H^2GNN: Graph Neural Networks with Homophilic and Heterophilic Feature Aggregations

Shixiong Jing[1], Lingwei Chen[2], Quan Li[1], and Dinghao Wu[1(✉)]

[1] The Pennsylvania State University, State College, University Park, PA 16802, USA
{jing,qbl5082,dinghao}@psu.edu
[2] Wright State University, Dayton, OH 45435, USA
lingwei.chen@wright.edu

Abstract. Graph neural networks (GNNs) rely on the assumption of graph homophily, which, however, does not hold in some real-world scenarios. Graph heterophily compromises them by smoothing node representations and degrading their discrimination capabilities. To address this limitation, we propose H^2GNN, which implements **H**omophilic and **H**eterophilic feature aggregations to advance **GNN**s in graphs with homophily or heterophily. H^2GNN proceeds by combining *local feature separation* and *adaptive message aggregation*, where each node separates local features into similar and dissimilar feature vectors, and aggregates similarities and dissimilarities from neighbors based on connection property. This allows both similar and dissimilar features for each node to be effectively preserved and propagated, and thus mitigates the impact of heterophily on graph learning process. As dual feature aggregations introduce extra model complexity, we also offer a simplified implementation of H^2GNN to reduce training time. Extensive experiments on seven benchmark datasets have demonstrated that H^2GNN can significantly improve node classification performance in graphs with different homophily ratios, which outperforms state-of-the-art GNN models.

Keywords: Graph Neural Networks · Heterophily · Node Classification

1 Introduction

Graph neural networks (GNNs) have emerged as prevalent models to address diverse graph-based tasks [10–12,16]. Their impressive learning capabilities can be attributed to the utilization of the message-passing mechanism, which allows for neighborhood aggregation through graph structure. In other words, GNNs rely on the assumption of graph *homophily* suggesting that nodes tend to connect with others sharing similar features [19,27,28]. However, this assumption does not hold in some real-world graph learning problems, such as fraud and bot detection [3,7], where attackers tend to build the relationships between malicious

M. Onizuka et al. (Eds.): DASFAA 2024, LNCS 14855, pp. 342–352, 2024.
https://doi.org/10.1007/978-981-97-5572-1_23

and benign nodes for stealth, leading to graphs with *heterophily*. When neighbors in a graph significantly differ, conventional aggregation mechanisms used by GNNs that directly blend neighborhood features may result in smoothed or indistinct representations, and degrade the discrimination capabilities of GNNs.

To address this limitation, recent methods have been proposed to handle graph heterophily, which broadly falls into three categories: neighbor extension [1,15,21], inter-layer connections [5,17,28], and adaptive message aggregation [4,8,18,26]. The neighbor-extension approach captures long-range dependencies to complement node representations, while the inter-layer connection approach capitalizes residuals to alleviate the disruptions caused by low-level abnormal connections. However, their efficacy is limited when tackling graphs with high levels of heterophily. In contrast, the adaptive message aggregation method trains separate filters to manipulate homophily and heterophily, and aggregates the outputs from these filters to derive node representations. This formulation is more adept at managing local homophily and heterophily but may compromise long-range information propagation in heterophilic graphs [2].

This naturally leads to a pivotal insight to address the heterophily issue: both similar and dissimilar features from neighbors contribute to node characterization, which need to be locally extracted, separately preserved, and further propagated to higher orders to prevent aggregations from blending distinguishable information associated with different labels. As such, we propose H^{2}GNN, which implements **H**omophilic and **H**eterophilic feature aggregations to advance **GNN**s in graphs with homophily or heterophily. Specifically, instead of mixing features from neighbors as a single feature vector, H^{2}GNN proceeds by combining the ideas of *local feature separation* and *adaptive message aggregation*. Each node separates local features into similar and dissimilar feature vectors, and aggregates corresponding features from neighbors based on connection property: when the connection is homophilic, the neighbor's similar and dissimilar features would respectively flow into the target node's similar and dissimilar feature vectors; conversely, in heterophilic connections, the neighbor's two sets of features would be amalgamated into the target node's opposite feature vectors.

By maintaining two distinct feature vectors side by side, both similar and dissimilar features for each node can be effectively preserved and propagated, and thus mitigate the impact of heterophily on the graph learning process. For each aggregation operation, we further elaborate an edge identifier that determines whether a given node pair is homophilic or heterophilic to selectively extract neighborhood features, promoting adaptive message aggregation. Moreover, considering that dual feature aggregations in H^{2}GNN introduce extra model complexity, we offer a simplified implementation to significantly reduce training time.

2 Problem Statement

We denote a given graph as $G = (V, E, \mathbf{X})$, where $V(n = |V|)$ is the set of nodes, E is the set of edges, and $\mathbf{X} \in \mathbb{R}^{n \times d}$ is the feature matrix. Edges E can be encoded as an adjacency matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ and $\mathbf{A}_{ij} = \{0, 1\}$. The

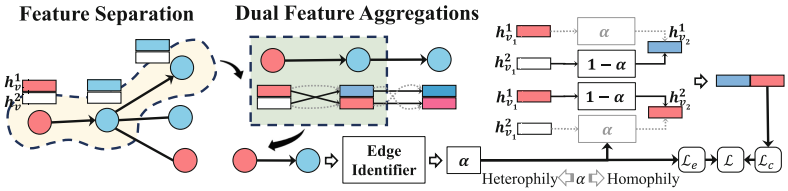


Fig. 1. Overview of our proposed model H^{2}GNN.

neighbors for v_i is represented as $\mathcal{N}(v_i) = \{v_j | (v_i, v_j) \in E)\}$. Each labeled node is associated with a ground truth $y \in Y = \{0, 1, \cdots, k-1\}$. GNN models enforce each node to aggregate information from its neighbors and generate higher-level embedding for downstream node classification. These graphs exhibit different degrees of homophily (or heterophily). Homophily is a concept that describes the tendency of nodes to associate with similar others [27]. In this paper, we focus on homophily in class labels [28], where a graph with high homophily indicates that connected nodes share the same label with a high probability. Heterophily is the opposite of homophily to describe the status of connected nodes belonging to different labels.

3 Proposed Model: H^{2}GNN

In this section, we present our proposed GNN model H^{2}GNN to deal with graph heterophily, the overview of which is illustrated in Fig. 1.

3.1 Feature Separation

To restore the discrimination capability in heterophilic graphs, GNN models need to harness the information of neighborhood differences [18]. Following this idea, various adaptive message aggregation methods [4,8,18,21] have been deployed that train separate filters to extract similar and dissimilar signals from neighbors, enriching node representations while mitigating the impact of heterophily. Nevertheless, these approaches have one limitation: though dissimilarities from neighbors are extracted, they are still mixed with similarities to be formulated as a single feature vector; such straightforward preservation may not be helpful for propagating dissimilarities to higher-order nodes [2].

To address this limitation, we propose to preserve these similarities and dissimilarities separately. Specifically, we devise two distinct feature vectors for each node $v \in V$ to facilitate feature aggregation: similar feature vector $\mathbf{h}_v^1$, and dissimilar feature vector $\mathbf{h}_v^2$. $\mathbf{h}_v^1$ is used to aggregate and preserve similar signals from neighborhood features, while $\mathbf{h}_v^2$ is used to preserve dissimilar signals extracted from neighbors. In this respect, the representation for each node $v \in V$ during aggregation can be formalized as $\mathbf{h}_v = \{\mathbf{h}_v^1, \mathbf{h}_v^2\}$, which is initialized as $\mathbf{h}_v = \{\mathbf{X}_v, \mathbf{0}\}$. Such a bi-vector feature setting can collaboratively preserve local

neighborhood similarities and dissimilarities, and promote long-range information propagation, which, in turn, boosts node representation expressiveness and model discrimination power. The details about the feature aggregations on $\mathbf{h}_v^1$ and $\mathbf{h}_v^2$ will be elaborated in the subsequent section.

3.2 Dual Feature Aggregations

Dual feature aggregations significantly rely on edge property specified by the labels of connected nodes, which is not always available in real-world graphs, as most of the nodes are not pre-annotated. To tackle this issue, we introduce an edge identifier to discriminate if a node pair is homophilic or heterophilic.

Edge Identifier. Similar to node classification using node representations, here we devise an edge identifier using the representations of its connected nodes to determine the property of the edge. Considering that node representations vary at each aggregation layer, edge identification is performed in a layer-wise manner. Specifically, at aggregation layer l, the edge identifier can be formalized as:

$$\alpha_{ij} = \sigma((\mathbf{h}_{v_i}^{(l-1)} || \mathbf{h}_{v_j}^{(l-1)})\mathbf{W}_e^{(l)}) \tag{1}$$

where $\sigma(\cdot)$ is a non-linear activation function and $\mathbf{W}_e^{(l)}$ is a learnable weight matrix at layer l. α_{ij} $(0 \leq \alpha_{ij} \leq 1)$ is a prediction score, indicating how possible an edge $e_{v_i v_j}$ is homophilic.

Feature Aggregation. Generally, the normalized adjacency matrix $\tilde{\mathbf{A}}$ is considered a low-lass filter in GNNs to retain the commonality of neighboring features [4,20], while $\mathbf{I} - \tilde{\mathbf{A}}$ provides diversification operation, which is considered as high-pass filter to extract neighborhood differences [18]. This inspires us to directly employ the soft prediction of edge identifier, i.e., prediction score α_{ij} $(0 \leq \alpha_{ij} \leq 1)$ instead of hard prediction to perform dual feature aggregation, which enables α_{ij} and $1 - \alpha_{ij}$ to act like advanced dual filters that flexibly control the number of similarities and dissimilarities from each neighbor to be aggregated and thus further refine the learned node representations. Formally, for a given node v_i, the dual feature aggregations are implemented as follows:

$$\mathbf{h}_{v_i}^{1,(l)} = \sigma(\mathbf{h}_{v_i}^{1,(l-1)}\mathbf{W}_0 + \frac{1}{|\mathcal{N}(v_i)|}\sum_{v_j \in \mathcal{N}(v_i)} \alpha \mathbf{h}_{v_j}^{1,(l-1)}\mathbf{W}_1 + (1-\alpha)\mathbf{h}_{v_j}^{2,(l-1)}\mathbf{W}_2) \tag{2}$$

$$\mathbf{h}_{v_i}^{2,(l)} = \sigma(\mathbf{h}_{v_i}^{2,(l-1)}\mathbf{W}_0 + \frac{1}{|\mathcal{N}(v_i)|}\sum_{v_j \in \mathcal{N}(v_i)} (1-\alpha)\mathbf{h}_{v_j}^{1,(l-1)}\mathbf{W}_1 + \alpha \mathbf{h}_{v_j}^{2,(l-1)}\mathbf{W}_2) \tag{3}$$

where $\sigma(\cdot)$ is specified as ReLU, $\mathbf{W}_0$, $\mathbf{W}_1$, and $\mathbf{W}_2$ are learnable weight matrices to promote feature aggregations. For node v_i at layer l, $\mathbf{h}_{v_i}^{1,(l)}$ aggregates neighboring features and produces its new similarity feature vector, while $\mathbf{h}_{v_i}^{2,(l)}$ produces its new dissimilarity feature vector, both of which contribute to the

new hidden representation. By leveraging dual feature aggregations, features contributing to the node's ground truth can be effectively preserved in $\mathbf{h}_v^1$, while features correlated to other classes can be explicitly saved in $\mathbf{h}_v^2$ to support higher-order enhancement, where heterophily can barely impact on the outputs from different classes.

3.3 Loss Optimization

After multi-layer dual feature aggregations, $\mathbf{h}_v^1$ and $\mathbf{h}_v^2$ are concatenated and fed to a fully-connected layer for node classification, which results in a cross-entropy loss $\mathcal{L}_c$ between predictions and nodes' true labels. In addition, for each feature aggregation layer l, edge identifier needs to be optimized as well. We select those edges whose connected nodes are labeled to construct homophilic and heterophilic edge samples for facilitating edge identifier training, which leads to another cross-entropy loss $\mathcal{L}_e^{(l)}$. Accordingly, the final training objective can be formulated as follows:

$$\mathcal{L} = \mathcal{L}_c + \gamma \sum_{l=1}^{L} \mathcal{L}_e^{(l)} \tag{4}$$

where γ is a balance parameter to trade off two losses, and L is the number of feature aggregation layers. All the weights can be updated by minimizing $\mathcal{L}$.

3.4 Simplified Model

We refer to our proposed GNN model as H^{2}GNN-W, since it resorts to different weight matrices for dual feature aggregations. This naturally introduces extra model complexity that increases the training time cost. To keep a better trade-off between model effectiveness and efficiency, we offer a simplified implementation of our framework called H^{2}GNN-S. H^{2}GNN-S reduces training time using a similarity metric $\alpha_{ij} = \text{sim}(\mathbf{h}_{v_i}^{(l-1)}, \mathbf{h}_{v_j}^{(l-1)})$ as the edge identifier. To exclude the weight matrix updates and multiplications during feature aggregations, we assemble all weights into a single linear layer that maps the original feature matrix $\mathbf{X}$ to $\mathbf{X}^{(0)} = f_\theta(\mathbf{X}) \in \mathrm{R}^{d \times m}$ $(m \ll d)$, such that $\mathbf{h}_v (v \in V)$ is initialized as $\mathbf{h}_v = \{\mathbf{X}_v^{(0)}, \mathbf{0}\}$, where $\mathbf{h}_v^1 = \mathbf{X}_v^{(0)}$, $\mathbf{h}_v^2 = \mathbf{0}$. Afterward, feature aggregations are performed without any weighted operations:

$$\mathbf{h}_{v_i}^{1,(l)} = \sigma(\mathbf{h}_{v_i}^{1,(l-1)} + \frac{1}{|\mathcal{N}(v_i)|} \sum_{v_j \in \mathcal{N}(v_i)} \alpha \mathbf{h}_{v_j}^{1,(l-1)} + (1-\alpha)\mathbf{h}_{v_j}^{2,(l-1)}) \tag{5}$$

$$\mathbf{h}_{v_i}^{2,(l)} = \sigma(\mathbf{h}_{v_i}^{2,(l-1)} + \frac{1}{|\mathcal{N}(v_i)|} \sum_{v_j \in \mathcal{N}(v_i)} (1-\alpha)\mathbf{h}_{v_j}^{1,(l-1)} + \alpha \mathbf{h}_{v_j}^{2,(l-1)}) \tag{6}$$

The outputs are also concatenated and fed to a fully connected layer for node classification. Since there's no necessity to train edge identifiers, the loss function $\mathcal{L}$ can be reduced to $\mathcal{L} = \mathcal{L}_c$. This simplified model sustains potent learning capability while also yielding the additional benefit of reduced time consumption.

4 Experimental Results and Analysis

4.1 Experimental Setup

Datasets. We evaluate H^2GNN on seven public datasets with different homophily ratios. Cora and Citeseer [25] are benchmark citation networks; Texas and Wisconsin [21] are subdatasets of WebKB collected from different universities; Actor [22] is a subgraph induced from the film-director-actor-writer network; Penn94 and Genius [14] are two large-scale online social networks. For Penn94 and Genius, we use 50%-25%-25% random split to match the experimental setting in related works [13,14]. For all other datasets, we use the 48%-32%-20% data-split as provided by [8,21]. Table 1 summarizes the data statistics.

Table 1. Statistics of the datasets

Dataset	#Nodes	#Edges	#Features	#Classes
Cora	2,078	5,429	1,433	7
Citeseer	3,327	9,228	3,703	6
Texas	183	309	1,703	5
Wisconsin	251	499	1,703	5
Actor	7,600	30,019	932	5
Penn94	41,554	1,362,229	5	2
Genius	421,961	984,979	12	2

Baselines. We use 3 GNN variants and 9 advanced GNNs for graph heterophily as baselines. Conventional GNNs includes GCN [9], GAT [23], and GIN [24]; advanced GNNs includes GCNII [5], Geom-GCN [21], CPGNN [27], MixHop [1], H2GCN [28], GPRGNN [6], GBK-GNN [8], LinkX [14], and GloGNN [13].

Implementation Details. The number of aggregation layers L is set to 2. Models are trained for 2,000 epochs with patience of 200 using Adam optimizer with learning rate $lr = 0.01$ and $5e-4$ L2 regularization. For H^2GNN-S, $\text{sim}(\cdot)$ is specified as cosine similarity, and the feature dimension is set to $m = 16$. We further introduce consistency loss to provide an advantage of smoothing decision boundaries, which is adjusted by a balance parameter λ. The impacts of λ and γ on the performance of H^2GNN are evaluated in Sect. 4.3.

4.2 Comparison with Baselines

We compare H^2GNN with 12 selected baselines over seven different datasets. Table 2 presents these comparative results. We observe that: (1) Traditional

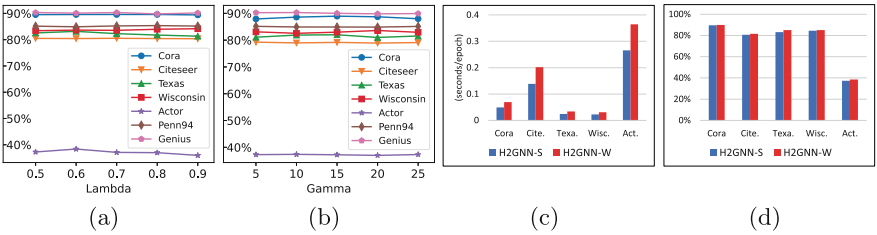
GNN models perform relatively well in homophilic graphs, but suffer from drastic drops in heterophilic graphs. (2) By contrast, enhanced models tailored for heterophilic graphs manage to bring up the accuracy to a higher level in graphs with extremely low homophily ratios, due to their enhancement in neighbor extension, inter-layer connections, or adaptive message aggregation, but some of these models sacrifice the performance in homophilic graphs. (3) Our proposed model delivers the best or second-best results, outperforming most of the baselines across all graph datasets. Among enhanced GNN models, H^{2}GNN-W achieves the best performance on most of the datasets. Even the simplified version of our model, H^{2}GNN-S, offers comparable results on all datasets. In summary, H^{2}GNN achieves state-of-the-art performance on all seven benchmarks (both homophily and heterophily) and outperforms the leading GNN models in most cases.

Table 2. Comparison of baselines (classification accuracy (%)) on both homophily and heterophily graph datasets. The best result for each dataset is highlighted in blue color.

	Cora	**Citeseer**	**Texas**	**Wisconsin**	**Actor**	**Penn94**	**Genius**
H. ratio r	**0.82**	**0.70**	**0.11**	**0.14**	**0.20**	**0.46**	**0.61**
GCN	86.81±0.79	73.96±1.02	49.52±5.29	46.78±5.36	26.96±1.25	82.47±0.27	87.42±0.37
GAT	84.31±1.21	71.89±1.40	41.94±5.33	63.04±5.33	26.56±0.98	81.53±0.55	55.80±0.87
GIN	81.93±0.57	68.08±0.62	29.03±4.42	45.65±4.71	23.61±0.76	81.78±0.39	87.16±0.51
GCNII	87.22±0.98	75.11±1.18	51.61±5.47	58.70±3.25	28.72±0.60	82.92±0.59	89.82±0.49
GeomGCN	84.10±1.12	76.28±2.06	67.57±5.35	68.63±4.92	30.00±1.26	83.01±0.38	89.75±0.41
CPGNN	79.40±1.39	69.41±0.45	75.68±5.12	76.47±6.16	35.59±0.86	79.76±0.39	85.33±0.92
H2GCN	82.70±0.87	43.70±0.24	72.97±4.53	70.59±5.76	35.39±1.34	81.31±0.60	OOM
MixHop	87.61±0.85	76.26±0.33	77.84±7.73	75.88±4.90	32.22±2.34	83.47±0.71	90.58±0.16
GBK-GNN	88.69±0.42	79.18±0.96	81.08±4.87	84.21±4.33	38.97±0.97	81.99±0.47	90.13±0.50
GPRGNN	87.95±1.18	77.13±1.67	78.38±4.36	82.94±4.21	34.63±1.22	81.38±0.16	90.05±0.31
LinkX	84.64±1.13	73.19±0.99	74.60±8.37	75.49±5.72	36.10±5.55	84.71±0.52	90.77±0.27
GloGNN	88.31±1.13	77.41±1.65	84.32±4.15	87.06±3.53	37.35±1.30	85.57±0.35	90.66±0.11
H^{2}GNN-S	89.64±1.20	80.59±1.37	83.15±5.67	84.43±5.32	37.41±1.86	85.03±0.41	90.25±0.46
H^{2}GNN-W	89.97±1.23	81.60±1.95	85.01±5.98	85.11±6.10	38.53±1.46	85.61±0.56	90.80±0.42

4.3 Parameter Evaluation

The performance of H^{2}GNN can be potentially affected by λ, which is the coefficient controlling the weight of consistency loss, and γ, which is the coefficient controlling the weight of edge identification loss. In this section, we evaluate the impacts of these parameter settings. The experimental results are reported in Fig. 2(a) and (b). It can be observed that different parameters contribute to slightly different results. However, these fluctuations are insignificant, implying that our model's performance is stable across various parameter settings.



(a) (b) (c) (d)

Fig. 2. Impact of parameters and model variants on model performance: (a) accuracy of H^2GNN-W with different λ; (b) accuracy of H^2GNN-W with different γ; (c) time cost of H^2GNN-W and H^2GNN-S; (d) accuracy of H^2GNN-W and H^2GNN-S.

4.4 Comparison Between H^2GNN-W and H^2GNN-S

In this section, we analyze the advantages and disadvantages between weighted H^2GNN (H^2GNN-W) and simplified H^2GNN (H^2GNN-S). We illustrate the comparative results in Fig. 2(c) and (d). Compared to H^2GNN-W, H^2GNN-S significantly reduces the training time per epoch on all tested datasets by over 20%, while not compromising the node classification performance. Specifically, the accuracy of node classification using H^2GNN-S only drops by less than 2% across all tested datasets. The reduced time consumption comes from the removal of weights in the aggregation layers and the exclusion of the edge identifier training. The preserved dual feature aggregations enable H^2GNN-S to maintain its potent learning capability. Overall, H^2GNN-S provides a better trade-off between effectiveness and training efficiency than H^2GNN-W.

4.5 Ablation Study

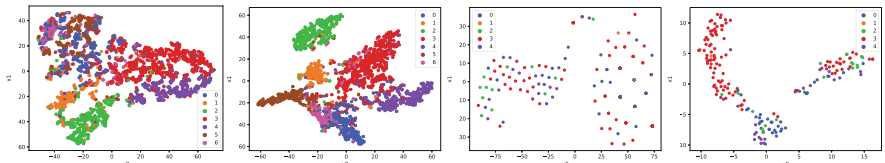
In this section, we set up an ablation study to investigate how different components contribute to the performance of our model. We investigate three components in our model design: (1) Feature separation, which refers to the combination of local feature separation and dual feature aggregation; (2) Edge identification, which refers to the utilization of α to identify homophilic and heterophilic edges; (3) Weighted operation, which refers to the inclusion of all weight matrices in feature aggregation operation. As illustrated in Table 3, all three components contribute to the performance of H^2GNN. Among these three components, the impact of weighted operation is the smallest. However, removing either feature separation or edge identification leads to significant performance drops on heterophilic graphs. This outcome aligns with our expectations. The core functionality of aggregation layers in H^2GNN relies on the collaboration of feature separation and edge identification. Therefore, the exclusions of either feature separation or edge identification may enforce a drastic performance drop.

Table 3. Ablation study: Feature Separation (FS), Edge Identification(EI), and Weighted Operation (WO)

FS	EI	WO	Cora	Cite.	Texa.	Wisc.
			86.59	73.04	48.03	48.20
		✓	88.04	70.16	48.35	65.41
	✓	✓	87.77	72.83	69.08	66.37
✓		✓	88.06	72.13	49.22	66.18
✓	✓		89.64	80.59	83.15	84.43
✓	✓	✓	89.97	81.60	85.01	85.11

4.6 Case Study

To validate our claim that H^{2}GNN provides more distinguishable node embeddings, we present a brief case study on two datasets, Cora (homophilic) and Texas (heterophilic), to showcase the difference between embeddings generated by GCN and H^{2}GNN. We map node embeddings into a two-dimensional space using t-SNE, and the resulting embeddings are visualized in Fig. 3. In both cases, H^{2}GNN either generates clusters with better-distinguished boundaries or redistributes nodes more cohesively. These observations reaffirm the effectiveness of H^{2}GNN in learning meaningful and distinguishable node representations.



(a) (b) (c) (d)

Fig. 3. Visualization of node embeddings: (a) Cora processed by GCN, (b) Cora processed by H^{2}GNN, (c) Texas processed by GCN, (d) Texas processed by H^{2}GNN.

5 Conclusion

In this paper, we introduce a new model H^{2}GNN for node classification in graphs with homophily or heterophily. H^{2}GNN employs feature separation and dual feature aggregation guided by edge identification to better preserve and propagate both similarities and dissimilarities for each node, which not only boosts the expressiveness of node representations but also mitigates the impact of heterophily on the graph learning process. Evaluation through extensive experiments demonstrates that our model achieves state-of-the-art performance, which

affirms its effectiveness in node classification, superiority over baselines, and practical significance in handling both heterophilic and homophilic graphs.

Acknowledgments. L. Chen's work is partially supported by the NSF under grant CNS-2245968.

References

1. Abu-El-Haija, S., et al.: Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In: International Conference on Machine Learning, pp. 21–29. PMLR (2019)
2. Alon, U., Yahav, E.: On the bottleneck of graph neural networks and its practical implications. In: International Conference on Learning Representations (ICLR) (2021)
3. Ashmore, B., Chen, L.: Hover: Homophilic oversampling via edge removal for class-imbalanced bot detection on graphs. In: CIKM, pp. 3728–3732 (2023)
4. Bo, D., Wang, X., Shi, C., Shen, H.: Beyond low-frequency information in graph convolutional networks. In: AAAI, pp. 3950–3957 (2021)
5. Chen, M., Wei, Z., Huang, Z., Ding, B., Li, Y.: Simple and deep graph convolutional networks. In: ICML (2020)
6. Chien, E., Peng, J., Li, P., Milenkovic, O.: Adaptive universal generalized pagerank graph neural network. In: ICLR (2021)
7. Dou, Y., Liu, Z., Sun, L., Deng, Y., Peng, H., Yu, P.S.: Enhancing graph neural network-based fraud detectors against camouflaged fraudsters. In: CIKM (2020)
8. Du, L., et al.: Gbk-gnn: gated bi-kernel graph neural networks for modeling both homophily and heterophily. In: Proceedings of the ACM Web Conference 2022, pp. 1550–1558 (2022)
9. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016)
10. Li, Q., Chen, L., Cai, Y., Wu, D.: Hierarchical graph neural network for patient treatment preference prediction with external knowledge. In: PAKDD, pp. 204–215 (2023)
11. Li, Q., Chen, L., Jing, S., Wu, D.: Pseudo-labeling with graph active learning for few-shot node classification. In: ICDM, pp. 1115–1120 (2023)
12. Li, Q., Li, X., Chen, L., Wu, D.: Distilling knowledge on text graph for social media attribute inference. In: SIGIR, pp. 2024–2028 (2022)
13. Li, X., et al.: Finding global homophily in graph neural networks when meeting heterophily. In: International Conference on Machine Learning, pp. 13242–13256. PMLR (2022)
14. Lim, D., et al.: Large scale learning on non-homophilous graphs: New benchmarks and strong simple methods. NeurIPS **34**, 20887–20902 (2021)
15. Liu, S., Chen, L., Dong, H., Wang, Z., Wu, D., Huang, Z.: Higher-order weighted graph convolutional networks. arXiv preprint arXiv:1911.04129 (2019)
16. Liu, S., et al.: Local augmentation for graph neural networks. In: International Conference on Machine Learning (2022)
17. Liu, S., et al.: Graph neural networks with adaptive residual. NeurIPS **34**, 9720–9733 (2021)
18. Luan, S., Hua, C., Lu, Q., et al.: Is heterophily a real nightmare for graph neural networks to do node classification? arXiv preprint arXiv:2109.05641 (2021)

19. Maurya, S.K., Liu, X., Murata, T.: Improving graph neural networks with simple architecture design. arXiv preprint arXiv:2105.07634 (2021)
20. Nt, H., Maehara, T.: Revisiting graph neural networks: All we have is low-pass filters. arXiv preprint arXiv:1905.09550 (2019)
21. Pei, H., Wei, B., Chang, K.C.C., Lei, Y., Yang, B.: Geom-gcn: geometric graph convolutional networks. arXiv preprint arXiv:2002.05287 (2020)
22. Tang, J., Sun, J., Wang, C., Yang, Z.: Social influence analysis in large-scale networks. In: KDD, pp. 807–816 (2009)
23. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y.: Graph attention networks. arXiv preprint arXiv:1710.10903 (2017)
24. Xu, K., Hu, W., Leskovec, J., Jegelka, S.: How powerful are graph neural networks? arXiv preprint arXiv:1810.00826 (2018)
25. Yang, Z., Cohen, W., Salakhudinov, R.: Revisiting semi-supervised learning with graph embeddings. In: ICML, pp. 40–48 (2016)
26. Zheng, X., Liu, Y., Pan, S., Zhang, M., Jin, D., Yu, P.S.: Graph neural networks for graphs with heterophily: a survey. arXiv preprint arXiv:2202.07082 (2022)
27. Zhu, J., et al.: Graph neural networks with heterophily. In: AAAI, pp. 11168–11176 (2021)
28. Zhu, J., Yan, Y., Zhao, L., et al.: Beyond homophily in graph neural networks: current limitations and effective designs. NeurIPS **33**, 7793–7804 (2020)