GRAPH CONTRASTIVE LEARNING UNDER HETEROPHILY: UTILIZING GRAPH FILTERS TO GENERATE GRAPH VIEWS

Anonymous authors
Paper under double-blind review

ABSTRACT

Graph Neural Networks have achieved tremendous success in (semi-)supervised tasks for which task-specific node labels are available. However, obtaining labels is expensive in many domains, specially as the graphs grow larger in size. Hence, there has been a growing interest in the application of self-supervised techniques, in particular contrastive learning (CL), to graph data. In general, CL methods work by maximizing the agreement between encoded augmentations of the same example, and minimizing agreement between encoded augmentations of different examples. However, we show that existing graph CL methods perform very poorly on graphs with heterophily, in which connected nodes tend to belong to different classes. First, we show that this is attributed to the ineffectiveness of existing graph augmentation methods. Then, we leverage graph filters to directly generate augmented graph views for graph CL under heterophily. In particular, instead of explicitly augmenting the graph topology and encoding the augmentations, we use a high-pass filter in the encoder to generate node representations only based on high-frequency graph signals. Then, we contrast the high-pass filtered representations with their low-pass counterparts produced by the same encoder, to generate representations. Our experimental results confirm that our proposed method, HLCL, outperforms state-of-the-art CL methods on benchmark graphs with heterophily, by up to 10%.

1 INTRODUCTION

Graph neural networks (GNNs) are powerful tools for learning graph-structured data in various domains, including social networks, biological compound structures, and citation networks (Kipf & Welling, 2016; Hamilton et al., 2017; Veličković et al., 2017). In general, GNNs leverage the graph's adjacency matrix to update the node representations by aggregating information from their neighbors. This can be seen as a low-pass filter that smooths the graph signals and produces similar node representations (Nt & Maehara, 2019). GNNs have achieved great success in supervised and semi-supervised learning, where task-specific labels are available. However, obtaining high-quality labels is very expensive in many domains, specially as graphs grow larger in size. This has motivated a recent body of work on self-supervised learning on graphs that learn the representations in an unsupervised manner (Velickovic et al., 2019; Peng et al., 2020; Qiu et al., 2020; Hassani & Khasahmadi, 2020; Zhu et al., 2020b). Among self-supervised methods, Contrastive Learning (CL) has shown a great success by achieving comparable performance with its supervised counterparts (Chen et al., 2020). Contrastive learning obtains representations by maximizing the mutual information between different augmented views of the same example, and minimizing agreement between differently augmented views of different examples. Despite being successful on graphs with homophily, where neighboring nodes tend to share the same label, existing graph CL methods cannot learn high-quality representations for graphs with heterophily, where connected nodes often belong to different classes Zhu et al. (2020b).

State-of-the-art graph CL methods work by contrasting the encoded node representations in two explicitly augmented graph views, generated by altering the graph topology or node features (Zhu et al., 2020c; 2021b;

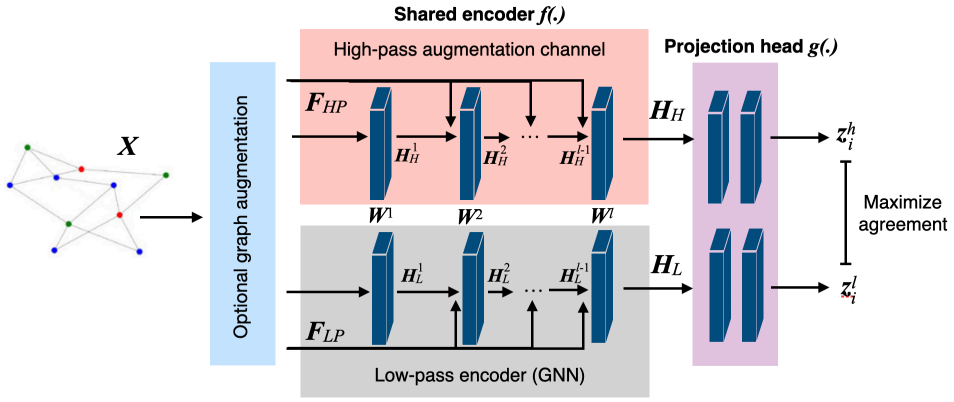

Figure 1: Our proposed HLCL framework leverages a high-pass graph filter in the encoder to generate a non-smooth graph view, and contrast it with its smooth counterparts generated by the same encoder.

Velickovic et al., 2019; Thakoor et al., 2021). Specifically, topology augmentation methods alter the graph structure by removing or adding nodes or edges. Feature augmentation methods alter the node features by masking particular columns, dropping features at random, or randomly shuffling the node features. The choice of the augmentation strategy has been shown to have a great influence on the performance of graph CL methods (Zhang & Ma, 2022; Zhu et al., 2021b). In particular, topology augmentations such as edge dropping that produce sparser graphs are the most effective for graphs with homophily (Zhu et al., 2021b).

In this work, we first study the effect of existing graph augmentation methods on the performance of graph CL under heterophily. We show that topology and feature augmentation methods that can effectively boost the performance of graph CL under homophily do not provide any considerable advantage under heterophily. In particular, we experimentally confirm that for graphs with homophily, topology augmentation effectively decreases the homophily ratio, and feature augmentation changes the variance of features in a neighborhood. Hence, explicit graph augmentation techniques enable the low-pass GNN encoder to better distinguish the smooth node representations—that are similar in a neighborhood—by adding more diversity to the graph. However, such methods cannot effectively diversify and alter the homophily ratio or variance of features in graphs with heterophily. Indeed, for such graphs, incorporating the graph structure is crucial for generating high-quality representations that can distinguish dissimilar nodes from their neighborhood.

Next, we address the above shortcoming by proposing an effective augmentation strategy for CL on graphs with heterophily, while ensuring high-quality representation learning on graphs with homophily. The key idea of our proposed method, HLCL, is to leverage a high-pass graph filter in a typical GNN encoder to directly generate non-smooth node representations, and contrast them with their smooth counterparts generated by the same encoder. More specifically, we use the normalized Lapcalian matrix as the high-pass filter in the GNN encoder to aggregate the node representations. The Laplacian matrix magnifies the differences in the node features in a neighborhood and makes the representations distinct. This is crucial for learning high-quality representation under heterophily. In contrast, a typical GNN encoder uses the normalized adjacency matrix as a low-pass filter to aggregate the node features with those of their neighbors, and enforce similar representations for the nodes in the same neighborhood. Maximizing the mutual information between the high-pass and low-pass filtered representations enables the encoder to learn rich representations on graphs with heterophily. Importantly, our method is complementary to explicit graph augmentation techniques, and such methods can be applied to further boost the performance, particularly on graphs with homophily. This allows our framework to achieve state-of-the-art under heterophily and comparable performance under homophily as we confirm by our experiments.

We show the effectiveness of our HLCL framework through extensive experiments on graphs with heterophily and homophily for unsupervised representations learning under the linear evaluation protocol. Our

results demonstrate that, on five popular public benchmark datasets, our framework can outperform existing graph CL methods by up to 10% for representation learning on graphs with heterophily, while ensuring a comparable performance to state-of-the-art graph CL methods on graphs with homophily.

2 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 & 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 & 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 often leads to over-smoothing issue, i.e., the learned node representations become indistinguishable in a neighborhood (Nt & Maehara, 2019). As a result, typical GNNs and their variants have been long criticized on their inability to capture graph heterophily and poor generalization performance on heterogeneous dataset (Balcilar et al., 2020), where most connected nodes are from different classes.

(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_2GCN 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. Most recently, 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) also leveraged the Laplacian matrix to capture the non-smooth signals discarded in typical GNNs. Two separate encoders are trained to capture the high-pass and low-pass graph signals separately, and the two outputs are combined with learnable parameters to balance their importance. Such methods achieve a superior performance under heterophily. 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.

Contrastive learning on graphs. Self-supervised contrastive learning methods learn representations of data points by maximizing the mutual information between different views of the same data point, and minimizing agreement between differently augmented views of different examples (Bachman et al., 2019; Ye et al., 2019; Wu et al., 2018; Chen et al., 2020; Sohn et al., 2020). For graphs, global graph-level and local node-level data are augmented and contrasted in different ways. DGI (Velickovic et al., 2019) and GMI (Peng et al., 2020) contrast graph and node representations within one augmented view of the original graph. More recent methods contrast global and local representations in two augmented views. GraphCL generates graph augmentations by subgraph sampling, node dropping, and edge perturbation and contrasts the augmented graph representations. GCC samples and contrasts subgraphs of the original graph (Qiu et al., 2020). MVGRL leverages node diffusion to augment the graph and contrasts the node representations (Hassani & Khasahmadi, 2020). More recently, contrasting the local node representations has been shown to

achieve state-of-the-art. GRACE contrasts the node representations in two graph views augmented with feature masking and edge removal (Zhu et al., 2020c). GCA extends this by dropping the less important edges and features, based on node centrality and feature importance metrics (Zhu et al., 2021c). A thorough empirical study on the combinatorial effect of different augmentations has been conducted by Zhu et al. (2021b). Due to the complexity of collecting negative samples in graph data, negative-samples-free contrastive objectives have been also studied. Among existing methods, BGRL that uses the Bootstrapping Latent loss (Thakoor et al., 2021), and GBT uses Barlow Twins loss (Bielak et al., 2021) are the most successful. Existing graph CL methods explicitly augment the input graph and contrast the augmented graph representations obtained with low-pass GNN-based encoders. Hence, they perform poorly on graphs with heterophily. In contrast, we leverage low-pass and high-pass graph filters to directly obtain different graph views, without relying on explicit graph augmentation. This allows achieving state-of-the-art under heterophily.

3 Contrastive Learning via Graph Filters

Here, we introduce our proposed contrastive learning pipeline, HLCL. The key idea of our method is to generate a non-smooth augmented representation and contrast it with the smooth representation generated using the same encoder. A typical GNN filters smooth graph frequencies by aggregating the node representations with those of their neighbors. Hence, it results in similar representations for the nodes in a neighborhood. In contrast, the high-pass filter only preserves the high-pass frequencies. In doing so, it magnifies the dissimilarities between the nodes and diversifies the representations of nodes in a neighborhood. Contrasting the two filtered representations results in learning rich representations, particularly under heterophily.

3.1 Preliminaries

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

High-pass and Low-pass graph filters. Multiplication of Laplacian with a graph signal $\hat{\boldsymbol{L}}_{sym}\boldsymbol{x} = \sum_i \lambda_i \boldsymbol{u}_i \boldsymbol{u}_i^T \boldsymbol{x}$, acts as a filtering operation over $\boldsymbol{x}$, adjusting the scale of the components of $\boldsymbol{x}$ in the frequency domain. The entries of every eigenvector, $\boldsymbol{u}_i$ aligns with a cluster of connected nodes in the graph. For the Laplacian matrix, a smaller eigenvalue λ_i corresponds to a smoother eigenvector $\boldsymbol{u}_i$, and corresponds to a larger cluster of connected nodes. On the other hand, a larger λ_i corresponds to non-smooth eigenvectors $\boldsymbol{u}_i$, which identify smaller clusters of closely connected nodes in the graph. A Laplacian filter magnifies the part of the signal that aligns well with basis functions corresponding to large eigenvalues $\lambda_i \in (1, 2)$ and suppresses the part the signal that aligns with basis functions corresponding to small eigenvalues $\lambda_i \in [0, 1]$. That means, for the small clusters of nodes that have a large alignment with $\boldsymbol{u}_i$ corresponding to $\lambda_i > 1$, the projection $\lambda_i \boldsymbol{u}_i \boldsymbol{u}_i^T \boldsymbol{x}$amplifies $\boldsymbol{x}$ within the cluster and consequently magnifies the difference in $\boldsymbol{x}$ among the nodes within that cluster. On the other hand, for the larger clusters that align well with $\boldsymbol{u}_i$ corresponding

to $\lambda_i < 1$, the projection $\lambda_i \boldsymbol{u}_i \boldsymbol{u}_i^T \boldsymbol{x}$ suppresses $\boldsymbol{x}$ within the cluster and reduces the differences in $\boldsymbol{x}$ 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. On the other hand, affinity matrices, such as the normalized adjacency matrix, can be treated as low-pass filters (Nt & 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 $\hat{\boldsymbol{L}}_{sym}$ and $\hat{\boldsymbol{A}}_{sym}$ filters with $\boldsymbol{x}$ can be understood as diversification and aggregation operations, respectively (Luan et al., 2020):

$$(\hat{\boldsymbol{L}}_{sym}\boldsymbol{x})_i = \sum_{j \in \mathcal{N}_i} \frac{1}{\boldsymbol{D}_{ii}}(\boldsymbol{x}_i - \boldsymbol{x}_j), \quad (\hat{\boldsymbol{A}}_{sym}\boldsymbol{x})_i = \sum_{j \in \mathcal{N}_i} \frac{1}{\boldsymbol{D}_{ii}}\boldsymbol{x}_j. \tag{1}$$

High-pass filters highlight the differences between the nodes and their neighbors, making them distinguishable. On the other hand, low-pass filters aggregate the features of the nodes in a neighborhood and make them similar. Since $\hat{\boldsymbol{L}}_{sym} + \hat{\boldsymbol{A}}_{sym} = \boldsymbol{I}$, both filters capture complementary information and their combination allow learning richer representations.

Homophily Ratio. To quantify how likely nodes with similar labels are connected in the graph, homophily ratio, β, is proposed (Pei et al., 2020), as follows:

$$\beta = \frac{1}{|V|} \sum_{v \in V} \frac{\text{Number of v's neighbors who have the same label as v}}{\text{Number of v's neighbors}}$$

For larger values of β, it is more likely that nodes with the same labels are connected together. As the nodes belonging to different classes often have different features, graphs with higher β are generally smoother.

3.2 Generating Graph Views via Graph Filters

As discussed, the key idea of our HLCL framework is to leverages a high-pass filter to generate a diverse set of node representations, and contrast them with smooth and similar node representations generated by a low-pass filter. To do so, we leverage the normalized Laplacian matrix $\hat{\boldsymbol{L}}_{sym}$ and adjacency matrix $\hat{\boldsymbol{A}}_{sym}$ as the diversification and aggregation operations in Eq. (1). We note that other types of high-pass and low-pass filters can be used in a similar way in our framework.

More specifically, we input the graph $\boldsymbol{X}$ into a graph encoder, and apply a high-pass filter $\boldsymbol{F}_{HP} = \hat{\boldsymbol{L}}_{sym}$ and a low-pass filter $\boldsymbol{F}_{LP} = \hat{\boldsymbol{A}}_{sym}$ to the encoder to generate high-pass node representations $\boldsymbol{H}_H$ and low-pass node representations $\boldsymbol{H}_L$ as follows:

$$\boldsymbol{H}_H^l = \sigma(\boldsymbol{F}_{HP}\boldsymbol{H}_H^{l-1}\boldsymbol{W}^{l-1}), \quad \boldsymbol{H}_L^l = \sigma(\boldsymbol{F}_{LP}\boldsymbol{H}_L^{l-1}\boldsymbol{W}^{l-1}), \quad \boldsymbol{H}_L^0 = \boldsymbol{H}_H^0 = \boldsymbol{X}. \tag{2}$$

where $\boldsymbol{H}_L^l$, and $\boldsymbol{H}_H^l$ are the low-pass filtered and high-pass filtered representations at layer l of the encoder, $\boldsymbol{W}^l \in \mathbb{R}^{d_l \times d_{l-1}}$ is the weight matrix in layer l of the encoder, and σ is the activation function.

The high-pass filter $\boldsymbol{F}_H$ filters out the low-frequency signals and preserves the high-frequency signals. In doing so, it captures the difference between features of every node with the features of the nodes in its neighborhood. 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. Hence, it makes the representations of nodes that have different features from their neighbors distinct in their multi-hop neighborhood.

On the other hand, the low-pass filter, $\boldsymbol{F}_L$, resembles a typical GNN, which only preserves the low-frequency signals by aggregating every node's features with the features of nodes in its immediate neighborhood.

Algorithm 1 Contrastive Learning with Graph Filters (HLCL)

1: **for** epoch=1,2,3,... **do**
2: Input graph $\mathcal{G}$ into the shared graph Encoder $f(\cdot)$
3: Generate high-frequency-signal node embeddings: $\mathbf{H}_H^l = \sigma(\boldsymbol{F}_{HP}\boldsymbol{H}_H^{l-1}\boldsymbol{W}^{l-1})$
4: Generate low-frequency-signal node embeddings: $\boldsymbol{H}_L^l = \sigma(\boldsymbol{F}_{LP}\boldsymbol{H}_L^{l-1}\boldsymbol{W}^{l-1})$
5: Compute the contrastive objective $\mathcal{L}$ with Eq. (5)
6: Update parameters by applying stochastic gradient ascent to maximize $\mathcal{L}$
7: **end for**

Using the low-pass filter within 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.

While the low-pass filter is essential for learning good representations in smooth graphs, it cannot produce distinguished representations for graphs that are non-smooth, specially under heterophily. For such graphs, the high-pass filter is crucial to provide distinct representations. The combinations of both filters provides complementary information and allows learning both smooth and non-smooth components of the graphs simultaneously. Hence, it enables learning richer representations, particularly under heterophily.

The high-pass and low-pass filtered representations can be obtained through message passing, according to Eq. (1). 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. Formally:

$$\hat{\boldsymbol{h}}_i^l = \sigma(\boldsymbol{W}^{l-1}\boldsymbol{h}_i^{l-1}), \quad (\boldsymbol{h}_i^l)_L = \Sigma_{j\in\{\mathcal{N}_i\cup\{i\}\}}(\hat{\boldsymbol{h}}_i^l + \hat{\boldsymbol{h}}_j^l), \quad (\boldsymbol{h}_i^l)_H = \Sigma_{j\in\{\mathcal{N}_i\cup\{i\}\}}(\hat{\boldsymbol{h}}_i^l - \hat{\boldsymbol{h}}_j^l). \tag{3}$$

Explicit graph augmentation. We note that the graph $\boldsymbol{X}$ can be optionally explicitly augmented before entering the encoder. While explicit augmentation do not provide a considerable benefit and can even be harmful under heterophily, it is crucial for graphs with homophily. We will study in details the effect of explicit augmentation under homophily and heterophily, in our experiments.

Our HLCL framework is demonstrated in Fig. 1. In Appendix, we study the effect of filters on a toy example.

3.3 The Contrastive Learning Framework

After generating the low-pass and high-pass filtered graph views, we pass them through a shared non-linear projection head and employ a contrastive objective that enforces the filtered representations of each node in the two views to agree with each other and can be discriminated from representations of other nodes.

Non-linear projection head. Before contrasting the high-pass and low-pass filtered representations, we pass them to a non-linear projection head $g(\cdot)$, which maps the filtered views to another latent space where the contrastive loss is calculated. This has been shown to improve the performance (Chen et al., 2020; You et al., 2020; Zhu et al., 2020c;a), and we also confirm its effect in our experiment. In our setting, we use a two-layer perceptron network to obtain $\boldsymbol{z}_h = g(\boldsymbol{H}_H)$ and $\boldsymbol{z}_l = g(\boldsymbol{H}_L)$. We use the projection head only during the contrastive training, and use the low-pass filtered representations $\boldsymbol{H}_L$ in the downstream task.

Contrastive loss. For every node i, its projected representation generated by the low-pass filter, $\boldsymbol{z}_l^i$, is treated as the anchor, and the one generated by the high-pass filter, $\boldsymbol{z}_l^i$, is treated as the corresponding positive sample. The other projected representations in the two views are treated as the negative samples.

For the contrastive loss function, we use InfoNCE proposed in (Van den Oord et al., 2018). Formally:

$$l(\boldsymbol{z}_l^i, \boldsymbol{z}_h^i) = \log \frac{e^{\text{sim}(\boldsymbol{z}_l^i, \boldsymbol{z}_h^i)/\tau}}{e^{\text{sim}(\boldsymbol{z}_l^i, \boldsymbol{z}_h^i)/\tau} + \sum_{\substack{k \in [N], \\ k \neq i}} e^{\text{sim}(\boldsymbol{z}_l^i, \boldsymbol{z}_h^k)/\tau} + \sum_{\substack{k \in [N], \\ k \neq i}} e^{\text{sim}(\boldsymbol{z}_l^i, \boldsymbol{z}_l^k)/\tau}}, \tag{4}$$

in which $\text{sim}(\boldsymbol{z}_l^i, \boldsymbol{z}_h^i)$ is the cosine similarity between $\boldsymbol{z}_l^i$ and $\boldsymbol{z}_h^i$, and τ is a temperature parameter. The second term in the denominator, $\sum_{k \neq i} e^{\text{sim}(\boldsymbol{z}_l^i, \boldsymbol{z}_h^k)}$, represents the inter-view negative pairs, i.e., between the low-pass view $\boldsymbol{z}_l^i$ of node i and the high-pass views of all the other nodes, and $\sum_{k \neq i} e^{\text{sim}(\boldsymbol{z}_h^i, \boldsymbol{z}_h^k)}$ represents the intra-view negative pairs within the low-pass view. Since two views are symmetric, the loss for another view, $l(\boldsymbol{z}_h^i, \boldsymbol{z}_l^i)$, is defined in a similar fashion. The overall objective to be maximized is then defined as the average over all positive pairs. Formally, we maximize:

$$\mathcal{L} = \frac{1}{2N} \sum_{i=1}^{N} [l(\boldsymbol{z}_l^i, \boldsymbol{z}_h^i) + l(\boldsymbol{z}_h^i, \boldsymbol{z}_h^l)]. \tag{5}$$

Effectively, maximizing the agreement between the low-pass and high-pass views pulls away the representation of nodes with different features from their neighborhood, and allows them to be distinguished from their neighbors. The pseudocode is illustrated in Alg. 1.

4 Experiments

In this section, we first study and show the ineffectiveness of existing graph augmentation methods under heterophily. Then, we evaluate our proposed framework for node representation learning under linear probe.

Datasets. We conduct experiments on five widely-used public benchmark datasets with different levels of homophily ratios, β. For graphs with homophily, we use the citation networks including Cora and Citeseer (Yang et al., 2016). For graphs with heterophily, we use the Wikipideia network and the web page networks including Chameleon, Squirrel, and Texas (Rozemberczki et al., 2021; Pei et al., 2020). Table 4 shows β, and the number of nodes, edges, and classes for the above datasets. We also report our results on a large-scale social network, namely Penn94 (Lim et al., 2021). Details can be found in Tables 4 and 5 of the Appendix.

HLCL setup. We employ a two-layer GCN (Kipf & Welling, 2016) as our low-pass encoder, and consider an additional high-pass channel with $\hat{\boldsymbol{L}}_{sym}$ as the message passing filter to capture the high-frequency signals from the graphs. The high-pass channel generates augmented node representations to be contrasted with those encoded by the GCN. The high-pass channel shares the same weight parameters with the GCN, but generate embedding with different filters. The final embedding is generated by the GCN encoder.

4.1 Graph Augmentation under Heterophily

First, we study existing data augmentation methods and their effectiveness for graph CL under heterophily. We use InfoNCE loss defined in Eq. (5) to contrast node representations in augmented graphs, using a standard 2-layer GCN as the graph encoder. We consider topology augmentation methods including Edge Removing (ER), Node Dropping (ND), Edge Adding (EA), Subgraph with Random Walk (RWS), and Personalized PageRank (PPR), as well as feature augmentation method including Feature Masking (FM) and Feature Dropping (FD) for generating different graph augmentations. We compare these methods with HLCL. Except for the proposed high-pass augmentation methods, we do not use any other explicit graph augmentation. We train the model 10 times with early-stopping, and report the average accuracy as the final result. For each run, we randomly select 10% nodes for training, 10% nodes for validation, 80% nodes for testing.

Table 1 shows that our method achieves a remarkable boost of up to 10% compared to the best graph augmentation method. This confirms the necessity of utilizing the non-smooth graph components under heterophily. Note that even though topology and feature augmentation methods benefit the homophily graphs,

Table 1: Existing graph augmentation methods are highly ineffective for graph CL under heterophily. In contrast, our HLCL method achieves up to 10% improvement over the best graph augmentation method.

		Chameleon	Squirrel	Texas
Ours	**HLCL (no-aug)**	**48.03** ±3.1	**33.44** ±1.7	**64.21** ±11.2
Topo	ER	34.23 ±3.8	25.14 ±2.3	52.63 ±10.7
	ND	36.02 ±1.9	26.70 ±1.8	50.53 ±12.2
	EA	38.20 ±3.1	26.92 ±1.6	54.73 ±9.7
	RWS	35.63 ±3.3	24.59 ±13.5	56.84 ±11.7
	PPR	34.14 ±3.5	24.43 ±2.9	50.00 ±6.7
Feat	FM	37.55 ±3.6	25.58 ±1.1	58.42 ±14.21
	FD	36.81 ±3.9	25.75 ±2.7	50.52 ±6.7

they are highly ineffective for graph CL under heterophily. This can be explained from the perspective of Homophily ratio (β) and graph smoothness, which is represented here as neighborhood feature variance (α).As shown in figure 2, augmentation methods that remove or add edges (ER, EA) decrease β of the graphs under homophily. On Cora with homophily, β drops rapidly given its initial high value, so it can generate a more diverse graph structures for aggregations among nodes that are more beneficial for the contrastive framework. However, for Chameleon with heterophily, β does not considerably change, diminishing the benefits of explicit augmentation. Similarly, for Feature augmentation methods that zero out feature entries, α decreases for both heterophily and homophily graphs, smoothing out the graphs. For homophily graphs that have smooth node neighborhoods, the change is not as rapid, and the masked features serve as dropout on the input layer, which improves the performance accordingly Zhu et al. (2021b). On the other hand, feature masking decreases α greatly on Chameleon with heterophily.It smoothes out its non-smooth graph features in a deteriorated manner, hence outweights its benefits as a dropout layer.

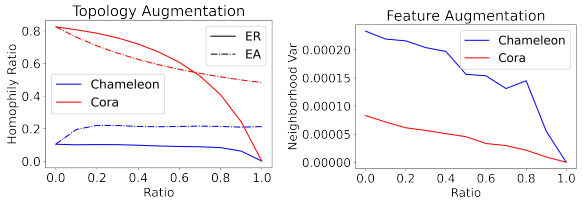

Figure 2: Left: Homophily ratio drops with the increasing edge removing ratio and edge adding ratio with Cora under homophily and stays approximately the same with Chameleon under heterophily. Right: neighborhood feature variance drops with the increasing feature masking ratio, with both Cora under homophily and Chameleon under heterophily, with the decrease more rapid on Chameleon.

4.2 Self-supervised Representation Learning

Next, we evaluate HLCL for self-supervised learning under linear probe 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. We train the model 10 times with early-stopping, and report the average accuracy as the final result. For each run, we randomly select 10% nodes for training, 10% nodes for validation, and 80% nodes for testing. We consider traditional representation learning methods like DeepWalk (Perozzi et al., 2014), deep learning methods including DGI (Velickovic et al., 2019), BGRL (Thakoor et al., 2021), and GRACE (Zhu et al., 2020c), and other popular representation learning baselines like Raw Features and DeepWalk + Raw Features as baselines. We also include MixHop (Abu-El-Haija et al., 2019) in our experiments as the supervised learning baseline. Table 2 compares the performance of HLCL when used on its own, HLCL (no-aug), and when it is applied to the graphs that are augmented with Edge Removing (ER) on the graphs with homophily and Feature Masking (FM) on the graphs with heterophily. We see that HLCL shows a significant boost on graphs with heterophily and a comparable performance on the graphs with homophily. Even without explicit augmentation, HLCL surpasses other baselines on graphs with heterophily, which confirms the effectiveness

Table 2: HLCL achieves state-of-the-art under heterophily, and a comparable performance under homophily when combined with explicit graph augmentation.

	Homophily		Heterophily		
	Cora	CiteSeer	Chameleon	Squirrel	Texas
HLCL	**82.34** ±2.7	**71.35** ±1.4	**49.78** ±1.8	**34.11** ±1.9	**65.26** ±7.8
HLCL(no-aug)	77.09 ±1.7	64.34 ±1.5	**48.04** ±3.1	**33.44** ±1.7	**64.21** ±11.2
Raw feature	64.8	64.6	31.61	24.43	52.63
DeepWalk	75.7 ±1.7	50.5 ±1.6	27.51 ±3.1	25.47 ±2.4	40.00 ±15.9
DeepWalk + features	73.1 ±2.3	47.6 ±1.5	36.21 ±4.6	26.23 ±1.8	53.68 ±10.2
DGI	82.60 ±0.4	68.80 ±0.7	40.48 ±1.9	27.95 ±0.9	60.00 ±10.8
BGRL	74.67 ±0.6	64.25 ±2.4	38.20 ±1.4	26.03 ±0.9	61.05 ±11.1
GRACE	**83.30** ±0.4	**72.10** ±0.5	37.37 ±0.6	28.67 ±0.9	62.6 ±7.2
MixHop	**85.34** ±0.4	**73.23** ±0.5	**46.84** ±3.5	36.42 ±3.4	62.15 ±2.5

of the high-pass channel. Importantly, applying HLCL to graphs that are explicitly augmented achieves a comparable performance under homophily. Thus, the high-pass filter does not harm the performance on such graphs. Comparing to supervised learning model such as MixHop trained in an end-to-end manner, HLCL achieves a comparable or superior performances. This, demonstrates the effectiveness of our design.

4.3 Ablation Studies: HLCL with Explicit Augmentation

Table 3: Combining HLCL with different graph view augmentations. LP represents augmentation on low-pass channel only, HL represents augmentation on both channels. Both topology and feature augmentations are included. The full table 6 presented in the Appendix. Worst results are highlighted in gray.

		Homophily		Heterophily		
		Cora	CiteSeer	Chameleon	Squirrel	Texas
	HLCL (no-aug)	77.09 ±1.7	64.34 ±1.5	48.03 ±3.1	33.44 ±1.7	64.21 ±11.2
LP Feat	FM	78.46 ±2.4	66.68 ±1.7	**49.78** ±1.8	**34.11** ±1.9	**65.26** ±7.8
	FD	78.42 ±2.7	67.75 ±1.7	47.46 ±3.2	33.65 ±1.7	63.15 ±7.4
HL Topo	ER	**82.34** ±2.7	**71.35** ±1.4	37.29 ±2.6	25.60 ±1.6	57.37 ±8.3
	ND	76.10 ±2.6	65.90 ±2.6	35.81 ±1.8	25.39 ±1.5	61.58 ±8.8
	EA	81.02 ±2.5	70.02 ±3.6	37.46 ±1.8	26.12 ±1.7	60.53 ±7.9
HL Feat	FM	78.60 ±2.3	66.08 ±2.0	49.52 ±3.6	33.90 ±1.5	63.68 ±8.9
	FD	77.50 ±3.0	66.59 ±1.7	48.42 ±4.0	33.49 ±1.8	63.68 ±6.8

To investigate the interactions between traditional graph augmentations and HLCL on graphs with different β, we perform ablation studies on explicit graph augmentation methods applied to both high-pass and the typical low-pass GCN channels. We include Edge Removing (ER), Node Dropping (ND), Edge Adding (EA), Feature Masking (FM), and Feature Dropout (FD) as graph augmentation methods. Table 3 shows that combined with HLCL, feature and topology augmentations considerably benefit the performance on graphs with homophily. Topology augmentation, however, may not be helpful under heterophily.

5 Conclusion

We proposed HLCL, a contrastive learning framework that leverages a high-pass graph filter to generate non-smooth augmented representations to be contrasted with their smooth counterparts, generated using the same encoder. This is particularly beneficial for graphs with heterophily, where existing graph CL methods are highly ineffective. Through extensive experiments, we demonstrated that our proposed framework achieves up to 10% boost in the performance for representation learning under the linear probe, for various graphs under heterophily. At the same time, it provides a comparable performance on graphs with homophily.

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*, pp. 21–29. PMLR, 2019.

Philip Bachman, R Devon Hjelm, and William Buchwalter. Learning representations by maximizing mutual information across views. *Advances in neural information processing systems*, 32, 2019.

Muhammet Balcilar, Guillaume Renton, Pierre Héroux, Benoit Gaüzère, Sébastien Adam, and Paul Honeine. Analyzing the expressive power of graph neural networks in a spectral perspective. In *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, pp. 3950–3957, 2021.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pp. 1597–1607. PMLR, 2020.

Fan RK Chung. *Spectral graph theory*, volume 92. American Mathematical Soc., 1997.

Michaël 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, Zhitao Ying, 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*, pp. 4116–4126. PMLR, 2020.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

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.

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.

Hoang Nt and Takanori Maehara. Revisiting graph neural networks: All we have is low-pass filters. *arXiv preprint arXiv:1905.09550*, 2019.

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*, pp. 259–270, 2020.

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 701–710, 2014.

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*, pp. 1150–1160, 2020.

Benedek Rozemberczki, Carl Allen, and Rik Sarkar. Multi-scale attributed node embedding. *Journal of Complex Networks*, 9(2):cnab014, 2021.

Kihyuk Sohn, David Berthelot, Nicholas Carlini, Zizhao Zhang, Han Zhang, Colin A Raffel, Ekin Dogus Cubuk, Alexey Kurakin, and Chun-Liang Li. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. *Advances in Neural Information Processing Systems*, 33:596–608, 2020.

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.

Aaron Van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv e-prints*, pp. arXiv–1807, 2018.

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.

Yewen Wang, Ziniu Hu, Yusong Ye, and Yizhou Sun. Demystifying graph neural network via graph filter assessment. 2019.

Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3733–3742, 2018.

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

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*, pp. 40–48. PMLR, 2016.

Mang Ye, Xu Zhang, Pong C Yuen, and Shih-Fu Chang. Unsupervised embedding learning via invariant and spreading instance feature. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6210–6219, 2019.

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.

Junbo Zhang and Kaisheng Ma. Rethinking the augmentation module in contrastive learning: Learning hierarchical augmentation invariance with expanded views. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16650–16659, 2022.

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*, pp. 11168–11176, 2020a.

Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman 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*, pp. 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*, pp. 2069–2080, 2021c.

A APPENDIX

A.1 BENCHMARK DATASETS

We conduct our experiments on five widely-used public benchmark datasets with different level of homophily ratios, β. We shows the specific statistics of the datasets including β, the number of nodes, edges, and classes, in table 4 In addition, to illustrate the scalability of our model, we run our models on realistic dataset like Penn94, proposed by Lim et al. (2021). We train the model 3 times with early-stopping, and report the average accuracy as the final result. For each run, we randomly select 10% nodes for training, 10% nodes for validation, and 80% nodes for testing. We present our result in table 5. As shown, our method works better in large-scale dataset than other self-supervised learning methods as well.

Table 4: Statistics for our datasets

Datasets	Cora	CiteSeer	Chameleon	Squirrel	Texas	Penn94
Hom. ratio (β)	0.83	0.71	0.25	0.22	0.06	0.47
Nodes	2708	3327	2277	5201	183	41,554
Edges	5278	4676	31421	198493	295	1,362,229
Classes	6	7	5	5	5	2

Table 5: HLCL achieves a noticeably higher accuracy on realistic large-scale dataset comparing to other GCL methods. HLCL outperforms SOTA by 8%.

	Penn94
HLCL	**70.97** ± 0.5
DGI	62.85 ± 0.3
BGRL	56.57 ± 0.8
GRACE	62.53 ± 0.1

A.2 ABLATION STUDIES: HLCL WITH EXPLICIT AUGMENTATION

To investigate the interactions between traditional graph augmentations and HLCL on graphs with different β, we perform ablation studies on explicit graph augmentation methods applied to both high-pass and the typical low-pass GCN channels. We include Edge Removing (ER), Node Dropping (ND), Edge Adding (EA), Feature Masking (FM), and Feature Dropout (FD) as graph augmentation methods. We present our result in table 6.

Table 6: The result on combining HLCL with different graph view augmentations. HP represents augmentation on high-pass channel only, LP represents augmentation on low-pass channel only, HL represents augmentation on both channels. Both topology and feature augmentations are included in the experiments.

		Homophily		Heterophily		
		Cora	CiteSeer	Chameleon	Squirrel	Texas
	HLCL (no-aug)	77.09 ±1.7	64.34 ±1.5	48.03 ±3.1	33.44 ±1.7	64.21 ±11.2
HP Topo	ER	80.80 ±3.0	71.20 ±1.3	36.11 ±4.1	26.26 ±2.3	57.36 ±8.9
	ND	78.42 ±2.0	70.24 ±2.0	35.80 ±2.7	26.79 ±1.6	61.05 ±12.0
	EA	74.04 ±2.9	65.75 ±3.2	35.81 ±3.3	26.79 ±2.0	60.53 ±7.9
HP Feat	FM	77.13 ±2.9	64.67 ±1.3	49.08 ±3.8	33.80 ±1.7	62.63 ±9.8
	FD	77.94 ±2.6	66.05 ±1.8	47.07 ±4.0	32.71 ±2.0	61.57 ±6.2
LP Topo	ER	77.94 ±3.1	65.18 ±1.9	37.07 ±3.4	25.36 ±2.0	57.37 ±7.6
	ND	75.66 ±1.6	61.28 ±2.6	36.15 ±2.0	26.53 ±1.6	59.47 ±8.5
	EA	80.73 ±2.2	68.35 ±2.0	39.08 ±2.0	26.75 ±1.5	61.05 ±11.8
LP Feat	FM	78.46 ±2.4	66.68 ±1.7	**49.78** ±1.8	**34.11** ±1.9	**65.26** ±7.8
	FD	78.42 ±2.7	67.75 ±1.7	47.46 ±3.2	33.65 ±1.7	63.15 ±7.4
HL Topo	ER	**82.34** ±2.7	**71.35** ±1.4	37.29 ±2.6	25.60 ±1.6	57.37 ±8.3
	ND	76.10 ±2.6	65.90 ±2.6	35.81 ±1.8	25.39 ±1.5	61.58 ±8.8
	EA	81.02 ±2.5	70.02 ±3.6	37.46 ±1.8	26.12 ±1.7	60.53 ±7.9
HL Feat	FM	78.60 ±2.3	66.08 ±2.0	49.52 ±3.6	33.90 ±1.5	63.68 ±8.9
	FD	77.50 ±3.0	66.59 ±1.7	48.42 ±4.0	33.49 ±1.8	63.68 ±6.8

A.3 Ablation studies: Output Encoder with Different Graph Filters

In our experiments, we use low-pass graph filter in our GNN encoder as the final representation encoder. It is important to note that the encoder that is trained with contrasting with the high-pass view does not suffer from the over-smoothing problem, even without using the high-pass filter during the inference. This is because the parameters of the encoder are learned by contrasting the low-pass representation with high-pass representations. Hence, we do not need to rely on the high-pass filter during the inference. To illustrate the effectiveness of our structural design, we present an ablation study on using different filters or combinations of filters in our final encoder. The result is shown in table 7. We see that on most datasets low-pass encoder achieves the highest accuracy with a large gap, and variation involving the high-pass encoder has worse performances across all datasets except Texas, which has a very low homophily ratio. This demonstrates the ineffectiveness of using high-pass filter directly in the representation encoder.

Table 7: HLCL with different graph filters in the final encoder. HP represents high-pass filter; LP represents low-pass filter; HP Concat LP represents using both high-pass filters and concatenate the output representations; HP Aggre LP represents using both high-pass filters and aggregate the output representations

	Homophily		Heterophily		
	Cora	CiteSeer	Chameleon	Squirrel	Texas
LP	**77.09** ±1.7	**64.34** ±1.5	**48.04** ±3.1	**33.44** ±1.7	64.21 ±11.2
HP	51.94 ±2.9	36.22 ±2.5	35.58 ±3.1	29.83 ±1.6	**65.26** ±10.3
HP Concat LP	66.47 ±3.6	55.96 ±2.6	41.09 ±3.6	30.86 ±1.9	65.21 ±9.4
HP Aggre LP	65.33 ±2.8	53.47 ±3.1	38.77 ±3.4	28.00 ±1.5	63.15 ±11.7

Table 8: Swapping low-pass encoders in GCL methods to high-pass encoders. Significant performance drops are highlighted in gray.

	Homophily		Heterophily		
	Cora	CiteSeer	Chameleon	Squirrel	Texas
HLCL	82.34 ±2.7	71.35 ±1.4	**49.78** ±1.8	**34.11** ±1.9	65.26 ±7.8
DGI:low	82.60 ±0.4	68.80 ±0.7	40.48 ±1.9	27.95 ±0.9	60.00 ±10.8
DGI:high	31.95 ±2.8	30.54 ±1.7	38.38 ±3.5	29.57 ±1.2	62.11 ±10.2
BGRL:low	74.67 ±0.6	64.25 ±2.4	38.20 ±1.4	26.03 ±0.9	61.05 ±11.1
BGRL:high	29.63 ±2.8	24.99 ±3.1	35.41 ±3.4	29.57 ±1.1	65.26 ±8.2
GRACE:low	**83.30** ±0.4	**72.10** ±0.5	37.37 ±0.6	28.67 ±0.9	62.6 ±7.2
GRACE:high	32.46 ±2.0	26.55 ±3.1	39.65 ±2.9	33.05 ±2.1	**72.63** ±9.3

A.4 Ablation studies: Other GCL methods with different encoders

In this section, we illustrate the importance of our contrastive structure. We show that the main accuracy gains on heterophily datasets are from contrasting the low- and high-pass filtered graph views, instead of the introduction of the high-pass filter alone. The high-pass filter alone cannot provide high-quality representations under heterophily. For different graphs, different combinations of low-pass and high-pass filters are required to provide good representations. Having access to labels, the best combination can be learned. However, this is indeed not possible without labels. We have shown that contrasting the low- and high-pass filtered graph views can provide high-quality representations in the unsupervised scenario. Here, We conduct a new ablation study by swapping the encoders from other graph CL methods with a high-pass encoder. The result is presented in 8. By replacing low-pass filter with high-pass filter, grph CL methods are able to achieve an better accuracy on some heterophily datasets like Squirrel and Texas. However, their performances deteriorate significantly under homophily cases, where aggregation among neighborhoods are desired. Our results show that when using high-pass or low-pass alone, the model cannot achieve good performance.

A.5 Illustration Examples on HP and LP filter under different homophily ratios

To explain the effect of high-pass and low-pass filters and motivates using both filters in a contrastive way, we construct a simple graph with seven nodes. Nodes are separated into two classes, labeled as red and green as shown in the figure. First, we generate features vectors with 200 dimensions for nodes from red and green classes based on two Gaussian distributions, centered at -5 and 5 respectively with a standard deviation of 1. Then, we do graph aggregation or diversification with low- and high-pass filters separately, following Eq. (1). For clearer illustration, we do not project the feature vectors into other dimensions.

We considered three different homophily ratios and ran the node features through high-pass and low-pass filters for a few iterations. We see that: (1) for high homophily ratio graph (Fig. 3), the low-pass filter reduces the variance of the features and pulls different classes towards each other, while the high-pass filter increases their variance and eventually pulls the classes away from each other. Hence, contrasting the two filtered representations prevents over-smoothing. The figure illustration is shown at Fig. 4 (2) for lower homophily ratio graph, where a part of the graph has homogeneous labels and the other part has heterogeneous labels (Fig. 5), the low-pass filter mainly reduces the variance of the features for the label-homogeneous part, while the high-pass filter unnecessarily mixes the class features. At the same time, the high-pass filter increases the variance of features for the label-heterogeneous part and pushes them away from the other class, while the low-pass filter unnecessarily mixes the class features. In this case, contrasting the two filters is crucial for learning good-quality representations. The figure illustration is shown at Fig. 6 (3) Finally, for very low

homophily ratio graph (Fig. 7), the low-pass filter reduces the variance of the distributions and mixes the distributions of the features in the two classes, making them indistinguishable. On the other hand, the high-pass filter pushes away the distributions of the features in different classes. By contrasting the high-pass and low-pass filtered features, HLCL is able to distinguish the feature distributions under heterophily. The figure illustration is shown at Fig. 8

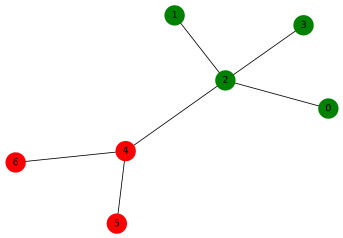

Figure 3: Case 1: High homophily ratio graph

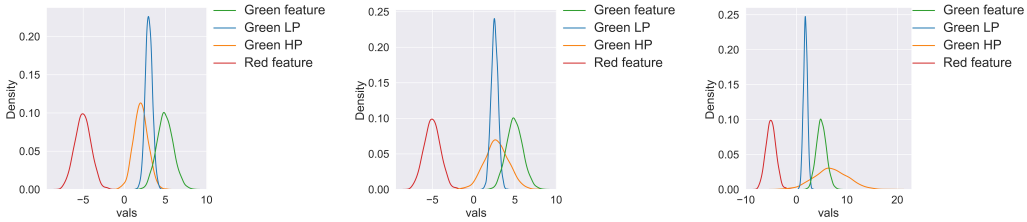

Figure 4: The change of distributions of node 2 (green class) in Fig. 3, at iterations 1, 3, and 6.

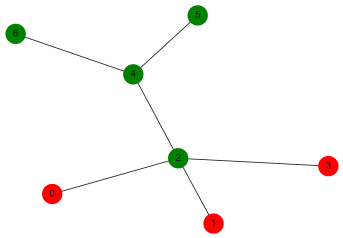

Figure 5: Case 2: A mixture of homophily and heterophily components

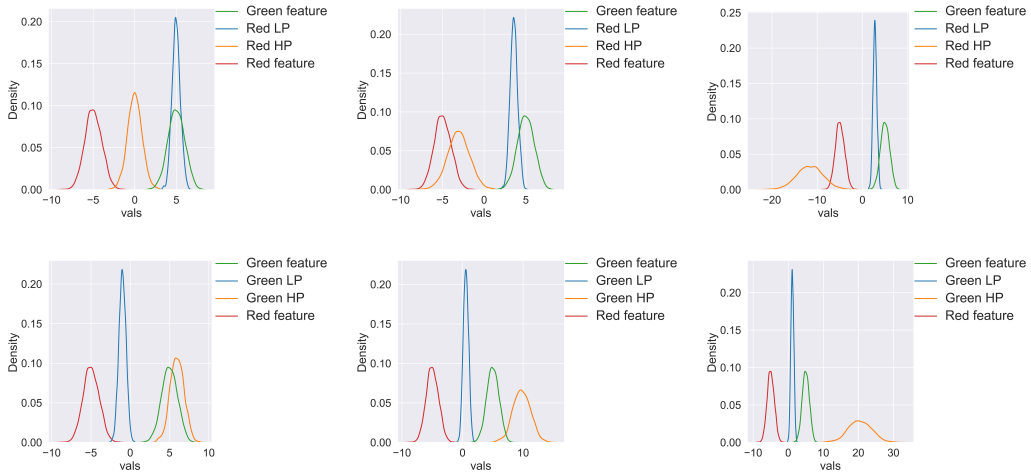

Figure 6: The change of distributions of node 2 (top row) and node 4 (bottom row) in Fig. 5, at iterations 1, 3, and 6. Node 2 is at a heterogeneous neighborhoods, while node 4 is at a homogeneous neighborhoods

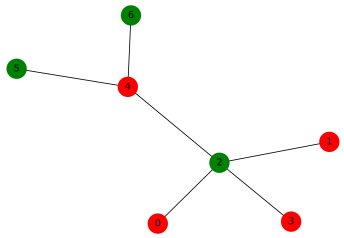

Figure 7: Case 3: Very low homophily ratio

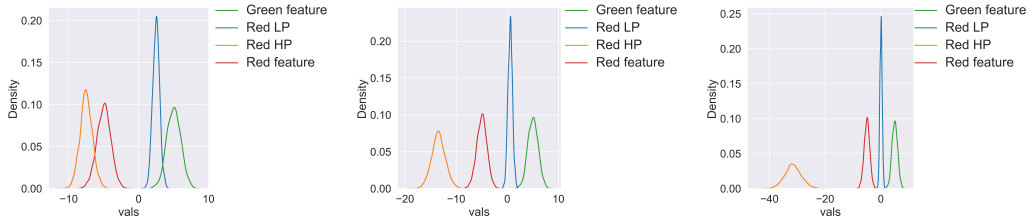

Figure 8: The change of distributions of node 2 (green class) in Fig. 7, at iterations 1, 3, and 6.