 5e-3	5e-3, 1e-3, 5e-4
Hidden size	16, 32, 64	128, 256
Dropout ratio	0.5, 0.6, 0.7	0.5, 0.6, 0.7
Scale level	-	2,3
Dilation	-	1.5, 2.0, 2.5

Table 6. Summary of the datasets for node classification tasks.

	Cora	**Citeseer**	**Pubmed**	**ogbn-arxiv**
# Nodes	$2,708$	$3,327$	$19,717$	$169,343$
# Edges	$5,429$	$4,732$	$44,338$	$1,166,243$
# Features	$1,433$	$3,703$	500	128
# Classes	7	6	3	40
# Training Nodes	140	120	60	$90,941$
# Validation Nodes	500	500	500	$29,799$
# Test Nodes	$1,000$	$1,000$	$1,000$	$48,603$
Label Rate	0.052	0.036	0.003	0.537

F.2. Robustness Analysis on Citeseer and Pubmed

This section supplements the perturbation analysis in Section 6. In this experiment, we compare our UFGConv (both shrinkage and ReLU models) against GCN (Kipf & Welling, 2017) and GAT (Veličković et al., 2018) on the perturbed citation networks. All the models consist of two convolutional layers with a dropout layer inserted in between. The hyperparameters are, unless further specified, tuned in the same way with the same searching spaces as illustrated in the first experiment of Section F.1. The shrinkage threshold σ is searched over $\{0.05, 0.10, 0.15\}$. For node attribute perturbation on **Pubmed**, the dropout ratio is searched from the set $\{0, 0.5\}$; the learning rate is searched from a larger set of {1e-2, 5e-3, 1e-3, 5e-4}.

The structure noise on all three datasets are generated analogously. We define the noise ratio as the number of connected node pairs in the new graph divided by the number of connected node pairs in the original graph. The ratio at 1 represents the undistorted graph. For node attribute perturbation, noise defined on **Cora** and **Citeseer** follows Bernoulli distribution, where we change a small portion of node features from 0 (or 1) to 1 (or 0). A selection of noise ratios in the range from 0.1 to 2 with a step size 0.1 are considered. On **Pubmed**, as the raw data are real values, we add a Gaussian noise with zero mean and standard deviation σ_s on the nodes, where σ_s is the noise ratio in the plot. The noise ratio ranges from 0.01 to 0.15 with step size 0.01 union $\{0.005, 0.015, 0.025\}$.

We refer the reader to Figure 3 in the main text for the experimental result on **Cora**, and Figures 7-8 for **Citeseer** and **Pubmed**.

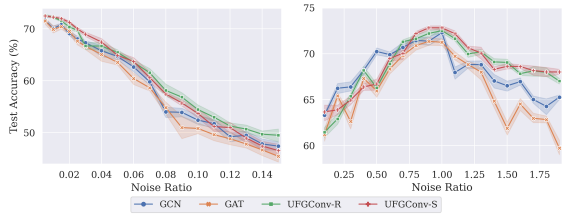

Figure 7. Node attribute (left) and graph structure (right) perturbation analysis on **Citeseer**.

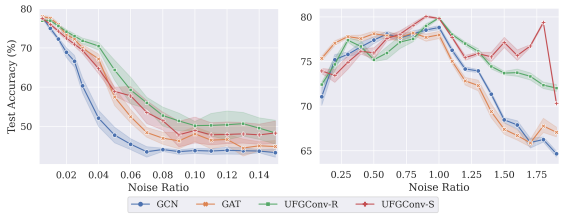

Figure 8. Node attribute (left) and graph structure (right) perturbation analysis on **Pubmed**.

The performance of our proposed UFGCONV (with either shrinkage or ReLU activation) on **Citeseer** and **Pubmed** aligns with the superior performance on **Cora**, which indicates that our models are more robust than the baseline methods, especially on datasets added with more noise.

F.3. Graph Classification and Regression

We supplement basic information of the datasets for graph property prediction tasks in Section 8.2. We report the descriptive statistics in Table 7 for the six benchmark datasets used in this experiment. The 'R' in the bracket of the last row of **QM7** represents regression task.

Table 7. Summary of the datasets for the graph property prediction tasks.

Datasets	**PROTEINS**	**Mutagenicity**	**D&D**	**NCI1**	**ogbg-molhiv**	**QM7**
# Graphs	1,113	4,337	1,178	4,110	41,127	7,165
Min # Nodes	4	4	30	3	2	4
Max # Nodes	620	417	5,748	111	222	23
Avg # Nodes	39	30	284	30	26	15
Avg # Edges	73	31	716	32	28	123
# Features	3	14	89	37	9	0
# Classes	2	2	2	2	2	1 (R)

Table 8. Hyperparameters for graph property prediction and sensitivity analysis.

Hyperparameters	**Graph classify and regress**	**Sensitivity analysis**
Learning rate	5e-3, 1e-3, 5e-4	5e-2, 1e-2, 5e-3
Weight decay (L_2)	5e-3, 1e-3, 5e-4	5e-2 1e-2, 5e-3
Hidden size	16, 32, 64	16, 32, 64
Dropout ratio	0, 0.5	0.6, 0.7

The model architecture is set to 2 GCN convolutional layers for **TU Dataset**s or 4 GIN convolutional layers for **OGB**, which is consistent with the general **OGB** framework. We summarize the searching spaces of the key hyperparameters in Table 8. Other hyperparameters, if not specified, are set to default values.

F.4. Sensitivity Analysis

The experiments documented in Section 8.3 analyze the sensitivity of our proposed UFGCONV on the hyperparameters *dilation* and *scale level*. For each dataset (**Cora**, **Citeseer** or **Pubmed**), we fix the optimal hidden size to 16, 32 and 64; the scale level to 2, 2 and 3 (for dilation analysis); the dilation to 2, 2 and 2 (for scale level analysis), respectively. We use shrinkage threshold $\sigma = 1$ for UFGCONV-S. We examine the sensitivity on dilation ranging from 1.25 to 4 with step 0.25, and on scale level ranging from 1 to 8 with step 1. The rest of the hyperparameters are searched over the spaces listed in Table 8.

F.5. Computational Complexity

In Section 3, we discuss the theoretical time and space complexities of the tensor-based framelet transforms which are the key procedures of our proposed UFGCONV. In this section, we empirically examine the time complexity of our models against some classic graph convolutions (GCN (Kipf & Welling, 2017), GAT (Veličković et al., 2018) with 8 attention heads, and CHEBYSHEV (Defferrard et al., 2016b) with polynomial degree 3) on the 12 random Erdös-Rényi graphs with different node sizes. In this experiment, we only time one forward propagation of each method and then repeat this procedure for 1,000 times. The experiment is conducted on an NVIDIA® Tesla V100 GPU card with 5,120 CUDA cores and 16GB HBM2 mounted on an HPC cluster.

From Figure 9, we can observe that UFGCONV-S has a slightly higher computational cost than its ReLU counterparts UFGCONV-R under the same scale level, due to the extra steps for calculating the shrinkage threshold and trimming the framelet coefficients. However, the sensitivity analysis in Section 8.3 suggests that a small scale level is sufficient to achieve

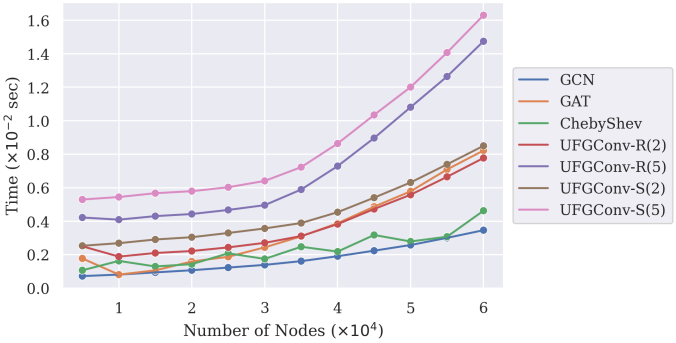

Figure 9. Computational cost (in seconds) against the node sizes of 12 randomly generated graphs. The result is the average of $1,000$ runs. The value reported in the brackets after the name of our model in the legend indicates the corresponding scale level J.

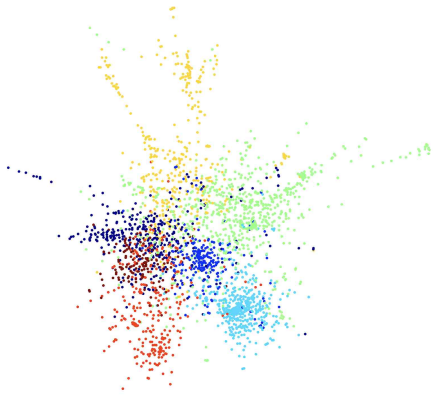

Figure 10. Visualization of UFGCONV-S on **Cora**, with each color indicates one of 7 classes that nodes come from.

a better performance. Both of our proposed models (when scale level set to 2) have a similar computational complexity as GAT (Veličković et al., 2018) (with 8 attention heads). As the node size of a graph increases, the speed of GAT degrades rapidly and our models are faster than GAT when the graph has more than 4×10^4 nodes.

F.6. Visualization for UFGCONV with Shrinkage on Cora

In this section, we visually demonstrate the effectiveness of the shrinkage activation on **Cora**. A complete graph of all seven classes in **Cora** is displayed in Figure 10. We adopt UFGCONV-S on **Cora** with a compression ratio of 47.7% from Table 1 in the main text.

The next seven figures visualize the framelet coefficients for all seven classes. In particular, we compare on each filter the coefficients at initialization, after two convolutional layers before shrinkage activation, and after shrinkage activation. All the values are retrieved under the evaluation mode, which means we leave the effect of random dropout. Also, the compressed results are only reported in the two high-pass filters, since the shrinkage activation does not apply on the low pass.

For all the seven sets of results, a vertical comparison between the initialization and the other two columns indicate clearly that all the three passes learn the coefficients according to the true class information. Nodes with respect to the corresponding classes usually have higher absolute values that are more distinct from 0. Also, coefficients on both high-passes compressed a critical part of coefficients after shrinkage (in green). The majority of leftovers, except for those from the highlighted class, have close-to-zero coefficients.

For a horizontal comparison, low-pass coefficients usually have higher but less distinctive values, while high-pass coefficients are more concentrated on the detailed information with respect to the individual classes.

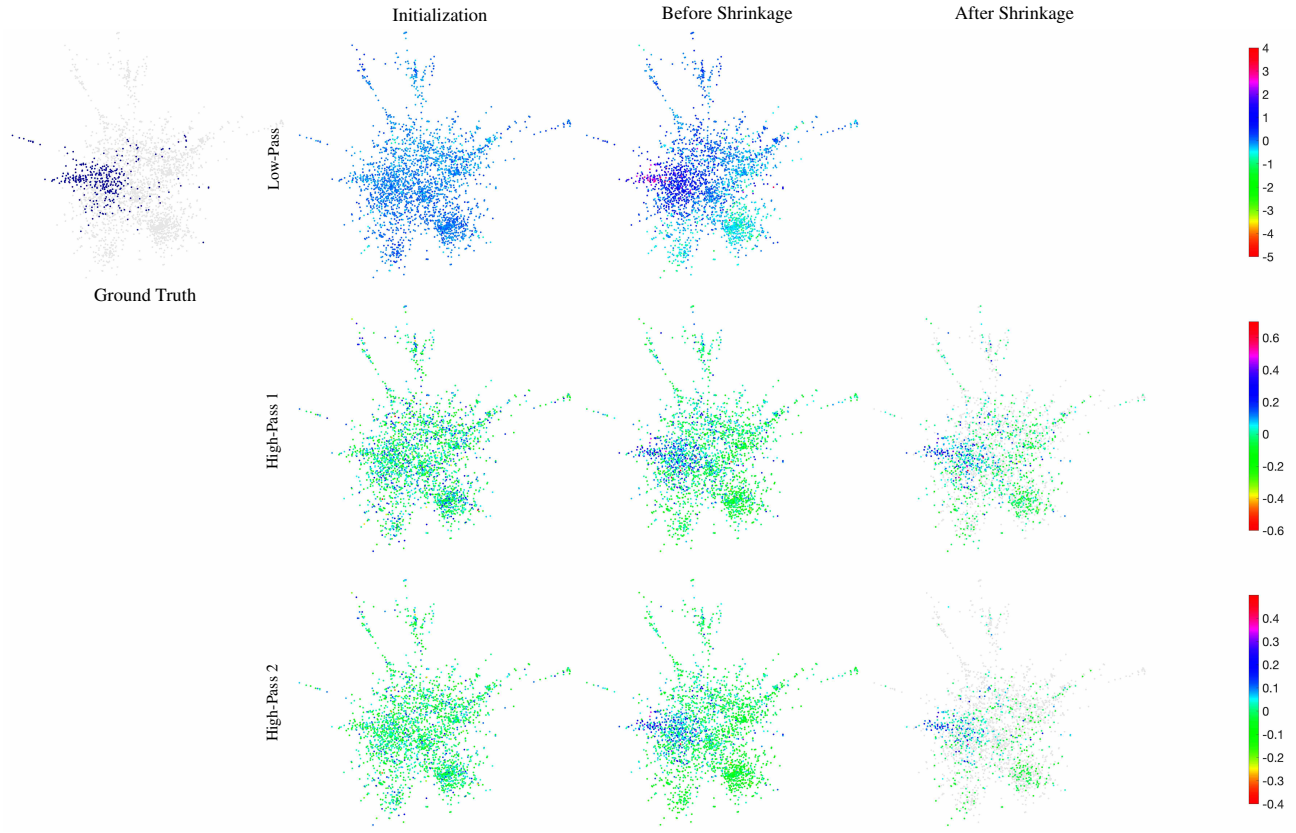

Figure 11. Framelet coefficients on **Cora**, Class 1.

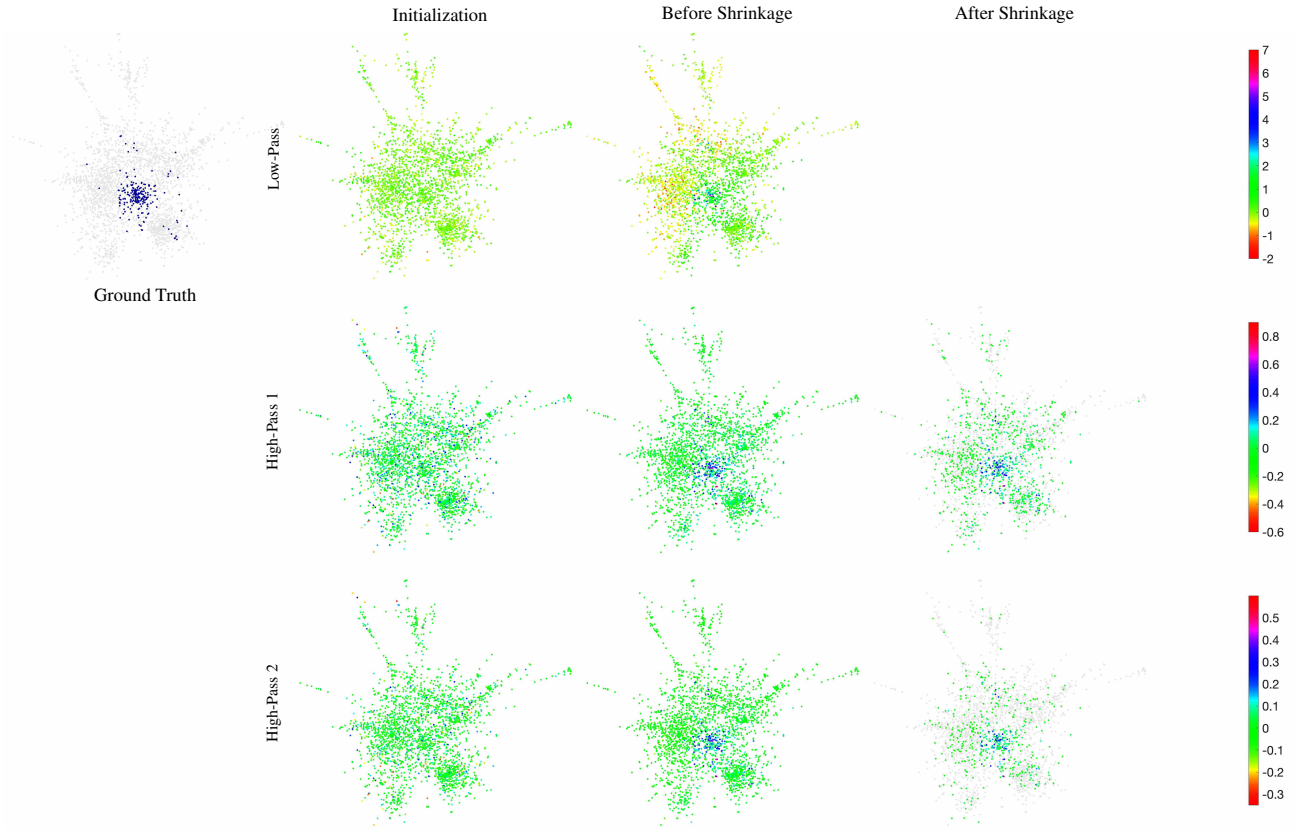

Figure 12. Framelet coefficients on **Cora**, Class 2.

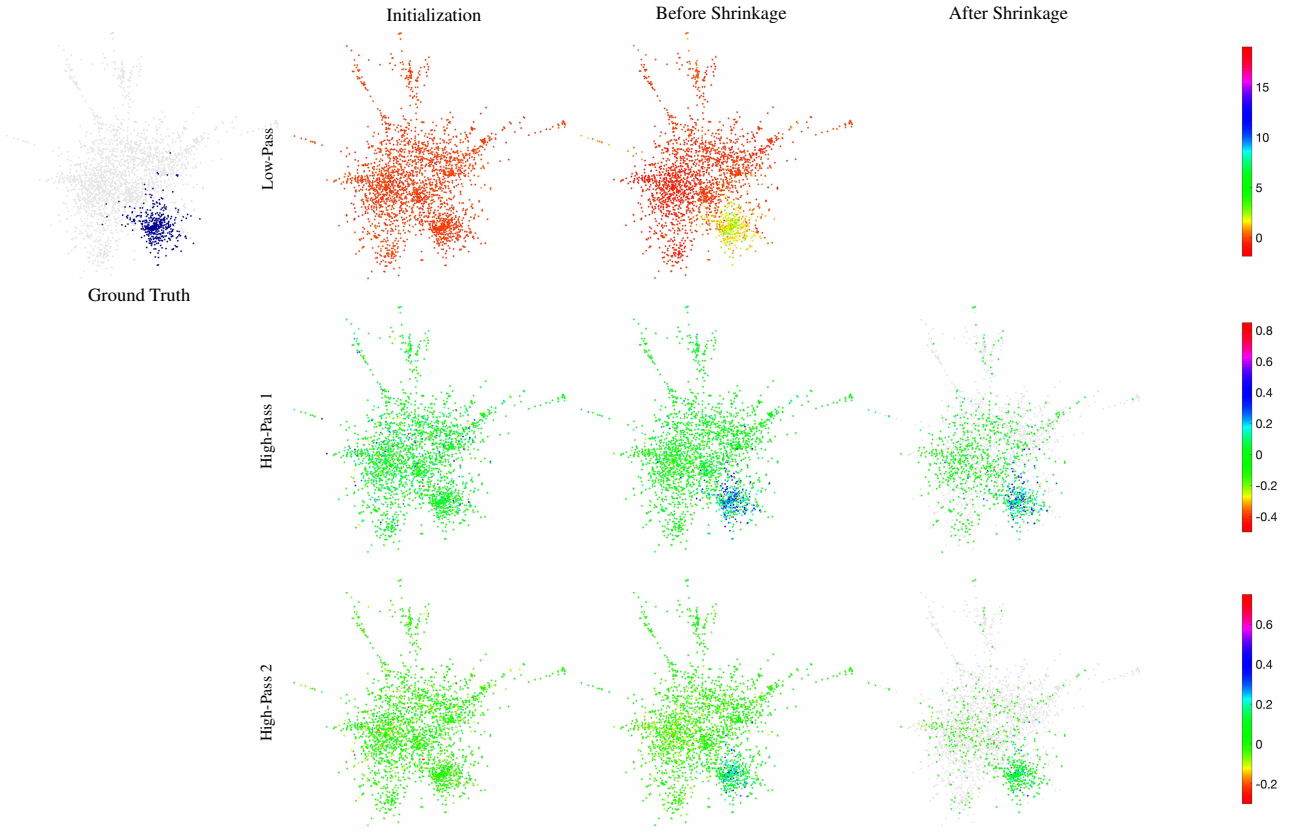

Figure 13. Framelet coefficients on **Cora**, Class 3.

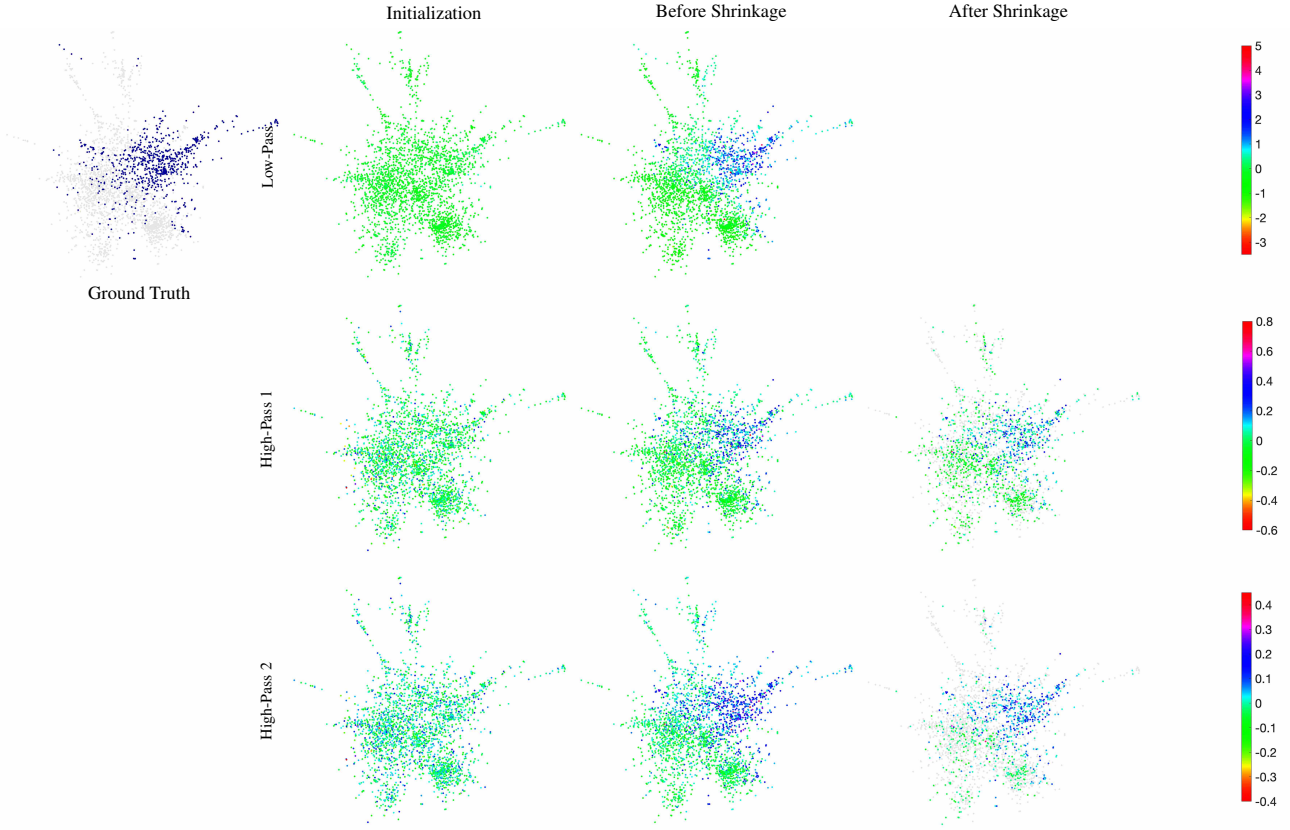

Figure 14. Framelet coefficients on **Cora**, Class 4.

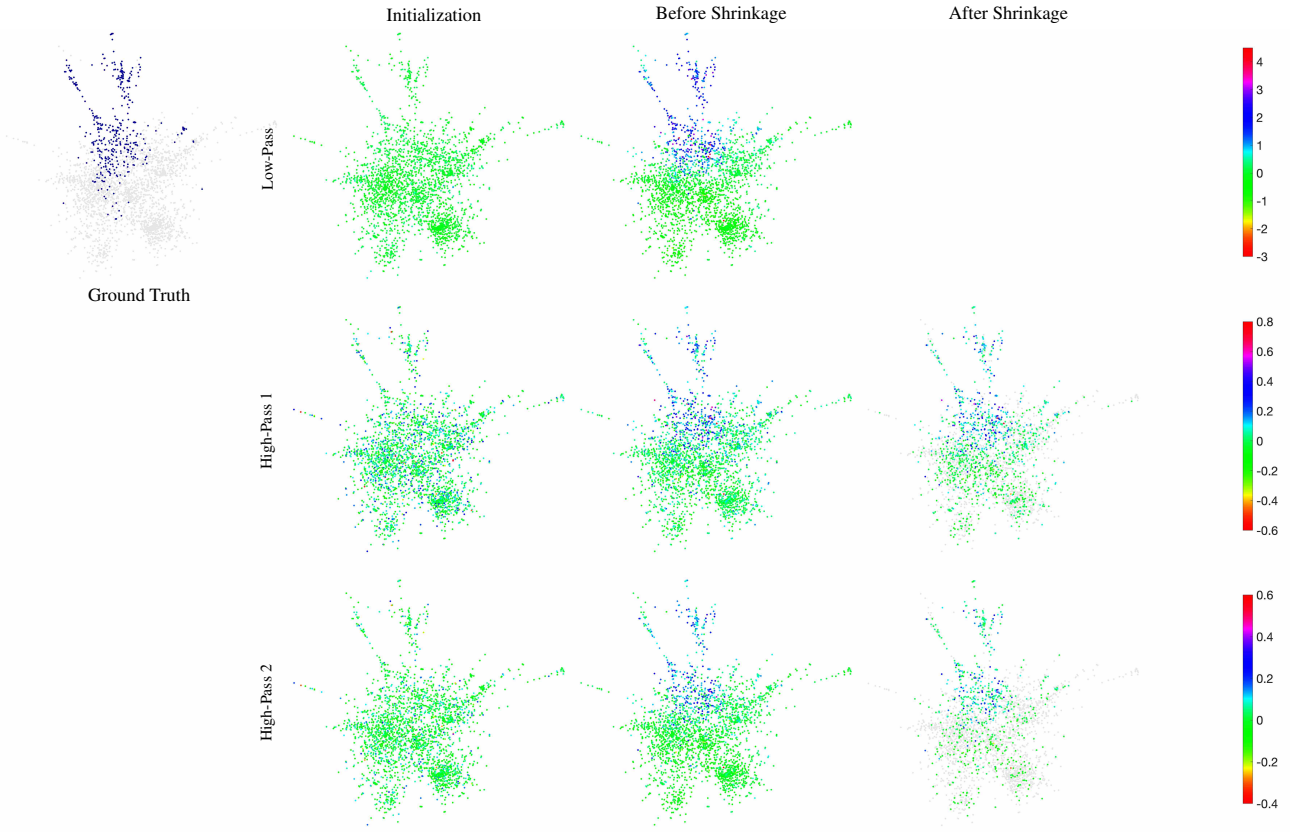

Figure 15. Framelet coefficients on **Cora**, Class 5.

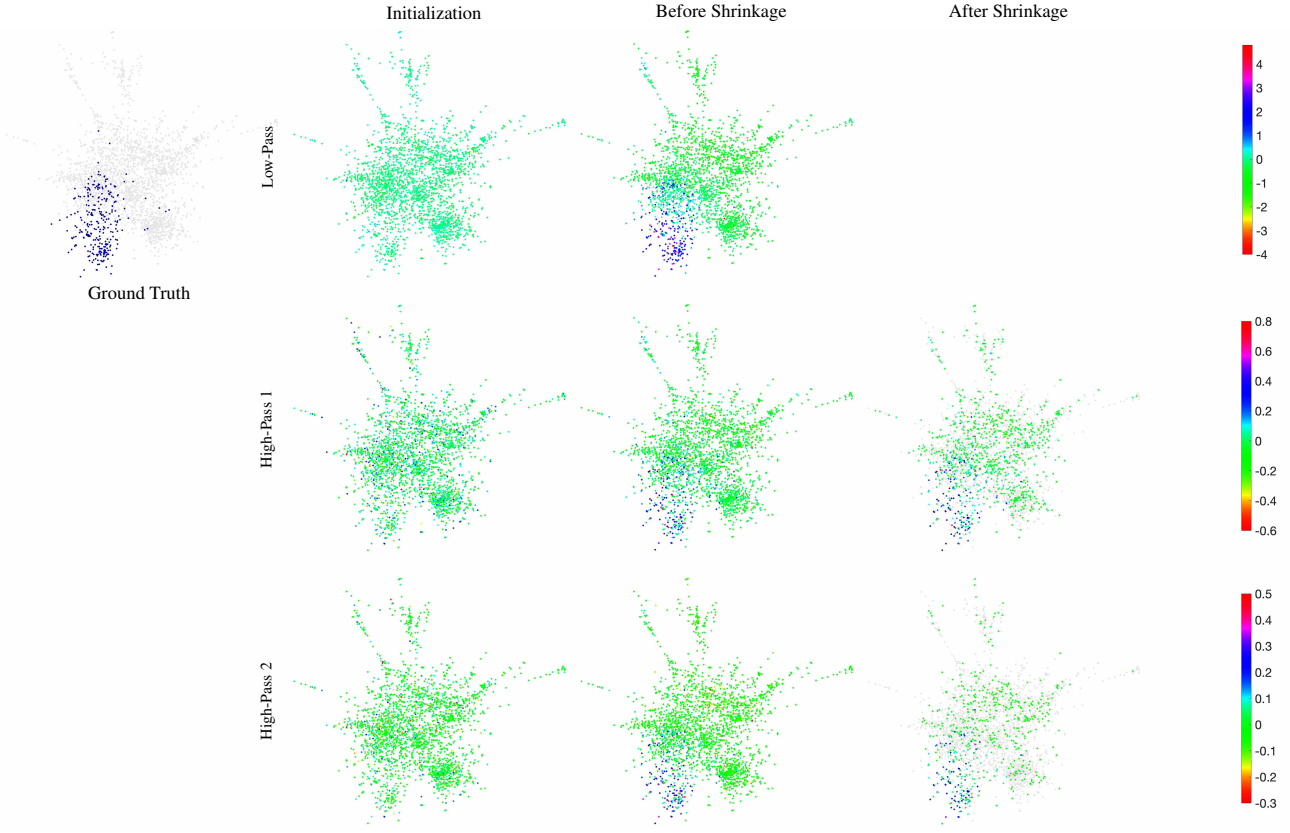

Figure 16. Framelet coefficients on **Cora**, Class 6.

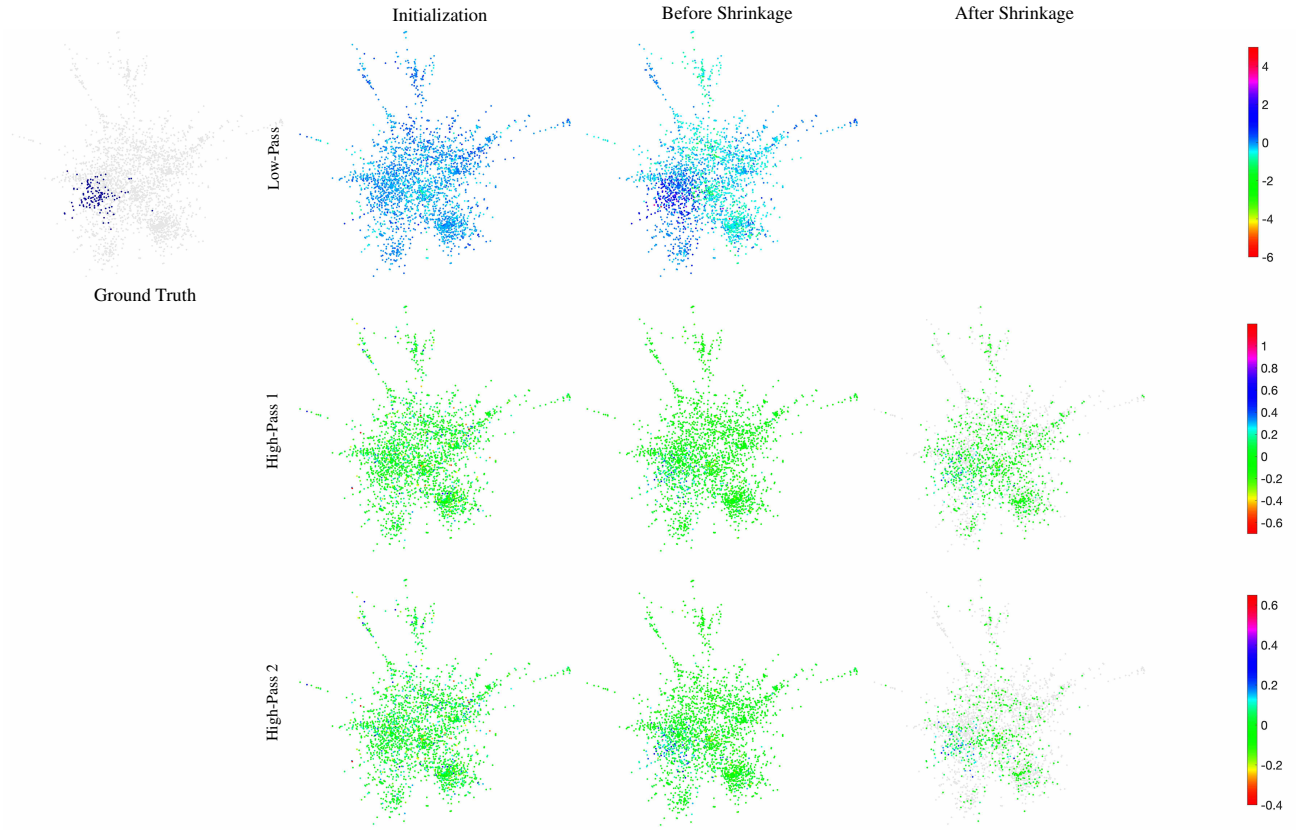

Figure 17. Framelet coefficients on **Cora**, Class 7.