Restructuring Graph for Higher Homophily via Learnable Spectral Clustering

Shouheng Li  
The Australian National University  
Data61, CSIRO  
Australia  
shouheng.li@anu.edu.au

Dongwoo Kim  
GSAI, POSTECH  
South Korea  
dongwokim@postech.ac.kr

Qing Wang  
The Australian National University  
Australia  
qing.wang@anu.edu.au

Abstract

While a growing body of literature has been studying new Graph Neural Networks (GNNs) that work on both homophilic and heterophilic graphs, little work has been done on adapting classical GNNs to less-homophilic graphs. Although lacking the ability to work with less-homophilic graphs, classical GNNs still stand out in some properties such as efficiency, simplicity and explainability. We propose a novel graph restructuring method to maximize the benefit of prevalent GNNs with the homophilic assumption. Our contribution is threefold: a) learning the weight of pseudo-eigenvectors for an adaptive spectral clustering that aligns well with known node labels, b) proposing a new homophilic metric that measures how two nodes with the same label are likely to be connected, and c) reconstructing the adjacency matrix based on the result of adaptive spectral clustering to maximize the homophilic scores. The experimental results show that our graph restructuring method can significantly boost the performance of six classical GNNs by an average of 25% on less-homophilic graphs. The boosted performance is comparable to state-of-the-art methods.

1 Introduction

Graph Neural Networks (GNNs) were originally inspired under the homophilic assumption - nodes of a graph with the same label are more likely to be connected than nodes with different labels. Recent studies reveal the limitations of this homophilic assumption when GNNs are applied on less-homophilic or heterophilic graphs (Pei et al., 2020). Ever since, a number of approaches have focused on developing deep learning architectures for heterophilic graphs (Zhu et al., 2021; Kim and Oh, 2021; Chien et al., 2021; Li et al., 2021; Bo et al., 2021; Lim et al., 2021). However, little work has been done on adapting classical GNNs to less-homophilic graphs. Although classical GNNs are lacking the ability to work with less-homophilic graphs, they still stand out in some properties such as efficiency (Zeng et al., 2020), simplicity (Wu et al., 2019) and explainability (Ying et al., 2019).

Our work studies a graph restructuring method that reconstructs the adjacency matrix of a graph from scratch to uplift the benefit of prevalent homophilic GNNs. Our work is partially inspired by Klicpera et al. (2019b), we extend the restructuring approach to heterophilic graphs and improve the performance of GNNs which do not work well on heterophilic graphs. Our observation from spectral clustering of heterophilic graphs shows that the eigenvectors corresponding to the leading eigenvalues

Preprint. Under review.
do not always align well with the node labels. Particularly, in a heterophilic graph, two adjacent nodes are unlikely to have the same label, which is in contradiction with the smoothness properties of leading eigenvectors. However, when we choose eigenvectors appropriately, the correlation between the similarity of spectral features and node labels increases. To generalize this observation, we propose a learnable spectral clustering algorithm that, a) divides the Laplacian spectrum into even slices called pseudo-eigenvalues, each slice corresponds to an embedding matrix called pseudo-eigenvector; b) learns the weights of pseudo-eigenvectors from existing node labels; c) restructures the graph according to node embedding distance to maximize homophily. To measure the homophilic level of graphs, we further introduce a new metric that addresses some limitations of the previous metrics. Our experimental results show that the performances of node-level prediction tasks with restructured graphs are greatly improved on classical GNNs.

2 Related Work

GNNs for heterophilic graphs. Early GNNs assume homophily implicitly. Such an inductive bias results in a degenerated performance on less-homophilic graphs (Lim et al., 2021). Recently, homophily is also shown to be an effective measure of a graph’s robustness to both over-smoothing and adversarial attacks. In the over-smoothing setting, node representations converge to a stationary state of similar values (“over-smoothed”) as a GNN goes deeper. A graph of low homophily is more prone to this issue as the stationary state is reached with fewer GNN layers (Yan et al., 2021). Similarly, homophilic graphs are more resilient to a graph injection attack than their heterophilic counterparts in the adversarial attack setting. Some techniques defend against such attacks by improving or retaining homophily under graph injection (Zhu et al., 2021a; Chen et al., 2022). Pei et al. (2020) firstly draw attention to the limitation of GNN on less-homophilic graphs. Since then, various GNNs have been proposed to improve performance on these graphs. H_{GCN} (Zhu et al., 2021b) show that proper utilization of ego-embedding, higher-order neighbourhoods, and intermediate embeddings can improve results in heterophilic graphs. A recent scalable model, LINKX (Lim et al., 2021), shows separating feature and structural embedding improves performance. Kim and Oh (2021) study this topic specifically for graph attention and finds improvements when an attention mechanism is chosen according to homophily and average degrees. Chien et al. (2021) propose to use a generalized PageRank method that learns the weights of a polynomial filter and show that the model can adapt to both homogeneous and heterophilic graphs. Similarly, Li et al. (2021) use learnable spectral filters for achieving an adaptive model on graphs of different homophilic levels. Zhu et al. (2021b) recently propose to incorporate a learnable compatibility matrix into GNN-based methods to handle heterophily of graphs.

Learning spectral clustering. From the deep learning perspective, most previous studies about spectral clustering aim at clustering using learning approaches, or building relations between a supervised learning model and spectral clustering. Law et al. (2017) and Bach and Jordan (2003) reveal that minimizing the loss of node similarity matrix can be seen as learning the leading eigenvector representations used for spectral clustering. Bianchi et al. (2020) train cluster assignment using a unsupervised MinCut objective and further use the learned clusters in a pooling layer to improve GNN performance. Instead of directly optimizing a similarity matrix, Tian et al. (2014) adopt an unsupervised autoencoder design that uses representations learned in hidden layers to perform K-means. Chowdhury and Needham (2021) show that Gromov-Wasserstein learning, an optimal transport-based clustering method, is connected with a two-way spectral clustering.

Graph restructuring and rewiring. GDC (Klicpera et al., 2019b) is one of the first works proposed to rewire edges in a graph. It uses diffusion kernels, such as heat kernel and personalized PageRank, to redirect messages passing beyond existing edges. Chamberlain et al. (2021) and Eliasof et al. (2021) extend the diffusion kernels to different classes of partial differential equations. In a slightly different setting where graph structures are not readily available, some try to construct graphs from scratch instead of modifying the existing edges. Fatemi et al. (2021) construct homophilic graphs via self-supervision on masked node features. Kalofolias (2016) learns a smooth graph by minimizing \( \text{tr}(X^T L X) \). These works belong to the “learning graphs from data” family (Dong et al., 2019) which is relevant but different to our work because a) these methods infer graphs from data where no graph topology is readily available, while in our setting, the original graphs are the key ingredients; b) as shown by Fatemi et al. (2021), without the initial graph, the performance of these methods are
not comparable to even a naive GCN; e) these methods are mostly used to solve graph generation problems instead of a node- or link-level task.

3 Preliminaries

Spectral filtering. Let $G = (V, E, A, X)$ be an undirected graph with $N$ nodes, where $V$, $E$, and $A \in \{0, 1\}^{N \times N}$ are the node set, edge set, and adjacency matrix of $G$, respectively, and $X \in \mathbb{R}^{N \times F}$ is the node feature matrix. The normalized Laplacian matrix of $G$ is defined as $L = I - D^{-1/2}AD^{-1/2}$, where $I$ is the identity matrix with $N$ diagonal entries and $D \in \mathbb{R}^{N \times N}$ is the diagonal degree matrix of $G$. In spectral graph theory, the eigenvalues $\lambda = \text{diag}(\lambda_1, ..., \lambda_N)$ and eigenvectors $U$ of $L = U\Lambda U^H$ are known as the graph’s spectrum and spectral basis, respectively, where $U^H$ is the Hermitian transpose of $U$. The graph Fourier transform takes the form of $\tilde{X} = U^H X$ and its inverse is $X = U\tilde{X}$.

It is known that the Laplacian spectrum and spectral basis carry important information on the connectivity of a graph (Shuman et al. 2013). Intuitively, lower frequencies correspond to global and smooth information on the graph, while higher frequencies correspond to local information, discontinuities and possible noise (Shuman et al. 2013). One can apply a spectral filter and use graph Fourier transform to manipulate signals on a graph in various ways, such as smoothing and denoising (Schaub and Segarra 2018), abnormally detection (Miller et al. 2011) and clustering (Wai et al. 2018). Spectral convolution on graphs is defined as the multiplication of a signal $x$ with a filter $g(\Lambda)$ in the Fourier domain, i.e.

$$ g(L)x = g(U^H U) x = U g(\Lambda) U^H x = U g(\Lambda) \tilde{x}. \quad (1) $$

Spectral Clustering (SC) as low-pass filtering. SC is a well-known method for clustering nodes in the spectral domain. A simplified SC algorithm is described in Appendix A.5. Spectral clustering can be interpreted as low-pass filtering on a one-hot node signal $\delta_i \in \mathbb{R}^N$, where the $i$-th element of $\delta_i$ is 1, i.e. $\delta_i(i) = 1$, and 0 elsewhere, for each node $i$. Filtering $\delta_i$ in the graph Fourier domain can be expressed as

$$ f_i = g_{\lambda_L}(\Lambda) U^H \delta_i, \quad (2) $$

where $g_{\lambda_L}$ is the low-pass filter that filter out components whose frequencies are greater than $\lambda_L$ as

$$ g_{\lambda_L}(\lambda) = \begin{cases} 1 & \text{if } \lambda \leq \lambda_L \\ 0 & \text{otherwise}. \end{cases} \quad (3) $$

As pointed out by Tremblay et al. (2016), Ramasamy and Madhow (2015), $\|f_i - f_j\|^2$ can be approximated by filtering random node features with $g_{\lambda_L}$. Consider a random node feature matrix $R = [r_1 | r_2 | ... | r_\eta] \in \mathbb{R}^{N \times \eta}$ consisting of $\eta$ random features $r_i \in \mathbb{R}^N$ of $V$ i.i.d sampled form the normal distribution with zero mean and $1/\eta$ variance. Let $H_{\lambda_L} = U g_{\lambda_L}(\Lambda) U^H$. We can define

$$ \tilde{f}_i = (H_{\lambda_L} R)^H \delta_i. \quad (4) $$

$R$ is a random Gaussian matrix of zero mean and $U$ is orthonormal. We apply the Johnson-Lindenstrauss lemma (Appendix A.3) to obtain the following error bounds.

**Proposition 3.1** (Tremblay et al. 2016). Let $\epsilon, \beta > 0$ be given. If $\eta$ is larger than:

$$ \eta_0 = \frac{4 + 2\beta}{\epsilon^2/2 - \epsilon^3/3} \log N, \quad (5) $$

then with probability at least $1 - N^{-\beta}$, we have: $\forall (i, j) \in [1, N]^2$,

$$ (1 - \epsilon) \|f_i - f_j\|^2 \leq \|\tilde{f}_i - \tilde{f}_j\|^2 \leq (1 + \epsilon) \|f_i - f_j\|^2. \quad (6) $$

Hence, $\|\tilde{f}_i - \tilde{f}_j\|$ is a close estimation of the Euclidean distance $\|f_i - f_j\|$. Note that the approximation error bounds in Equation 6 hold for any band-pass filter $g_{c_1, c_2}$:

$$ g_{c_1, c_2}(\lambda) = \begin{cases} 1 & \text{if } c_1 < \lambda \leq c_2 \\ 0 & \text{otherwise}. \end{cases} \quad (7) $$

where $0 \leq c_1 \leq c_2 \leq 2$ since the spectrum of a normalized Laplacian matrix has the range $[0, 2]$. This fact is especially useful when the band pass $g_{c_1, c_2}$ can be expressed by a specific functional form such as polynomials, since $H_{c_1, c_2} = U g_{c_1, c_2}(\Lambda) U^H$ can be computed without eigen-decomposition.
4 Learnable Spectral Clustering

Now we propose a learnable method for spectral clustering which empowers us to restructure a graph to improve graph homophily while preserving the original graph structures as much as possible.

4.1 Discrepancy between SC and labels

Spectral clustering aims to cluster nodes such that the edges between different clusters have low weights and the edges within a cluster have high weights. Such clusters are likely to align with the labels of nodes when a graph is homophilic, i.e., nodes with the same labels are likely to be connected. Whereas this may not hold in the case of less-homophilic graphs. Figure 1 visualizes the nodes in the Wisconsin and Europe Airport datasets based on the eigenvectors corresponding to the five smallest eigenvalues. As shown in Figure 1a and 1c, the labels are not aligned with the node clusters. However, if we choose the eigenvectors carefully, nodes clusters can align better with their labels, as evidenced by Figure 1b and 1d, which are visualized using manually chosen eigenvectors. Below, we discuss how to develop a learnable SC method that aligns the clustering structure with node labels by learning the underlying frequency patterns. The spectral expressive power of this method is discussed separately in Appendix A.7.

4.2 Learning eigenvector coefficients

Let \( f_i^Z \) be the representation of a node \( i \) obtained from an arbitrary set \( Z \subseteq [N] \) of eigenvectors in place of the ones with the leading eigenvalues. We cast the problem of finding eigenvectors to align with the known node labels into a minimization problem of computing the distance between the representations of nodes \( i \) and \( j \) when their labels are the same. Let \( d(\cdot) \) be a distance metric between two node representations. Then, our objective is formalized as

\[
\arg\min_Z \sum_{i,j \in V_Y} \ell(d(f_i^Z, f_j^Z), \mathbb{1}(y_i, y_j)),
\]

where \( V_Y \) is a collection of nodes whose labels are available, \( \mathbb{1} \) is an indicator function, \( y_i \) is the label of node \( i \), and \( \ell \) is a loss function. The loss function penalizes the objective when two nodes with the same label are far from each other, as well as when two nodes with different labels are close. In this work, we use Euclidean distance and a contrastive loss function.

Solving the above objective requires iterating over all possible combinations of eigenvectors, which is infeasible in general. It also requires performing expensive eigendecomposition with \( O(N^3) \) time complexity. In addition, SC does not consider feature similarity between nodes. To address these challenges, we introduce two key ideas to generalize SC in the next section.
4.3 Eigendecomposition-free SC

As explained in Section 3, \( \| f_i - f_j \| \) can be approximated by a filtering operation under the Johnson-Lindenstrauss lemma. The same holds true for the generalized case \( f_i^2 \). However, the operation still requires expensive eigendecomposition as Equation 7 takes eigenvalues explicitly. To mitigate this issue, we propose to use a series of rectangular functions, each serving as a band-pass filter that “slices” the Laplacian spectrum into a finite set of equal-length and equal-magnitude ranges. Each filter takes the same form of Equation 7 but is relaxed on the continuous domain. This formulation comes with two major advantages. Firstly, rectangular functions can be efficiently approximated with polynomial rational functions, thus bypassing expensive eigendecomposition. Secondly, each \( g_{e_1,e_2} \) groups frequencies within the range \([e_1, e_2]\) to form a “coarsened” pseudo-eigenvalue. Because nearby eigenvalues capture similar structural information, the “coarsening” operation reduces the size of representations, while still largely preserving the representation power.

Spectrum slicers. We approximate the band-pass rectangular filters in Equation 7 using a rational function

\[
\hat{g}_{s,a} = \frac{1}{s^{2m}} \left( \frac{2m \lambda - a}{s} \right)^{-1}
\]

where \( s \geq 2 \) is a parameter that controls the width of the passing window on the Laplacian spectrum, \( a \in [0, 2] \) is another parameter that controls the horizontal center of the function, and \( m \) is the approximation order. A larger \( m \) can reduce the approximate error but is more expensive to compute. Figure 2 shows an example of these functions.

With properly chosen \( s \) and \( a \), the Laplacian spectrum can be evenly sliced into chunks of range \([\lambda_i, \lambda_{i+1}]\). Each chunk is a “coarsened eigenvalue” that umbrellas eigenvalues within the range. Substituting \( g(\lambda) \) in Equation 1 with \( \hat{g}_{s,a} \), the spectral filtering operation becomes

\[
U \hat{g}_{s,a} (\Lambda) U^H x
\]

\[
= \frac{1}{s^{2m}} \left( \frac{L - aI}{2 + \hat{\epsilon}} \right)^{2m} + \frac{I}{s^{2m}}
\]

(10)

where \( U \Lambda U^H = L \).

An important property of Equation 10 is that the matrix inversion can be computed via truncated Neumann series (Wu et al., 2013). This can bring the computation cost of \( O(N^3) \) down to \( O(pN^2) \).

**Lemma 1.** For all \( \hat{\epsilon} > \frac{2s^{2m}}{s^{2m} - 1} - 2 \), the inverse of \( T = \left( \frac{L - aI}{2 + \hat{\epsilon}} \right)^{2m} + \frac{I}{s^{2m}} \) can be expressed by a Neumann series with guaranteed convergence. (A proof is given in Appendix A.4)

4.4 SC with node features

Traditionally, SC does not use node features. However, independent from graph structure, node features can provide extra information that is valuable for a clustering task. This is especially true where node features are in disagreement with representations obtained from eigenvectors. Therefore, we incorporate node features into the SC filtering operation by concatenating it with the random signal before the filtering operation

\[
\Gamma_{s,a} = \hat{g}_{s,a}(L)(R \parallel X),
\]

(11)

where \( \parallel \) is a column-wise concatenation operation. \( \Gamma_{s,a} \) has the shape of \( N \times (P + F) \) and is sometimes referred as "dictionary" (Thanou et al., 2014). When using a series of \( g_{s,a} \), we have

\[
\Gamma = (\Gamma_{s_1,a_1} \parallel \Gamma_{s_2,a_2} \parallel ...) \]

(12)
Let $P'$ be the dimension of embeddings and $\Theta$ represent a learnable weight matrix or a feed-forward neural network. The concatenated dictionary is then fed into a learnable function to obtain a node embedding matrix $H \in \mathbb{R}^{N \times P'}$ as

$$H = \Theta(\Gamma)$$

(13)

Our objective in Equation 8 can then be realized as

$$\mathcal{L}(\Theta) = \sum_{i,j \in V} \left[ ||H_i - H_j||_2^2 - ||H_i - H_k||_2^2 + \epsilon \right]_+$$

(14)

where $N_Y(k)$ is a set of negative samples whose labels are different from node $i$, i.e. $y_i \neq y_k$. The intuition is if the labels of nodes $i$ and $j$ are the same, then the distance between the two nodes needs to be less than the distance between $i$ and $k$, to minimize the loss.

4.5 Restructure graphs to maximize homophily

After training, we obtain a distance matrix $D'$ where $D'_{ij} = ||H_i - H_j||_2$. As suggested by Klicpera et al. (2019b), there are two ways to restructure a graph from the matrix. One way is to apply a threshold on $D'$ and entries below the threshold are kept as edges; another way is to draw edges between $K$ nodes pairs of the smallest distance. As the $D'$ is learned with labels, the restructured graph is likely to have higher homophily than the original graph. In practice, for better control of sparsity, we draw edges between node pairs with a small distance. Specifically, $\hat{A}$ is the adjacency matrix whose entries are defined as

$$\hat{A}_{ij} = \begin{cases} 
1, & \text{if } (i,j) \in \text{topK}^{-1}(S) \\
0, & \text{otherwise}
\end{cases}$$

(15)

where $\text{topK}^{-1}$ returns node pairs of the $k$ smallest entries in $D'$. In the implementation, we start with an empty adjacency matrix and iteratively add edges until the homophily score starts to drop. A simplified workflow is illustrated in Figure 3. A detailed algorithm can be found in Appendix A.6.

4.6 Complexity analysis

The most expensive operation of our method is the matrix inversion in Equation 10 which has the time complexity of $O(pN^2)$. A small $p \leq 4$ is sufficient because the Neumann series is a geometric sum so exponential acceleration tricks can be applied. We use Equation 10 because it is closer to a rectangular function and better illustrates the spectrum slicing idea, but in practice, it can be replaced by other slicer functions that do not require matrix inversion, such as a quadratic function $1 - (sL - a)^2$, to further reduce cost. It is also worth noting that the matrix inversion and multiplication only need to be computed once and can be pre-computed offline. The training step can be mini-batched easily. We randomly sample $8 - 64$ negative samples per node so the cost of computing Equation 14 is low.

5 Homophily Measure

Various methods have been proposed to measure the homophily of a graph (Zhu et al., 2020; Lim et al., 2021). The two most used are edge homophily $h_{edge}$ and node homophily $h_{node}$: the former uses
As pointed out by Lim et al. (2021), both edge and node homophily suffer from sensitivity to both (inter-class). For a graph of \( K \) labels, the following five propositions hold for our new metric:

1. A dense homophilic graph of a complete set of intra-class edges and zero inter-class edges has a score of 1.
2. A dense heterophilic graph of a complete set of inter-class edges and zero intra-class edges has a score of 0.
3. An Erdos-Renyi random graph \( G(n, p) \) of \( n \) nodes and the edge inclusion probability \( p \) has the score of \( \approx 0.5 \), i.e. a graph that is uniformly random.
4. A totally disconnected graph and a complete graph have the same score of 0.5.
5. For graphs with the same intra-class and inter-class edge ratios, the denser graph has a relatively higher score.

Figure 4: Examples of graphs with different label-topology relationships and comparison of different homophily measures. The node colour represents the node labels. The red edges connect nodes of different labels, while the green edges connect nodes of the same labels. Figure 4a - Figure 4c shows homophilic graphs of different densities. Figure 4d and Figure 4e are two graphs that only consist of inter-class edges, but are of different densities. Figure 4f is a label-imbalanced graph. Figure 4g and Figure 4f are two regular graphs, where Figure 4g has an intra-class/inter-class edge ratio of 0.5, and Figure 4h is an example of an Erdos-Reyi graph sampled with uniform edge probability.
Table 1: Node classification accuracy. The results are obtained from [Pei et al. 2020]. The * results are obtained from [Zhu et al. 2020].

| Datasets and models | ACTOR | CHAMELEON | SQUIRREL | WISCONSIN | CORNELL | TEXAS |
|---------------------|-------|-----------|-----------|-----------|--------|-------|
| GCN                 | 30.7±0.5 | 59.4±2.6  | 36.9±1.3  | 64.1±6.3  | 59.2±3.2 | 64.1±4.9 |
| GCN (GDC)           | 35.0±0.5 (+1.3) | 62.2±1.2 (+2.4) | 45.3±1.3 (+8.4) | 53.9±2.6 (+10.2) | 57.6±4.1 (-1.6) | 57.8±4.1 (-6.3) |
| GCN (ours)          | 36.2±1.0 (+5.5) | 66.9±3.1 (+7.1) | 55.7±2.4 (+18.8) | 83.1±3.2 (+19.0) | 79.2±6.3 (+20.0) | 74.8±5.4 (+14.3) |
| CHEV                | 34.5±1.3  | 66.0±2.3  | 39.6±3.0  | 82.5±2.8  | 76.5±9.4  | 79.7±5.0  |
| CHEV(GDC)           | 35.0±0.6 (+0.5) | 63.0±1.3 (-3.0) | 48.2±0.7 (+8.6) | 83.5±2.9 (+10.1) | 81.1±3.2 (+4.6) | 79.2±3.0 (-0.5) |
| CHEV(ours)          | 36.0±1.1 (+1.5) | 66.8±1.8 (+0.8) | 55.0±2.0 (+15.4) | 84.3±3.2 (+1.8) | 80.8±4.1 (+4.3) | 80.0±4.8 (+0.3) |
| ARMA (GD)           | 34.9±0.8  | 62.1±4.6  | 47.8±3.5  | 78.4±4.6  | 74.9±2.9  | 62.2±5.1  |
| ARMA (GDC)          | 35.0±0.5 (+1.0) | 60.2±0.6 (-1.9) | 47.8±0.8 (+0.0) | 79.8±2.6 (+1.4) | 78.4±4.1 (+3.5) | 78.4±3.2 (-3.8) |
| ARMA (ours)         | 35.2±0.7 (+0.3) | 68.4±2.3 (+6.3) | 55.6±1.7 (+7.8) | 84.5±0.5 (+6.1) | 81.1±6.1 (+6.2) | 81.1±4.2 (-1.1) |
| GAT                 | 25.9±1.8  | 54.7±2.0  | 30.6±2.1  | 62.0±5.2  | 58.9±3.3  | 60.0±5.7  |
| GAT (GDC)           | 35.0±0.6 (+9.1) | 63.8±1.2 (+9.1) | 48.6±2.1 (+18.0) | 51.4±4.5 (+10.6) | 58.9±2.2 (+0.0) | 77.1±8.3 (+19.1) |
| GAT (ours)          | 35.6±0.7 (+9.7) | 66.5±2.6 (+11.8) | 56.3±2.2 (+25.7) | 84.3±3.7 (+22.3) | 81.9±5.4 (+23.0) | 79.8±4.3 (+19.8) |
| SGC                 | 28.7±1.2  | 33.7±3.5  | 46.9±1.7  | 51.8±5.9  | 58.1±4.6  | 58.9±6.1  |
| SGC (GDC)           | 34.3±0.6 (+5.6) | 60.6±1.5 (+26.9) | 51.4±1.6 (+4.5) | 53.7±5.1 (+1.9) | 56.2±3.8 (-1.9) | 60.3±6.3 (+1.4) |
| SGC (ours)          | 34.9±0.7 (+6.2) | 67.1±2.9 (+33.4) | 52.3±2.3 (+5.4) | 77.8±4.7 (+26.0) | 73.5±4.3 (-15.4) | 74.4±6.0 (-15.5) |
| APPNP               | 35.0±1.4  | 45.3±1.6  | 31.0±1.6  | 81.2±2.5  | 70.3±9.3  | 79.0±4.6  |
| APPNP (GDC)         | 35.7±0.5 (+0.7) | 52.3±1.4 (+7.0) | 40.5±0.8 (+9.5) | 80.2±2.4 (+10.0) | 77.8±3.5 (+7.5) | 79.4±4.6 (-3.3) |
| APPNP (ours)        | 35.9±1.1 (+0.9) | 66.7±2.7 (+21.4) | 55.9±2.9 (+24.9) | 84.3±4.2 (+3.1) | 81.6±5.4 (+11.3) | 80.4±4.8 (+0.8) |
| GPRGNN              | 33.4±1.4  | 64.4±1.6  | 41.9±2.2  | 85.5±5.0  | 79.5±7.0  | 84.6±4.0  |
| GPRGNN (GDC)        | 34.4±1.0 (+1.0) | 61.0±1.7 (-2.5) | 39.2±1.5 (-1.7) | 85.1±5.0 (-0.4) | 82.4±4.7 (-2.9) | 80.8±4.9 (-3.8) |
| GPRGNN (ours)       | 34.1±1.1 (+0.7) | 65.5±2.2 (+1.1) | 47.1±2.4 (+5.2) | 85.3±4.1 (-0.4) | 80.3±6.3 (+0.8) | 84.3±5.1 (-0.3) |
| Geom-GCN²           | 31.6     | 60.9      | 38.1      | 64.1      | 60.8      | 67.8      |
| H²-GCN²             | 35.9±1.0  | 59.4±2.0  | 37.9±2.0  | 86.7±4.7  | 82.2±6.0  | 84.9±6.8  |

Proposition 3 states that a uniformly random graph, which has no label preference on edges, has homophily score of 0.5. Proposition 5 considers edge density: for graphs with the same tendencies of connecting inter- and intra-class nodes, the denser one has a higher absolute score value. The metric is defined as below.

$$ h_{\text{den}} = \min \{d_k - \hat{d}_k\}_{k=0}^{K-1} $$

where $d_k$ is the edge density of the subgraph formed by nodes from the label $k$, i.e., the intra-class edge density of $k$ including self-loops, and $\hat{d}_k$ is the maximum inter-class edge density of label $k$.

$$ d_k = \frac{2|\{(u,v) \in E : k_u = k_v = k\}|}{|Y_k||Y_k| + 1}, \quad \hat{d}_k = \max\{d_{kj} : j = 0, \ldots, K-1; j \neq k\}, $$

where $d_{kj}$ is the inter-class edge density of label $j$ and $k$, i.e. edge density of the subgraph formed by nodes from label $k$ and $j$.

$$ d_{kj} = \frac{|\{(u,v) \in E : k_u = k, k_v = j\}|}{|Y_k||Y_j|}. $$

Equation 17 has the range $(-1, 1)$. To make it comparable with the other homophily metrics, we scale it to the range $(0, 1)$ using

$$ h_{\text{den}} = \frac{1 + \hat{h}_{\text{den}}}{2}. $$

Proposition 1, 2, 4 and 5 are easy to prove. We hereby give a brief description and proof for Proposition 3.

**Lemma 2.** \(\forall K > 1, \mathbb{E}[h_{\text{den}}] = 0.5\) for the Erdos-Renyi random graph \(G(n, p)\). (A proof is given in Appendix A.)

Figure 4 shows some example graphs with four different metrics. Compared with $h_{\text{node}}$ and $h_{\text{edge}}$, $h_{\text{den}}$ is not sensitive to the number of labels and label imbalance. Compared with $h_{\text{norm}}$, $h_{\text{den}}$ is able to detect neutral graphs. $h_{\text{den}}$ gives scores in the range $(0, 0.5)$ for graphs of low-homophily, allowing direct comparison between them. $h_{\text{den}}$ also considers edge density and therefore is more robust to disconnectivity.

6 Empirical results

**Datasets and models.** We compare six classic GNNS, GCN [Kipf and Welling 2017], SGC [Wu et al. 2019], ChevNet [Defferrard et al. 2016], ARMA Net [Bianchi et al. 2021], GAT [Velickovic...
et al., 2018), and APPNP (Klicpera et al., 2019a), on their performance before and after graph restructuring. For comparison, we report the performance using an additional restructuring methods: GDC (Klicpera et al., 2019b). Three recent GNNs that target heterophilic graphs are also listed as baselines: GPRGNN (Chien et al., 2021), H$_2$GCN (Zhu et al., 2020) and Geom-GCN (Pei et al., 2020). We run experiments on six heterogeneous graphs: TEXAS, CORNELL, WISCONSIN, ACTOR, CHAMELEON and SQUIRREL (Rozemberczki et al., 2021; Pei et al., 2020). Details of these datasets are given in Appendix A.1.

Experimental setup. Hyperparameters are tuned using grid search for all models on the unmodified and restructured graphs of each dataset. We record prediction accuracy on the test set averaged over 10 runs with different random initializations. We use the same split setting as Pei et al. (2020); Zhu et al. (2020). The results are averaged over all splits. We adopt early stopping and record the results from the epoch with highest validation accuracy. Homophily scores are computed using validation set only in the validation step. We report the averaged accuracy as well as the standard deviation. For the spectrum slicer in Equation 10, we use a set of 20 slicers with $s = 40$ and $m = 4$ so that the spectrum is sliced into 20 even range of 0.1. The graphs are restructured to have edges that give the highest $h_{den}$ before it starts to decrease. All experiments are run on a single NVIDIA RTX A6000 48GB GPU unless otherwise noted.

Node classification results. Node classification tasks predict labels of nodes based on graph structure and node features. We aim to improve the prediction accuracy of GNN models by restructuring edges via the learnable SC method, particularly for heterophilic graphs. The evaluation results are shown in Table 1. On average, the performance of GNN models is improved by an average of 25%. We attribute this improvement to the increased homophily in the restructured graphs.

Performance vs. Homophily Figure 5b shows an ideal scenario where the "optimal edges" point corresponds to both the highest homophily and accuracy. Homophily scores are computed using validation set only in the validation step. We report the averaged accuracy as well as the standard deviation. For the spectrum slicer in Equation 10, we use a set of 20 slicers with $s = 40$ and $m = 4$ so that the spectrum is sliced into 20 even range of 0.1. The graphs are restructured to have edges that give the highest $h_{den}$ before it starts to decrease. All experiments are run on a single NVIDIA RTX A6000 48GB GPU unless otherwise noted.

Figure 5: Homophily and accuracy of GCN as per edges numbers. The optimal number of edges are chosen based on $h_{den}$ on validation set.
7 Conclusion

We propose an approach to enhance GNN performance on less-homophilic graphs by restructuring the graph to maximize its homophily. Our method is inspired and closely related to Spectral Clustering (SC). It extends SC beyond the leading eigenvalues and learns the frequencies that are best suited to cluster a graph. To achieve this, we use rectangular spectral filters expressed in the Neumann series to slice the graph spectrum into chunks that umbrellas small ranges of frequency. We also proposed a new homophily metric that is density-aware and is a better homophily indicator for the quality of graph restructuring. There are many promising extensions of this work, such as using it to guard against over-smoothing and adversarial attacks, by monitoring changes in homophily and adopting tactics to maintain it.

References

Dimitris Achlioptas. Database-friendly random projections: Johnson-Lindenstrauss with binary coins. *Journal of Computer and System Sciences*, 66(4):671–687, June 2003.

Francis Bach and Michael Jordan. Learning spectral clustering. *Advances in neural information processing systems*, 16, 2003.

Muhammet Balcilar, Guillaume Renton, Pierre Héroux, Benoit Gaüzère, Sébastien Adam, and Paul Honeine. Analyzing the expressive power of graph neural networks in a spectral perspective. In *9th International Conference on Learning Representations, ICLR*, 2021.

Filippo Maria Bianchi, Daniele Grattarola, and Cesare Alippi. Spectral clustering with graph neural networks for graph pooling. In *International Conference on Machine Learning*, pages 874–883. PMLR, 2020.

Filippo Maria Bianchi, Daniele Grattarola, Lorenzo Livi, and Cesare Alippi. Graph neural networks with convolutional arma filters. *IEEE transactions on pattern analysis and machine intelligence*, 2021.

Deyu Bo, Xiao Wang, Chuan Shi, and Huaweii Shen. Beyond low-frequency information in graph convolutional networks. In *Thirty-Fifth AAAI Conference on Artificial Intelligence*, 2021.

Ben Chamberlain, James Rowbottom, Maria I. Gorinova, Michael M. Bronstein, Stefan Webb, and Emanuele Rossi. GRAND: graph neural diffusion. In *Proceedings of the 38th International Conference on Machine Learning, ICML*, 2021.

Yongqiang Chen, Han Yang, Yonggang Zhang, Kaili Ma, Tongliang Liu, Bo Han, and James Cheng. Understanding and improving graph injection attack by promoting unnoticeability. *CoRR*, abs/2202.08057, 2022.

Eli Chien, Jianhao Peng, Pan Li, and Olgica Milenkovic. Adaptive universal generalized pagerank graph neural network. In *9th International Conference on Learning Representations, ICLR*, 2021.

Samir Chowdhury and Tom Needham. Generalized spectral clustering via gromov-wasserstein learning. In *The 24th International Conference on Artificial Intelligence and Statistics, AISTATS*, 2021.

Sanjoy Dasgupta and Anupam Gupta. An elementary proof of a theorem of johnson and lindenstrauss. *Random Struct. Algorithms*, 22(1):60–65, 2003.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 3837–3845, 2016.

Xiaowen Dong, Dorina Thanou, Michael Rabbat, and Pascal Frossard. Learning graphs from data: A signal representation perspective. *IEEE Signal Process. Mag.*, 36(3):44–63, 2019.

Moshe Eliasof, Eldad Haber, and Eran Treister. PDE-GCN: Novel architectures for graph neural networks motivated by partial differential equations. *Advances in Neural Information Processing Systems*, 34, 2021.
Bahare Fatemi, Layla El Asri, and Seyed Mehran Kazemi. SLAPS: self-supervision improves structure learning for graph neural networks. In Advances in Neural Information Processing Systems 34: NeurIPS 2021, 2021.

Yilun Jin, Guojie Song, and Chuan Shi. Gralsp: Graph neural networks with local structural patterns. In The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI, 2020.

Vassilis Kalofolias. How to learn a graph from smooth signals. Proceedings of the 19th International Conference on Artificial Intelligence and Statistics, AISTATS, 2016.

Dongkwan Kim and Alice Oh. How to find your friendly neighborhood: Graph attention design with Self-Supervision. In 9th International Conference on Learning Representations, ICLR, 2021.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In Proceedings of the 5th International Conference on Learning Representations (ICLR), 2017.

Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. In Proceedings of the 7th International Conference on Learning Representations (ICLR), 2019a.

Johannes Klicpera, Stefan Weißenberger, and Stephan Günnemann. Diffusion improves graph learning. In Advances in Neural Information Processing Systems (NeurIPS), pages 13333–13345, 2019b.

Marc T. Law, Raquel Urtasun, and Richard S. Zemel. Deep spectral clustering learning. In Proceedings of the 34th International Conference on Machine Learning, ICML, PMLR, 2017.

Shouheng Li, Dongwoo Kim, and Qing Wang. Beyond Low-Pass filters: Adaptive feature propagation on graphs. In Machine Learning and Knowledge Discovery in Databases. Research Track - European Conference, ECML PKDD, 2021.

Derek Lim, Felix Hohne, Xiuyu Li, Sijia Linda Huang, Vaishnavi Gupta, Omkar Bhalerao, and Ser-Nam Lim. Large scale learning on non-homophilous graphs: New benchmarks and strong simple methods. In Advances in Neural Information Processing Systems 34: NeurIPS, May 2021.

B. A. Miller, M. S. Beard, and N. T. Bliss. Matched filtering for subgraph detection in dynamic networks. In 2011 IEEE Statistical Signal Processing Workshop (SSP), pages 509–512, 2011.

Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. Geom-gcn: Geometric graph convolutional networks. In 8th International Conference on Learning Representations, ICLR, 2020.

Dinesh Ramasamy and Upamanyu Madhow. Compressive spectral embedding: sidestepping the SVD. In Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems, 2015.

Leonardo F.R. Ribeiro, Pedro H.P. Saverese, and Daniel R. Figueiredo. Struc2vec: Learning node representations from structural identity. In Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2017.

Benedek Rozemberczki, Carl Allen, and Rik Sarkar. Multi-scale attributed node embedding. J. Complex Networks, 9(2), 2021.

Michael T. Schaub and Santiago Segarra. Flow smoothing and denoising: Graph signal processing in the edge-space. In 2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP), pages 735–739, 2018.

Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. Collective classification in network data. AI Magazine, 29(3):93–106, 2008.

David I. Shuman, Sunil K. Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. IEEE Signal Process. Mag., 30(3):83–98, 2013.
Jie Tang, Jimeng Sun, Chi Wang, and Zi Yang. Social influence analysis in large-scale networks. In Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2009.

Dorina Thanou, David I Shuman, and Pascal Frossard. Learning parametric dictionaries for signals on graphs. IEEE Trans. Signal Process., 62(15):3849–3862, 2014.

Fei Tian, Bin Gao, Qing Cui, Enhong Chen, and Tie Yan Liu. Learning deep representations for graph clustering. Proceedings of the National Conference on Artificial Intelligence, 2:1293–1299, 2014.

Nicolas Tremblay, Gilles Puy, Pierre Borgnat, Rémi Gribonval, and Pierre Vandergheynst. Accelerated spectral clustering using graph filtering of random signals. In 2016 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP, 2016.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In Proceedings of the 6th International Conference on Learning Representations (ICLR), 2018.

H. Wai, S. Segarra, A. E. Ozdaglar, A. Scaglione, and A. Jadbabaie. Community detection from low-rank excitations of a graph filter. In 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 4044–4048, 2018.

Felix Wu, Amauri H. Souza Jr., Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Q. Weinberger. Simplifying graph convolutional networks. In Proceedings of the 36th International Conference on Machine Learning (ICML), volume 97, pages 6861–6871, 2019.

Michael Wu, Bei Yin, Aida Vosoughi, Christoph Studer, Joseph R Cavallaro, and Chris Dick. Approximate matrix inversion for high-throughput data detection in the large-scale MIMO uplink. In 2013 IEEE International Symposium on Circuits and Systems (ISCAS), pages 2155–2158, May 2013.

Yujun Yan, Milad Hashemi, Kevin Swersky, Yaoqing Yang, and Danai Koutra. Two sides of the same coin: Heterophily and oversmoothing in graph convolutional neural networks. arXiv preprint arXiv:2102.06462, 2021.

Rex Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. GNNExplainer: Generating explanations for graph neural networks. Adv. Neural Inf. Process. Syst., 32(July 2017), 2019.

Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor K. Prasanna. GraphSAINT: Graph sampling based inductive learning method. In 8th International Conference on Learning Representations, ICLR, 2020.

Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. Beyond homophily in graph neural networks: Current limitations and effective designs. In Advances in Neural Information Processing Systems 33: NeurIPS, 2020.

Jiong Zhu, Junchen Jin, Michael T. Schaub, and Danai Koutra. Improving robustness of graph neural networks with heterophily-inspired designs. CoRR, abs/2106.07767, 2021a.

Jiong Zhu, Ryan A Rossi, Anup B Rao, Tung Mai, Nedim Lipka, Nesreen K Ahmed, and Danai Koutra. Graph neural networks with heterophily. In Proceedings of the AAAI Conference on Artificial Intelligence, 2021b.
A Appendix

A.1 Dataset Details

We use 6 datasets listed in Table 2. TEXAS, WISCONSIN and CORNELL are graphs of web page links between universities, known as the CMU WebKB datasets. We use the pre-processed version in Pei et al. (2020), where nodes are classify into 5 categories of course, faculty, student, project and staff. SQUIRREL and CHAMELEON are graphs of web pages in Wikipedia, originally collected by Sen et al. (2008), then Pei et al. (2020) classifies nodes into 5 classes according to their average traffic. ACTOR is a graph of actor co-occurrence in films based on Wikipedia, modified by Pei et al. (2020) based on Tang et al. (2009).

| Dataset     | Classes | Nodes  | Edges  | Features |
|-------------|---------|--------|--------|----------|
| CHAMELEON   | 5       | 2,277  | 36,101 | 2,325    |
| SQUIRREL    | 5       | 5,201  | 217,073| 2,089    |
| ACTOR       | 5       | 7,600  | 26,752 | 931      |
| TEXAS       | 5       | 183    | 325    | 1,703    |
| CORNELL     | 5       | 183    | 298    | 1,703    |
| WISCONSIN   | 5       | 251    | 515    | 1,703    |

A.2 Experiment Results

Figure 6 shows the node classification accuracy from Table 1. The performance of classical GNNs before and after restructuring are plotted as histograms. Performance of three GNNs that target less-homophilic datasets, Geom-GCN, $H_2GCN$ and GPRGNN, are plotted as horizontal lines.

A.3 Johnson-Lindenstrauss Theorem

**Theorem A.1** (Johnson-Lindenstrauss Theorem (Dasgupta and Gupta 2003), For any $0 < \epsilon < 1$ and any integer $n$, let $k$ be a positive integer such that
\[
k \leq 4(\epsilon^2/2 - \epsilon^3/3)^{-1} \ln n
\]
Then for any set \( S \) of \( n \) points in \( \mathbb{R}^d \), there is a map \( f: \mathbb{R}^d \rightarrow \mathbb{R}^k \) such that for all \( u, v \in S \),
\[
(1 - \epsilon)\|u - v\|^2 \leq \|f(u) - f(v)\|^2 \leq (1 + \epsilon)\|u - v\|^2.
\]

Furthermore, this map can be found in randomized polynomial time.

We refer the readers to Dasgupta and Gupta (2003) for proof. In this work, we adopt a side result of their proof, which shows the map \( f = \mathbf{R}x \) is an instance of such mapping that satisfies Equation 22, where \( \mathbf{R} \) is a matrix of random Gaussian variables with zero mean. Achlioptas (2003) further extends the proof and shows that a random matrix drawn from \([1, 0, -1]\), or \([1, -1]\) also satisfies the theorem. We leave these projections for future study.

A.4 Proofs

We recall Lemma 1

**Lemma 1.** For all \( \hat{\epsilon} > \frac{2s^2m}{s^{2m} - 1} - 2 \), the inverse of \( T = \left( \frac{L - aI}{2 + \hat{\epsilon}} \right)^{2m} + \frac{I}{s^{2m}} \) can be expressed by a Neumann series with guaranteed convergence. (A proof is given in Appendix A.4.)

*Proof.* With a slight abuse of notation, we use \( \lambda(*) = \lambda_1, \lambda_2, \ldots \) to denote eigenvalues of a matrix. The spectral radius of a matrix is the largest absolute value of its eigenvalues \( \rho(*) = \max(|\lambda(*)|) \). \( L \) is the normalized Laplacian matrix of \( G \), therefore \( 0 \leq \lambda(L) \leq 2 \). According to eigenvalue properties, we have
\[
-1 < \frac{\lambda(L) - a}{2 + \hat{\epsilon}} < 1,
\]
thus
\[
-1 < \lambda\left(\frac{L - aI}{2 + \hat{\epsilon}}\right) < 1
\]
because \( \hat{\epsilon} > 0 \) and \( a \in [0, 2] \). Because the power of the eigenvalues of a matrix is the eigenvalues of the matrix power, i.e. \( \lambda(*^{2m}) = \lambda(*)^{2m} \), we have
\[
0 < \lambda\left(\frac{L - aI}{2 + \hat{\epsilon}}\right)^{2m} < \lambda\left(\frac{L - aI}{2 + \hat{\epsilon}}\right) < \frac{2}{2 + \hat{\epsilon}}.
\]
Therefore,
\[
\frac{1}{s^{2m}} < \lambda\left(\frac{L - aI}{2 + \hat{\epsilon}}\right)^{2m} + \frac{I}{s^{2m}} < \frac{2}{2 + \hat{\epsilon}} + \frac{1}{s^{2m}}.
\]
Hence, \( \forall \hat{\epsilon} > \frac{2s^{2m}}{s^{2m} - 1} - 2 \),
\[
0 < \lambda\left(\frac{L - aI}{2 + \hat{\epsilon}}\right)^{2m} + \frac{I}{s^{2m}} < 1.
\]
In another word, \( \rho(I - T) < 1 \). Gelfand’s formula shows that if \( \rho(I - T) < 1 \), then \( \lim_{p \to \infty} (I - T)^p = 0 \) and the inverse of \( T \) can be expressed by a Neumann series \( T^{-1} = \sum_{p=0}^{\infty} (I - T)^p \). \( \square \)

We recall Lemma 2

**Lemma 2.** \( \forall K > 1, \mathbb{E}[h_{\text{den}}] = 0.5 \) for the Erdos-Renyi random graph \( G(n, p) \). (A proof is given in Appendix A.4.)

*Proof.* For each node label \( k \) of \( |Y_k| \) nodes, the are at most \( \binom{|Y_k|+1}{2} = |Y_k|||Y_k||1 + 1)/2 \) intra-class edges (including self-loops). For each pair of label \( (k, j) \), there are at most \( |Y_k|||Y_j| \) inter-class edges. On average \( G(n, p) \) has \( \binom{n+1}{2}p \) edges, among which \( \binom{|Y_k|+1}{2}p = \frac{|Y_k|||Y_k||1 + 1)p}{2} \) are intra-class for \( k \), and \( |Y_k|||Y_j||p \) are inter-class for the class pair \( (k, j) \). Hence from Equation 18 we have \( \mathbb{E}[d_k] = p \), from Equation 19 we have \( \mathbb{E}[d_{kj}] = p \). Substitute \( d_k \) and \( d_{kj} \) in Equation 17 we have \( \mathbb{E}[d_{\text{den}}] = 0 \) and \( \mathbb{E}[d_{\text{den}}] = 0.5 \). \( \square \)
A.5 Spectral Clustering

A simplified Spectral Clustering (SC) algorithm involves the following four steps:

1. Perform eigendecomposition for the Laplacian matrix to obtain eigenvalues \((\lambda_1, \lambda_2, ..., \lambda_N)\) sorted in ascending order.
2. Pick \(L (1 < L \leq N)\) eigenvectors \(u_1, ..., u_L\) associated with the leading \(L\) eigenvalues.
3. Represent a node \(i\) with a vector \(f_i\) whose element are from the chosen eigenvectors:
   \[ f_i = [u_1(i), u_2(i), ..., u_L(i)]^T \in \mathbb{R}^L. \]
4. Perform K-means with a distance measurement, such as the Euclidean distance \(||f_i - f_j||\) or dot product similarity \(f_i^T f_j\), to partition the nodes into \(K\) clusters.

A.6 Graph Restructuring Algorithm

The restructuring algorithm is illustrated in Algorithm 1.

Algorithm 1 Graph Restructuring Algorithm

Input: Graph \(G\), homophily metric \(h\), number of random sample \(P\), spectrum band length \(s\), edge step size \(n\)
Output: Restructured adjacency matrix \(\hat{A}\)

Sample \(r_1, ..., r_M \sim \mathcal{N}(0, 1/P)\)
\(R \leftarrow (r_1 \| r_2 \| ... \| r_P)\)
for each spectrum band \((s, a)\) do
   \(\Gamma_{s,a} \leftarrow g_{s,a}(L)(R \| X)\) \(\triangleright \) Equation 11
end for
\(\Theta \leftarrow \arg \min_{\Theta} L(\Theta)\) \(\triangleright \) Equation 14
\(H \leftarrow \Theta(\Gamma)\)
Compute \(D'\) where \(D'_{ij} = ||H_i - H_j||\)
\(\pi \leftarrow \) sorted index of \(D'\) in descending order \(\triangleright\) only use the lower triangular part
\(A' \leftarrow 0\)
\(\lambda, \lambda^{\text{old}} \leftarrow 0.5\)
while \(\lambda \geq \lambda^{\text{old}}\) do
   \(\lambda^{\text{old}} \leftarrow \lambda\)
   \((i_1,j_1), ..., (i_n,j_n) \leftarrow \text{Pop}(\pi, n)\) \(\triangleright\) keep the old homophily score
   \(A'_{i_1,j_1} \leftarrow 1, ..., A'_{i_n,j_n} \leftarrow 1\) \(\triangleright\) next \(n\) edges from sorted index
   \(\lambda \leftarrow h(A')\) \(\triangleright\) add new edge
end while
return \(A' + A'^\top\)

A.7 Spectral Expressive Power

In this section, we analyze the ability of the learnable spectral clustering to learn specific frequency patterns. Being able to adjust to different frequency patterns, to an extent, demonstrates the expressive power of a model in the spectral domain. As pointed out by Balcilar et al. (2021), the majority of GNNs are limited to only low-pass filters and thus have limited expressive power, while only a few are able to capture high-pass and band-pass patterns.

To evaluate this, we adopt the experimental setup of Balcilar et al. (2021) using filtered images. A real 100x100 image is filtered by three pre-defined low-pass, band-pass and high-pass filters: \(\phi_1(\rho) = \exp(-100\rho^2), \phi_2(\rho) = \exp(-1000(\rho - 0.5)^2)\) and \(\phi_3(\rho) = 1 - \exp(-10\rho^2)\), where \(\rho = \rho_1^2 + \rho_2^2\) and \(\rho_1\) and \(\rho_2\) are the normalized frequencies in each direction of an image. The original image and the three filtered versions are shown in Figure 7. The task is framed as a node regression problem, where
we minimize the square error between $H \in \mathbb{R}^{N \times 1}$ in Equation 13 and the target pixel values, i.e.

$$
\mathcal{L}'(\Theta) = \sum_{i=1}^{N} (H_i - Y_i)^2
$$

where $Y_i$ is the target pixel value of node $i$. We train the models with 3000 iterations and stop early if the loss is not improving in 100 consecutive epochs.

| Task     | MLP  | GCN  | GIN  | GAT  | ChevNet | Ours  |
|----------|------|------|------|------|---------|-------|
| Low-pass | 43.42| 5.79 | 1.44 | 2.30 | 0.17    | 0.07  |
| Band-pass| 71.81| 74.31| 46.80| 74.04| 27.70   | 2.94  |
| High-pass| 19.95| 24.74| 17.80| 24.57| 2.16    | 1.36  |

Table 3 shows the square loss of our method along MLP and 2 GNNs. Our method consistently outperforms other models. Some output images are shown in Figure 8. As expected, MLP fails to learn the frequency pattern across all three categories. GCN, GAT and GIN are able to learn the low-pass pattern but failed in learning the band and high-frequency patterns. Although ChevNet shows comparable results in the high-pass task, it is achieved with 41,537 trainable parameters while our method only requires 2,050 parameters. Lastly, our method is the only one that can learn and accurately resemble the band-pass image, demonstrating a better flexibility in learning frequency patterns.

A.8 Code

The code of our experiments is released under the MIT license at https://anonymous.4open.science/r/graph_restructure-1BBB.
Figure 8: The low-pass, band-pass and high-pass images learned using MLP, GIN, ChevNet and our method. The images learned using our method better resemble the images shown in Figure 7 across all three categories.