

---

# Graph Contrastive Learning under Heterophily via Graph Filters

---

Wenhan Yang<sup>1</sup>

Baharan Mirzasoleiman<sup>1</sup>

<sup>1</sup>Computer Science Department, University of California Los Angeles (UCLA)

## Abstract

Graph contrastive learning (CL) methods learn node representations in a self-supervised manner by maximizing the similarity between the augmented node representations obtained via a GNN-based encoder. However, CL methods perform poorly on graphs with heterophily, where connected nodes tend to belong to different classes. In this work, we address this problem by proposing an effective graph CL method, namely HLCL, for learning graph representations under heterophily. HLCL first identifies a homophilic and a heterophilic subgraph based on the cosine similarity of node features. It then uses a low-pass and a high-pass graph filter to aggregate representations of nodes connected in the homophilic subgraph and differentiate representations of nodes in the heterophilic subgraph. The final node representations are learned by contrasting both the augmented high-pass filtered views and the augmented low-pass filtered node views. Our extensive experiments show that HLCL outperforms state-of-the-art graph CL methods on benchmark datasets with heterophily, as well as large-scale real-world graphs, by up to 7%, and outperforms graph supervised learning methods on datasets with heterophily by up to 10%.

## 1 INTRODUCTION

Graph neural networks (GNNs) are powerful tools for learning graph-structured data in various domains [Kipf and Welling, 2016, Veličković et al., 2017]. GNNs use the graph’s adjacency matrix to aggregate node information from their neighbors, effectively acting as a low-pass filter that smooths graph signals [Nt and Maehara, 2019]. They have shown remarkable success in supervised and semi-

supervised learning, where task-specific labels are available. However, obtaining high-quality labels can be costly in many domains, spurring interest in self-supervised learning on graphs to learn representations without supervision [Velickovic et al., 2019, Peng et al., 2020, Qiu et al., 2020, Hassani and Khasahmadi, 2020, Zhu et al., 2020b].

Among these self-supervised methods, Contrastive Learning (CL) has demonstrated remarkable success [Velickovic et al., 2019, Peng et al., 2020, Qiu et al., 2020, Hassani and Khasahmadi, 2020, Zhu et al., 2020b]. Graph CL methods first augment the input graph, either by altering node features or the graph topology. Then, they learn representations by contrasting the augmented graph views encoded with a GNN-based encoder. Existing graph CL methods perform well under homophily, where neighboring nodes often share the same label. However, they perform poorly on heterophilic graphs, where connected nodes tend to belong to different classes [Zhu et al., 2020b]. Indeed, for learning rich representations in graphs with heterophily, contrasting augmented views of every node is not enough, but it is crucial to *differentiate* representation of node with different labels [Bo et al., 2021, Luan et al., 2020]. However, *without* label information, it is not clear how this can be achieved.

In this work, we propose an effective graph CL method, namely HLCL, for learning node representations under heterophily. HLCL first uses nodes’ feature similarity to identify a homophilic and heterophilic subgraph in the original graph. Then, for each subgraph, it generates two augmented graph views, and applies a high-pass filter to the heterophilic subgraphs and a low-pass filter to the homophilic subgraphs. The final representations are learned by contrasting the augmented high-pass filtered views and contrasting the augmented low-pass filtered views of each node, using the *same* GNN encoder, as illustrated in Fig. 1. In doing so, HLCL achieves state-of-the-art performance under heterophily, surpassing graph *supervised learning* methods and yielding comparable performance to state-of-the-art graph CL methods under homophily. In addition, we prove that the learned representations by HLCL encode both low-frequency and

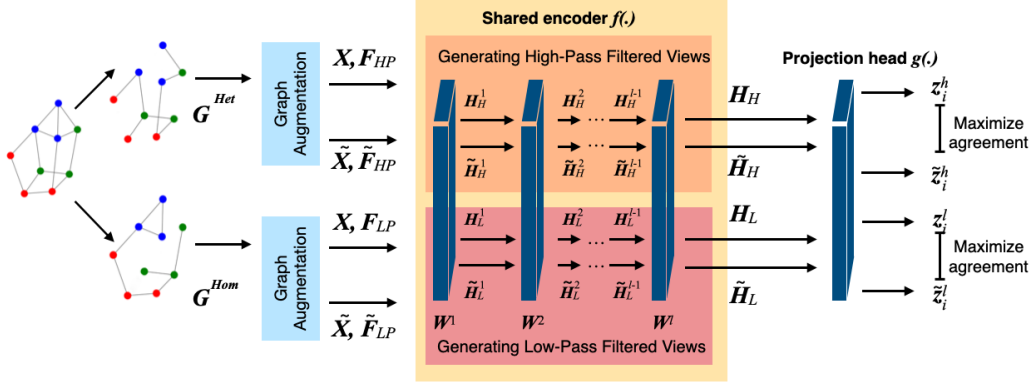


Figure 1: HLCL identifies a homophilic and a heterophilic subgraph  $\mathcal{G}^{hom}, \mathcal{G}^{het}$ , and generates two augmentations for each subgraph. Then, it applies low-pass filters  $\mathbf{F}_{LP}, \tilde{\mathbf{F}}_{LP}$  to the augmented homophilic subgraphs and high-pass filters  $\mathbf{F}_{HP}, \tilde{\mathbf{F}}_{HP}$  to the augmented heterophilic subgraphs, to generate low-pass  $\mathbf{H}_L, \tilde{\mathbf{H}}_L$  and high-pass  $\mathbf{H}_H, \tilde{\mathbf{H}}_H$  filtered views, using the same encoder  $\mathbf{W}$ . HLCL learns the final representations by contrasting the projected low-pass filtered augmented views  $\mathbf{z}^L, \tilde{\mathbf{z}}^L$  and the high-pass filtered augmented views  $\mathbf{z}^h, \tilde{\mathbf{z}}^h$  of every node.

high-frequency information.

Our extensive experiments show that on seven benchmark datasets, HLCL outperforms existing graph CL methods by up to 7% and graph supervised learning methods by up to 10% under heterophily, while maintaining comparable performance under homophily. Additionally, HLCL scales well to large graphs like Penn94, outperforming other graph CL methods by up to 5%.

In summary, our contributions are as follows:

- *Graph CL with graph filters.* HLCL is the first graph CL method that utilizes graph filters, and combines high-pass and low-pass filtered representations using contrastive losses. This approach enables learning rich representations under heterophily.
- *Careful aggregation.* HLCL identifies a homophilic and a heterophilic subgraph based on node features or representations, for effective information aggregation.
- *Theoretical analysis.* By analyzing HLCL, we theoretically prove that HLCL learns the invariance information from both homophilic and heterophilic subgraphs.
- *Extensive experiments.* Empirically, we confirm that HLCL achieves state-of-the-art under heterophily and a competitive performance under homophily.

## 2 RELATED WORK

**Graph self-supervised learning.** Graph self-supervised learning (SSL) methods have become a powerful tool for learning representations without any labels. Graph contrastive learning (CL) is among the most successful graph SSL methods. Numerous methods have been proposed in the field: [Velickovic et al., 2019, Peng et al., 2020, Hasani and Khasahmadi, 2020, Zhu et al., 2021c] focus on contrasting the global augmented representation with the local augmented representation, while [Zhu et al., 2020c,

You et al., 2020, Qiu et al., 2020, Liu et al., 2022] contrast same-scale representation, global or local, in two augmented views. Due to the complexity of collecting negative samples in graph data, negative-sample free contrastive objectives have also been studied [Thakoor et al., 2021, Bielak et al., 2021]. However, such work focus on encoding the homophilic graphs and perform poorly under heterophily. Recently, a stream of SSL methods have been proposed to learn the node representations of the heterophilic graphs without labels. HGRL [Chen et al., 2022] improves the node representations on heterophilic graphs by rewiring non-local neighbors based on feature information before training. SP-GCL [Wang et al., 2022] considers nodes from the  $T$ -hop neighborhood of a node with high feature similarities as positive pairs, without using any explicit augmentations. DSSL [Xiao et al., 2022] separates the heterogeneous patterns in local neighborhood distributions to capture both homophilic and heterophilic information globally. GREET [Liu et al., 2023] discriminates homophilic edges from heterophilic edges using random walk based graph diffusion and contrasts the projected representations of the two graph views directly via a dual-channel contrastive loss. MUSE [Yuan et al., 2023] creates two views to capture information from the node itself and its neighborhood, and fuses these views to enhance node representations. NeCo [He et al., 2023] proposes a new pretext task, group discrimination, which divides the nodes into  $k$  groups and keeps the consistent representation of nodes within a group.

**Graph (semi-)supervised learning under heterophily.** In the supervised setting, recent methods propose to use other types of aggregation that better fit graphs with heterophily. Zhu et al. [2021a] analyzed and designed a uniform framework for GNNs’ propagations and proposed GNN-LF and GNN-HF that preserve information of different frequency separately by using different filtering kernels with learnable weights. FAGCN [Bo et al., 2021] and FBGNN [Luan

et al., 2020] train two *separate* encoders to capture the high-pass and low-pass graph signals separately. Then they rely on labels to learn relatively complex mechanisms to combine the outputs of the encoders. However, learning how to combine the encoder outputs is highly sensitive to having high-quality labels. This makes such methods highly impractical for self-supervised contrastive learning, where the label information is not available. Unlike the above supervised methods, we apply the high-pass and low-pass filters to different subgraphs, contrasting the resulting high-pass filtered node views and low-pass filtered node views in a self-supervised manner, without any label. This is in contrast to learning the best combination of filtered signals of different encoders based on labels. A more comprehensive overview of related work are provided in Appendix A.6.

### 3 PRELIMINARIES

**Notations.** We denote by  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$  an undirected graph, where  $\mathcal{V} = \{v_1, v_2, \dots, v_N\}$  represents the node set, and  $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$  represents the edge set. We denote by  $\mathbf{A} \in \{0, 1\}^{N \times N}$  the symmetric adjacency matrix of graph  $\mathcal{G}$ , where  $\mathbf{A}_{ij} = 1$  if and only if  $(v_i, v_j) \in \mathcal{E}$ , and  $\mathbf{A}_{ij} = 0$  otherwise. We denote the feature matrix by  $\mathbf{X}$ , where  $\mathbf{X}_i \in \mathbb{R}^m$  is the feature vector of the  $i^{\text{th}}$  node, and  $\mathbf{x} \in \mathbb{R}^N$  is a column of the matrix and represents a graph signal.  $\mathbf{D}$  is the degree matrix of the graph, with  $\mathbf{D}_{ii} = \sum_j \mathbf{A}_{ij}$ , and  $\mathcal{N}_i = \{j : \mathbf{A}_{ij} = 1\}$  is the neighborhood of node  $i$ .  $\mathbf{L}$  is the Laplacian matrix of the graph, defined as  $\mathbf{L} = \mathbf{D} - \mathbf{A}$ . The normalized Laplacian matrix is denoted by  $\mathbf{L}_{sym} = \mathbf{D}^{-\frac{1}{2}} \mathbf{L} \mathbf{D}^{-\frac{1}{2}}$ , and the normalized adjacency matrix is defined as  $\mathbf{A}_{sym} = \mathbf{D}^{-\frac{1}{2}} \mathbf{A} \mathbf{D}^{-\frac{1}{2}}$ . Here, we use the renormalized version of the adjacency matrix  $\hat{\mathbf{A}}_{sym} = \bar{\mathbf{D}}^{-\frac{1}{2}} \bar{\mathbf{A}} \bar{\mathbf{D}}^{-\frac{1}{2}}$  as introduced in [Kipf and Welling, 2016], where  $\bar{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ ,  $\bar{\mathbf{D}} = \mathbf{D} + \mathbf{I}$ . Similarly, the renormalized Laplacian matrix is defined as  $\hat{\mathbf{L}}_{sym} = \mathbf{I} - \hat{\mathbf{A}}_{sym}$ .  $\hat{\mathbf{L}}_{sym}$  is a real symmetric matrix, with orthonormal eigenvectors  $\{\mathbf{u}_i\}_{i=1}^n \in \mathbb{R}^n$ , and corresponding eigenvalues  $\lambda_i \in [0, 2)$  [Chung, 1997]. For  $\hat{\mathbf{A}}_{sym}$  we have  $\lambda_i(\hat{\mathbf{A}}_{sym}) \in (-1, 1]$ .

#### 3.1 GRAPH CL UNDER HOMOPHILY

State-of-the-art graph CL methods explicitly augment the input graph using feature or topology augmentations, encode the augmented graphs using a GNN-based encoder, and contrast the encoded node representations [Zhu et al., 2020c, 2021b, Velickovic et al., 2019, Thakoor et al., 2021, Qiu et al., 2020], as we will discuss in more detail next.

**Graph Augmentation.** First, the input graph is explicitly augmented, by altering its topology or node features. Topology augmentation methods remove or add nodes or edges, and feature augmentation methods alter the node features by masking particular columns, dropping features at random, or randomly shuffling the node features [Zhu et al., 2020c,

2021b, Velickovic et al., 2019, Thakoor et al., 2021].

**GNN Encoder.** The augmented graphs are then passed through a GNN-based encoder to obtain the augmented node views. The GNN encoder produces node representations by aggregating the node features in a neighborhood as follows:

$$\mathbf{H}^l = \sigma(\tilde{\mathbf{A}}_{sym} \mathbf{H}^{l-1} \mathbf{W}^{l-1}), \quad \mathbf{H}^0 = \mathbf{X}, \quad (1)$$

where  $\mathbf{H}_l^l$  is the node representations at layer  $l$  of the encoder,  $\mathbf{W}^l \in \mathbb{R}^{d_l \times d_{l-1}}$  is the weight matrix in layer  $l$  of the encoder, and  $\sigma$  is the activation function. Crucially, the Adjacency matrix  $\tilde{\mathbf{A}}_{sym}$  is a low-pass filter that aggregates every node’s features with the features of nodes in its immediate neighborhood. For a multi-layer graph encoder, it iteratively aggregates features in a multi-hop neighborhood of every node to learn its representation. Hence, it smooths out the node representations and produces similar representations for the nodes within the same multi-hop neighborhood.

**Contrastive Loss.** Finally, the contrastive loss distinguishes the representations of the same node in two different augmented views, from other node representations. For example the commonly used InfoNCE loss [Oord et al., 2018] is:

$$-\log \frac{e^{\text{sim}_\tau(\mathbf{u}^i, \mathbf{v}^i)}}{e^{\text{sim}_\tau(\mathbf{u}^i, \mathbf{v}^i)} + \sum_{k \neq i} e^{\text{sim}_\tau(\mathbf{u}^i, \mathbf{v}^k)}}, \quad (2)$$

where  $\mathbf{u}_i, \mathbf{v}_i$  are representations of two different augmented views of node  $i$ ,  $\text{sim}(\mathbf{u}^i, \mathbf{v}^k)$  is the cosine similarity between  $\mathbf{u}^i$  and  $\mathbf{v}^k$ , and  $\tau$  is a temperature parameter.

#### 3.2 HIGH-PASS AND LOW-PASS GRAPH FILTERS

The adjacency and Laplacian matrices can be leveraged to filter the smooth and non-smooth graph components, and capture similarity and dissimilarity of node features to their neighborhoods. Specifically, multiplication of Laplacian with a graph signal  $\hat{\mathbf{L}}_{sym} \mathbf{x} = \sum_i \lambda_i \mathbf{u}_i \mathbf{u}_i^T \mathbf{x}$ , acts as a filtering operation over  $\mathbf{x}$ , adjusting the scale of the components of  $\mathbf{x}$  in the frequency domain. The entries of every eigenvector,  $\mathbf{u}_i$  align with a cluster of connected nodes in the graph. For the Laplacian matrix, a smaller eigenvalue  $\lambda_i$  corresponds to a lower frequency (smoother) eigenvectors  $\mathbf{u}_i$ , and a larger cluster of connected nodes. On the other hand, a larger  $\lambda_i$  corresponds to a high frequency (non-smooth) eigenvectors  $\mathbf{u}_i$ , which identify smaller clusters of closely connected nodes in the graph. A Laplacian filter magnifies the high frequency signals that align well with basis functions corresponding to large eigenvalues  $\lambda_i \in (1, 2)$  and suppresses the low frequency signal that aligns with basis functions corresponding to small eigenvalues  $\lambda_i \in [0, 1]$ . That means, for small clusters of nodes that have a large alignment with  $\mathbf{u}_i$  corresponding to  $\lambda_i > 1$ , the projection  $\lambda_i \mathbf{u}_i \mathbf{u}_i^T \mathbf{x}$  amplifies  $\mathbf{x}$  within the cluster and consequently magnifies the difference in  $\mathbf{x}$  among the nodes within that cluster. On the other hand, for the larger clusters

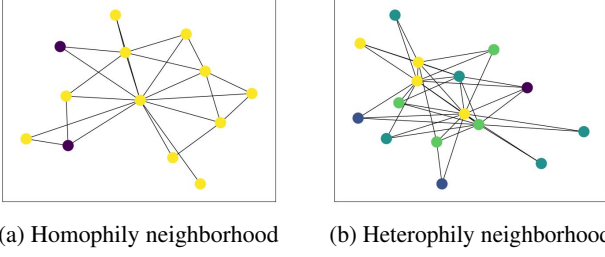


Figure 2: Chameleon ( $\beta=0.23$ ). Heterophilic graphs contain neighborhoods with homogeneous & heterogeneous labels.

that align well with  $\mathbf{u}_i$  corresponding to  $\lambda_i < 1$ , the projection  $\lambda_i \mathbf{u}_i \mathbf{u}_i^T \mathbf{x}$  suppresses  $\mathbf{x}$  within the cluster and reduces the differences in  $\mathbf{x}$  among the nodes within that cluster. Hence the Laplacian matrices can be generally regarded as high-pass filters [Ekambaram, 2014], that enlarge the differences in node features over small clusters, and smooths out the differences over larger clusters in the graph. In contrast, affinity matrices, such as the normalized adjacency matrix, can be treated as low-pass filters [Nt and Maehara, 2019], which suppress and filter out non-smooth components of the signals. This is because all of the eigenvalues of the affinity matrices are smaller than 1, i.e.,  $\lambda_i \in (-1, 1]$ .

On the node level, left multiplying  $\hat{\mathbf{L}}_{sym}$  and  $\hat{\mathbf{A}}_{sym}$  filters with  $\mathbf{x}$  can be understood as diversification and aggregation operations, respectively [Luan et al., 2020]. In particular, a typical GNN filters smooth graph frequencies by aggregating the node representations with those of their neighbors, using the adjacency matrix, i.e.,

$$(\hat{\mathbf{A}}_{sym} \mathbf{x})_i = \sum_{j \in \mathcal{N}_i} \frac{1}{\bar{D}_{ii}} \mathbf{x}_j. \quad (3)$$

Hence, it results in similar representations for the nodes in a neighborhood. In contrast, the high-pass filter only preserves the high-pass frequencies, using the Laplacian matrix, i.e.

$$(\hat{\mathbf{L}}_{sym} \mathbf{x})_i = \sum_{j \in \mathcal{N}_i} \frac{1}{\bar{D}_{ii}} (\mathbf{x}_i - \mathbf{x}_j). \quad (4)$$

In doing so, it magnifies the dissimilarities between the nodes and make the representations of nodes in a neighborhood distinguishable.

**Homophily Ratio** Homophily ratio quantifies how likely nodes with same labels are connected in the graph. Formally, homophily ratio,  $\beta$ , is defined as follows [Pei et al., 2020]:

$$\beta = \frac{1}{|V|} \sum_{v \in V} \frac{\text{No. of similar neighbors of } v}{\text{No. of neighbors of } v}. \quad (5)$$

## 4 GRAPH CL UNDER HETEROPHILY

In this section, we first discuss the challenges of having a universal method for graph CL under heterophily and

homophily. Then, we present our approach to overcome these challenges and learn high-quality representations.

**Challenges.** Under heterophily, where nodes in a neighborhood may have different labels, aggregating the node representations in a neighborhood fades out the dissimilarity between representations of node in different classes, and contrasting those augmented representations further makes them indistinguishable. Labels can help guide an appropriate aggregation in the neighborhood. However, without labels, it is not clear how the neighborhood information should be aggregated. Additionally, even if one can identify homophilic edges, the number of such edges may be too small to learn high quality representations via GNNs, under heterophily. To achieve rich representations in such graphs, it is crucial to not only aggregate representations of neighbors with the same label, but also push away representations of neighbors with different labels. This allows learning richer node representations based on both similarities and dissimilarities of the nodes in different neighborhoods.

Next, we present our method, HLCL, that can learn high-quality representations under heterophily.

### 4.1 HIGH-PASS & LOW-PASS GRAPH CL (HLCL)

As discussed, under heterophily, leveraging node feature similarities is not enough for learning high-quality representations. It is crucial to capture the *dissimilarities* between the neighboring nodes to separate different classes. A high-pass filter like the Laplacian matrix (c.f. Sec. 3.2) filters the non-smooth graph component and captures the dissimilarity of the node features in a neighborhood. However, without labels, we cannot know whether the graph is homophilic or heterophilic, and naively using a high-pass filter instead of a low-pass filter significantly harms the performance under homophily. Moreover, most heterophilic graphs also consist of several neighborhoods with homogeneous labels, as illustrated in Fig.2. Hence, simply applying a high-pass filter to an unlabeled graph may result in poor performance.

**Idea.** To learn rich node representations for both graph types, our main idea is to first identify a homophilic subgraph and a heterophilic subgraph in the original graph. Then, we augment each subgraph, and apply a low-pass filter to the augmented homophilic subgraphs and a high-pass filter to the augmented heterophilic subgraphs to obtain two high-pass and two low-pass filtered views for every node, using the *same encoder*. The final representations are learned by contrasting the two high-pass filtered views and the two low-pass filtered views of every node.

Next, we introduce our method, HLCL, which works based on the above idea.

**Separating Subgraphs.** HLCL first identifies two subgraphs in the original graph: a homophilic subgraph with

edges connecting nodes with homogeneous labels, and a heterophilic subgraph with edges connecting nodes with heterogeneous labels.

Formally, given a graph  $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ , the heterophilic subgraph  $\mathcal{G}^{het} = (\mathcal{V}, \mathcal{E}^{het})$  and the homophilic subgraph  $\mathcal{G}^{hom} = (\mathcal{V}, \mathcal{E}^{hom})$  each contain all the nodes  $\mathcal{V}$ , and a subset of the edges of the original graph, i.e.,  $\mathcal{E}^{het}, \mathcal{E}^{hom} \subseteq \mathcal{E}$ . We denote by  $\mathbf{A}^{het}, \mathbf{A}^{hom} \in \{0, 1\}^{N \times N}$  the symmetric adjacency matrix of subgraphs  $\mathcal{G}^{het}, \mathcal{G}^{hom}$ , respectively. Note that the feature matrix  $\mathbf{X}$  for  $\mathcal{G}$  is the same as  $\mathcal{G}^{het}$  and  $\mathcal{G}^{hom}$ . However, the neighborhood for a given node  $i$  can be different in the two subgraphs. We define  $\mathcal{N}_i^{het} = \{j : \mathbf{A}_{ij}^{het} = 1\}$  and  $\mathcal{N}_i^{hom} = \{j : \mathbf{A}_{ij}^{hom} = 1\}$  as the neighborhood of node  $i$  in  $\mathcal{G}^{het}, \mathcal{G}^{hom}$ , respectively. Without any label supervision, we rely on the important observation that for graphs with different homophily ratios, the original features can approximately indicate the label information [Jin et al., 2021, Chen et al., 2022, Wang et al., 2020, Zhu et al., 2020b]. Based on this observation, we calculate pairwise feature similarities  $s_{ij} = \langle \mathbf{x}_i, \mathbf{x}_j \rangle$  for all  $i, j \in [n] = |\mathcal{V}|$ , where  $\langle \cdot, \cdot \rangle$  is the cosine similarity. Then, we first form the homophilic subgraph by selecting  $k_1$  fraction of edges in neighborhood of every node  $i$  with largest cosine similarities. Formally,  $\mathcal{E}^{hom} = \{(i, j) | i \in [n], j \in \arg \max_{P \subseteq \mathcal{N}_i, |P|=\lceil k_1 \cdot |\mathcal{N}_i| \rceil} \sum_{p \in P} \{s_{i,p}\}\}$ . Next, we form the heterophilic subgraph using  $k_2$  fraction of the edges in neighborhood of every node with lowest cosine similarities, i.e.,  $\mathcal{E}^{het} = \{(i, j) | i \in [n], j \in \arg \min_{P \subseteq \mathcal{N}_i, |P|=\lceil k_2 \cdot |\mathcal{N}_i| \rceil} \sum_{p \in P} \{s_{i,p}\}\}$ .

The initial subgraphs are constructed via original node features. However, the subgraphs are updated every  $T$  epochs with the learned node representations during training. Note that, in contrast to prior work [Chen et al., 2022], we do not introduce new edges based on feature similarities throughout the training, which could change the semantic information of the graph [He et al., 2023].

We note that all the nodes may not be connected in both subgraphs. However, as long as one subgraph is mostly connected, the information can be aggregated effectively and a satisfactory performance is obtained. For example, under homophily the heterophilic subgraph is small, but the homophilic subgraph contains almost all the nodes in the largest connected component of the original graph. Similarly, under extreme heterophily almost all the nodes are in the heterophilic subgraph and the homophilic subgraph is small and minimally affects the performance. As HLCL contrasts augmented views of the homophilic subgraph and heterophilic separately (it does not contrast the subgraphs with each other), only one subgraph needs to be mostly connected to achieve satisfactory performance. In our ablation studies in Sec. 6.5, we confirm that in real-world graphs at least one of the subgraphs are mostly connected.

**Augmenting the Subgraphs.** Next, HLCL generates two

augmented views for each subgraph via random graph perturbations. We denote the two augmented graph views as  $\tilde{\mathcal{G}}$  and  $\tilde{\mathcal{G}}$ . For the homophilic subgraph, we follow [Zhu et al., 2020c] and apply edge removal and feature masks as our graph augmentations. For the heterophilic subgraph, we apply node dropping and feature masks as our augmentations. We study the effects of different augmentation techniques on the heterophilic subgraph in Sec. 6.1.

**Producing the Filtered Representations.** Subsequently, HLCL applies a high-pass filter to the two augmented views of the heterophilic subgraph, and a low-pass filter to the two augmented views of the homophilic subgraph, using the *same encoder*. The shared encoder is crucial to ensure a good performance under both homophily and heterophily.

Specifically, to generate the low-pass and high-pass filtered node views, HLCL leverages the renormalized adjacency matrices of the augmented heterophilic subgraph and the renormalized Laplacian matrices of the augmented heterophilic subgraph. Formally,  $\mathbf{F}_{LP} = \hat{\mathbf{A}}_{sym}^{hom}$ , and  $\mathbf{F}_{HP} = \hat{\mathbf{L}}_{sym}^{het} = \mathbf{I} - \hat{\mathbf{A}}_{sym}^{hom}$ , are the low-pass and high-pass filters corresponding to the first augmented view of the homophilic and heterophilic subgraphs, and  $\tilde{\mathbf{F}}_{LP}, \tilde{\mathbf{F}}_{HP}$  are the low-pass and high-pass filters corresponding to the second augmented view of the homophilic and heterophilic subgraphs. Effectively,  $\mathbf{F}_{LP}, \tilde{\mathbf{F}}_{LP}$  are the aggregation operations in Eq. (3) and  $\mathbf{F}_{HP}, \tilde{\mathbf{F}}_{HP}$  are diversification operations in Eq. (4). Then, the two low-pass filtered views of the homophilies subgraph are obtained as follows:

$$\mathbf{H}_L^l = \sigma(\mathbf{F}_{LP} \mathbf{H}_H^{l-1} \mathbf{W}^{l-1}), \quad (6)$$

$$\tilde{\mathbf{H}}_L^l = \sigma(\tilde{\mathbf{F}}_{LP} \mathbf{H}_H^{l-1} \mathbf{W}^{l-1}), \quad (7)$$

and the two high-pass filtered views of the heterophilic subgraph are obtained as follows:

$$\mathbf{H}_H^l = \sigma(\mathbf{F}_{HP} \mathbf{H}_H^{l-1} \mathbf{W}^{l-1}), \quad (8)$$

$$\tilde{\mathbf{H}}_H^l = \sigma(\tilde{\mathbf{F}}_{HP} \mathbf{H}_H^{l-1} \mathbf{W}^{l-1}). \quad (9)$$

$\mathbf{H}_L^l, \tilde{\mathbf{H}}_L^l$  are the low-pass filtered augmented views at layer  $l$  of the encoder,  $\mathbf{H}_H^l, \tilde{\mathbf{H}}_H^l$  are the high-pass filtered augmented views at layer  $l$  of the encoder,  $\mathbf{W}^l \in \mathbb{R}^{d_l \times d_{l-1}}$  is the weight matrix in layer  $l$  of the encoder,  $\sigma$  is the activation function, and we have  $\mathbf{H}_L^0 = \mathbf{H}_H^0 = \mathbf{X}$ , and  $\tilde{\mathbf{H}}_L^0 = \tilde{\mathbf{H}}_H^0 = \tilde{\mathbf{X}}$  where  $\mathbf{X}, \tilde{\mathbf{X}}$  are augmented feature matrices.

$\mathbf{F}_{HP}, \tilde{\mathbf{F}}_{HP}$  filter out the low-frequency signals and preserve the high-frequency signals. In doing so, they capture the difference in feature of each node and its neighbors. Using a high-pass encoder within a multi-layer encoder iteratively captures the difference between features of the nodes in a multi-hop neighborhood of a node in the heterophilic subgraph. Hence, it makes the representations of nodes that have different features from their neighbors distinct in their multi-hop neighborhood. On the other hand,

$F_{LP}, \tilde{F}_{LP}$ , only preserve the low-frequency signals by aggregating every node’s features with those of its immediate neighborhood. Using the low-pass filter within a multi-layer graph encoder iteratively aggregates features in a multi-hop neighborhood of every node in the homophilic subgraph to learn its representation. Hence, they smooth out the node representations and produces similar representations for the nodes within the same multi-hop neighborhood.

Note that, we use the Laplacian and adjacency matrices of the augmented subgraphs instead of those of the original graphs, as they indicate how the information in different neighborhoods should be aggregated by the GCN encoder. Indeed, it is important to use the corresponding matrices in the subgraphs. In doing so, we pull together representations of nodes within label-homogeneous neighborhoods by applying low-pass filters to homophilic subgraphs, and push away representations of nodes within label-heterogeneous neighborhoods by applying high-pass filter to heterophilic subgraphs. If both filters are applied to the original graph, representations of the nodes within each neighborhood will be pulled together and pushed apart at the same time.

Using both high-pass and low-pass filters provide complementary information and allow learning both smooth and non-smooth components of the graphs simultaneously, which is particularly useful for graphs under heterophily. We note that other types of high-pass and low-pass filters can be used in a similar way in our framework.

**Contrasting the Filtered Representations.** Finally, by contrastive the augmented views of each subgraph, HLCL learns high-quality representations. The augmented views  $H, \tilde{H}$  are first projected via a 2-layer non-linear MLP, named projection head, to another latent space  $z, \tilde{z}$  where the contrastive losses are calculated, as advocated by [Chen et al., 2020, Chen and He, 2021, Zhu et al., 2020c, 2021b].

Then, for each subgraph, we first consider every node  $i$  in the first augmented subgraph view as the anchor, and contrast it with all the nodes in the second augmented subgraph view. This yields the following contrastive losses for the homophilic and heterophilic subgraphs, respectively:

$$l(z_l^i, \tilde{z}_l^i) = \log \frac{e^{\text{sim}(z_l^i, \tilde{z}_l^i)/\tau}}{e^{\text{sim}(z_l^i, \tilde{z}_l^i)/\tau} + \sum_{\substack{k \in [N], \\ k \neq i}} e^{\text{sim}(z_l^i, \tilde{z}_l^k)/\tau}} \quad (10)$$

$$l(z_h^i, \tilde{z}_h^i) = \log \frac{e^{\text{sim}(z_h^i, \tilde{z}_h^i)/\tau}}{e^{\text{sim}(z_h^i, \tilde{z}_h^i)/\tau} + \sum_{\substack{k \in [N], \\ k \neq i}} e^{\text{sim}(z_h^i, \tilde{z}_h^k)/\tau}}, \quad (11)$$

where  $\text{sim}$  is the cosine similarity between the projected representations, and  $\tau$  is a temperature parameter. The second term in the denominator represent the inter-view negative pairs, which are between the anchored view of node  $i$  and the views of all other nodes from the other view.

Similarly, for each subgraph we also consider the second augmented view of node  $i$  as the anchor and contrast it with

---

**Algorithm 1** High-pass and Low-pass Graph CL (HLCL)

---

- 1: Infer subgraph  $\mathcal{G}^{hom}$  by selecting the top  $\lceil k_1 \times |\mathcal{N}_i| \rceil$  edges with highest cosine similarity for every node  $i$ .
  - 2: Infer subgraph  $\mathcal{G}^{het}$  by selecting the top  $\lceil k_2 \times |\mathcal{N}_i| \rceil$  edges with lowest cosine similarity for every node  $i$ .
  - 3: **for** epoch = 1, 2, 3, ... **do**
  - 4:   Obtain augmented graph views  $\mathcal{G}^{hom}, \tilde{\mathcal{G}}^{hom}, \mathcal{G}^{het}, \tilde{\mathcal{G}}^{het}$  via random perturbations.
  - 5:   Generate high-pass node representations  $H_H, \tilde{H}_H$  based on Eq. (8), (9), using encoder weights  $W$ .
  - 6:   Generate low-pass node representations based on  $H_L, \tilde{H}_L$  based on Eq. (6), (7) using encoder weights  $W$ .
  - 7:   Compute the contrastive objective  $\mathcal{L}_{HLCL}$  in Eq. (12).
  - 8:   Update the encoder weights  $W$  by applying stochastic gradient ascent to minimize  $\mathcal{L}_{HLCL}$ .
  - 9:   **if** epoch %  $T = 0$  **then**
  - 10:     update  $\mathcal{G}^{het}, \mathcal{G}^{hom}, \tilde{\mathcal{G}}^{het}, \tilde{\mathcal{G}}^{hom}$  based on current node representations.
  - 11:   **end if**
  - 12: **end for**
- 

all the nodes in the first augmented subgraph view. Since two views are symmetric, the loss for using the other view as anchor is defined in a similar fashion. The overall objective to be minimized is then defined as the average over all the four contrastive losses. Formally, we minimize:

$$\mathcal{L}_{HLCL} = -\frac{1}{4N} \sum_{i=1}^N [l(z_l^i, \tilde{z}_l^i) + l(z_h^i, \tilde{z}_h^i) + l(\tilde{z}_l^i, z_l^i) + l(\tilde{z}_h^i, z_h^i)]. \quad (12)$$

Effectively, by maximizing the agreement between the low-pass views and between the high-pass views, HLCL pulls away the representation of nodes with different features from their neighborhood, and allows them to be distinguished from their neighbors.

**Final representations.** After minimizing the contrastive loss in Eq. (12), we use the low-pass filtered representations as the final output.

The pseudocode is illustrated in Alg. 1.

**Scalability to Large Graphs via Message Passing.** The high-pass and low-pass filtered representations can be obtained through message passing in an inductive manner, according to Eq. (3), (4), without the need to explicitly calculate the normalized Adjacency and Laplacian matrix. In particular, the high-pass filtered representations can be obtained by iteratively differentiating the representations of a node and those of its neighbors, and the low-pass filtered representations can be obtained by aggregating the node’s

representation with those of its neighbors:

$$\mathbf{h}_i^l = \sigma(\mathbf{W}^{l-1} \mathbf{h}_i^{l-1}), \quad (13)$$

$$(\mathbf{h}_i^l)_L = \sum_{j \in \{\mathcal{N}_i^{\text{hom}} \cup \{i\}\}} (\mathbf{h}_i^l + \mathbf{h}_j^l), \quad (14)$$

$$(\mathbf{h}_i^l)_H = \sum_{j \in \{\mathcal{N}_i^{\text{het}} \cup \{i\}\}} (\mathbf{h}_i^l - \mathbf{h}_j^l). \quad (15)$$

The above update rules can be applied to both augmented subgraphs. This is the same approach used to train GNNs on large graphs. Hence, HLCL will have the same complexity as conducting a normal GNN message passing with an additional message being passed to generate the high-pass filtered views. This makes HLCL scalable to large graphs, as we will also confirm in our experiments.

In addition, we will empirically confirm in Appendix A.4 that directly contrasting the high-pass and low-pass filtered representations can produce comparable results to HLCL, while speeding up the algorithm by 2x, as it requires minimizing only one pair of contrastive losses.

## 4.2 THEORETICAL ANALYSIS

Next, we theoretically prove that by separating the graph into homophilic and heterophilic subgraphs and applying low-pass and high-pass filters on them respectively, HLCL can encode both low-frequency and high-frequency information in the learned representations.

Following [Liu et al., 2022], we simplify the contrastive losses (10), (11) by assuming  $\tau = 1$  and using inner product for *sim*. Additionally, we assume a one-layer linear encoder.

**Theorem 1 (HLCL: Spectral Invariance).** *Under the above assumptions and given ideal subgraphs  $G_{\text{hom}}$  and  $G_{\text{het}}$ , the HLCL loss can be lower-bounded as follows:*

$$\begin{aligned} \mathcal{L}_{\text{HLCL}} \geq & \frac{-1 - N}{2} \sum_i \left( \alpha_{A_i} (2 - (\lambda_{A_i^{\text{hom}}} - \lambda_{\tilde{A}_i^{\text{hom}}})^2) \right. \\ & \left. + \alpha_{L_i} (4 - (\lambda_{L_i^{\text{het}}} - \lambda_{\tilde{L}_i^{\text{het}}})^2) \right), \end{aligned}$$

where  $\lambda_{A^{\text{hom}}}, \lambda_{\tilde{A}^{\text{hom}}}$  denote the eigenvalues of the low-pass filters corresponding to augmented homophilic subgraph,  $\lambda_{L^{\text{het}}}, \lambda_{\tilde{L}^{\text{het}}}$  denote the eigenvalues of the high-pass filters corresponding to augmented heterophilic subgraph, and  $\alpha_{A^{\text{hom}}}, \alpha_{L^{\text{het}}}$  are adaptive weights that change during the training as the parameters of the encoder changes.

Theorem 1 provides a lower-bound for the HLCL loss. The lower-bound is in the form of a summation of two terms: the first term is the sum of the difference between the low-frequency components of the two low-pass filtered augmented views of the homophilic subgraph, and the second term is the sum of the difference between the two high-pass filtered augmented views of the heterophilic subgraph. Minimizing the HLCL loss ensures a small value for the lower bound. In doing so, the encoder changes such that it

assigns a larger weight ( $\alpha_{A_i}$  and  $\alpha_{L_i}$ ) to invariant frequencies  $i$ , for which  $\lambda_{A_i^{\text{hom}}} \sim \lambda_{\tilde{A}_i^{\text{hom}}}$  and  $\lambda_{L_i^{\text{het}}} \sim \lambda_{\tilde{L}_i^{\text{het}}}$ . Notably, ( $\lambda_{A_i^{\text{hom}}} \sim \lambda_{\tilde{A}_i^{\text{hom}}}$ ) implies that the two contrasted augmentations are invariant at  $i^{\text{th}}$  frequency. Same reasoning holds for the second term. Therefore, during training with HLCL, the encoder will emphasize the invariance between two contrasted augmentations from the spectrum domain, for both the homophilic and heterophilic subgraphs.

The proof is given in the Appendix. B

## 5 EXPERIMENTS

In this section, we evaluate the node representations learned with HLCL, under linear probe. We compare HLCL with existing graph CL, graph SSL and graph supervised learning methods, and conduct an extensive ablation study to evaluate the effect of each of HLCL’s components.

**Datasets.** We consider nine widely-used public benchmark datasets with different homophily ratios,  $\beta$ . The details of the datasets are shown in Sec. A.2. We repeat the experiments 10 times for smaller benchmark datasets, and 3 times for large real-world datasets, and report the early-stopped average accuracy as the final result. For small graphs, we follow CPGNN [Zhu et al., 2020a], GRACE [Zhu et al., 2020c], and HGRL [Chen et al., 2022] and randomly select 10% of nodes for training, 10% of nodes for validation, and 80% of nodes for testing. For large graphs, following [Lim et al., 2021] we randomly select 25% of nodes for training, 25% of nodes for validation, and 50% of nodes for testing.

**Linear Probe Evaluation.** For SSL methods, we follow the evaluation protocol used in [Zhu et al., 2020c]. Models are first trained in a self-supervised manner without labels. Then, we fed the final node embeddings into a  $l_2$ -regularized logistic regression classifier to fit the labeled data.

### 5.1 RESULTS

**HLCL vs Self-supervised Baselines.** We compare HLCL with existing baselines for self-supervised representation learning. We consider general graph self-supervised learning methods like DGI [Velickovic et al., 2019], BGRL [Thakoor et al., 2021], and GRACE [Zhu et al., 2020c] as well as graph self-supervised learning methods that focus on learning under heterophily like HGRL [Chen et al., 2022], and SP-GCL [Wang et al., 2022]. In addition, we also include popular general graph supervised learning methods like GCN [Kipf and Welling [2016], and graph supervised learning method targeting graphs under heterophily like MixHop, H2GCN, GloGNN, and CPGNN [Abu-El-Haija et al., 2019, Zhu et al., 2020b, 2021c, Li et al., 2022]. We record the hyperparameters for our experiments in Sec. A.1 Table 1 shows that HLCL achieves a significant boost on graphs with heterophily and a comparable performance



Table 1: HLCL vs baselines. Methods identified with  $\dagger$  and  $*$  are supervised methods and SSL methods for graphs under heterophily, respectively. HLCL achieves state-of-the-art under heterophily, and a comparable performance under homophily.

	Homophily			Heterophily					
	Cora	CiteSeer	Pubmed	Actor	Chameleon	Squirrel	Penn94	Twitch-gamers	Genius
Hom.( $\beta$ )	.83	.71	.79	.09	.23	.19	.48	.56	.51
Nodes	2,708	3,327	19,717	5,201	2,277	5,201	41,554	168,114	421,961
Edges	5,278	4,676	44,324	198,493	8,854	46,998	1,362,229	6,797,557	984,979
Classes	6	7	3	5	5	5	2	2	2
<b>HLCL</b>	<b>84.1 <math>\pm</math> 1.0</b>	<b>70.1 <math>\pm</math> 0.8</b>	<b>84.5 <math>\pm</math> 0.4</b>	<b>34.0 <math>\pm</math> 0.2</b>	<b>50.9 <math>\pm</math> 1.0</b>	<b>42.9 <math>\pm</math> 2.6</b>	<b>68.1 <math>\pm</math> 3.5</b>	<b>67.0 <math>\pm</math> 0.9</b>	<b>84.3 <math>\pm</math> 0.1</b>
DGI	<b>84.5 <math>\pm</math> 1.1</b>	<b>71.9 <math>\pm</math> 0.7</b>	86.0 $\pm$ 0.1	28.0 $\pm$ 1.4	32.6 $\pm$ 2.9	38.8 $\pm$ 2.3	62.9 $\pm$ 0.4	61.5 $\pm$ 0.6	OOM
BGRL	83.0 $\pm$ 0.7	69.8 $\pm$ 0.6	80.2 $\pm$ 0.6	28.3 $\pm$ 0.9	32.6 $\pm$ 4.7	35.7 $\pm$ 1.4	58.8 $\pm$ 0.6	60.9 $\pm$ 0.3	76.4 $\pm$ 3.0
GRACE	83.7 $\pm$ 0.7	71.4 $\pm$ 1.0	86.7 $\pm$ 0.1	<u>34.5 <math>\pm</math> 1.1</u>	35.4 $\pm$ 3.6	36.2 $\pm$ 2.8	62.5 $\pm$ 0.4	57.1 $\pm$ 0.1	79.6 $\pm$ 2.9
SP-GCL*	83.2 $\pm$ 0.1	72.0 $\pm$ 0.4	79.2 $\pm$ 0.7	27.7 $\pm$ 0.7	36.5 $\pm$ 1.9	33.7 $\pm$ 1.3	-	62.0 $\pm$ 0.2	<b>90.1 <math>\pm</math> 0.2</b>
HGRL*	82.1 $\pm$ 0.8	71.0 $\pm$ 0.7	84.2 $\pm$ 0.2	35.4 $\pm$ 0.9	43.9 $\pm$ 1.7	38.7 $\pm$ 1.7	OOM	OOM	OOM
GCN $\dagger$	82.3 $\pm$ 1.2	70.2 $\pm$ 0.9	86.4 $\pm$ 0.3	28.2 $\pm$ 0.4	40.9 $\pm$ 4.1	39.5 $\pm$ 1.5	82.5 $\pm$ 0.3	62.2 $\pm$ 0.3	87.4 $\pm$ 0.4
MixHop $\dagger$	81.0 $\pm$ 1.6	66.4 $\pm$ 1.7	85.1 $\pm$ 0.3	29.0 $\pm$ 1.0	33.8 $\pm$ 1.2	33.4 $\pm$ 1.6	83.5 $\pm$ 0.7	65.6 $\pm$ 0.3	90.6 $\pm$ 0.2
H2GCN $\dagger$	81.4 $\pm$ 1.2	71.8 $\pm$ 0.9	85.9 $\pm$ 0.4	33.6 $\pm$ 0.8	26.8 $\pm$ 3.6	35.1 $\pm$ 1.2	OOM	OOM	OOM
GloGNN $\dagger$	<b>88.3 <math>\pm</math> 1.1</b>	<b>77.4 <math>\pm</math> 1.7</b>	<b>89.6 <math>\pm</math> 0.4</b>	<b>37.4 <math>\pm</math> 0.8</b>	25.9 $\pm$ 3.6	35.1 $\pm$ 1.2	<b>85.6 <math>\pm</math> 0.4</b>	66.4 $\pm$ 0.3	<b>90.7 <math>\pm</math> 0.1</b>
CPGNN $\dagger$	83.6 $\pm$ 1.3	72.0 $\pm$ 0.5	86.7 $\pm$ 0.2	35.6 $\pm$ 0.9	33.0 $\pm$ 3.2	30.0 $\pm$ 2.0	OOM	OOM	OOM

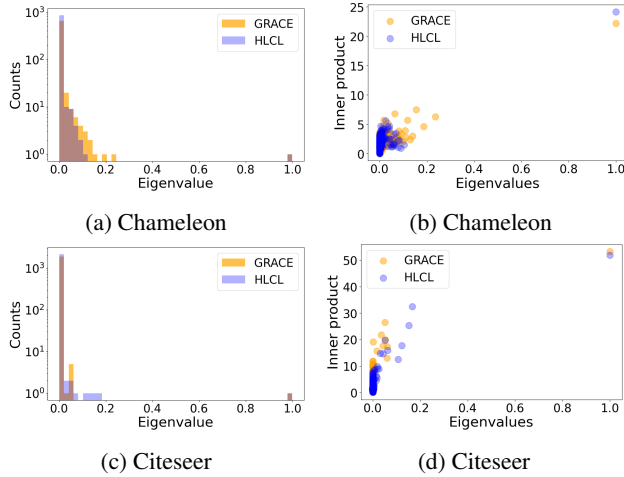


Figure 3: GRACE vs HLCL representations. (a), (c) distribution of eigenvalues in the representation matrix. (b), (d) alignment of the labels with the eigenvectors of the representation matrix. HLCL produces higher quality representations with lower rank and higher alignment with the label vector.

on graphs with homophily compared to the popular graph CL methods, showing up to 7% performance boost on Chameleon and 5% boost on Penn94. Compared to supervised methods such as H2GCN trained in an end-to-end manner, HLCL achieves a comparable performances under homophily and superior performance on heterophilic graphs like Chameleon by 10% and Squirrel by 3%. This confirms the effectiveness of HLCL.

**HLCL learns superior representations under heterophily.** Next, we compare the quality of representations learned by HLCL with that of GRACE, which only uses the low-pass filter for graph CL. We study Chameleon, a popular heterophily dataset [Platonov et al., 2023], and Citeseer,

a well-known homophily dataset [Yang et al., 2016]. Fig. 3 compares the distribution of normalized eigenvalues of the representation matrices and the alignment of their eigenvectors with the label vector. Lower-ranked representations that have a higher alignment between their prominent eigenvalues and label vector yield superior performance [Xue et al., 2022]. Fig. 3a confirms that HLCL’s representations of Chameleon (heterophily) have a lower-rank structure compare to that of GRACE. Fig. 3b further confirms a strong alignment between the eigenvectors of the representation matrix and the label vector. Both factors contribute to higher classification accuracy of HLCL compared to GRACE. On the other hand, on Citeseer (homophily), HLCL exhibits a similar but only slightly higher rank spectrum than GRACE. Thus, contrasting the low-pass and high-pass views does not significantly harm the performances under homophily.

## 6 ABLATION STUDIES

### 6.1 HLCL WITH EXPLICIT AUGMENTATION

Graph augmentation methods are well studied for graph CL under homophily [Zhu et al., 2020c, You et al., 2020, Liu et al., 2022]. However, it is unclear if the same techniques are effective when a high-pass filter is applied to the graph. Here, we study the effects of different structural and feature augmentations applied to the heterophilic subgraphs of HLCL. We keep the graph augmentations on the homophilic subgraph constant, as mentioned in Sec. 4.1 and only investigate the effect of augmentation on the heterophilic subgraph. We consider popular graph augmentation methods including edge dropping, feature masking, node dropping, edge adding, and diffusion [You et al., 2020, Hassani and Khasahmadi, 2020]. As shown in Table 2, applying node dropping is



Table 2: The effect of applying different augmentations to the heterophilic subgraph.

	Homophily		Heterophily		
	Cora	CiteSeer	Chameleon	Squirrel	Actor
EdgeRemoving	80.0 $\pm$ 0.5	65.8 $\pm$ 0.3	47.6 $\pm$ 1.1	40.5 $\pm$ 1.1	32.4 $\pm$ 0.9
NodeDropping	<b>82.1</b> $\pm$ 0.5	<b>66.7</b> $\pm$ 0.8	48.0 $\pm$ 3.4	42.0 $\pm$ 0.4	32.9 $\pm$ 0.5
EdgeAdding	81.5 $\pm$ 1.7	66.2 $\pm$ 0.8	49.1 $\pm$ 0.5	<b>42.9</b> $\pm$ 2.6	32.8 $\pm$ 1.1
FeatureMasking	81.9 $\pm$ 1.8	65.6 $\pm$ 1.5	48.3 $\pm$ 1.8	40.8 $\pm$ 1.3	<b>33.8</b> $\pm$ 1.5
PPRDiffusion	75.1 $\pm$ 1.8	62.0 $\pm$ 1.3	<b>50.2</b> $\pm$ 4.7	40.7 $\pm$ 0.2	33.2 $\pm$ 1.8

Table 3: Using high-pass (HP) only, low-pass only (LP) filter or both filters (HLCL) with inferred and ideal homophilic and heterophilic subgraphs (found using actual labels).

	Homophily	Heterophily	
	Cora	Chameleon	Squirrel
<b>HLCL</b>	<b>84.1</b>	<b>50.9</b>	<b>42.9</b>
LP	83.7	35.4	36.2
HP	32.5	33.1	33.1
HLCL:ideal	89.7	61.6	47.4
LP:ideal	87.1	53.7	44.9
HP:ideal	63.6	58.9	39.9

Table 4: Performance for different update intervals.

Dataset	$T=10$	$T=50$	$T=250$	No Update
Cora	80.5	<b>84.1</b>	83.1	82.1
Chameleon	41.6	<b>50.9</b>	48.3	42.7

more effective on improving the performance on homophilic graph, while feature perturbation is more effective on improving the performance on heterophilic graph. Overall, HLCL’s performance is stable across all augmentations.

## 6.2 HLCL WITH SINGLE GRAPH FILTER

Next, to confirm that both filters are necessary for HLCL’s superior performance, we examine the performance of HLCL while applying contrastive loss to either low-pass or high-pass filtered representations during training. The results are shown in Table 3. To rule out the possibility for poor subgraph sampling influencing the results, we also consider ideal subgraphs obtained via the true labels. That is, in  $\mathcal{G}^{hom}$ , only nodes of the same labels are connected, while in  $\mathcal{G}^{het}$ , only nodes of different labels are connected. First, we observe that applying contrastive loss to both high-pass and low-pass filtered representations yields the best performance, both on regular subgraphs and ideal subgraphs. This demonstrate that utilizing both representations from both frequency terms is crucial for HLCL’s success. Besides, we see that more precise homophilic and heterophilic subgraphs considerably improves the performance, and finding them more accurately is a promising direction for future work.

In Appendix A.3, we demonstrate the performance of existing graph CL methods using only high-pass filters.

## 6.3 SUBGRAPH UPDATE INTERVAL

We also conduct an ablation study on the interval between subgraph updates. The results are shown in Table 4. We see that frequent updates ( $T=10$ ) or no updates can both harm the performance on both homophilic and heterophilic graphs. Not updating the subgraphs leads to overfitting their inaccuracies, and updating them too frequently does not allow aggregating and learning the information effectively. A moderate amount of updates yields best performance.

## 6.4 HOMOPHILY RATIO CAN GUIDE TUNING

Next, we conduct a detailed study to explore the impact of varying  $k_1$  on HLCL’s performance. As observed in Table 5, different  $k_1$  (and  $k_2 = 1 - k_1$ ) values can have a significant influence on the performance. The performance on the Cora dataset varies by 30% with different  $k_1$  values, while the performance of the Chameleon dataset varies by 7%. Based on the results, the homophily ratio of the graph is a good indicator of the appropriate  $k_1$  (and  $k_2$ ) values. On both Cora and Chameleon, the best performances are achieved when  $k_1 \approx$  homophily ratio. Similar results are observed in Citeseer ( $k_1 = 0.9$ ; homophily ratio = 0.71) and Actor ( $k_1 = 0.09$ , homophily ratio = 0.09). In practice, one can sample a small subgraph and measure its homophily ratio for easier tuning.

Table 5: Performance for different values of  $k_1$ .

Dataset	0.9	0.8	0.5	0.2	0.1
Cora ( $\beta=83$ )	<b>84.1</b>	80.2	72.1	54.8	53.7
Chameleon ( $\beta=23$ )	42.0	42.0	42.0	<b>50.9</b>	45.0

## 6.5 HLCL SUBGRAPH INFERENCE

We also investigate the connectivity of the homophilic and heterophilic subgraphs inferred by HLCL. Specifically, we measured the fraction of nodes in the largest connected component of the original graph that are in the largest connected

Table 6: Connectivity of the inferred homophilic and heterophilic subgraphs.

Data	homophilic	heterophilic
Cora (.83)	1	8.5%
Citeseer (.71)	1	7.7%
Chameleon (.23)	25.6%	98.2%
Squirrel (.19)	94.4%	95%
Actor (.09)	4.7%	99.8%

component of each subgraph after sampling in Table 6. We see that under homophily (Cora, Citeseer), all the nodes are in the homophilic subgraphs and the heterophilic subgraph is small and minimally affects the performance. Under extreme heterophily (Actor) almost all the nodes are in the heterophilic subgraph and the homophilic subgraph is small and minimally affects the performance. For other graphs, depending on the tuned value of  $k_1$ , the size of the largest connected component of the two subgraphs changes.

## 6.6 USING DIFFERENT FILTERED REPRESENTATIONS AS OUTPUT

Finally, we study the performance of using different filters to produce final representations. We consider using low-pass filtered only, high-pass filtered only, and concatenating the low-pass filtered and high-pass filtered representations. The results are shown in Table 7. We observe that using low-pass filtered representations can yield better performances for both homophilic and heterophilic graphs. It is important to note that the encoder is trained using contrastive loss on both high-pass filtered heterophilic subgraphs and low-pass filtered homophilic subgraphs. This ensures that nodes in the same class have similar representations when a low-pass filter is applied, and nodes in different classes have distinct representations with a high-pass filter. During inference, the goal is to identify nodes with similar representations. Hence, using low-pass filtered representatives work better in practice, than using the high-pass filtered representations or a combination of both.

## 7 LIMITATIONS

The performance of learning from heterophily graphs heavily depends on how information is aggregated in different neighborhoods. Label availability is crucial for guiding this aggregation. In the absence of labels, a significant challenge for any graph SSL method, including HLCL, arises when nodes with similar labels cannot be approximately identified. For instance, in the Penn94 dataset, all SSL methods in Table 1, including HLCL, underperform compared to supervised methods. This is due to the lack of correlation between node feature similarity and label similarity in this dataset. In contrast, in the Cora dataset, nodes belonging to the same

Table 7: Producing final representations with different graph filters. Low-pass filtered representations has the highest performance on both homophilic and heterophilic graphs.

Model	Cora	Citeseer	Chameleon	Squirrel
LP	<b>84.1</b> $\pm$ 1.0	<b>70.1</b> $\pm$ 0.8	<b>50.9</b> $\pm$ 1.0	<b>42.9</b> $\pm$ 2.6
HP	51.9 $\pm$ 2.9	36.2 $\pm$ 2.5	35.6 $\pm$ 3.1	29.8 $\pm$ 1.6
LP+HP	74.2 $\pm$ 2.0	57.6 $\pm$ 2.0	48.7 $\pm$ 1.0	40.8 $\pm$ 2.0

class exhibit an average of 31% higher feature cosine similarity than nodes from different classes, while in Chameleon, this difference is 11%. However, in Penn94, the difference is only 1.4% on average, indicating a high similarity in node features across different classes. Consequently, SSL methods, including HLCL, face challenges in learning high-quality node representations. Despite this, HLCL outperforms other SSL baselines on Penn94, as shown in Table 1.

Additionally, on graphs where node features cannot differentiate different classes, HLCL can face challenges. For instance, for Actor dataset, nodes from different classes have similar connectivity patterns to other classes [Ma et al.]. In this case, the graph structure is not useful to classify the nodes (and may hurt the performance), and only relying on node features achieve a better performance. As shown in [Ma et al., Chen et al., 2022], models like MLP which do not use any graph structure can outperform GNN methods like GCN and even H2GCN on the Actor dataset. HGRL uses MLP as its encoder and does not leverage the graph structure, hence it can slightly outperform HLCL on Actor, but is outperformed by HLCL on other datasets.

In Sec. A.5, we discuss the effectiveness of node features in inferring subgraphs, and their limitations if used directly for classification without incorporating the graph structure.

## 8 CONCLUSION

We proposed HLCL, a contrastive learning framework that finds a homophilic and a heterophilic subgraph in a graph, applies high-pass and low-pass filters to the augmented subgraph views, and learns node representations by contrasting the filtered augmented views. This is particularly beneficial for graphs with heterophily. Through extensive experiments, we demonstrated that our proposed framework achieves up to 7% boost graphs under heterophily and outperforms popular graph supervised learning methods by up to 10%. HLCL also provides a comparable performance under homophily. We believe our work provides an important direction for future work on contrastive learning under heterophily.

**Acknowledgments.** This research was partially supported by the National Science Foundation CAREER Award 2146492.

## References

- Sami Abu-El-Haija, Bryan Perozzi, Amol Kapoor, Nazanin Alipourfard, Kristina Lerman, Hrayr Harutyunyan, Greg Ver Steeg, and Aram Galstyan. Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In *international conference on machine learning*, pages 21–29. PMLR, 2019.
- Muhammet Balcilar, Guillaume Renton, Pierre H  roux, Benoit Ga  z  re, S  bastien Adam, and Paul Honeine. Analyzing the expressive power of graph neural networks in a spectral perspective. In *International Conference on Learning Representations*, 2020.
- Piotr Bielak, Tomasz Kajdanowicz, and Nitesh V Chawla. Graph barlow twins: A self-supervised representation learning framework for graphs. *arXiv preprint arXiv:2106.02466*, 2021.
- Deyu Bo, Xiao Wang, Chuan Shi, and Huawei Shen. Beyond low-frequency information in graph convolutional networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 3950–3957, 2021.
- Jingfan Chen, Guanghui Zhu, Yifan Qi, Chunfeng Yuan, and Yihua Huang. Towards self-supervised learning on graphs with heterophily. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pages 201–211, 2022.
- Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.
- Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 15750–15758, 2021.
- Fan RK Chung. *Spectral graph theory*, volume 92. American Mathematical Soc., 1997.
- Micha  l Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29, 2016.
- Venkatesan Nallampatti Ekambaram. *Graph-structured data viewed through a Fourier lens*. University of California, Berkeley, 2014.
- Will Hamilton, Zhitaoying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.
- Kaveh Hassani and Amir Hosein Khasahmadi. Contrastive multi-view representation learning on graphs. In *International Conference on Machine Learning*, pages 4116–4126. PMLR, 2020.
- Dongxiao He, Jitao Zhao, Rui Guo, Zhiyong Feng, Di Jin, Yuxiao Huang, Zhen Wang, and Weixiong Zhang. Contrastive learning meets homophily: two birds with one stone. In *International Conference on Machine Learning*, pages 12775–12789. PMLR, 2023.
- Di Jin, Zhizhi Yu, Cuiying Huo, Rui Wang, Xiao Wang, Dongxiao He, and Jiawei Han. Universal graph convolutional networks. *Advances in Neural Information Processing Systems*, 34:10654–10664, 2021.
- Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.
- Xiang Li, Renyu Zhu, Yao Cheng, Caihua Shan, Siqiang Luo, Dongsheng Li, and Weining Qian. Finding global homophily in graph neural networks when meeting heterophily. In *International Conference on Machine Learning*, pages 13242–13256. PMLR, 2022.
- 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. *Advances in Neural Information Processing Systems*, 34:20887–20902, 2021.
- Nian Liu, Xiao Wang, Deyu Bo, Chuan Shi, and Jian Pei. Revisiting graph contrastive learning from the perspective of graph spectrum. *Advances in Neural Information Processing Systems*, 35:2972–2983, 2022.
- Yixin Liu, Yizhen Zheng, Daokun Zhang, Vincent CS Lee, and Shirui Pan. Beyond smoothing: Unsupervised graph representation learning with edge heterophily discriminating. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pages 4516–4524, 2023.
- Sitao Luan, Mingde Zhao, Chenqing Hua, Xiao-Wen Chang, and Doina Precup. Complete the missing half: Augmenting aggregation filtering with diversification for graph convolutional networks. *arXiv preprint arXiv:2008.08844*, 2020.
- Y Ma, X Liu, N Shah, and J Tang. Is homophily a necessity for graph neural networks? arxiv 2021. *arXiv preprint arXiv:2106.06134*.
- Miller McPherson, Lynn Smith-Lovin, and James M Cook. Birds of a feather: Homophily in social networks. *Annual review of sociology*, 27(1):415–444, 2001.

- Hoang Nt and Takanori Maehara. Revisiting graph neural networks: All we have is low-pass filters. *arXiv preprint arXiv:1905.09550*, 2019.
- Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.
- Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. Geom-gcn: Geometric graph convolutional networks. *arXiv preprint arXiv:2002.05287*, 2020.
- Zhen Peng, Wenbing Huang, Minnan Luo, Qinghua Zheng, Yu Rong, Tingyang Xu, and Junzhou Huang. Graph representation learning via graphical mutual information maximization. In *Proceedings of The Web Conference 2020*, pages 259–270, 2020.
- Oleg Platonov, Denis Kuznedelev, Michael Diskin, Artem Babenko, and Liudmila Prokhorenkova. A critical look at the evaluation of gnns under heterophily: are we really making progress? *arXiv preprint arXiv:2302.11640*, 2023.
- Jiezhong Qiu, Qibin Chen, Yuxiao Dong, Jing Zhang, Hongxia Yang, Ming Ding, Kuansan Wang, and Jie Tang. Gcc: Graph contrastive coding for graph neural network pre-training. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1150–1160, 2020.
- Benedek Rozemberczki, Carl Allen, and Rik Sarkar. Multi-scale attributed node embedding. *Journal of Complex Networks*, 9(2):cnab014, 2021.
- Shantanu Thakoor, Corentin Tallec, Mohammad Gheshlaghi Azar, Mehdi Azabou, Eva L Dyer, Remi Munos, Petar Veličković, and Michal Valko. Large-scale representation learning on graphs via bootstrapping. *arXiv preprint arXiv:2102.06514*, 2021.
- Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.
- Petar Velickovic, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. Deep graph infomax. *ICLR (Poster)*, 2(3):4, 2019.
- Haonan Wang, Jieyu Zhang, Qi Zhu, and Wei Huang. Can single-pass contrastive learning work for both homophilic and heterophilic graph? *arXiv preprint arXiv:2211.10890*, 2022.
- Xiao Wang, Meiqi Zhu, Deyu Bo, Peng Cui, Chuan Shi, and Jian Pei. Am-gcn: Adaptive multi-channel graph convolutional networks. In *Proceedings of the 26th ACM SIGKDD International conference on knowledge discovery & data mining*, pages 1243–1253, 2020.
- Yewen Wang, Ziniu Hu, Yusong Ye, and Yizhou Sun. Demystifying graph neural network via graph filter assessment. 2019.
- Teng Xiao, Zhengyu Chen, Zhimeng Guo, Zeyang Zhuang, and Suhang Wang. Decoupled self-supervised learning for graphs. *Advances in Neural Information Processing Systems*, 35:620–634, 2022.
- Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.
- Yihao Xue, Kyle Whitecross, and Baharan Mirzasoleiman. Investigating why contrastive learning benefits robustness against label noise. In *International Conference on Machine Learning*, pages 24851–24871. PMLR, 2022.
- 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.
- Zhilin Yang, William Cohen, and Ruslan Salakhudinov. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*, pages 40–48. PMLR, 2016.
- Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. Graph contrastive learning with augmentations. *Advances in Neural Information Processing Systems*, 33:5812–5823, 2020.
- Mengyi Yuan, Minjie Chen, and Xiang Li. Muse: Multi-view contrastive learning for heterophilic graphs. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, pages 3094–3103, 2023.
- Ziwei Zhang, Peng Cui, Xiao Wang, Jian Pei, Xuanrong Yao, and Wenwu Zhu. Arbitrary-order proximity preserved network embedding. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2778–2786, 2018.
- Jiong Zhu, Ryan A Rossi, Anup Rao, Tung Mai, Nedim Lipka, Nesreen K Ahmed, and Danai Koutra. Graph neural networks with heterophily. *arXiv preprint arXiv:2009.13566*, pages 11168–11176, 2020a.
- Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Le-man Akoglu, and Danai Koutra. Beyond homophily in graph neural networks: Current limitations and effective designs. *Advances in Neural Information Processing Systems*, 33:7793–7804, 2020b.
- Meiqi Zhu, Xiao Wang, Chuan Shi, Houye Ji, and Peng Cui. Interpreting and unifying graph neural networks with an optimization framework. In *Proceedings of the Web Conference 2021*, pages 1215–1226, 2021a.

Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. Deep graph contrastive representation learning. *arXiv preprint arXiv:2006.04131*, 2020c.

Yanqiao Zhu, Yichen Xu, Qiang Liu, and Shu Wu. An empirical study of graph contrastive learning. *arXiv preprint arXiv:2109.01116*, 2021b.

Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. Graph contrastive learning with adaptive augmentation. In *Proceedings of the Web Conference 2021*, pages 2069–2080, 2021c.

---

# Graph Contrastive Learning under Heterophily via Graph Filters (Supplementary Material)

---

Wenhan Yang<sup>1</sup>

Baharan Mirzasoleiman<sup>1</sup>

<sup>1</sup>Computer Science Department, University of California Los Angeles (UCLA)

## A APPENDIX

### A.1 HYPERPARAMETERS DETAILS

We list the details of our model hyperparameters for each datasets in Table. 8.

Table 8: HLCL hyperparameters for each dataset

	Cora	CiteS	Pubmed	Actor	Cham	Squir	Penn	TwitchG	Genius
$k_2$	0.9	0.9	0.8	0.1	0.2	0.5	1.0	1.0	1.0
$k_1$	0.1	0.1	0.2	0.9	0.8	0.5	1.0	1.0	1.0
lr	1e-3	1e-3	1e-3	1e-3	1e-3	1e-3	1e-3	1e-3	1e-3
$T$	50	50	50	50	50	50	50	50	50

### A.2 DATASET DETAILS

For graphs with homophily, we use the citation networks including Cora, Citeseer, and Pubmed [Yang et al., 2016]. For graphs with heterophily, we use the Wikipedia network and the web page networks including Chameleon, Squirrel, and Actor [Rozemberczki et al., 2021, Pei et al., 2020]. Note that, for fair comparison, we adopt the Chameleon and Squirrel from [Platonov et al., 2023] with duplicated nodes removed. To illustrate the scalability of HLCL, we also include three large-scale real-world datasets, Penn94, Genius, and Twitch-gamers provided by [Lim et al., 2021].

### A.3 EXISTING GCL METHODS WITH HP FILTERS

In this section, we illustrate the importance of our contrastive structure in achieving performance gains on heterophily datasets. We show that contrasting both the low-pass filtered graph views and high-pass filtered graph views is crucial to obtain high-quality representation under heterophily, as opposed to applying high-pass filter. To do so, we replace the LP filter with HP filter in other popular graph CL methods. The results are shown in Table 9. As demonstrated, while there are performance gains on some heterophily datasets, accuracy significantly deteriorates in homophily settings. For larger values of  $\beta$ , it is more likely that nodes with the same labels are connected together. In graphs with a large homophily ratio, most of the neighborhoods have homogeneous labels. On the other hand, graphs with a small homophily ratio contain neighborhoods with homogeneous and heterogeneous labels, as illustrated in Fig. 2. Existing graph CL methods have a very poor performance under heterophily, or low homophily ratio, and cannot learn high-quality representations.



Table 9: Using high-pass filter in existing Graph CL methods. HLCL denotes our method

	Homophily		Heterophily		
	Cora	CiteSeer	Chameleon	Squirrel	Actor
<b>HLCL</b>	84.1 $\pm$ 1.0	70.1 $\pm$ 0.8	50.9 $\pm$ 1.0	42.9 $\pm$ 2.6	34.0 $\pm$ 0.2
DGI:low	84.53 $\pm$ 1.1	71.88 $\pm$ 0.7	32.58 $\pm$ 2.9	38.83 $\pm$ 2.3	28.00 $\pm$ 1.4
DGI:high	31.95 $\pm$ 2.8	30.54 $\pm$ 1.7	29.89 $\pm$ 3.0	36.86 $\pm$ 3.0	32.03 $\pm$ 0.9
BGRL:low	83.01 $\pm$ 0.7	69.81 $\pm$ 0.6	32.58 $\pm$ 4.7	35.70 $\pm$ 1.4	28.32 $\pm$ 0.9
BGRL:high	29.63 $\pm$ 2.8	24.99 $\pm$ 3.1	37.30 $\pm$ 6.0	38.03 $\pm$ 1.1	33.87 $\pm$ 1.9
GRACE:low	83.69 $\pm$ 0.7	71.37 $\pm$ 1.0	35.39 $\pm$ 3.6	36.18 $\pm$ 2.8	34.5 $\pm$ 1.1
GRACE:high	32.46 $\pm$ 2.0	26.55 $\pm$ 3.1	33.03 $\pm$ 3.9	33.05 $\pm$ 2.1	32.00 $\pm$ 1.3

#### A.4 SIMPLIFIED HLCL

Empirically, we observed that directly contrasting the high-pass filtered representations with the low-pass filtered representations can produce comparable results to HLCL, as shown in Table 10. This simplified version can speed up the algorithm by  $2\times$ , as it requires only one contrasting learning process.

Table 10: Comparison between HLCL and Simplified HLCL

	Homophily		Heterophily		
	Cora	CiteSeer	Chameleon	Squirrel	Actor
<b>HLCL</b>	84.1 $\pm$ 1.0	70.1 $\pm$ 0.8	50.9 $\pm$ 1.0	42.9 $\pm$ 2.6	34.0 $\pm$ 0.2
<b>HLCL<sub>Simplified</sub></b>	83.5 $\pm$ 2.7	71.8 $\pm$ 1.4	48.3 $\pm$ 6.8	39.5 $\pm$ 5.3	35.5 $\pm$ 1.9

#### A.5 USING FEATURES TO INFER LABEL INFORMATION

HLCL uses feature information to approximately estimate the label information. Here, we justify this choice empirically and demonstrate that while feature information can help in inferring subgraphs approximately, it cannot be used for accurate node classification. First, we show that the node features are sufficient to give approximate neighborhood information, which is helpful in splitting the subgraph. We provide the homophily ratios of the original graph, the homophilic subgraph, and the heterophilic subgraph selected based on feature similarity across different datasets. As shown in Table 11, using feature cosine similarity, HLCL can approximately create homophilic and heterophilic subgraphs from the original graph. However, while features can approximately indicate if neighboring nodes are of the same class, they are insufficient for accurate (multi-class) node classification, and the graph structure is crucial to take into account. Otherwise, one could simply use an MLP classifier on node features. It is important to note that approximately identifying a homophilic and a heterophilic subgraph is a binary classification task, which is significantly easier than multi-class node classification. We show the insufficiency of node features for accurate classification without graph structure in Table 12. We conducted additional experiments with an MLP classifier on various homophily and heterophily datasets, which showed that MLP yields very poor performances, particularly under heterophily.

Table 11: Homophily ratios of the subgraphs sampled via node features. After sampling, homophilic subgraph has a higher homophily ratio, while the heterophilic subgraph has a lower homophily ratio compared to the original graph.

	Cora(hom)	Citeseer(hom)	Chameleon(het)	Squirrel(het)
<b>orig graph hom %</b>	0.83	0.71	0.23	0.19
<b>hom subgraph hom %</b>	0.87	0.82	0.74	0.42
<b>het subgraph hom %</b>	0.08	0.05	0.24	0.19

Table 12: Using node feature only (MLP) to classify the nodes. As shown, without graph structures, the model can only achieve sub-optimal performances.

	Cora (6 classes)	Citeseer (7 classes)	Chameleon (5 classes)	Squirrel (5 classes)
<b>MLP</b>	$64.8 \pm 1.2$	$66.5 \pm 1.0$	$37.4 \pm 2.1$	$25.5 \pm 0.9$
<b>HLCL</b>	$84.1 \pm 1.0$	$70.1 \pm 0.8$	$50.9 \pm 1.0$	$42.9 \pm 2.6$

## A.6 EXTENDED RELATED WORK

**(Semi-)supervised learning on graphs.** In recent years, GNNs have become one of the most prominent tools for processing graph-structured data. In general, GNNs utilize the adjacency matrix to learn the node representations, by aggregating information within every node’s neighborhood [Defferrard et al., 2016, Kipf and Welling, 2016]. Existing variants, including GraphSAGE [Hamilton et al., 2017], Graph Attention (GAT) [Veličković et al., 2017], MixHop [Abu-El-Haija et al., 2019], SGC [Nt and Maehara, 2019], GAT [Velickovic et al., 2019], and GIN [Xu et al., 2018], learn a more general class of neighborhood mixing relationships, by aggregating weighted information within a multi-hop neighborhood of every node. GNNs can be generally seen as applying a fix, or a parametric and learnable (e.g. GAT) low-pass graph filter to graph signals. Those with trainable parameters can adapt to a wider range of frequency levels on different graphs. However, they still have a higher emphasis on lower-frequency signals and discard the high-frequency signals in a graph. While the aggregation operation makes GNNs powerful tools for semi-supervised learning, it can make the learned node representations indistinguishable in a neighborhood [Nt and Maehara, 2019]. As a result, typical GNNs and their variants have been long criticized for their poor generalization performance under heterophily [Balcilar et al., 2020].

**Graph self-supervised learning.** Graph self-supervised learning methods have become a powerful tool for learning representations without any labels, and graph contrastive learning is the most successful and popular model structure. Numerous methods have been proposed in the field: [Velickovic et al., 2019, Peng et al., 2020, Hassani and Khasahmadi, 2020, Zhu et al., 2021c] focus on contrasting the global augmented representation with the local augmented representation, while [Zhu et al., 2020c, You et al., 2020, Qiu et al., 2020, Liu et al., 2022] contrast same-scale representation, global or local, in two augmented views. Due to the complexity of collecting negative samples in graph data, negative-sample free contrastive objectives have also been studied [Thakoor et al., 2021, Bielak et al., 2021]. However, works mentioned above focus on encoding the homophily graphs and perform poorly on graphs with heterophily. Recently, a stream of self-supervised learning methods have been proposed to learn effectively the node representations of the heterophily graphs without any labels. HGRL [Chen et al., 2022] improves the node representations on heterophilic graphs by preserving the node original features and rewiring informative nodes that are not in the local neighborhood. SP-GCL [Wang et al., 2022] proposed using nodes from the T-hop neighborhood of a node with high feature similarities as positive pairs, without using any explicit augmentations. DSSL [Xiao et al., 2022] separates the heterogeneous patterns in local neighborhood distributions to capture both homophilic and heterophilic information globally. GREET [Liu et al., 2023] discriminates homophilic edges from heterophilic edges using random walk based graph diffusion and contrasts the projected representations of the two graph views directly via a dual-channel contrastive loss. MUSE [Yuan et al., 2023] utilize semantic view contrast based on ego node feature perturbations and contextual view contrast based on topology perturbations. Then, it integrates the representations learned from both contrasting views to construct a fusion contrast that combines both structural and semantic information. NeCo [He et al., 2023] proposes a new pretext task, group discrimination, which divides the nodes into k groups and keeps the consistent representation of nodes within a group.

**Graph (semi-)supervised learning under heterophily.** To address over-smoothing issue of GNNs, recent methods propose to use other types of aggregation that better fit graphs with heterophily. Geom-GCN uses geometric aggregation in place of the typical aggregation [Pei et al., 2020], H<sub>2</sub>GCN uses several special model designs including separate aggregation and higher-neighborhood aggregation to train the model for handling graphs with heterophily, and CPGNN trains a compatibility matrix to model the heterophily level [Zhu et al., 2020a]. More recently, Wang et al. [2019] proposed to learn an aggregation filter for every graph from a set of based filters designed based on different ways of normalizing the adjacency matrix. GGCN introduced degree corrections and signed message passing on GCN to address both oversmoothing problems and the model’s poor performances on heterophily graphs [Yan et al., 2021]. Zhu et al. [2021a] analyzed and designed a uniform framework for GNNs propagations and proposed GNN-LF and GNN-HF that preserve information of different frequency separately by using different filtering kernels with learnable weights. FAGCN [Bo et al., 2021] and FBGNN [Luan et al., 2020] train two *separate* encoders to capture the high-pass and low-pass graph signals separately. Then they rely on labels to learn relatively complex mechanisms to combine the outputs of the encoders. However, learning how to combine the encoder outputs is highly sensitive to having high-quality labels. This makes such methods highly impractical for unsupervised

contrastive learning, where the label information is not available.

Unlike the above supervised methods, we apply the high-pass and low-pass filters to different subgraphs, contrasting the resulting high-pass filtered node views and low-pass filtered node views in a self-supervised manner, without any label. This is in contrast to learning the best combination of filtered signals of different encoders based on labels.

## B PROOF

**Assumption 1.** Let  $\mathbf{X}$  be the feature matrix of  $\mathcal{G}^{hom}$  and  $\mathbf{W}$  be the learnable weights of the GNN encoder. Then,

$$\mathbf{X}\mathbf{W}\mathbf{W}\mathbf{X} = w_0 + w_1\mathbf{A}^{hom} + w_2(\mathbf{A}^{hom})^2 + \dots + w_j(\mathbf{A}^{hom})^j.$$

$\mathbf{X}\mathbf{W}\mathbf{W}\mathbf{X}$  under homophily captures the similarities of features between every two nodes in the subgraph after passing through the low-pass graph filter. Assumption 1 aims to expand  $\mathbf{X}\mathbf{W}\mathbf{W}\mathbf{X}$  with the weighted sum of different orders of  $\mathbf{A}$ . Here,  $w_i$  s are the weights of different orders of  $\mathbf{A}$ . That is  $w_i$  is the weight of  $i$ -th order of  $\mathbf{A}$ , representing the number of length- $i$  paths between nodes  $i$  and  $j$  in its  $(i, j)$  entry. For homophilic subgraphs, which adhere to the homophily principle, the weights for closer-hop connections (represented by  $\mathbf{A}$ ,  $\mathbf{A}^2$ , etc.) are higher, since the closer the nodes are, the more similar they are. This is based on the homophily principle [McPherson et al., 2001, Luan et al., 2020]. This principle suggests that, in homophily graphs, nodes within closer neighborhoods exhibit greater feature similarities. After projection, the similarities also become higher [Zhang et al., 2018].

**Assumption 2.** Let  $\mathbf{X}$  be the feature matrix of  $\mathcal{G}^{het}$  and  $\mathbf{W}$  be the learnable weights of the GNN encoder. Then,

$$\mathbf{X}\mathbf{W}\mathbf{W}\mathbf{X} = w_0 + w_1\mathbf{L}^{het} + w_2(\mathbf{L}^{het})^2 + \dots + w_j(\mathbf{L}^{het})^j.$$

$\mathbf{X}\mathbf{W}\mathbf{W}\mathbf{X}$  under heterophily captures the dissimilarities of features between every two nodes in the subgraph after passing through the high-pass graph filters. Assumption 2 aims to expand  $\mathbf{X}\mathbf{W}\mathbf{W}\mathbf{X}$  with the weighted sum of different orders of  $\mathbf{L}$ . Here,  $w_i$  is the weight of  $i$ -th order of  $\mathbf{L}$ . In contrast to homophilic graphs, for heterophilic subgraphs, the closer the nodes are, the more dissimilar they are [Zhu et al., 2020c].

**Lemma 1.** Let  $\mathbf{A}$  and  $\tilde{\mathbf{A}}$  be adjacency matrices of the target graph and its augmented counterpart. Suppose that  $\mathbf{A}$  and  $\tilde{\mathbf{A}}$  have the same eigenspaces, and let  $\mathbf{D}$  and  $\tilde{\mathbf{D}}$  be the corresponding degree matrices, where  $\mathbf{D} = \tilde{\mathbf{D}}$ . Then the Laplacian matrices  $\mathbf{L}$  and  $\tilde{\mathbf{L}}$  have the same eigenspaces.

*Proof.* Given that  $\mathbf{A}$  and  $\tilde{\mathbf{A}}$  have the same eigenspaces, there exists an orthogonal matrix  $\mathbf{Q}$  such that:

$$\mathbf{A} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T \quad \text{and} \quad \tilde{\mathbf{A}} = \mathbf{Q}\tilde{\mathbf{\Lambda}}\mathbf{Q}^T$$

where  $\mathbf{\Lambda}$  and  $\tilde{\mathbf{\Lambda}}$  are diagonal matrices containing the eigenvalues of  $\mathbf{A}$  and  $\tilde{\mathbf{A}}$ , respectively. Since  $\mathbf{D} = \tilde{\mathbf{D}}$ , let  $\mathbf{D} = \tilde{\mathbf{D}}$ . The Laplacian matrices are defined as:

$$\mathbf{L} = \mathbf{D} - \mathbf{A} \quad \text{and} \quad \tilde{\mathbf{L}} = \mathbf{D} - \tilde{\mathbf{A}}$$

Substituting the spectral decompositions of  $\mathbf{A}$  and  $\tilde{\mathbf{A}}$ , we have:

$$\mathbf{L} = \mathbf{D} - \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^T$$

$$\tilde{\mathbf{L}} = \mathbf{D} - \mathbf{Q}\tilde{\mathbf{\Lambda}}\mathbf{Q}^T$$

Both  $\mathbf{L}$  and  $\tilde{\mathbf{L}}$  can be written as:

$$\mathbf{L} = \mathbf{Q}(\mathbf{Q}^T\mathbf{D}\mathbf{Q} - \mathbf{\Lambda})\mathbf{Q}^T$$

$$\tilde{\mathbf{L}} = \mathbf{Q}(\mathbf{Q}^T\mathbf{D}\mathbf{Q} - \tilde{\mathbf{\Lambda}})\mathbf{Q}^T$$

Since  $\mathbf{D}$  is diagonal,  $\mathbf{Q}^T\mathbf{D}\mathbf{Q}$  remains a diagonal matrix (as the orthogonal transformation of a diagonal matrix preserves diagonal structure). Let  $\mathbf{D}' = \mathbf{Q}^T\mathbf{D}\mathbf{Q}$ , then:

$$\mathbf{L} = \mathbf{Q}(\mathbf{D}' - \mathbf{\Lambda})\mathbf{Q}^T$$

$$\tilde{\mathbf{L}} = \mathbf{Q}(\mathbf{D}' - \tilde{\mathbf{\Lambda}})\mathbf{Q}^T$$

The eigenvalues of  $\mathbf{L}$  and  $\tilde{\mathbf{L}}$  are given by the diagonal entries of  $\mathbf{D}' - \mathbf{\Lambda}$  and  $\mathbf{D}' - \tilde{\mathbf{\Lambda}}$ , respectively. Since  $\mathbf{Q}$  is the same for both  $\mathbf{L}$  and  $\tilde{\mathbf{L}}$ , they have the same eigenspaces. Thus,  $\mathbf{L}$  and  $\tilde{\mathbf{L}}$  have the same eigenspaces.  $\square$

## B.1 THEOREM 1 [HLCL: SPECTRAL INVARIANCE]

Given a graph  $\mathcal{G}$ , we infer a homophilic and a heterophilic subgraph from it, denoted as  $\mathcal{G}_{\text{hom}}$  and  $\mathcal{G}_{\text{het}}$ , respectively. Their augmented counterparts are denoted as  $\tilde{\mathcal{G}}_{\text{hom}}$  and  $\tilde{\mathcal{G}}_{\text{het}}$ . For graph augmentations, we follow [Liu et al., 2022], where the adjacency matrix of the homophilic subgraph and the augmented homophilic subgraph share the same eigenspaces ( $\mathbf{A}_{\text{hom}}$  and  $\tilde{\mathbf{A}}_{\text{hom}}$ ). Similarly, the adjacency matrix of the heterophilic subgraph and the augmented heterophilic subgraph share the same eigenspaces ( $\mathbf{A}_{\text{het}}$  and  $\tilde{\mathbf{A}}_{\text{het}}$ ). By Lemma 1, the Laplacian matrix of the homophilic subgraph and the augmented homophilic subgraph share the same eigenspaces ( $\mathbf{L}_{\text{hom}}$  and  $\tilde{\mathbf{L}}_{\text{hom}}$ ), and the Laplacian matrix of the heterophilic subgraph and the augmented heterophilic subgraph share the same eigenspaces ( $\mathbf{L}_{\text{het}}$  and  $\tilde{\mathbf{L}}_{\text{het}}$ ).

We establish the following lower bound:

$$\mathcal{L}_{\text{HLCL}} \geq \frac{-1-N}{2} \sum_i \left( \alpha_{\mathbf{A}_i} \left( 2 - (\lambda_{\mathbf{A}_i^{\text{hom}}} - \lambda_{\tilde{\mathbf{A}}_i^{\text{hom}}})^2 \right) + \alpha_{\mathbf{L}_i} \left( 4 - (\lambda_{\mathbf{L}_i^{\text{het}}} - \lambda_{\tilde{\mathbf{L}}_i^{\text{het}}})^2 \right) \right)$$

where  $\lambda_{\mathbf{A}^{\text{hom}}}$  and  $\lambda_{\mathbf{L}^{\text{het}}}$  denote the eigenvalues of the homophilic subgraph low-pass filter and the heterophilic subgraph high-pass filter, respectively, and  $\alpha_{\mathbf{A}^{\text{hom}}}$  and  $\alpha_{\mathbf{L}^{\text{het}}}$  denote the adaptive weights for the  $i$ -th adjacency and Laplacian matrix components.

*Proof.* By minimizing the HLCL loss, we minimize the losses for contrasting augmented views of both heterophilic and homophilic subgraphs. Hence we discuss each in our proof.

For simplification, since the HLCL loss is symmetric, we only choose one graph view as the anchor view.

$$\mathcal{L} = -\frac{1}{2N} \sum_{i=1}^N (l(\mathbf{z}_l^i, \tilde{\mathbf{z}}_l^i) + l(\mathbf{z}_h^i, \tilde{\mathbf{z}}_h^i)) \quad (16)$$

$$= -\frac{1}{2N} \sum_{i=1}^N \left( \log \frac{e^{\mathbf{z}_l^i \tilde{\mathbf{z}}_l^{iT}}}{e^{\mathbf{z}_h^i \tilde{\mathbf{z}}_h^{iT}} + \sum_{\substack{k \in [N], \\ k \neq i}} e^{\mathbf{z}_l^i \tilde{\mathbf{z}}_l^{kT}}} + \log \frac{e^{\mathbf{z}_h^i \tilde{\mathbf{z}}_h^{iT}}}{e^{\mathbf{z}_l^i \tilde{\mathbf{z}}_l^{iT}} + \sum_{\substack{k \in [N], \\ k \neq i}} e^{\mathbf{z}_h^i \tilde{\mathbf{z}}_h^{kT}}} \right) \quad (17)$$

$$= -\frac{1}{2N} \sum_{i=1}^N (z_l^i \tilde{z}_l^{iT} + z_h^i \tilde{z}_h^{iT} - \log \sum_k e^{\mathbf{z}_l^i \tilde{\mathbf{z}}_l^{kT}} - \log \sum_k e^{\mathbf{z}_h^i \tilde{\mathbf{z}}_h^{kT}}) \quad (18)$$

$$\geq -\frac{1}{2N} \sum_{i=1}^N (z_l^i \tilde{z}_l^{iT} + z_h^i \tilde{z}_h^{iT} - \log N \cdot e^{\sum_k \mathbf{z}_l^i \tilde{\mathbf{z}}_l^{kT} / N} - \log N \cdot e^{\sum_k \mathbf{z}_h^i \tilde{\mathbf{z}}_h^{kT} / N}) \quad (19)$$

$$\equiv -\sum_{i=1}^N (z_l^i \tilde{z}_l^{iT} + z_h^i \tilde{z}_h^{iT} - \frac{1}{N} \sum_N \mathbf{z}_l^i \tilde{\mathbf{z}}_l^{iT} + \mathbf{z}_h^i \tilde{\mathbf{z}}_h^{iT}) \quad (20)$$

$$= -(tr(\mathbf{Z}_l \tilde{\mathbf{Z}}_l^T) + tr(\mathbf{Z}_h \tilde{\mathbf{Z}}_h^T) - \frac{1}{N} sum(\mathbf{Z}_l \tilde{\mathbf{Z}}_l^T) - \frac{1}{N} sum(\mathbf{Z}_h \tilde{\mathbf{Z}}_h^T)) \quad (21)$$

$\mathbf{Z}_l$  is the projected representation of  $\mathcal{G}_{\text{hom}}$ ,  $\tilde{\mathbf{Z}}_l$  is the projected representation of  $\tilde{\mathcal{G}}_{\text{hom}}$ ,  $\mathbf{Z}_h$  is the projected representation of  $\mathcal{G}_{\text{het}}$ , and  $\tilde{\mathbf{Z}}_h$  is the projected representation of  $\tilde{\mathcal{G}}_{\text{het}}$ . As mentioned before,  $\mathbf{A}_{\text{hom}}$  and  $\tilde{\mathbf{A}}_{\text{hom}}$  share the same eigenspaces, so we have that  $\mathbf{A}_{\text{hom}} = \mathbf{Q}_{\text{hom}} \Lambda_{\text{hom}} \mathbf{Q}_{\text{hom}}^T$  and  $\tilde{\mathbf{A}}_{\text{hom}} = \mathbf{Q}_{\text{hom}} \tilde{\Lambda}_{\text{hom}} \mathbf{Q}_{\text{hom}}^T$ , where  $\mathbf{Q}_{\text{hom}}$  is the collection of eigenspaces, and  $\Lambda_{\text{hom}} = \text{diag}(\lambda_{\mathbf{A}_{\text{hom}}})$  and  $\tilde{\Lambda}_{\text{hom}} = \text{diag}(\lambda_{\tilde{\mathbf{A}}_{\text{hom}}})$  are their diagonal weight matrices. Similarly,  $\mathbf{A}_{\text{het}} = \mathbf{Q}_{\text{het}} \Lambda_{\text{het}} \mathbf{Q}_{\text{het}}^T$  and  $\tilde{\mathbf{A}}_{\text{het}} = \mathbf{Q}_{\text{het}} \tilde{\Lambda}_{\text{het}} \mathbf{Q}_{\text{het}}^T$ , where  $\mathbf{Q}_{\text{het}}$  is the collection of eigenspaces, and  $\Lambda_{\text{het}} = \text{diag}(\lambda_{\mathbf{L}_{\text{het}}})$  and  $\tilde{\Lambda}_{\text{het}} = \text{diag}(\lambda_{\tilde{\mathbf{L}}_{\text{het}}})$ . With the simplification of the HLCL loss, we have  $\mathbf{Z}_h \tilde{\mathbf{Z}}_h^T = \mathbf{LXWWX}\tilde{\mathbf{L}}$  and  $\mathbf{Z}_l \tilde{\mathbf{Z}}_l^T = \mathbf{AXWWX}\tilde{\mathbf{A}}$ , where  $\mathbf{W}$  is learnable parameters of the encoder.

**Lemma 2.** With assumption 1, for homophilic subgraph  $\mathcal{G}^{\text{hom}}$ , when  $j \geq N-1$ ,  $\mathbf{XWWX} = w_0 + w_1 \mathbf{A}^{\text{hom}} + w_2 (\mathbf{A}^{\text{hom}})^2 + \dots + w_j (\mathbf{A}^{\text{hom}})^j = \mathbf{Q}_{\text{hom}} \mathbf{A}_{\text{hom}} \mathbf{Q}_{\text{hom}}^T$ , where  $\mathbf{A}_{\text{hom}} = \text{diag}(\alpha_{\mathbf{A}_1} \dots \alpha_{\mathbf{A}_N})$ .  $\alpha_{\mathbf{A}_1} \dots \alpha_{\mathbf{A}_N}$  are  $N$  different parameters, if  $\lambda_{\mathbf{A}_1^{\text{hom}}} \dots \lambda_{\mathbf{A}_N^{\text{hom}}}$  are  $N$  different frequency amplitudes.

*Proof.* The proof can be found in Theorem 4 of [Liu et al., 2022]. □

**Lemma 3.** With assumption 2, for heterophilic subgraph  $\mathcal{G}^{het}$ , when  $j \geq N-1$ ,  $\mathbf{XWWX} = w_0 + w_1 \mathbf{L}^{het} + w_2 (\mathbf{L}^{het})^2 + \dots + w_j (\mathbf{L}^{het})^j = \mathbf{Q}_{het} \mathbf{A}_{het} \mathbf{Q}_{het}^T$ , where  $\mathbf{A}_{het} = \text{diag}(\alpha_{\mathbf{L}_1} \dots \alpha_{\mathbf{L}_N})$ .  $\alpha_{\mathbf{L}_1} \dots \alpha_{\mathbf{L}_N}$  are  $N$  different parameters, if  $\lambda_{\mathbf{A}_1^{het}} \dots \lambda_{\mathbf{A}_N^{het}}$  are  $N$  different frequency amplitudes.

*Proof.* The proof can be found in Theorem 4 of [Liu et al., 2022], by replacing  $\mathbf{L}$  as the decomposing matrix.  $\square$

For  $\mathbf{Z}_l \tilde{\mathbf{Z}}_l^T$ , using Lemma 2, we have:

$$\begin{aligned}
\mathbf{Z}_l \tilde{\mathbf{Z}}_l^T &= \mathbf{A} \mathbf{X} \mathbf{W} \mathbf{W} \mathbf{X} \tilde{\mathbf{A}} \\
&= \mathbf{Q}_{hom} \mathbf{\Lambda}_{hom} \mathbf{Q}_{hom}^T \mathbf{Q}_{hom} \mathbf{A}_{hom} \mathbf{Q}_{hom}^T \mathbf{Q}_{hom} \tilde{\mathbf{\Lambda}}_{hom} \mathbf{Q}_{hom}^T \\
&= \mathbf{Q}_{hom} \mathbf{\Lambda}_{hom} \mathbf{A}_{hom} \tilde{\mathbf{\Lambda}}_{hom} \mathbf{Q}_{hom}^T \\
&= \mathbf{Q}_{hom} \begin{bmatrix} \lambda_{\mathbf{A}_1^{hom}} \alpha_{\mathbf{A}_1} \lambda_{\tilde{\mathbf{A}}_1^{hom}} & 0 & \dots & 0 \\ 0 & \lambda_{\mathbf{A}_2^{hom}} \alpha_{\mathbf{A}_2} \lambda_{\tilde{\mathbf{A}}_2^{hom}} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_{\mathbf{A}_N^{hom}} \alpha_{\mathbf{A}_N} \lambda_{\tilde{\mathbf{A}}_N^{hom}} \end{bmatrix} \mathbf{Q}_{hom}^T \\
&= \sum_{i=1}^N \lambda_{\mathbf{A}_i^{hom}} \alpha_{\mathbf{A}_i} \lambda_{\tilde{\mathbf{A}}_i^{hom}} q_{\mathbf{A}_i} q_{\tilde{\mathbf{A}}_i}^T,
\end{aligned}$$

where  $q_{\mathbf{A}_i}$  is the  $i^{th}$  column of the matrix  $\mathbf{Q}_{hom}$ .

For  $\mathbf{Z}_h \tilde{\mathbf{Z}}_h^T$ , using Lemma 3, we have:

$$\begin{aligned}
\mathbf{Z}_h \tilde{\mathbf{Z}}_h^T &= \mathbf{L} \mathbf{X} \mathbf{W} \mathbf{W} \mathbf{X} \tilde{\mathbf{L}} \\
&= \mathbf{Q}_{het} \mathbf{\Lambda}_{het} \mathbf{Q}_{het}^T \mathbf{Q}_{het} \mathbf{A}_{het} \mathbf{Q}_{het}^T \mathbf{Q}_{het} \tilde{\mathbf{\Lambda}}_{het} \mathbf{Q}_{het}^T \\
&= \mathbf{Q}_{het} \mathbf{\Lambda}_{het} \mathbf{A}_{het} \tilde{\mathbf{\Lambda}}_{het} \mathbf{Q}_{het}^T \\
&= \mathbf{Q}_{het} \begin{bmatrix} \lambda_{\mathbf{L}_1^{het}} \alpha_{\mathbf{L}_1} \lambda_{\tilde{\mathbf{L}}_1^{het}} & 0 & \dots & 0 \\ 0 & \lambda_{\mathbf{L}_2^{het}} \alpha_{\mathbf{L}_2} \lambda_{\tilde{\mathbf{L}}_2^{het}} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_{\mathbf{L}_N^{het}} \alpha_{\mathbf{L}_N} \lambda_{\tilde{\mathbf{L}}_N^{het}} \end{bmatrix} \mathbf{Q}_{het}^T \\
&= \sum_{i=1}^N \lambda_{\mathbf{L}_i^{het}} \alpha_{\mathbf{L}_i} \lambda_{\tilde{\mathbf{L}}_i^{het}} q_{\mathbf{L}_i} q_{\tilde{\mathbf{L}}_i}^T,
\end{aligned}$$

where  $q_{\mathbf{L}_i}$  is the  $i^{th}$  column of the matrix  $\mathbf{Q}_{het}$ . Therefore, we have:

$$\begin{aligned}
\text{tr}(\mathbf{Z}_l \tilde{\mathbf{Z}}_l^T) &= \sum_{i=1}^N \lambda_{\mathbf{A}_i^{hom}} \alpha_{\mathbf{A}_i} \lambda_{\tilde{\mathbf{A}}_i^{hom}}, \quad \text{sum}(\mathbf{Z}_l \tilde{\mathbf{Z}}_l^T) = \sum_i \lambda_{\mathbf{A}_i^{hom}} \alpha_{\mathbf{A}_i} \lambda_{\tilde{\mathbf{A}}_i^{hom}} \text{sum}(q_{\mathbf{A}_i} q_{\tilde{\mathbf{A}}_i}^T) \\
\text{tr}(\mathbf{Z}_h \tilde{\mathbf{Z}}_h^T) &= \sum_{i=1}^N \lambda_{\mathbf{L}_i^{het}} \alpha_{\mathbf{L}_i} \lambda_{\tilde{\mathbf{L}}_i^{het}}, \quad \text{sum}(\mathbf{Z}_h \tilde{\mathbf{Z}}_h^T) = \sum_i \lambda_{\mathbf{L}_i^{het}} \alpha_{\mathbf{L}_i} \lambda_{\tilde{\mathbf{L}}_i^{het}} \text{sum}(q_{\mathbf{L}_i} q_{\tilde{\mathbf{L}}_i}^T)
\end{aligned}$$

By substituting this into Eq. (21), we have

$$\begin{aligned}
\mathcal{L}_{HLCL} &\geq - \left( \sum_{i=1}^N \left( \lambda_{\mathbf{A}_i^{hom}} \alpha_{\mathbf{A}_i} \lambda_{\tilde{\mathbf{A}}_i^{hom}} + \lambda_{\mathbf{L}_i^{het}} \alpha_{\mathbf{L}_i} \lambda_{\tilde{\mathbf{L}}_i^{het}} \right) - \frac{1}{N} \sum_{i=1}^N \left( \lambda_{\mathbf{A}_i^{hom}} \alpha_{\mathbf{A}_i} \lambda_{\tilde{\mathbf{A}}_i^{hom}} \sum (q_{\mathbf{A}_i} q_{\tilde{\mathbf{A}}_i}^T) + \lambda_{\mathbf{L}_i^{het}} \alpha_{\mathbf{L}_i} \lambda_{\tilde{\mathbf{L}}_i^{het}} \sum (q_{\mathbf{L}_i} q_{\tilde{\mathbf{L}}_i}^T) \right) \right) \\
&= - \left( \sum_{i=1}^N \lambda_{\mathbf{A}_i^{hom}} \alpha_{\mathbf{A}_i} \lambda_{\tilde{\mathbf{A}}_i^{hom}} \left( 1 - \frac{1}{N} \sum (q_{\mathbf{A}_i} q_{\tilde{\mathbf{A}}_i}^T) \right) + \lambda_{\mathbf{L}_i^{het}} \alpha_{\mathbf{L}_i} \lambda_{\tilde{\mathbf{L}}_i^{het}} \left( 1 - \frac{1}{N} \sum (q_{\mathbf{L}_i} q_{\tilde{\mathbf{L}}_i}^T) \right) \right)
\end{aligned}$$

Since  $q_i^T q_i = 1$ ,  $|q_{ij}| < 1$ ,  $\sum (q_i q_i^T) > -N^2$ , we have

$$\mathcal{L}_{HLCL} \geq (-1 - N) \sum_{i=1}^N \left( \lambda_{\mathbf{A}_i^{\text{hom}}} \alpha_{\mathbf{A}_i} \lambda_{\bar{\mathbf{A}}_i^{\text{hom}}} + \lambda_{\mathbf{A}_i^{\text{het}}} \alpha_{\mathbf{L}_i} \lambda_{\bar{\mathbf{A}}_i^{\text{het}}} \right).$$

Since  $\lambda_{\mathbf{A}_i^{\text{hom}}} \in (-1, 1]$ , and  $\lambda_{\mathbf{L}_i^{\text{het}}} \in [0, 2)$ , we have

$$\mathcal{L}_{HLCL} \geq \frac{-1 - N}{2} \sum_{i=1}^N \left( \alpha_{\mathbf{A}_i} \left( 2 - (\lambda_{\mathbf{A}_i^{\text{hom}}} - \lambda_{\bar{\mathbf{A}}_i^{\text{hom}}})^2 \right) + \alpha_{\mathbf{L}_i} \left( 4 - (\lambda_{\mathbf{L}_i^{\text{het}}} - \lambda_{\bar{\mathbf{L}}_i^{\text{het}}})^2 \right) \right).$$

□