Revisiting the Role of Heterophily in Graph Representation Learning: An Edge Classification Perspective

JINCHENG HUANG, PING LI, RUI HUANG, NA CHEN, and ACONG ZHANG, School of Computer Science, Southwest Petroleum University, China

Graph representation learning aims at integrating node contents with graph structure to learn nodes/graph representations. Nevertheless, it is found that many existing graph learning methods do not work well on data with high heterophily level that accounts for a large proportion of edges between different class labels. Recent efforts to this problem focus on improving the message passing mechanism. However, it remains unclear whether heterophily truly does harm to the performance of graph neural networks (GNNs). The key is to unfold the relationship between a node and its immediate neighbors, e.g., are they heterophilous or homophilous? From this perspective, here we study the role of heterophily in graph representation learning before/after the relationships between connected nodes are disclosed. In particular, we propose an end-to-end framework that both learns the type of edges (i.e., heterophilous/homophilous) and leverage edge type information to improve the expressiveness of graph neural networks. We implement this framework in two different ways. Specifically, to avoid messages passing through heterophilous edges, we can optimize the graph structure to be homophilous by dropping heterophilous edges identified by an edge classifier. Alternatively, it is possible to exploit the information about the presence of heterophilous neighbors for feature learning, so a hybrid message passing approach is devised to aggregate homophilous neighbors and diversify heterophilous neighbors based on edge classification. Extensive experiments demonstrate the remarkable performance improvement of GNNs with the proposed framework on multiple datasets across the full spectrum of homophily level.

CCS Concepts: • Mathematics of computing → Graph theory; • Computing methodologies → Artificial intelligence;

Additional Key Words and Phrases: Graph neural networks, heterophily, edge type, hybrid message passing

ACM Reference format:
Jincheng Huang, Ping Li, Rui Huang, Na Chen, and Acong Zhang. 2023. Revisiting the Role of Heterophily in Graph Representation Learning: An Edge Classification Perspective. ACM Trans. Knowl. Discov. Data. 18, 1, Article 13 (September 2023), 17 pages.
https://doi.org/10.1145/3603378
1 INTRODUCTION

The surprising expressiveness of graph neural networks (GNNs) has aroused an explosion of interests in graph representation learning [7], resulting in extensive applications ranging from social network analysis [19] to molecular biology [4, 11], even to regular data processing like text mining [18] and image processing [29]. Despite their practical success, it is not yet guaranteed that GNNs can be effective for arbitrary graph data.

One notable feature in graphs is heterophily [33]: connected nodes may have different class labels or properties. In real-world systems, many graphs exhibit strong heterophily. For instance, the matchmaking website tends to connect people to those with the opposite gender, the hierarchical organization structure of many natural and man-made systems shows the leader-member relations. When applying GNNs on such graph data, the implicit assumption underlying GNNs that node features should be similar (smooth) among neighboring nodes is violated, which makes it very possible to learn undiscriminating features for different classes.

To tackle this issue, a few recent work has made efforts to capture node feature encoded in the topology from different respects. For example, by looking into the distribution of class labels in a graph, Zhu et al. [33] find that the 2-hop neighborhood is less heterophilious than the nearest neighborhood, thus it is expected to improve the power of graph learning by leveraging the second-order neighbors. Geom-GCN [17] maps a graph to a continuous space to capture long-range dependencies between nodes such that the information on distant nodes with the same label is able to be exploited in node representation learning. On the other hand, it is possible to aggregate messages from different neighbors (i.e., homophilious neighbors and heterophilious neighbors) in different ways based on the semantic similarity between neighboring nodes [6, 15]. Though the prior work has carefully crafted the way of information integration, to optimize representation learning strategy needs to exhaustively examine the characteristics of class label distribution in a graph, which remains a great challenge.

Clearly, the block is the presence of heterophilious edges. As shown in Figure 1, even by simply removing all heterophilious edges in graph data, the performance of a state-of-the-art model SGC [26] can be improved by about 30 percentage points, compared to the original data. This result suggests that under the classical message passing scheme, heterophilious edges are indeed detrimental to representation learning. Therefore, an intuitive solution is to use only homophilious edges for message passing, which however requires that edge type is available. Furthermore, once edge type is available, considering that the presence of heterophilious edges can be seen as a feature and play a role in bridging long distance nodes, it will be beneficial to exploit heterophilious edges in feature representation. In effect, previous study [6] shows that even a rough estimate on edge type by attention mechanism facilitates the mixing of incompatible features. However, different from existing feature-based attention models [6], the use of edge type makes it more direct and efficient to learn appropriate feature aggregation on the neighborhood. Thus, it is meaningful to identify edge types in graphs.

In this work, we propose an end-to-end “spotting-then-aggregating” scheme for graph representation learning, which consists of two components: edge type identification (i.e., “spotting”) and node feature representation learning (i.e., “aggregating”). Specifically, we categorize the edges into two types, namely, homophilious edge and heterophilious edge, where homophilious edges refer to the edges connecting nodes in the same class, and heterophilious edges represent the edges connecting nodes between different classes. Based on the edge labeling on the training set, a binary classification model is learned to discriminate between heterophilious and homophilious edges, with which the heterophilious edges are spotted from the unlabeled set. On top of that, representation learning can be improved. Here we provide two strategies for improving graph representation learning, namely, removing detected heterophilious edges from the graph,
or aggregating messages using two channels (corresponding to homophilious edges and heterophilous edges, respectively).

The whole process is implemented in an end-to-end manner, which allows the edge type identification to be dynamically adjusted according to the prediction of GNNs. Despite its adaptability, end-to-end learning will suffer from discrete meta-outputs. In particular, the output of edge classifier is binary values, which means that continuous gradients cannot be propagated back to the edge classifier. To resolve this problem, we devise a back-max method based on Gumbel-softmax \cite{10} to tune the parameters of the edge classifier by the task-specific loss.

The contribution of this article is summarized as follows:

– We present a perspective from edge type identification to study the role of heterophily in graph representation learning, which makes it possible to leverage the presence of heterophily to boost GNNs.
– We propose the general “spotting-then-aggregating” framework to learn the heterophily of the edges and explore the use of heterophilous edges. In particular, we devise two simple yet effective methods to implement the framework.
– Extensive experiments on the benchmark graph data with low homophily validate the superiority of the proposed method over the state-of-the-art models.

2 PRELIMINARIES

Let $G = (V, E)$ be an undirected and unweighted graph with node set $V$ and edge set $E$. The nodes are described by the feature matrix $X \in \mathbb{R}^{n \times f}$, where $f$ denotes the number of features per node and $n$ is the number of nodes. Each node is associated with a class label, which is depicted in the label matrix $Y \in \mathbb{R}^{n \times c}$ with a total of $c$ classes. We represent the graph by its adjacency matrix $A \in \mathbb{R}^{n \times n}$ and the normalized adjacency matrix $\tilde{A} = D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$, where $D$ is a diagonal degree matrix with $D_{i,i} = \sum_j A_{i,j}$.

**Graph Convolutional Network and Its Simplification.** Message passing graph convolutional networks have been proved to be powerful on a number of graph data, among which **graph convolutional network** (GCN) proposed by Kipf et al. \cite{13} is a widely used one. A typical GCN
makes prediction according to a series of operations on node features:

\[ \hat{Y} = \text{softmax}(\tilde{A}\text{ReLU}(\tilde{A}XW_0)W_1), \]

(1)

where \( \hat{Y} \in \mathbb{R}^{n \times c} \) are the predicted node labels, \( W_0 \) and \( W_1 \) are for feature mapping at the first and second layer, respectively.

The above Equation (1) reveals three core components in GCN, namely, feature propagation, linear transformation, non-linear activation. However, Wu et al. [26] have shown that the latter two components are redundant and detrimental to performance, so they simplify the vanilla GCN as

\[ \hat{Y} = \text{softmax}(\tilde{A}KXW), \]

(2)

where \( K \) is the number of graph convolutions, and \( W \in \mathbb{R}^{d \times c} \) denotes the learnable parameters of a logistic regression classifier. The simplified GCN is referred to as SGC.

**Graph Fourier Transform and Graph Signals.** In the light of graph signal theory [23], graph convolution is equivalent to the Laplacian transform of graph signals from the time domain to the frequency domain. Let \( L = I_n - D^{-\frac{1}{2}}AD^{-\frac{1}{2}} \) be the normalized graph Laplacian matrix, which is positive semi-definite and has a complete set of orthogonal eigenvectors \( \{u_l\}^n_{l=1} \in \mathbb{R}^n \) corresponding to the eigenvalues \( \lambda_l \in [0, 2] \). Similar to the Laplacian operator to the basis function \( e^{-i\omega t} \) of Fourier transform in the time domain, i.e., \( \Delta e^{-i\omega t} = \frac{\partial^2 e^{-i\omega t}}{\partial t^2} = -\omega^2 e^{-i\omega t} \), the eigenvectors of Laplacian matrix is analogous to the basis functions of Fourier transform. That is, the Fourier transform on the graph can be defined as \( \hat{x} = U^T x \), associated with its inverse transform \( x = U \hat{x} \). Hence, the convolution of graph signal \( x \) with the convolution kernel \( f \) reads as

\[ (f \ast g)_G = U((U^T f) \odot (U^T x)) = U g_\theta U^T x, \]

(3)

where \( \odot \) is Hadamard product, and \( g_\theta \) is a diagonal matrix, which represents the convolutional kernel in the frequency domain, replacing \( U^T f \).

**Homophily and heterophily.** In this work, we use the homophily measure given in [33] to quantify the homophily level of a graph data, as described in Definition 2. Moreover, we define homophilious and heterophilous edge as

**Definition 1.** Homophilious edge: Edge connecting nodes that have the same label. On the contrary, heterophilious edges are the edges that connect nodes with different labels. We say a graph is homophilious if there are no heterophilious edges in the graph.

**Definition 2 ([33]).** The edge homophily ratio is defined as \( h = \frac{|\{(u,v) \in \xi : y_u = y_v\}|}{|\xi|} \), where \( |\xi| \) denotes the total number of edges in the graph.

## 3 THE IMPACT OF HETEROPHILY

To encode topological interaction between nodes into feature representations, GNNs generally integrate information from various neighborhoods via message passing. The consequence is that nodes in close proximity are more likely to be similar in feature representation. In other words, GNNs attempt to smooth out the difference between connected nodes, which is expected to favor the downstream classification tasks. However, in some settings where graphs have low edge homophily ratios, the vanilla GNNs have been found to underperform MLPs [33]. Here, we tap into the impact of heterophilous edges on several popular GNN models by comparing the performance change of GNNs when heterophilous edges are perturbed.

We first conduct an empirical study to test the node classification performance of vanilla GNNs in the ideal setting where the homophilious edges in the graph are retained, while all
Table 1. Summary of Node Classification Results (in Percent)

| Dataset | Cora | Cite. | Film. | Texas | Wisc. | Corn. | Cham. | Squi. |
|---------|------|-------|-------|-------|-------|-------|-------|-------|
| Homo.ratio h | 0.81 | 0.74 | 0.22 | 0.06 | 0.1  | 0.2  | 0.23  | 0.22  |
| GCN     | 87.2 | 76.4 | 30.1 | 59.5 | 59.8 | 57.3 | 60.3  | 36.7  |
| GCN∗    | 95.1 | 84.1 | 51.4 | 82.7 | 87.5 | 84.1 | 86.6  | 81.9  |
| GAT     | 87.6 | 76.3 | 29.0 | 59.1 | 53.1 | 58.4 | 45.1  | 28.3  |
| GAT∗    | 95.4 | 84.0 | 53.6 | 84.9 | 88.4 | 83.8 | 87.0  | 82.5  |
| SGC2    | 86.9 | 76.3 | 26.3 | 59.2 | 54.2 | 63.7 | 61.7  | 42.7  |
| SGC2∗(l = 2) | 95.6 | 84.2 | 59.4 | 83.5 | 87.7 | 83.0 | 87.2  | 81.9  |
| SGC2∗(l = 50) | 96.2 | 84.9 | 61.9 | 82.5 | 87.5 | 84.1 | 87.9  | 81.5  |

The * represents the perfectly homophilious situation, and $l$ represents the number of convolution layer. SGC2 uses a two-layer MLPs with linear map as the predictor.

Fig. 2. Training and validation loss on Cornell under the heterophily edge deletion rates of 0%, 50%, and 100% configurations.

heterophilious edges are deleted. From the Table 1 (see Section 5 for the details of datasets and data split), we have the following observations:

— When the graph becomes perfectly homophilious, the performance of the GNNs is greatly improved, compared to that on original structure. This observation suggests that a large number of heterophilious edges in graphs with weak homophily (i.e., strong heterophily) can be noisy and interfere in feature representation learning, since they allow the “messages” to pass between classes.

— Deeply stacking convolution layers (e.g., SGC with 50 convolution layers) does not result in drops in GNNs’ performance, implying that the over-smoothing effect is only negative for inter-class messaging, but helpful for intra-class messaging.

We further study the impact of heterophilious edges on graph learning via the training processes. Figure 2 shows the variations of training losses of SGC under the configurations where heterophilious edges are randomly removed with deletion rates of 0%, 50%, and 100%, respectively. Note that 100% deletion corresponds to the ideal situation. It is shown from the figure that the lowest loss of the model is achieved when there are no heterophilious edges, while the highest loss corresponds to the original graph structure, and deleting even a portion of heterophilious edges
can help to better fit the training samples. The experimental results show the chances of exploiting heterogeneous edges to make a positive contribution to GNNs.

4 METHODOLOGY

We propose **Spotting-then-Aggregating (SA)**, a novel scheme that aims at resolving the major limitation of message-passing-based GNNs when confronted with graph data that has strong heterophily. Instead of resorting to high-order neighbors [33] and estimating dissimilar neighbors [28] (i.e., those connected with heterophilious edges), SA directly spots potential heterophilious edges, with which graph learning either performs on optimized structure (e.g., removing heterophilious edges from the graph such that message passing along those edges is not allowed), or integrates information from neighbors in a different way. To achieve this, we introduce an edge classifier that is pre-trained on training set at the first place, and then join it with an off-the-shelf or refined GNN model for further end-to-end training. The overall framework is diagrammatically illustrated in Figure 3.

4.1 Model Pretraining

We first pre-train a binary classifier to initialize the graph update procedure.

**Pretraining task.** We introduce a supervised edge classification task that predicts which type a given edge belongs to, where edges are divided into two categories, namely, homophilous and heterophilous edges. The edges among labeled nodes (i.e., the training set) are used for edge classifier pre-training, whose labels are obtained according to Definition 1. In a similar way, the validation edge set and test edge set can also be obtained.

**Pre-training model.** The representation of the edge between node $i$ and node $j$ is determined by the representations of the two endpoints:

$$ e_{ij} = R(Wx_i, Wx_j, γ), $$

(4)
where \( \mathbf{x}_i \) denotes the feature representation of node \( i \), \( W \in \mathbb{R}^{f' \times f} \) is a shared parameter matrix that maps node representations into a lower dimensional subspace, and \( R(\cdot) \) can be any permutation invariance function. Here we consider the squared difference,

\[
R(W\mathbf{x}_i, W\mathbf{x}_j) = (W\mathbf{x}_i - W\mathbf{x}_j)^2,
\]

(5)

Using edge representations as input features, a binary classifier is learned:

\[
\hat{y}_{ij} = \text{softmax}(W_c e_{ij} + b_c),
\]

(6)

where \( \hat{y}_{ij} \in \mathbb{R}^2 \), and \( W_c, b_c \) are learnable parameters. For simplicity, here we employ a single layer feedforward network. However, we note that any binary classifiers that are readily available can be used for edge classification. We use the cross-entropy loss to train the model. The prediction score can be further binarized to indicate the presence or absence of an edge in the changed graph, where the presence of an edge implies that it is identified to be homophilous edge.

Remark. The pre-training of the edge classifier generally requires the availability of a certain number of labeled edges. However, we note that if the training edges are quite limited or even if no training samples are available, for example, the labeled nodes do not share edges, the model will reckon without pre-training and just randomly initialize the parameters of the edge classifier, which is equivalent to classifying edges merely based on their feature similarity. We will show that the model still performs well without pre-training in the experimental section (i.e., Section 5.2).

After pre-training, our edge classifier has obtained an expected initialization weight. Next, we present two ways to exploit the inferred edge types in the follow-up graph representation learning, both of which attempt to optimize the message passing scheme. The first method is to reduce the messages passing through heterophilous edges because they can introduce inconsistent features (which we call noise) from neighbors with class labels different from central nodes'. Alternatively, we modify the message passing scheme by integrating the signals from the inconsistent neighbors. We demonstrate these two methods as follows.

4.2 Method 1: Structure Optimization (SA-SGC)

This method is inspired by the above investigation on the impacts of heterophily, heterophilous edges identified by the edge classifier are removed to produce a homophilious graph, on top of which GNNs are expected to be improved. So the pre-trained edge classification model is joint with a GNN model, to update the parameters simultaneously. Here, we design an end-to-end training scheme. By Equation (6), we can get the kind of each edge, so we set the adjacency matrix as

\[
A_{o(i,j)} = \phi(\hat{y}_{i,j}) \quad \text{s.t.} \quad A_{i,j} = 1,
\]

(7)

where \( \phi = 1 \) if the result of \( \hat{y}_{i,j} \) is a homophilous edge, and 0 otherwise, and \( A_o \) represent a symmetric normalized adjacency matrix composed of homophilous edges. We adopt SGC to learn graph structure. For SGC, 1-layer perceptrons can behave much like linear mappings, so the embedding expression ability of a layer of MLP is insufficient [27], especially when the original features of the node have a high dimensionality. Therefore, we use the SGC of two-layer MLPs as the base model and record it as SGC2, it can be expressed as the following formula,

\[
\hat{Y} = \text{softmax}(\tilde{A}_o^K X W_0 W_1).
\]

(8)

where \( \tilde{A} \) is a symmetric normalized adjacency matrix composed of classified homophilous edges. In this scheme, GNNs’ downstream task is used to supervise the feature learning and structure update(edge classifier parameter update). Specifically, task-relevant training loss will be back propagated to the edge classifier and GNN at the same time.
However, to propagate the loss to the edge classifier is challenging. The reason is that as the output of edge classifier, the discrete graph structure is not differentiable. We tackle this problem by proposing a BackMax method inspired by the Gumbel-Softmax \cite{10}.

**BackMax.** Our BackMax is detailed as follows.

1. For any arbitrary edge with endpoints \( \{u, v\} \), the probability that the edge belongs to one of two classes is given by Equation (6), i.e.,
   \[
   z_{u,v} = \text{edge\_classifier}(x_u, x_v),
   \]
   where \( z_{u,v} \in \mathbb{R}^2 \).

2. The softmax function is employed to amplify the probability distribution,
   \[
   \pi_{u,v} = \frac{\exp(z_{u,v})}{\sum \exp(z_{u,v})},
   \]
   (10)

3. Then in the forward propagation phase,
   \[
   y_{\text{hard}} = \text{one\_hot}(\text{argmax}(\pi_{u,v})),
   \]
   which is adopted to update the graph structure, while in the backward propagation phase, the error will be fed back to the continuous counterpart of \( y_{\text{hard}} \): \( y = \pi_{u,v} \) for updating the parameters of the edge classifier. The following expression combines two variables together:
   \[
   \hat{y} = \text{detach}(y_{\text{hard}} - \pi_{u,v}) + \pi_{u,v},
   \]
   (12)

where the function \( \text{detach}() \) is to disconnect the backpropagation. In comparison to Gumbel-softmax estimator \cite{10}, BackMax does not involve any sampling for reparametrization and hyper-parameter for annealing, which can reduce the computational complexity.

**Proposition 1.** \( \text{SA}(\cdot) \) unifies MLP(\( \cdot \)) and GNN(\( \cdot \)) in supervised graph learning.

**Proof.** Given a GNN \( \hat{Y} = f(A, X, W) \), in one extreme case where the edges among training nodes are all heterophilious, SA will learn to delete all connections in the graph, i.e., the adjacency matrix \( A = I \) at the convergence. Then SA and MLP are equivalent (i.e., \( \hat{Y} = f(A, X, W) = f(X, W) \)). On the other end of the spectrum, there are only homophilous edges in the training set, which makes the edge classifier bias to homophilous edges, so SA will not remove any links in the graph. In this case, SA equals GNN. When the training set is in between, SA will adaptively drop heterophilous edges that are assumed to be adverse to GNN’s message passing. After structure optimization, for the nodes that are disconnected from the graph, SA behaves like MLP, while for the connected components, GNN will be performed. \( \square \)

### 4.3 Method 2: Aggregation Optimization (SA*-SGC)

Though the heterophilous edges will introduce the inconsistent information to central nodes when applying the standard message passing mechanism, heterophilous edges convey information about the fact that “some neighbors are from different classes”, which also can be a constituent part of node feature. To enhance the feature discrimination between nodes in different classes, we introduce the negative term to the vanilla message passing for heterophilous edges associated with central node:

\[
\mathbf{h}_i^{l+1} = \mathbf{h}_i^l + \sum_{j: \hat{y}_{ij}^{[1]} = 1} \frac{1}{\sqrt{d_i} \sqrt{d_j}} \mathbf{h}_j - \alpha \sum_{j: \hat{y}_{ij}^{[2]} = 1} \frac{1}{\sqrt{d_i} \sqrt{d_j}} \mathbf{h}_j,
\]

(13)

where \( \hat{y}_{ij} \) is the one-hot edge classification vector, whose first dimension indicates the homophilous edge between node \( i \) and \( j \) and second dimension indicates the heterophilous relationship.
between \(i\) and \(j\), \(d_i^o\) denotes the number of homophilous neighbors of node \(i\) and \(d_i^e\) is the number of heterophilous neighbors of node \(i\). The second term in Equation (13) will force the central node \(i\) approaching to the mean field of the neighborhood consisting of the nodes with the same label, that is, similar nodes will become more similar (i.e., smoothing effect). Meanwhile, the last term will distance the central node away from dissimilar neighbors [2] (connected by heterophilous edges). So the messages passing through heterophilous edges can be deemed to diversify the representations of the nodes with different class labels.

We can understand our new message passing mechanism from the angle of graph signal denoising. Considering the node features \(S \in \mathbb{R}^{n \times f}\) as the noisy signals on a graph \(G\), the goal is to obtain clean signals \(H \in \mathbb{R}^{n \times f}\) which can be smoothed over \(G\), by optimizing the following objective:

\[
\arg \min_H \mathcal{L} = \| H - S \|_F^2 + \lambda_o \cdot \sum_{(i,j) \in E_o} \| H_i - H_j \|_2^2 - \lambda_e \cdot \sum_{(i,j) \in E_e} \| H_i - H_j \|_2^2, \tag{14}
\]

where \(E_o\) and \(E_e\) are the set of homophilous edges and the set of heterophilous edges, respectively. The first term will guide the target \(H\) to be close to the original signal \(S\), while the second term regularizes the feature similarity between nodes connected by homophilous edges (i.e., smoothing effect), and the third term encourages the dissimilarity between two nodes connected by heterophilous edges (i.e., discriminating effect). This way, the connected nodes in the same class will share similar features while the connected nodes from different classes will retain distinct features. The Equation (14) is equivalent to

\[
\arg \min_H \mathcal{L} = \| H - S \|_F^2 + \lambda_o \cdot tr(H^T L_o H) - \lambda_e \cdot tr(H^T L_e H), \tag{15}
\]

where \(L_o = I - \hat{A}_o\) is the normalized symmetric Laplacian on the subgraph consisting of all homophilous edges denoted by \(E_o\) and \(L_e = I - \hat{A}_e\) is for the heterophilous subgraph \(A_e\) defined on \(E_e\).

The gradient with respect to \(H\) at \(X\) is

\[
\frac{\partial \mathcal{L}}{\partial H} \mid_{H=S} = 2 \lambda_o L_o S - 2 \lambda_e L_e S.
\]

Therefore, the one-step update can reads as

\[
H = S - \eta \frac{\partial \mathcal{L}}{\partial H} \mid_{H=S} = S - 2\eta (\lambda_o L_o S - \lambda_e L_e) S
\]

By letting step size \(\eta = \frac{1}{2\lambda_o}\) and \(\alpha = \frac{\lambda_e}{\lambda_o}\), we have

\[
H = S + A_o S - \alpha A_e S. \tag{17}
\]

Taking the original signal \(S\) as the \(l\)th step node feature, then the \((l+1)\)-th iterative step is as follows:

\[
H^{l+1} = H^l + \hat{A}_o H^l - \alpha \hat{A}_e H^l, \tag{18}
\]

which is the compact form of Equation (13), i.e., the new message passing mechanism. Based on the above graph signal filtering modeling, we can view the two components in the proposed message passing \(\hat{A}_o = D^{-\frac{1}{2}} A_o D^{-\frac{1}{2}}\) as the low-frequency filtering (similarity learning) and \(\hat{A}_e = D^{-\frac{1}{2}} A_e D^{-\frac{1}{2}}\) as the high-frequency filtering (distinctiveness learning), respectively. Regarding the value of \(\alpha\), we choose to set it as a hyperparameter and discuss its value in Section 5.5.

**Discussion.** It is noteworthy that our work focuses on optimizing message passing with SA* framework, so the simplified GCN is used as the base model. However, techniques such as residual connections or decay aggregation [28] can be leveraged to improve the performance. For instance, similar to decay aggregation, we can let the signal in previous step be amplified with a tunable factor when updating the representations: \(H^{l+1} = \beta H^l + \hat{A}_o H^l - \alpha \hat{A}_e H^l\), where \(\beta\) is the amplification factor. Such amplification works as the tradeoff between the memory (i.e., the last-step
Table 2. The Statistics of the Datasets

| Datasets    | Nodes | Edges | Features | Classes |
|-------------|-------|-------|----------|---------|
| Cora        | 2,708 | 5,429 | 1,433    | 7       |
| Citeseer    | 3,327 | 4,732 | 3,703    | 6       |
| Pubmed      | 19,717| 44,338| 500      | 3       |
| Chameleon   | 2,277 | 36,101| 2,325    | 5       |
| Squirrel    | 5,201 | 217,073| 2,089    | 5       |
| Film        | 7,600 | 33,544| 931      | 5       |
| Cornell     | 183   | 295   | 1,703    | 5       |
| Texas       | 183   | 309   | 1,703    | 5       |
| Wisconsin   | 251   | 499   | 1,703    | 5       |

feature) and the new aggregated messages. In particular, when $\beta = 2$ this formula is equivalent to the SA*-SGC with residual connections. We find in the experiment that when $\beta > 1$ (denoted SA*-SGC$_2$+), the model performance can be considerably improved.

5 EXPERIMENTS

In this section, we evaluate the performance of SA on transductive node classification task on a wide variety of benchmark graph datasets. We use “SA-base model” and “SA*-base model” to represent the two implementations of the proposed framework with GNNs: structure optimization and aggregation optimization, respectively.

Datasets. We conduct experiments on nine open graph datasets [17, 22, 24, 30] across the full spectrum of homophily ratio $h$. The statistics of the datasets is listed in Table 2. For fair comparison, we follow the data partition in [17] (i.e., 48%/32%/20% of nodes per class for train/validation/test) and adopt the shared 10 random splits for each dataset.

Baseline Models. We compare our method to strong baselines and state-of-the-art approaches, including GCN [13], SGC [26], GAT [25], GCN-Cheby [5], graphSAGE [8], and Mixhop [1]. We also compare our model with heterophily-oriented methods, namely, two variants of H2GCN (i.e., H2GCN-1 and H2GCN-2) [33], Geom-GCN [17] which quantitatively analyzes the homogeneity of a graph for the first time and, every recent work GPRGNN, CPGNN [3, 32]. In particular, we choose the best two among the four variants of CPGNN for comparison. Moreover, we compare our model with recently proposed FAGCN [2] and GBK-GNN [6] that also leverage high-frequency signals. Specifically, FAGCN aims at relieving the over-smoothing effect on disassortative networks, because similar to heterophily, disassortativity accounts for the feature that nodes from different classes tend to connect with each other. In contrast to our two-channel method, GBK-GNN adaptively learns the dissimilarity between neighboring nodes to apply different filtering strategies (i.e., low-frequency or high-frequency filtering).

Setup. We implement models in Pytorch and use Adam optimizer for parameter updates. We set learning rate to 0.005 for the pretraining module, dimension to 64 for linear transformation and use L2 regularizer with regularization factor of 0.0005 on the weights of the linear layer. For the base models SGC, SGC$_2$, and GCN, we use two-layer MLPs as the classification model on all datasets, whose hidden units are 64, learning rate is 0.01, and weight decay is 0.0005. Furthermore, dropout ratio of 0.6 is applied to both layers. For scaling hyper-parameter $\alpha$, we searched in the range [0, 1] spacing with 0.1 and amplification factor $\beta$, we searched in {1, 2, 3, 4}.

5.1 Results

The experimental results on real data are reported in Table 3. We observe that SA using SGC as base model shows consistently strong performance across the full spectrum of homophily.
that uses a model using heterophilous edge message passing and decay with 2-layer MLPs remarkably outperforms the original SGC.

It is noteworthy that compared to the more recent GBBK-GNN that (prior work). We note that SGC

Revisiting the Role of Heterophily in Graph Representation Learning 13:11

Specifically, The SA*-SGC+ model using heterophilous edge message passing and decay aggregation mechanism achieved the highest ranking among all baselines, SA*-SGC; that uses heterophilous edges for high-frequency message passing achieves state-of-the-art performance on average, and SA-SGC with structure optimization is competitive to the strong state-of-the-art method H2GCN. Note that graphSAGE and GCN-Cheb perform well on the datasets with high heterophily, demonstrating the benefits of separate embedding of ego and its neighbors and learning higher order neighbors, which consists of design factors of H2GCN [33]. Compared to vanilla SGC, it is shown that our design for structure optimization is able to significantly improve the expressiveness of GNNs and thus boost the base model. Another observation on graph data with low homophily levels is that among several heterophily-oriented methods, namely, GEOM-GCN, two best H2GCN models, FAGCN, CPGNN, GPRGNN and ours, optimizing structure or message passing (ours) is more effective than integrating high-order features as a whole. The best results are in bold and the second best results are underlined.

**Table 3. Test Node Classification Accuracies (in Percent) on Graph Datasets**

| Datasets | Homo ratio | Cora | Cite. | Pubm. | Cham. | Squi. | Film | Corn. | Texas | Avg. Rank |
|----------|------------|------|-------|-------|-------|-------|------|-------|-------|-----------|
| SA-SGC2 | 0.81 | 76.85±1.5 | 88.50±0.6 | 31.84±1.2 | 33.75±1.3 | 71.46±1.9 | 35.47±0.6 | 85.50±1.6 | 86.48±1.6 | 86.68±1.6 |

Average accuracy and standard deviation over different splits is reported. “+” denotes results obtained from [33]. The best results are in bold and the second best results are underlined.
Table 4. Accuracy and Recall of Edge Classification

| Methods      | Metrics | Cora   | Cite. | Pubm. | Cham. | Squi. | Film | Corn. | Texa. | Wisc. |
|--------------|---------|--------|-------|-------|-------|-------|------|-------|-------|-------|
|              | Homo. ratio h | 0.81   | 0.74  | 0.8   | 0.23  | 0.23  | 0.25 | 0.3   | 0.23  | 0.16  |
| GGCN         | Accuracy | 82.45±1.1 | 72.75±1.5 | 70.11±1.0 | 42.54±1.5 | 38.09±1.9 | 46.23±2.1 | 56.25±4.8 | 9.83±5.3 | 44.53±4.0 |
|              | Recall   | 9.75±1.0  | 5.46±1.1  | 15.48±0.9  | 25.82±1.3  | 21.07±1.5  | 37.14±1.6  | 56.25±4.5  | 5.09±6.7  | 44.56±4.8  |
| w/o pre-train| Accuracy | 72.56±1.1 | 66.74±0.9 | 75.61±0.9 | 32.50±1.1 | 75.12±1.1 | 80.21±1.7 | 85.28±1.7 | 95.12±3.1 | 83.92±2.3 |
|              | Recall   | 20.02±0.8 | 5.51±0.3 | 20.30±0.7 | 36.95±2.1 | 22.22±1.4 | 95.55±1.6 | 100.00±0.0 | 100.00±0.0 | 100.00±0.0 |
| pre-train    | Accuracy | 76.89±1.2 | 77.88±1.1 | 70.15±0.6 | 74.02±1.2 | 77.11±1.5 | 85.41±2.0 | 85.79±1.8 | 97.52±5.8 | 85.92±2.0 |
|              | Recall   | 24.56±1.1 | 3.65±0.1 | 20.30±0.8 | 25.65±2.2 | 26.00±2.4 | 94.2±1.8  | 100.00±0.0 | 100.00±0.0 | 100.00±0.0 |

Connection to GGCN. GGCN also addresses the heterophily with the idea of negating message passing on heterophilous edges. However, there is a significant difference in identifying heterophilous edges between our method and GGCN. To be specific, GGCN uses the cosine similarity between node features to differentiate between homophilous (positive) and heterophilous (negative) edges so that the negating message passing can be applied to certain neighbors. The effectiveness of this manner depends heavily on the consistency of the neighbors’ features and their labels (i.e., nodes with similar features having the same label), a basic assumption of graph smoothing.

However, this assumption is not applicable for graphs with strong heterophily. To see this, we have empirically studied the learned positive and negative edges after GGCN converges. As shown in Table 4, though GGCN is able to identify positive/negative edges (i.e., the homophilous and the heterophilous) with high accuracy on graphs with strong homophily, its performance on graphs with strong heterophily drops significantly. In fact, the recall results suggest that, cosine-based edge classification adopted by GGCN tends to identify edges as homophilous on all datasets. In contrast, our edge classifier shows high accuracy as well as recall values for spotting heterophilous edges when dealing with heterophilous graphs.

To further explore the effects of different ways of differentiating edges on the downstream task, namely, node classification, we have performed comparison experiments on nine benchmark datasets. To get a clear picture of the difference between the effects of two negating methods on prediction performance, we compare GGCN to our method with signal amplification (i.e., \( \beta \geq 1 \)), as similar signal enhancement technique is also used in GGCN. The results in Table 3 demonstrates that edge classifier-based method (i.e., \( \text{SA}^*-\text{SGGC}^{±} \)) is competitive with GGCN on homophilous graphs, while surpasses GGCN on almost all heterophilous graphs. The above observations suggest the effectiveness of our edge classification design in tackling heterophily issues. It is noteworthy that in our framework SGC is used as the base model, which can be replaced by most of the existing end-to-end GNN models. Moreover, though compared to GGCN the performance gains of our model on heterophilous graphs are not remarkable, it is very simple and easy to train, with fewer parameters and lower time complexity.

5.2 Performance of Edge Classification

Edge classification is critical to the effectiveness of our framework. To look into the relationship between edge classification and model performance on downstream task (i.e., node classification), we measure to what degree the edges are correctly classified after the model converges, i.e., the accuracy of edge type identification. Moreover, to validate whether SA can effectively screen out heterophilous edges, we consider the ratio of true heterophilous edge to the total of identified heterophilous edges, termed heterophilous edge recall.

The results in Table 4 suggest that the homophily level has an impact on edge classification bias, that is, the edge type classifier performs better on graphs with low homophily level than on graphs with strong homophily.

---

1 The total time complexity of GGCN is \( O(|V|^2 FL) \) [28], while ours is \( O(|E|FL) \).
Table 5. Test Classification Accuracies (in Percent)

| Drop rate | Datasets | Edge | 6%  | 17%  | 92%  | 98%  |
|-----------|----------|------|-----|-----|-----|-----|
|           | Cora     | SQUI | Film| Texas|
| SGC2      | 86.9±1.6 | 42.4±1.5 | 25.3±1.0 | 59.2±4.4 |
| GCN       | 87.2±1.3 | 36.7±1.7 | 30.1±1.0 | 59.5±1.6 |
| GAT       | 87.5±1.2 | 28.39±1.4 | 29.03±0.9 | 59.10±4.3 |
| Dropedge-SGC | 85.5±1.6 | 39.0±2.0 | 30.9±0.7 | 65.7±8.6 |
| Dropedge-GCN | 84.9±1.3 | 38.8±1.8 | 27.4±1.1 | 58.9±5.0 |
| Dropedge-GAT | 82.11±1.4 | 22.35±1.8 | 21.06±4.1 | 58.92±4.3 |
| SA-SGC   | 87.2±1.6 | 44.2±1.6 | 35.2±1.0 | 81.2±4.8 |
| SA-GCN   | 87.4±1.2 | 41.9±1.7 | 35.0±1.0 | 80.0±5.2 |
| SA-GAT   | 87.6±1.1 | 40.51±2.1 | 31.04±3.5 | 79.43±3.7 |

Bold represents the best result.

highly homophilious graphs. More precisely, in graphs that have strong homophily/heterophily, homophilious/heterophilious edges are far more than heterophilious/homophilious edges. This imbalance is favorable when it is biased towards heterophilious edges. On the other hand, it will be undesirable for structure optimization when homophilious edges dominate, as in this setting the classifier has a higher chance to mistakenly infer an edge to be homophilious, whose ground-truth actually is heterophilious. As the recall index in the table suggests, heterophilious edges are successfully spotted in graphs with strong heterophily, while their identification suffers from homophily bias.

Recall that method 2 involves two channels of message passing: positive aggregation for homophilious neighbors (i.e., neighbors with the label same as central node’s), and negative aggregation for heterophilious neighbors (i.e., neighbors with the labels different from central node’s). So high-accuracy edge classification favors the positive message passing between intra-class nodes and negative message passing between inter-class nodes, which explains why SA performs better on graphs with low homophily.

5.3 Comparing Method 1 to DropEdge

Our structure change strategy (i.e., method 1) has a connection to the previous method DropEdge [21], as both of them are to remove links in graphs. However, different from DropEdge [21] that randomly delete some links, structure optimization aims at removing some heterophilous edges that are optimal-related to performance. In this section, we compare our method with DropEdge to verify the effectiveness of our method. Note that DropEdge randomly removes a portion of edges in the graph during training, so for fair comparison, we set the drop ratio in DropEdge to be equal to the total edge deletion ratio obtained by our method on each dataset.

Table 5 shows the change of accuracy when we apply two methods of dropping edge on two base models (SGC2, GCN, and GAT). We can see that our goal-directed method is evidently superior to random deletion, while random deletion (i.e., DropEdge) may even degrade the performance of base models, especially GAT. Another advantage of our method compared to DropEdge is that SA-GNNs do not introduce any additional hyperparameters, e.g., edge deletion ratio, the model learns target edges to drop adaptively.

5.4 Edge Classifier Analysis

To demonstrate the necessity of learning a separate classifier to distinguish homophilous/ heterophilous edges, we compare two heuristic edge classifiers based on cosine similarity and
Table 6. Test Node Classification Accuracies (in Percent) on Graph Datasets

| Datasets | Cora | Cite. | Pubm. | Cham. | Squi. | Film | Corn. | Texa. | Wisc. |
|----------|------|-------|-------|-------|-------|------|-------|-------|-------|
| Homo.ratio $h$ | 0.81 | 0.74 | 0.8 | 0.22 | 0.22 | 0.22 | 0.2 | 0.06 | 0.1 |
| Cosine-SGC | 77.71±1.3 | 73.15±1.7 | 83.36±1.1 | 30.31±2.4 | 26.35±1.0 | 31.16±1.2 | 66.64±6.4 | 70.83±6.3 | 65.29±4.2 |
| Euclidean-SGC | 87.10±1.0 | 76.30±1.3 | 86.63±1.2 | 32.71±2.6 | 26.56±1.2 | 29.94±1.1 | 70.27±3.8 | 76.75±4.7 | 69.80±4.3 |
| SA*-SGC | 87.36±1.2 | 76.88±1.7 | 87.55±0.6 | 64.94±2.5 | 48.97±1.1 | 38.28±2.2 | 81.12±5.8 | 83.52±5.8 | 84.52±4.0 |

Average accuracy and standard deviation over different splits is reported. And the best results are in bold.

Euclidean distance. We choose an appropriate threshold to classify edges to replace our learned edge classifier. The results are shown in Table 6. Our learned edge classifier shows the best results on nine datasets compared to the two heuristic classifiers.

5.5 Hyper-parameter in Method 2

In method 2 (i.e., aggregation optimization), hyper-parameter $\alpha$ weighs the influence of the features from heterophilous neighbors on node representation. It is interesting to explore the impact of $\alpha$ on the final performance. Results in Figure 4 demonstrate that a high weighting on heterophilous features might hurt performance. In particular, for Cora that shows strong homophily, a large proportion of features from heterophilous neighbors remarkably impairs the performance of GNNs, while for Texas and Chameleon with lower homophily levels, they are more robust to the combination of high-frequency features. In general, a small $\alpha$ around 0.1 is appropriate for graphs.

Besides manual tuning, we also implement a trainable parameterized aggregation optimization, which allows $\alpha$ to be learned. Table 7 reports the performance of method 2 with two kinds of parameter searching techniques on three benchmark datasets. The results show that empirical setting of $\alpha$ is better than adaptive learning.

Table 7. Comparison of Manual Setting and Adaptive Learning of $\alpha$

| Dataset | $\alpha$(learning) | $\alpha$(manual) |
|---------|-------------------|-----------------|
| Cora    | 86.82±1.2         | 87.36±1.2       |
| Cham.   | 59.65±2.0         | 64.94±2.2       |
| Texas   | 78.23±1.0         | 83.52±5.8       |

Bold represents the best result.
model to be aware of the information not only about node features but about class labels, graph Markov neural networks [20] models the joint label distribution with conditional random field, which can be effectively trained with the variational EM algorithm. On the other hand, to mitigate the negative effect of heterophily on performance, FB-GNN [16] performs both low-pass filtering and high-pass filtering on the same graph (the original graph), without differentiating edge types. H2GCN [33] studies the distribution of heterophilious edges in graph and finds that on a graph with a high level of heterophily, the 2-hop neighborhood is always dominated by homophily nodes, so it can be used for feature representation learning. GPRGNN [3] adaptively learns generalized pagerank weights so as to jointly optimize node feature and topological information extraction. In addition, CPGNN [32] shares similar motivations to ours, i.e., both works attempt to identify connected label pairs that are incompatible/compatible. But unlike CPGNN that learns compatibility for all label pairs via belief diffusion, our method directly learns an incompatible/compatible classifier connections. A recent work FAGCN [2] can be regarded as another revelant work. FAGCN aims at disassortative data, in which nodes belonging to different communities (or classes) are connected to each other. To enrich the feature representation, FAGCN utilizes the differences between nodes (the so-called high frequency part of signal). HOG-GCN measures the homophily between node pairs through topology and feature information, and designs GNN that can automatically change the promotion and aggregation process. GBK GNN [6] proposes a GNN model based on a bi-kernel feature transformation and a selection gate. This method shares some similarity with ours, but GBK-GNN is to adaptively learn different kernels, and it aggregates different types of edges by adding, while we explicitly distance heterophilous neighbors by subtracting the messages from those nodes. LINKX [14] is the first method proposed in large-scale heterophilious graph datasets. It can achieve better results on large-scale heterophilious datasets simply by sending the feature matrix and adjacency matrix into MLP for learning and fusion. Generally, previous work mainly focuses on enhancing the expressivity of feature representations. To the best of our knowledge, this work is the first attempt to resolve the heterophily challenge from the angle of edge type identification.

Graph Sparsification. The operation of discarding heterophilious edges in our work can be viewed as a way of graph sparsification. A popular graph sparsification model is graphSAGE [8], whose goal is to allow graph convolution to be available in large-scale graphs. Likewise, Dropedge [21] randomly drops edges to overcome the oversmoothness of graph networks. However, most of the existing work on graph sparsification ignores the relatedness between edges and tasks, except for NeuralSparse [31] that learns k-neighbor subgraphs for deleting task-irrelevant edges. In contrast, our model performs task-oriented sparsification.

7 CONCLUSION AND DISCUSSION

We have proposed a general framework to leverage the heterophily of graphs for boosting GNNs on graph data with low homophily levels from the perspective of edge type. On the conjecture that heterophilious edges mislead GNNs to aggregate information from different classes, we have first devised an optimal structure learning method based on edge type classification. Also based on edge classification, we have proposed to not optimize the graph structure but optimize the way of feature aggregation. Towards this end, we have introduced additional message passing channel to convey information about heterophilous neighbors. Our experiments on a variety of benchmark datasets show the significantly strong performance of the two proposed methods on graphs with high heterophily.

However, it should be noted that since the edge classification will be biased towards the majority of a certain edge type, it remains an open challenge for improving the accuracy of edge classification on graphs with strong homophily, which we leave as future work.
REFERENCES

[1] Sami Abu-El-Haija, Bryan Perozzi, Amol Kapoor, Nazanin Alipourfard, Kristina Lerman, Hrayr Harutyunyan, Greg Ver Steeg, and Aram Galstyan. 2019. Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In Proceedings of the International Conference on Machine Learning. PMLR, 21–29.

[2] Deyu Bo, Xiao Wang, Chuan Shi, and Huawei Shen. 2021. Beyond low-frequency information in graph convolutional networks. In Proceedings of the 55th AAAI Conference on Artificial Intelligence, AAAI 2021, 33rd Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The 11th Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2–9, 2021. AAAI, 3950–3957.

[3] Eli Chien, Jianhao Peng, Pan Li, and Olga Milenkovic. 2021. Adaptive universal generalized PageRank graph neural network. In Proceedings of the 9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3–7, 2021.

[4] Connor W. Coley, Wengong Jin, Luke Rogers, Timothy F. Jamison, Tommi S. Jaakkola, William H. Green, Regina Barzilay, and Klavs F. Jensen. 2019. A graph-convolutional neural network model for the prediction of chemical reactivity. Chemical Science 10, 2 (2019), 370–377.

[5] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. Advances in Neural Information Processing Systems 29 (2016), 3844–3852.

[6] Lun Du, Xiaozhou Shi, Qiang Fu, Xiaojun Ma, Hengyu Liu, Shi Han, and Dongmei Zhang. 2022. GBK-GNN: Gated bi-kernel graph neural networks for modeling both homophily and heterophily. In Proceedings of the ACM Web Conference 2022. 1550–1558.

[7] William L. Hamilton. 2020. Graph representation learning. Synthesis Lectures on Artificial Intelligence and Machine Learning 14, 3 (2020), 1–159.

[8] William L. Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In Proceedings of the 31st International Conference on Neural Information Processing Systems. 1025–1035.

[9] Yifan Hou, Jian Zhang, James Cheng, Kaili Ma, Richard T. B. Ma, Hongzhi Chen, and Ming-Chang Yang. [n.d.]. Measuring and improving the use of graph information in graph neural networks. In Proceedings of the 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26–30, 2020.

[10] Eric Jang, Shixiang Gu, and Ben Poole. [n. d.]. Categorical reparameterization with gumbel-softmax. In Proceedings of the 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24–26, 2017, Conference Track Proceedings. OpenReview.net.

[11] Wengong Jin, Connor W. Coley, Regina Barzilay, and Tommi S. Jaakkola. [n. d.]. Predicting organic reaction outcomes with weisfeiler-lehman network. In Proceedings of the Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4–9, 2017, Long Beach, CA. 2607–2616.

[12] Dongkwon Kim and Alice Oh. 2020. How to find your friendly neighborhood: Graph attention design with self-supervision. In Proceedings of the International Conference on Learning Representations.

[13] Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In Proceedings of the 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24–26, 2017, Conference Track Proceedings.

[14] Derek Lim, Felix Hohne, Xiuyu Li, Sijia Linda Huang, Vaishnavi Gupta, Omkar Bhalaria, and Ser Nam Lim. 2021. Large scale learning on non-homophilous graphs: New benchmarks and strong simple methods. Advances in Neural Information Processing Systems 34 (2021), 20887–20902.

[15] Sitao Luan, Chenqing Hua, Qincheng Lu, Jiaqi Zhu, Mingde Zhao, Shuyuan Zhang, Xiao-Wen Chang, and Doina Precup. 2021. Is heterophily a real nightmare for graph neural networks to do node classification? arXiv preprint arXiv:2109.05641 (2021).

[16] Sitao Luan, Mingde Zhao, Chenqing Hua, Xiao-Wen Chang, and Doina Precup. 2020. Complete the missing half: Augmenting aggregation filtering with diversification for graph convolutional networks. arXiv:2008.08844. Retrieved from https://arxiv.org/abs/2008.08844.

[17] Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. 2020. Geom-GCN: Geometric Graph Convolutional Networks. In 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26–30, 2020.

[18] Hao Peng, Jianxin Li, Yu He, Yaopeng Liu, Mengjiao Bao, Lihong Wang, Yangqiu Song, and Qiang Yang. 2018. Large-scale hierarchical text classification with recursively regularized deep graph-cnn. In Proceedings of the 2018 World Wide Web Conference. 1063–1072.

[19] Jiezhihuo Qiu, Jian Tang, Hao Ma, Yuxiao Dong, Kuansan Wang, and Jie Tang. 2018. Deepinf: Social influence prediction with deep learning. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2110–2119.

[20] Meng Qu, Yoshua Bengio, and Jie Tang. 2019. Gmm: Graph markov neural networks. In Proceedings of the International Conference on Machine Learning. PMLR, 5241–5250.
[21] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. [n. d.]. DropEdge: Towards deep graph convolutional networks on node classification. In Proceedings of the 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26–30, 2020.

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

[23] David I. Shuman, Sunil K. Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. 2013. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. IEEE Signal Processing Magazine 30, 3 (2013), 83–98. DOI: https://doi.org/10.1109/MSP.2012.2235192

[24] Jie Tang, Jimeng Sun, Chi Wang, and Zi Yang. 2009. Social influence analysis in large-scale networks. In Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 807–816.

[25] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In Proceedings of the 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30–May 3, 2018, Conference Track Proceedings. OpenReview.net.

[26] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. 2019. Simplifying graph convolutional networks. In Proceedings of the International Conference on Machine Learning, PMLR, 6861–6871.

[27] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How powerful are graph neural networks? In Proceedings of the 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, May 6–9, 2019. OpenReview.net.

[28] Yujun Yan, Milad Hashemi, Kevin Swersky, Yaoqing Yang, and Danai Koutra. 2021. Two sides of the same coin: Heterophily and oversmoothing in graph convolutional neural networks. arXiv:2102.06462. Retrieved from https://arxiv.org/abs/2102.06462.

[29] Jianwei Yang, Jiasen Lu, Stefan Lee, Dhruv Batra, and Devi Parikh. 2018. Graph r-cnn for scene graph generation. In Proceedings of the European Conference on Computer Vision (ECCV’18), 670–685.

[30] Zhilin Yang, William Cohen, and Ruslan Salakhudinov. 2016. Revisiting semi-supervised learning with graph embeddings. In Proceedings of the International Conference on Machine Learning, PMLR, 40–48.

[31] Cheng Zheng, Bo Zong, Wei Cheng, Dongjin Song, Jingchao Ni, Wenchao Yu, Haifeng Chen, and Wei Wang. 2020. Robust graph representation learning via neural sparsification. In Proceedings of the International Conference on Machine Learning, PMLR, 11458–11468.

[32] Jiong Zhu, Ryan A. Rossi, Anup Rao, Tung Mai, Nedim Lipka, Nesreen K. Ahmed, and Danai Koutra. 2021. Graph neural networks with heterophily. In Proceedings of the 35th AAAI Conference on Artificial Intelligence, AAAI 2021, 33rd Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The 11th Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2–9, 2021. AAAI, 11168–11176.

[33] Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. 2020. Beyond homophily in graph neural networks: Current limitations and effective designs. In Proceedings of the Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6–12, 2020, virtual.

Received 29 June 2022; revised 13 March 2023; accepted 30 May 2023.