Exploiting Neighbor Effect: Conv-Agnostic GNN Framework for Graphs With Heterophily

Jie Chen, Graduate Student Member, IEEE, Shouzhen Chen, Junbin Gao, Zengfeng Huang, Junping Zhang, Senior Member, IEEE, and Jian Pu, Member, IEEE

Abstract—Due to the homophily assumption in graph convolution networks (GCNs), a common consensus in the graph node classification task is that graph neural networks (GNNs) perform well on homophilic graphs but may fail on heterophilic graphs with many interclass edges. However, the previous interclass edges’ perspective and related homo-ratio metrics cannot well explain the GNNs’ performance under some heterophilic datasets, which implies that not all the interclass edges are harmful to GNNs. In this work, we propose a new metric based on the von Neumann entropy to reexamine the heterophily problem of GNNs and investigate the feature aggregation of interclass edges from a neighbor identifiable perspective. Moreover, we propose a simple yet effective Conv-Agnostic GNN framework (CAGNNs) to enhance the performance of most GNNs on the heterophily datasets by learning the neighbor effect for each node. Specifically, we first decouple the feature of each node into the discriminative feature for downstream tasks and the aggregation feature for graph convolution (GC). Then, we propose a shared mixer module to adaptively evaluate the neighbor effect of each node to incorporate the neighbor information. The proposed framework can be regarded as a plug-in component and is compatible with most GNNs. The experimental results over nine well-known benchmark datasets indicate that our framework can significantly improve performance, especially for the heterophily graphs. The average performance gain is 9.81%, 25.81%, and 20.61% compared with graph isomorphism network (GIN), graph attention network (GAT), and GCN, respectively. Extensive ablation studies and robustness analysis further verify the effectiveness, robustness, and interpretability of our framework. Code is available at https://github.com/JC-202/CAGNN.

Index Terms—Graph neural networks (GNNs), heterophily, homophily, node classification, representation learning.
still distinguishable after a simple mean aggregation operator since their neighbor’s distribution is identifiable [19], [20].

In this work, we first present a novel metric to quantify the identifiability of the neighbor’s distribution. Compared with the previous interclass edges metrics, ours can better indicate the graph-level neighbor effect for classifications and understand the heterophily problem. Then, motivated by this metric, we propose a simple yet effective Conv-Agnostic GNN framework (CAGNNs) to improve GNNs’ performance on heterophilic graphs by learning the node-level neighbor effect. Our framework is Conv-Agnostic and more general.

1) We present a new perspective of heterophily with neighbor identifiability and quantify it with the metric inspired by the von Neumann entropy. Specifically, we extract the label distribution matrix of neighbors at the class level and extend the von Neumann entropy [21] as a new metric to measure the identifiability of neighbors for the graph. Unlike the edge perspective, this metric demonstrates the importance of the neighbor effect for classifications in understanding the heterophily problems. It can be used to explain the performance differences in GNNs among various datasets with the same homo-ratio. Furthermore, it motivates us to consider the node-level neighbor effect to improve classical GNNs.

2) We propose CAGNNs as a general framework to improve classical GNNs by learning the neighbor effect for each node. To consider the neighbor effect at the node level, we first decouple the features of a node into two parts: discriminative features used for downstream tasks and aggregated features from its neighbors via GC. Then, we introduce a mixer module to fuse these two features into new discriminative features for classification via learning the node-level neighbor effect. Our framework can improve the performance of most GNNs under heterophily without modifying the GC kernel due to the generic node-level neighbor effect learning mechanism. Thus, unlike other heterophilic-oriented GNNs that need to modify the convolution kernel [7], [14], [17], our framework is Conv-Agnostic and more general.

3) We conduct extensive experiments on nine well-known benchmark datasets to verify the effectiveness, interpretability, and robustness of CAGNNs. We show that a simple shared-parameter mixer module can substantially enhance the performance of most GNNs on heterophilic graphs while maintaining the performance on homophilic graphs. The average performance gain is 9.81%, 25.81%, and 20.61% compared with graph isomorphism network (GIN), graph attention network (GAT), and graph convolution networks (GCNs), respectively. Moreover, our framework shows good interpretability for determining whether neighbor information is helpful for node classification tasks, and the framework is robust to over-smoothing and noisy edges.

The article is organized as follows. In Section II, we survey the related works. Section III briefly introduces the notations and background. In Section IV, we present the proposed novel metric for measuring neighbor identifiability for graphs with heterophily from a neighbor perspective. In Section V, we describe the proposed CAGNN framework and implementation in detail. Evaluation results on nine benchmark datasets and ablation studies are presented in Section VI to verify the effectiveness, interpretability, and robustness of the proposed framework. The final section concludes the article.

II. RELATED WORKS

A. Graph Neural Networks

The GNN models can be roughly categorized into spectral and spatial methods. Early on, Bruna et al. [22] first propose a spectral graph-based extension of convolutional networks to graphs. In a follow-up work, ChebNet [23] define GCs using the Chebyshev polynomials to remove the computationally expensive Laplacian eigendecomposition. GCNs [6] further simplify GCs by stacking layers of the first-order Chebyshev polynomial filters with a redefined propagation matrix. Also, the GCN bridges the spectral and spatial domain gap since it can also be regarded as a mean aggregator to aggregate neighbor information to each node. Furthermore, in the spatial domain, GAT applies the attention mechanism to learn edge weights to improve the aggregation step. Xu et al. [24] study the expressiveness of graph neural networks and introduce GIN, which is proven to be as powerful as the Weisfeiler–Lehman (WL) test. There are many other graph neural models [2], [25], [26]; we refer to [1], [3], and [27] for a more comprehensive review.

To investigate why and when graph neural networks work well for node classification, some researchers aim to understand the behavior of GNNs. Nt and Maehara [5] indicate that graph neural networks only perform low-pass filtering on feature vectors and do not have the nonlinear manifold learning property from a signal processing perspective. Li et al. [4] point out that the GCN model’s GC is actually a special form of Laplacian smoothing, which is consistent with the homophily assumption. However, it also brings potential concerns about making the features of connected nodes from different labels indistinguishable. On the other hand, Ma et al. [20] theoretically reveal that homophily is not a necessary assumption for the GCN model. Moreover, for attention-based GNNs, Wang et al. [28] find that stacking multiple attention layers causes excessive smoothing of node features due to information exchange over interclass edges. In [29] and [30], the current popular message-passing scheme in GNNs is summarized and it is argued that the message...
between intraclass edges would help nodes receive information gain. In contrast, the interclass edges may introduce negative disturbance.

B. GNNs for Heterophily

Recently, heterophilic graph learning has become an upward-trending research topic, and various specific structured GNNs have been proposed. Most argue that message-passing during interclass edges is harmful to the node classification task and try to avoid harmfulness. Current kinds of literature can be divided into three lines.

1) Some works deal with heterophily from the spectral domain. FAGCN [14] divides the message from each edge into low- and high-frequency signals and shows that both the low- and high-frequency signals are necessary for heterophilic graph learning. In addition, some works aim at extracting high-order approximation with graph spectral filters. GPRGNN [19] modifies the convolution to the generalized page rank and learns an arbitrary $K$-order polynomial graph filter. GCNII [31] proposes the initial residual and identity mapping for vanilla GCN and theoretically proves that it can express a $K$-order polynomial filter with arbitrary coefficients. BernNet [32] learns arbitrary graph spectral filters via the Bernstein approximation to oversimplified or ill-posed filters. ACM-GCN [18] modifies GC by explicitly dividing it into low-pass, high-pass, and identity filters in each layer and adaptively fusing them for each node. However, these spectral-based methods need to specifically design and modify the GC, which is not trivial to generalize to broader spatial GNNs like GAT and GIN.

2) Some works reorganize the graph structure to obtain a more homophilic signal. Geom-GCN [17] uses geometric mapping to capture structural similarity in the latent space and long-range dependencies. NLGNN [33] leverages attention-guide sorting to generate a connected graph and conducts nonlocal aggregation. SLAPS [34] combines the self-supervised technique and GAT [37]. The formulations of three well-known GC layers are summarized in Table I.

3) Some works aim to capture high-order neighbor information, which was proven to be homophily-dominant [14]. MixHop [35] repeatedly mixes representations of multihop neighbors to achieve higher order message passing. JK-Nets [36] jumps the intermediate representations to the last layer for better structure-aware representation. H2GCN [7] proposes three designs with separated ego and neighbors, higher order neighbors, and a combination of intermediate representations to combine the message from neighbors.

Unlike these specific GNN architectures to avoid the harmfulness of interclass edges, we consider the interclass edges from an identifiable neighbor distribution perspective. Furthermore, we propose a simple and general Conv-Agnostic framework. This framework can be regarded as a plug-in component and is compatible with most GNNs to improve their performance on heterophilic graphs.

III. PRELIMINARY

A. Problem Setup

Consider an undirected graph $G = (V, E)$ with adjacency matrix $A \in \mathbb{R}^{N \times N}$ and the diagonal degree matrix $D$ of $A$, where $V$ and $E$ are the sets of nodes and edges, respectively. For each node $v_i \in V$, we denote $\mathcal{N}(v_i) = \{j : (i, j) \in E\}$ as its neighbor set. Each node is given an $m$-dimensional feature representation $x_i$, and a $c$-dimensional one-hot class label $y_i$. The feature inputs are then formed by $X = [x_1, \ldots, x_N]$, and the labels are $Y = [y_1, \ldots, y_N]$. Given the labels $Y_L$ of a subset of nodes $L \subset V$, the task of semi-supervised node classification is to predict the labels $Y_U$ of the unlabeled nodes $U = V \setminus L$ by exploiting the graph structure $E$ and the features of all the nodes $X$.

B. Graph Neural Networks

From a probabilistic view, most GNNs assume the local Markov property on node features, i.e., for each node $v_i$, the label $y_i$ only depends on the node self-feature $x_i$ and its neighbor features $x_j : j \in \mathcal{N}(v_i)$. We use superscript $l$ to indicate the layer index. For the $l$th layer of a GNN, we use $h_i^l$ to represent the embedding of node $v_i$ and $h_i^0$ to represent $x_i$ or a projection of $x_i$, for dimension reduction. Then, the general $l$th layer GraphConvolution($A$, $H^{l-1}$) for node $i$ can be formulated as

$$h_i^l = f\left(h_i^{l-1}, \left\{h_j^{l-1} : j \in \mathcal{N}(v_i)\right\}\right)$$

where the GC operator $f$ can be implemented by a weighted sum of each node based on the adjacent matrix $A$ as in GCN [6] and GIN [24] or the attention mechanism in GAT [37]. The formulations of three well-known GC layers are summarized in Table I.

The final output $Z \in \mathbb{R}^{N \times c}$ of the label prediction is evaluated using a softmax function to embed the last layer $H^c$. The objective function is the cross-entropy of the ground-truth labels $Y$ and the output of the network $Z$.

$$O = - \sum_{i \in L} \sum_{j=1}^{c} Y_{ij} \ln Z_{ij}. \quad (2)$$
C. Homophily/Heterophily Metrics on Graphs

The homophily ratio $h$ aims to measure the overall homophily level in a graph. The commonly used node-level [17] and edge-level [7] homophily metrics are usually defined by

$$\mathcal{H}_{\text{node}}(G) = \frac{1}{|V|} \sum_{v \in V} |\{u \in N_v, Y_u = Y_v\}| d_v$$

$$\mathcal{H}_{\text{edge}}(G) = \frac{|\{e_{av} \in E, Y_u = Y_v\}|}{|E|}.$$  

Such metrics measure the proportion of interclass edges in a graph based on label consistency. There is another metric $\mathcal{H}_{\text{agg}}(G)$ to consider the node features similarity from post-aggregation perspective [18]. However, it is limited to the scenario where nodes have features. By definition, the ratio $h \in [0, 1]$, graphs with $h$ closer to 1 tend to have more intraclass edges indicating stronger homophily; on the other hand, graphs with $h$ closer to 0 have more edges connecting different classes, which indicates stronger heterophily.

However, as reported in previous literature [7], [17], [18], these metrics are not significantly relevant to the prediction performance of GCNs. For example, for the well-known heterophily datasets Chameleon and Actor, their homo-ratios are all 0.22. However, on one hand, the reported accuracy of node classification for GCN varies, i.e., 60% and 30%, respectively [7]. On the other hand, the accuracy of the corresponding multilayer perceptrons (MLPs) is 46% and 35%. The opposite performance gap between GCN and MLP in the low homophilic datasets shows that interclass edges can be either beneficial or harmful to classification, which motivates us to consider a new metric beyond the edge perspective to measure heterophily.

IV. REVIEW HETEROPHILY FROM AN ENTIRE NEIGHBOR PERSPECTIVE

A. Measure the Graph-Level Neighbor Effect for Heterophily

In this section, we present a new metric to better understand the heterophily problem and answer whether all interclass edges are harmful. As noted in Fig. 1, when the label distribution of neighbors is random, i.e., every node is connected to other neighbors with random labels, there is no helpful information we can learn from the aggregation step. However, when the neighbor distribution of each class’s nodes forms a certain identifiable distribution, regardless of whether the connected edges are intra- or interclass, GC can extract useful information from this nonrandom neighbor distribution for the downstream tasks. For instance, GC can still achieve perfect performance on a bipartite graph [19], [20]. Therefore, instead of simply calculating the proportion of interclass edges as a measurement for graphs with heterophily, we need to measure the randomness/identifiability of the entire neighbor distribution.

We define the identifiability of neighbors as the information of the nodes’ neighbor distribution, which can be seen as the graph-level neighbor effect for heterophilic datasets. As Fig. 2 shows, to measure the identifiability of neighbors of each class, we group the nodes by class and form a class-level neighbor’s label distribution matrix $A^k_N \in \mathbb{R}^{n_k \times C}$ for each class $k$, where $k = 1, \ldots, C$ for different classes and $n_k$ indicates the number of nodes with label $k$. Then, our task is to evaluate the information of the neighbors’ label distribution matrix to quantify the identifiability of neighbors. Inspired by the von Neumann entropy in quantum statistical mechanics [21], which measures the purity/information of a quantum-mechanical system by calculating the entropy of the eigenvalue distribution of a positive definite symmetric density matrix, we generalize this idea to our task of evaluating neighbors’ identifiability. Specifically, since the neighbor distribution matrix $A^k_N$ is not symmetric, we consider the entropy of the singular values’ distribution rather than the eigenvalues as an indicator of identifiability. This can be understood as a measurement of how many vectors (patterns of neighbors) are needed for an adequate explanation of the neighbors’ label distribution matrix, indicating the richness/randomness of the neighbor distribution.

Suppose $\sigma^k_1, \sigma^k_2, \ldots, \sigma^k_{n_k}$ denotes singular values of $A^k_N$, we then normalize them so that $\sum_{i=1}^{C} \sigma^k_i = 1$, where $i = 1, \ldots, C$ for index of singular values. Then we calculate the entropy of class $K$ by

$$\mathcal{H}_{\text{neigh}}^k = -\frac{1}{\log(C)} \sum_{i=1}^{n_k} \sigma^k_i \log(\sigma^k_i).$$

The above metric ranges from $[0, 1]$ and can be used to quantify the identifiability of neighbors for a specific class (the lower, the more identifiable). Considering the problem of class imbalance, we compute the weighted sum of class-level entropy to evaluate the neighbors’ identifiability of a graph

$$\mathcal{H}_{\text{neigh}}(G) = \sum_{k=1}^{C} \frac{n_k}{N} \mathcal{H}_{\text{neigh}}^k.$$  

Our metric sheds new light on understanding the heterophily problem from an identifiable neighbor perspective. As shown in Fig. 3, compared with the node/edge-level homophily metrics, our measurement and the GCNs’ performance for different datasets are more monotonous. Specifically, we can observe that the Actor, Chameleon, and Squirrel datasets have similar node/edge-level homophily metrics (proportion...
of interclass edges) in the first and second parts of Fig. 3. However, the performance of these datasets is inconsistent, especially for the Actor dataset, which has the poorest performance (accuracy ≈ 30%) and is totally different from the others. As we can see in the third part of Fig. 3, \( H_{\text{agg}}(\mathcal{G}) \) still fails to distinguish the Actor from others. In contrast, as shown in the right of Fig. 3, our metric can distinguish the Actor dataset from others since it has a nearly random neighbor distribution (\( H_{\text{neighbor}} = 0.98 \)). This indicates that the proposed metric can better explain the difference in model performance from a neighbor perspective. Furthermore, our metric reveals that the interclass edges are not always harmful for node classification during aggregation, and the entire local neighbor perspective can provide more information.

**B. Improve GNNs via Learning Node-Level Neighbor Effect**

Unlike other specific heterophilic-oriented GNNs that need to modify GC kernels, our neighbor perspective motivates us to learn the effectiveness of each node’s local neighbors during aggregation to help traditional GNNs deal with heterophily in a general way. Most GNNs straightforwardly feed the current aggregation features to the following GC layer and adopt the last layer representation for the downstream node classification task [6], [37]. However, the entanglement of aggregation and classification may lead to over-smoothing of node representations due to interclass neighbors, resulting in a loss of discrimination in heterophilic graphs, despite the fact that neighbor information may be useful for downstream node classification. To adaptively combine neighbor information and enhance traditional GNNs, we propose decoupling discriminant representations of nodes from the aggregation. Then, it allows us to guide the aggregation and generate suitable representations of nodes for classification by learning the node-level neighbor effect.

However, to guide the aggregation during the training process, we cannot directly use the entropy measurement and need to evaluate the node-level neighbor effect in another way. The reasons are twofold: 1) similar to \( H_{\text{edge}} \) and \( H_{\text{node}} \), the computation of the entropy \( H_{\text{neighbor}} \) also requires the labels of all the nodes, which are unavailable in the training process; and 2) the class-level neighbor distribution identifiability \( H_{\text{neighbor}}^{c} \) is not consistent with the entropy of the node-level label distribution. Namely, the entropy of a node does not represent the identifiability of the neighbor distribution of this class. In Section V, we will elaborate on how to adaptively learn the node-level neighbor effect from the downstream supervision signal and the features of a node with its neighbor.

**V. PROPOSED METHOD**

In this section, we propose the CAGNNs to improve traditional GNNs’ performance by adaptively learning the node-level neighbor effect. Then, we provide a spectral analysis to show the expressive power of the proposed framework on the node classification task. Finally, we describe the difference between our framework and the GNNs with decoupling design and skip connection.

**A. Conv-Agnostic GNN Framework**

The proposed CAGNNs aims to empower traditional GNNs to generate suitable representations for each node for both homophilic and heterophilic graphs. The core idea is to treat the node-local neighbor as an entirety and determine its efficacy during aggregation. To this end, we first decouple the representation of nodes into discrimination and aggregation. Then, we learn a mixer module that can adaptively evaluate each node’s neighbor effect based on these two representations and determine whether to incorporate the information from neighbors. As shown in Fig. 4, our framework is composed of four major components: Encoder, GC, Mixer, and Decoder. Below, we elaborate on each component in order. Moreover, we append the normalization operation after the Encoder, GC, and Mixer to maintain the numerical stability, which will be discussed later.

1) Encoder: We use a linear layer as the encoder to transform the node features \( X \). Then, we feed it into two streams. One is the node’s own feature \( S_0 \in \mathbb{R}^{N \times d} \) for downstream task discrimination, and the other is the aggregation feature \( H^0 \in \mathbb{R}^{N \times d} \) for GC. These two streams decouple the discriminant feature and the information from neighbors, which may prevent representations of nodes for classification from being over-smooth by their neighbors during aggregation. Moreover, this decoupling operation allows each node to evaluate the neighbor effect for downstream discrimination tasks via the mixer module in the following aggregation step:

\[
H^0 = S^0 = \text{Norm}(f_{\text{encoder}}(X)).
\]
2) **Graph Convolution:** Since our framework is Conv-Agnostic, in this part, any standard GC layers (e.g., GCN, GAT, and GIN) can be applied to aggregate each node’s neighborhood information to update the aggregation representation $H$

$$H^{l+1} = \text{Norm}(\text{GraphConvolution}(A, H^l)).$$

(8)

Moreover, this GC layer can be stacked multiple times to enhance the receptive field of each node by considering the information of more neighbors. The embedding $H^l$ can also be regarded as the $l$-hop neighbors’ information. However, most GNNs ignore the fact that the information from neighbors is not always beneficial for the classification task of each node. Hence, we propose the following mixer operator to determine the neighbor effect of each hop.

3) **Mixer:** From the view of node $v_i$ at layer $l$, it needs to combine the discrimination feature $S^l$ and $l$-hop neighbors’ feature $H_i^l$ according to the neighbor effect to update the representative embedding for the downstream task. Therefore, the goal of the mixer function is to evaluate the neighbor effect of each node and then to selectively incorporate the neighbors’ information. As discussed in Section IV, we do not need to inject the entropy function to evaluate the node-level neighbor effect. We have already encoded the neighbor information of each node into $H^l$ by the GC. According to the universal approximation theorem [38],[39], we hypothesize that the MLP can adaptively learn the node-level neighbor effect based on $S^l \in \mathbb{R}^{N \times d}$ and $H^l \in \mathbb{R}^{N \times d}$ and the downstream objective $O$. For simplicity, we implement the $\alpha^l$ by

$$\alpha^l = \sigma(f_{\text{mixer}}(S^l \| H^l))$$

(9)

$$S^l = \text{Norm}((1 - \alpha^l) * S^l + \alpha^l * H^l)$$

(10)

where $||$ is the concat operator and the function $f_{\text{mixer}}$ is a linear layer $\mathbb{R}^{N \times d} \rightarrow \mathbb{R}^{N \times 1}$. It maps the discrimination feature $S^l$ and neighbor feature $H_i^l$ at layer $l$ to a vector $\alpha^l \in \mathbb{R}^{N \times 1}$. With the sigmoid function $\sigma$, each element of $\alpha^l$ is normalized to an importance score ranging from 0 to 1 and can be regarded as the neighbor effect of the node. Based on the importance score $\alpha^l$, we use the convex combination of discriminant feature $S^l$ and neighbor information $H_i^l$ at each layer to adaptively update the discriminant feature $S^l$. This convex combination strategy has the advantages of numerical stability and interpretability and has been widely used in modern deep learning, such as the highway network [40] and attention mechanism [41].

Note that the parameters of all $f_{\text{mixer}}$ functions are shared across layers to learn the neighbor effect of each node. Such a parameter-efficient sharing mechanism can help reduce overfitting and improve generalization, similar to the gating function in long short-term memory [42]. Moreover, the mixer function maintains the expressive power of the chosen GC, since it can easily degenerate to the normal GC when $\alpha^l = 1$.

As shown in the ablation study, our implementation of this mixer function is very effective with minimal parameter cost.

4) **Decoder:** With the last layer discriminant $S^L$ at hand, the task of the decoder is to produce the final prediction $Z$ for classification. For simplicity, we use the linear layer with softmax operator as our decoder $f_{\text{decoder}}$

$$Z = \text{softmax}(f_{\text{decoder}}(S^L)).$$

(11)

Note that we also apply the Norm layer to maintain numerical stability at the end of the Encoder, GC, and Mixer modules. Specifically, we use L2 normalization for each node. Compared with the widely used BatchNorm and LayerNorm, we experimentally show that the parameter-free L2 normalization can achieve better performance for our task.

In summary, unlike other GNNs that explicitly modify graph filters or GC kernels for heterophily [14],[18],[19], our node-level mixer is a convolution-agnostic technique. We divide the representation of the node into two parts: node-self for discrimination and neighbor information from arbitrary GC. Hence, the original GC operator may not directly interact with the node’s discrimination feature and over-smooth its representation. Moreover, we use only one shared-parameterized layer to adaptively learn node-level neighbor effect and mix these two representations, which can be seen as adding a plug-in side layer to standard GNNs for generating more discriminative node representations. Therefore, our method is compatible with most GC layers (GCN, GAT, GIN, etc.) and enhances their performance in a more general and parameter-efficient way.

B. **Spectral Analysis of CAGNN**

We then study the expressive power of CAGNNs from a spectral perspective. Recall the $K$-order polynomial graph filters with graph signal $X$ and propagation matrix $P$ as $(\sum_{k=0}^{K} \theta^P)X$, where $P = D^{-1/2}AD^{-1/2}$ is the normalized adjacency matrix, and the scalar $\theta^P$ are the polynomial coefficients. Note that using such a polynomial graph filter can derive either high- or low-pass filters [32],[43], maintaining the model’s ability to deal with various label-connected patterns, which is essential to prevent over-smoothing [31] and to learn from heterophilic graphs [19]. However, most spectral GNNs learn the shared $K$-order polynomial scalar coefficients $\theta^P$’s for all the nodes, which may limit the expressive power. We demonstrate that CAGNNs corresponds to a polynomial graph filter with different coefficients for each node on the graph spectral domain.

**Theorem 1:** Considering the propagation matrix $P$ used for the basic GC layer and a graph signal $X$, a $K$-layer CAGNNs has the ability to express a $K$-order polynomial filter $(\sum_{k=0}^{K} \theta^P)X$ with different coefficients $\theta^P$ for each node.

The above theorem indicates the expressive power on node classification of our framework. Intuitively, the importance score $\alpha^l$ evaluated by the Mixer allows CAGNNs to simulate different coefficient $\theta^P$ of the polynomial graph filter for each node. Note that with a proper choice of $\theta^P$, the discriminant feature $S^K$ of each node can carry information from both the input feature and the high-order neighbor’s information adaptively with the increment of the order $K$. Compared with other spectral GNNs that learn shared polynomial filters $(\sum_{k=0}^{K} \theta^P)X$ for all nodes [19],[23],[31], our models can empower each node with distinct polynomial coefficients by vectorized $\theta^P$. This property can capture the nodes’ distinct
complex connected patterns; as in Section VI-B, our framework achieves higher performance than other spectral GNNs experimentally. The detailed proof is presented below.

Proof: For simplicity, we neglect the L2 normalization at each layer because the simplified version also produces comparable performance. Moreover, we assume the input feature matrix $X$ to be nonnegative and remove the nonnegative rectified linear unit (ReLU) activation in the GC layer [31], [44]. Then, for the simplified GC layer, we have

$$H' = PH^{-1}W^l$$

$$= P^lH^0 \prod_{j=0}^l W^j$$

$$= P^lX\tilde{W}^l,$$ where $\tilde{W}^l = \prod_{j=0}^l W^j$.  

(14)

Furthermore, we can express the $K$-layer representation as

$$S^K = (1 - \alpha^K) \ast S^{K-1} + \alpha^K \ast H^K$$

$$= \sum_{l=0}^{K} \alpha^l \prod_{k=l+1}^K (1 - \alpha^k) \ast H^l$$

$$= \sum_{l=0}^{K} \theta^l \ast P^lX\tilde{W}^l,$$ where $\theta^l = \alpha^l \prod_{k=l+1}^K (1 - \alpha^k)$.  

(17)

However, $\theta^l \in [0, 1]^{N \times 1}$ which limits the expressive power of the polynomial filter. Thanks to the weight matrix $\tilde{W}^l$, the coefficients of the polynomial can be extended to arbitrary values. Inspired by [31], we consider a weaker version of CAGNNs by fixing the weight matrix $W^l$ to be $\gamma^l$, where $\gamma^l$ is a learnable parameter. We have

$$S^K = \sum_{l=0}^{K} \theta^l \ast P^lX\gamma^l$$

$$= \sum_{l=0}^{K} \theta^l \ast P^lX,$$ where $\theta^l = \theta^l \gamma^l$.  

(19)

The polynomial coefficient $\theta^l$ for each layer $l$ can be set to desired values with the help of the scalable parameter $\gamma^l$, which concludes the proof.

C. Relationship With Other Decouple or Skip Connection GNNs

In this section, we discuss the relationship of our two techniques (decoupled operation and mixer module) in the framework with other models.

1) Relationship With GNNs With Decouple Design: We remark that the decoupling operation, which we study in this work, is a distinct concept from other decoupling GNNs. Well-known decoupled GNNs, such as APPNP [25], S$^2$GC [45], and GPRGNN [19], aim to decouple the propagation and transformation of GC. Without the loss of generalizability, they can be formulated as follows:

$$H^K = \left( \sum_{l=0}^{K} \theta^l P^l \right)^{\gamma^l} \tilde{H} \tilde{W}$$

(20)

where $\theta^l$ is a scalar and $P$ is the propagation matrix. This operation makes the node have the ability to receive high-order neighbor information in one layer. In addition, the learnable parameter $\theta^l$ ensures that it can learn the arbitrary coefficient of polynomial graph filters beyond the low-pass filters and perform well in heterophily [19]. However, it needs to decouple the weight matrix $W$ to reformulate the GC operator, which restricts the adaptation to other GCs.

In contrast, our approach aims to decouple the discriminant feature and the aggregation feature, which can be easily compatible with most standard GNNs. Moreover, we theoretically prove that our framework can learn different arbitrary coefficients of $K$-order polynomial graph filters for each node and achieve higher performance experimentally.

2) Relationship With GNNs With Skip Connection: To better understand the mixer module, we compare it with respect to the DeepGNNs with skip connections. Note that we aim at the heterophily problem, and our approach is different from the DeepGNNs designed to reduce the over-smoothing problem [4]. To alleviate the over-smoothing problem when the model becomes deeper, most DeepGNNs equip the residual connection or initial connection to combine previous layers’ features to prevent forgetting the original feature when models become deeper [31], [46]. The standard DeepGNN with skip connection can be formulated as follows:

$$H' = \sigma (P^lX\tilde{W}^l + \{H^{-1}/H^l\}) \text{(residual/init connection)}.$$  

(21)

However, the DeepGNNs with skip connections are not designed for heterophilic graphs. Most skip connections combine previous features and cannot adaptively aggregate information from neighbors for each node. For instance, the recent well-known SOTA of this type model is GCNNI [31], where

$$H' = \sigma \left( (1 - \alpha^l)P^lX + \alpha^lH^0 \right) \left( (1 - \beta^l)P^lX + \beta^l\tilde{W}^l \right).$$  

(22)

The scalar $\alpha^l$ in GCNNI is a manually chosen hyperparameter that controls the strength of initial connection. However, since $\alpha^l$ is shared for all the nodes and not learnable, it is not the best choice to guide how to aggregate neighbors’ information on heterophilic datasets.

In contrast, as shown in (9), we use the mixer function to learn the importance score $\alpha$ to explicitly evaluate the node-level neighbor effect for feature fusion. Moreover, compared with the skip connection, we separate the discriminant features $S$ of nodes at each layer and do not feed them into the next layer GC component.

VI. EXPERIMENTS

In this section, we report and compare the results for node classification on both the real-world heterophily and homophily datasets to investigate the effectiveness, robustness,
and interpretability of the proposed heterophily GNN framework CAGNNs. We also show the importance of the neighbor effect from the relationship between the model performance and metrics.

A. Experimental Setup

1) Datasets: We evaluate the performance on nine well-known real-world node classification datasets, including three homophily datasets (Citeseer, Pubmed, and Cora) and six heterophily datasets (Texas, Wisconsin, Actor, Squirrel, Chameleon, and Cornell). For all the benchmarks, we use the same feature vectors, graph structure, class labels, and standard ten fixed random splits (48%/32%/20% of nodes per class for train/validation/test) provided in literature [7], [17].

1) Homophily Datasets

a) Citeseer, Pubmed, Cora [6]: For the basic citation datasets [47], nodes correspond to papers; edges correspond to citation links; the sparse bag-of-words are the feature representation of each node. Finally, the label of each node represents the topic of the article.

2) Heterophily Datasets

a) Texas, Wisconsin, Cornell [17]: Cornell, Texas, and Wisconsin are the web page networks captured from the computer science departments of these universities in the WebKB dataset. In these networks, nodes and edges represent the web pages and hyperlinks, respectively. Similar to the citations networks, words in the web page represent the node features in the bag-of-word form. The web pages are labeled into five categories: student, project, course, staff, and faculty.

b) Squirrel, Chameleon [17]: Chameleon and Squirrel are web pages extracted from different topics in Wikipedia [48]. Similar to WebKB, nodes and edges denote the web pages and hyperlinks among them, respectively, and informative nouns in the web pages are used to construct the node features in the bag-of-word form. Webpages are labeled in terms of the average monthly traffic level.

c) Actor [17]: The actor network contains the co-occurrences of actors in films, which are extracted from the heterogeneous information networks. It describes the complex relationships among films, directors, actors, and writers [49]. In this network, nodes and edges represent actors and their co-occurrences in films, respectively. The actor’s Wikipedia page is exploited to extract features and node labels.

2) Baselines: We compare our method with the following baselines: 1) classical standard GNNs: GCN [6], GAT [37], and GIN [24]; 2) recent state-of-the-art GNNs of specific structure tackling heterophily: Geom-GCN [17], MixHop [35], H2GCN [7], GPRGNN [19], FAGCN [14], GCN-Cheby [23], JK-Net [36], and GCNII [31]; and 3) standard two-layer MLP.

To show the effectiveness and generalizability of our framework, we choose three simple GCN, GAT, and GIN as the GC component in CAGNNs. For ease of comparison, we use the reported results of baselines in the literature [7], [17]. Moreover, for the missing results under these splits, we rerun the released code ten times and report the mean and standard deviation.

1) Classical GNNs

a) GCN [6]: GCN can be seen as a Laplacian smoother since it uses the mean aggregator to smooth each node and its neighbor’s features.

b) GIN [24]: GIN uses the MLP to model the injective function when aggregation and generalizes the WL test.

c) GAT [37]: GAT is a graph neural network that applies the attention mechanism on node features to learn edge weights for aggregation.

2) Heterophily GNNs

a) H2GCN [14]: H2GCN proposed three designs with separate ego and neighbors, higher order neighbors, and a combination of intermediate representations to combine the message from neighbors.

b) FAGCN [14]: FAGCN divides the message from each edge into low- and high-frequency signals during aggregation.

c) GCN-Cheby [23]: GCN-Cheby combines higher order neighbor information with the Chebyshev polynomials from the spectral domain.

d) GEOM-GCN [17]: GEOM-GCN uses structural similarity to capture the smooth structure in the latent space and long-range dependencies.

e) MixHop [35]: MixHop repeatedly mixes feature representations of neighbors at various distances to achieve higher order message passing.

f) GPRGNN [19]: GPRGNN modifies the convolution to the generalized page rank and learned an arbitrary polynomial graph filter to incorporate multiscale information.

g) JK-Net [36]: JK-Net combines intermediate node representations from each layer by concatenating them in the final layer.

h) GCNII [31]: The state-of-the-art deep model combines initial connections and identity mapping to train a very deep GCN.

3) Hyperparameters: We adopt the same set of default hyperparameters (two layers and 64 hidden dimensions) for GCN, GAT, and GIN and the corresponding CAGNNs, which is the most widely used hyperparameters’ setting as [14], [31], and [37]. For CAGNNs, we add only one linear layer with 128 hidden units. We use the Adam optimizer and select the learning rate $\in \{0.001, 0.01, 0.05\}$, weight decay $\in \{0.0005, 0.0005\}$, and dropout rate $\in \{0.0, 0.5\}$ based on the validation sets. For other models, we use their best default parameters in the original articles.

B. Performance Comparison With SOTA

We report and compare the performance for the standard node classification task in Table II. For the classic GNNs (GCN, GAT, and GIN), we first note that they outperform the
MLP in the homophily datasets. It indicates that the homophily assumption and connected intraclass neighbors provide helpful node classification information. However, we also note that they sometimes also perform better than the MLP under the heterophily datasets. For instance, in the Actor dataset, the performance of MLP is approximately 36%, but the traditional GCN only obtains 30%. However, in the Chameleon dataset, the performance comparison between GCN and MLP is (60% versus 46%). Hence, it implies that not all the interclass edges in the heterophily datasets are harmful to node classification.

In addition, we note that the GNNs with specific designs for the heterophily datasets outperform the traditional GNNs with a large margin for six heterophily datasets. Most of them (e.g., MixHop, JK-Net, and H2GCN) explicitly aggregate high-order neighbors’ information to avoid the harmfulness of interclass edges. For instance, the strong baseline H2GCN proves that the neighbor’s high-order information is expectedly homophily-dominant and achieves 70.87% average performance. In comparison, the average performance of the two-layer GCN is only 61.62%.

In contrast, instead of considering that the interclass edges are all harmful, we take each node’s entire neighbor effect into account. As a result, compared with recent state-of-the-art heterophily GNNs, our CAGNNs can help traditional standard GNNs achieve competitive results while maintaining the performance on three homophily datasets. Moreover, from the perspective of the spectral domain, CAGNN can help traditional GNNs adaptively learn different coefficients of K-order polynomial graph filters for each node. Therefore, our framework performs better than the spectral GNNs sharing the same polynomial filter coefficients for all the nodes (e.g., GPRGNN and GCNII). Under our framework, the average performance on nine datasets of GIN, GAT, and GCN outperforms all the baselines, and the average performance gains are 9.81%, 25.81%, and 20.61%, respectively. Among them, the proposed CAGNNs with two-layer GCN achieves the best average performance (74.32%) over all the datasets. It verifies the effectiveness of decoupling design and consideration of the neighbor effect when performing GC.

C. Relationship Between the Metrics and Performance

Table II also shows the different metrics for all the datasets. All the metrics range from [0, 1] and a higher score of $H_{\text{node}}$, $H_{\text{edge}}$, and $H_{\text{agg}}$ denotes higher homophily. However, a higher $H_{\text{neighbor}}$ means lower identifiability of neighbors’ distribution and a more challenging dataset, i.e., the neighbor distribution provides less useful information for classification. Neighbor distribution identifiability offers an alternative perspective to the interclass edges’ approach ($H_{\text{node}}$ and $H_{\text{edge}}$) for understanding the heterophily problem in GNNs. Note that only our $H_{\text{neighbor}}$ can distinguish the dataset Actor (0.98) from others, in which the neighbors’ distribution is nearly random and the best test classification accuracy is very low, i.e., 35.86. Our metric shows that the GNN does not outperform MLP (35.73)

| Nodes | Texas | Wisconsin | Actor | Squirrel | Chameleon | Cornell | Citeseer | Pubmed | Cora | Average |
|-------|-------|-----------|-------|----------|-----------|---------|---------|--------|------|---------|
| Edges | 1,703 | 1,703 | 931 | 2,089 | 3,252 | 1,703 | 3,703 | 500 | 1,433 | - |
| Features | 5 | 5 | 5 | 5 | 5 | 7 | 3 | 6 | - | - |

$H_{\text{node}}$ | 0.06 | 0.16 | 0.24 | 0.22 | 0.22 | 0.23 | 0.30 | 0.74 | 0.80 | 0.81 |

$H_{\text{edge}}$ | 0.09 | 0.78 | 0.68 | 0.36 | 0.61 | 0.80 | 0.98 | 0.94 | 0.99 | - |

$H_{\text{agg}}$ | 0.45 | 0.98 | 0.92 | 0.55 | 0.87 | 0.85 | 0.72 | - | - | - |

$H_{\text{neighbor}}$ | 67.57 | 64.12 | 31.63 | 38.14 | 60.90 | 68.10 | 77.99 | 90.05 | 85.27 | 64.05 |

| MLN* | 81.89 | 78.49 | 78.29 | 65.19 | 56.18 | 75.94 | 70.47 | 70.68 | 89.64 | 49.19 |

| GIN | 58.32 | 59.25 | 59.32 | 59.32 | 59.32 | 59.32 | 59.32 | 59.32 | 59.32 | 59.32 |

| GAT | 63.06 | 64.78 | 65.77 | 66.77 | 67.77 | 68.77 | 69.77 | 70.77 | 71.77 | 72.77 |

| GCN | 59.46 | 60.36 | 61.26 | 62.16 | 63.06 | 64.06 | 65.06 | 66.06 | 67.06 | 68.06 |

| GIN | 85.13 | 85.13 | 85.13 | 85.13 | 85.13 | 85.13 | 85.13 | 85.13 | 85.13 | 85.13 |

| GAT | 63.17 | 64.07 | 65.08 | 66.09 | 67.10 | 68.11 | 69.12 | 70.13 | 71.14 | 72.15 |

| GCN | 63.17 | 64.07 | 65.08 | 66.09 | 67.10 | 68.11 | 69.12 | 70.13 | 71.14 | 72.15 |

**Table II**

**Performance Comparison on Various Real-World Heterophily and Homophily Datasets. Mean Test Accuracy and Standard Deviation Are Reported Over Ten Random Data Splits. The Best Performance Is Highlighted.

*”* Denotes the Results Obtained From [7]
TABLE III
KENDALL CORRELATION BETWEEN DIFFERENT METRICS AND THE PERFORMANCE OF CAGNN\textsubscript{GCN}. THE HIGHER COEFFICIENT AND THE LOWER p-VALUE ARE BETTER AND MORE SIGNIFICANT

| Datasets | Kendall | H\textsubscript{node} | H\textsubscript{edge} | H\textsubscript{agg} | H\textsubscript{neighbor} |
|----------|---------|----------------|----------------|----------------|---------------------|
| >500 nodes | coefficient | 0.733 | 0.828 | 0.467 | 0.867 |
| | p-value | 0.056 | 0.022 | 0.27 | 0.017 |
| All datasets | coefficient | 0.11 | 0.25 | 0.44 | 0.59 |
| | p-value | 0.7 | 0.34 | 0.12 | 0.02 |

Fig. 5. Class-level comparison between our CAGNN\textsubscript{GCN} performance (red) and the negative class-level H\textsubscript{neighbor} (blue) for homophily (Cora, Citeseer, and Pubmed) and heterophily (Chameleon, Squirrel, and Actor). The class-level performance of CAGNN\textsubscript{GCN} is highly consistent with the neighbor distribution identifiability for most datasets. (a) Cora. (b) Chameleon. (c) Citeseer. (d) Squirrel. (e) Pubmed. (f) Actor.

D. Ablation Study

In this section, we compare CAGNN with its variants for mixer and normalization to validate the effectiveness of each component. When testing different mixer variants, we fix the normalization to L2. Also, we set the mixer to linear to test different normalization variants. We select the best hyperparameters of each variant and run experiments for each dataset under ten random splits to report the average performance on all the datasets.

1) Variants of Mixer: The mixer module is a critical part of our framework to evaluate the neighbor effect and feature fusion. We compared our results with six variants.

1) Add: The Add mixer is implemented by $S\textsubscript{l} = \text{Norm}(S\textsubscript{l} - 1 + H\textsubscript{l})$.

2) Concat: The Concat mixer is implemented by $S\textsubscript{l} = \text{Norm}([S\textsubscript{l} - 1 || H\textsubscript{l}])$. Note that we only apply the Norm at the first and last layers for the Concat variant to maintain stability.

3) Global: The Global mixer is $S\textsubscript{l} = \text{Norm}(1 - \alpha l S\textsubscript{l} - 1 + \alpha l H\textsubscript{l})$, where the learnable scalar $\alpha l$ is shared for all the nodes.

4) Unshared: It replaces the shared-parameterized f\textsubscript{mixer} to unshared version for each layer.

5) MLP-2/3: It replaces the linear layer in f\textsubscript{mixer} with two or three layers of shared-parameterized MLP.

From the results of these mixer variants in Table IV, we have the following observations.

1) All these variants of mixer can substantially improve the performance of standard GNNs, which verifies the effectiveness of the decoupled design.

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
2) The results of the global mixer are better than those of the Add and Concat mixer, indicating that the neighbor effect is diverse under different datasets.

3) Our one-layer mixer and multilayer MLP variants achieve consistently better performance than the global model. These mixers can adaptively learn each node’s neighbor effect, which verifies that the learnable node-level neighbor effect plays a more vital role than the global mixer.

4) Compared with the unshared-parameterized version and the multiple layer MLPs in the current \( f_{\text{mixer}} \) module, the result shows that the current shared-parameterized one-layer mixer is simple yet effective.

2) Variants of Normalization: We also compare the L2 norm for each node with three variants (None, BatchNorm [50], and LayerNorm [51]) to show its effectiveness. We first observe that the None normalization version achieves comparable performance, demonstrating that our framework is stable and competitive. However, the BatchNorm usually assumes the independent and identical distribution of each sample in deep learning, which may not be reasonable for the heterophilic graph and results in worse performance. Moreover, LayerNorm is also not beneficial since the layer norm introduces learning parameters that may be redundant for GNNs. Compared with these variants, our simple yet effective L2 norm achieves the best average performance.

E. Robustness Analysis

To investigate whether the proposed CAGNN framework can help basic GNNs become more robust, we report the performance comparison between the basic GNNs and the corresponding CAGNN in the over-smoothing and noisy edges’ scenario.

1) Alleviating Oversmooth: It is well-known that the traditional GC is sensitive to the number of convolution layers due to the over-smoothing problem [4]. Our CAGNNs is able to increase the robustness of traditional GC to avoid over-smoothing. As shown in Fig. 6(a), when the number of layers increases, the performance of traditional GNNs drops rapidly due to over-smoothing. Moreover, due to the massive inter-class edges, the over-smoothing phenomenon occurs earlier in the heterophilic datasets. In contrast, the methods under the proposed framework are more stable and more consistent on both the homophilic and heterophilic datasets. The reason is that our framework has the ability to avoid incorporating the over-smoothing features to maintain the discrimination power for each node.

2) Alleviating Noisy Edges: Most GNNs are also sensitive to the noisy edges in graphs [52], [53]. To evaluate the robustness of the proposed framework on noisy graphs, we construct graphs with random edge addition according to the literature [54]. Specifically, we randomly add 25%~500% edges in the original graphs. As shown in Fig. 6(b), our CAGNNs achieves significantly better prediction for noisy graphs compared with the basic GNNs. It also demonstrates that our decouple design and the mixer module are able to learn to discard the noisy features from neighbors.

F. Visualization and Interpretability

To verify whether CAGNNs can adaptively learn the different neighbor effect of each node, we visualize the neighbor importance score \( \alpha \) distribution of each layers on both the homophily and heterophily datasets, where the results are shown in Fig. 7. We have the following observation.

1) As we can see, for the homophily datasets (Cora, Citeseer, and Pubmed), the coefficients are near 0.4 for most nodes. This indicates that the information from neighbors is helpful for downstream classification, which is consistent with the homophily assumption. item However, we observe similar trends in the heterophily datasets (Chameleon and Squirrel) that most nodes still absorb the neighbors’ information. Moreover, the distribution of the second layer score shows that one-hop neighbors are more important than two-hop neighbors. This phenomenon is contrary to the previous studies that the interclass edges are all harmful, which implies that GC can still extract classification information from an inte-class neighbor with a nonrandom distribution. item The Actor dataset has a similar proportion of interclass edges with Chameleon and Squirrel, but the neighbor distribution is nearly random \( H_{\text{neighbor}} \) is 0.98. The neighbor importance score of nodes tends to be 0, which verifies that the interclass edges with random distribution are harmful and guide the model to discard neighbors’ information when aggregation.

G. Efficiency

Finally, we investigate the complexity of the proposed framework. The complexity of computing the neighbor importance score and mixing for all the nodes is \( O(Nd) \) where \( N \)
that interclass edges can be helpful when the neighborhood distribution is identifiable. Furthermore, instead of computing node/edge-level metrics for a graph with heterophilic properties, we proposed a new measurement from the entire neighbor-level perspective via the von Neumann entropy. The proposed new metric sheds light on the heterophilic problem. It enables us to explain the performance variation in GNNs for different datasets and can be used to guide the application of GNNs.

We also proposed a simple yet effective heterophilic GNN framework, CAGNN, which adds just one mixer layer to enhance the performance of conventional GNNs. The node features are first decoupled into discrimination and aggregation parts, and then adaptively fused according to the neighborhood effect of each node. Our experiments on nine well-known benchmark datasets not only demonstrate the effectiveness of the proposed framework as a plug-in to consistently improve the existing GNNs on heterophilic graphs but also indicate good interpretability for determining whether neighbor information is helpful for the downstream node classification task.

Similar to other node- and edge-level metrics, our metrics are constrained by the requirement for node label information. Therefore, it is nontrivial to explicitly combine these metrics to guide the aggregation process during training without label information. One of our future works is to incorporate the pseudolabel with these metrics to improve the aggregation of GNNs. Besides, beyond node classification, exploiting the connection between neighbor identifiability with the edge or graph-level task is also an interesting topic for future work.

VII. CONCLUSION

In this article, we have investigated the neighbor effect on the heterophilic datasets. Unlike previous works that argue that interclass edges are harmful for node classification, we find

is the total number of nodes, and $d$ is the dimension of the hidden nodes. The computational complexity is on par with the neighborhood aggregation operation in GNNs, which is also $O(Nd)$. Hence, the complexity of our method and classic GNNs is on the same level. We also report the average training time of each epoch over all the datasets of the standard GC and our method in Fig. 8. It indicates that our model scales similar to the basic GNNs with a small computation cost for the mixer module.

REFERENCES

[1] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, “A comprehensive survey on graph neural networks,” IEEE Trans. Neural Netw. Learn. Syst., vol. 32, no. 1, pp. 4–24, Jan. 2020.
[2] W. L. Hamilton, R. Ying, and J. Leskovec, “Inductive representation learning on large graphs,” in Proc. Adv. Neural Inf. Process. Syst., 2017, pp. 1025–1035.
[3] D. Bacciu, F. Errica, A. Micheli, and M. Podda, “A gentle introduction to deep learning for graphs,” Neural Netw., vol. 129, pp. 203–221, Sep. 2020.
[4] Q. Li, Z. Han, and X.-M. Wu, “Deeper insights into graph convolutional networks for semi-supervised learning,” in Proc. AAAI Conf. Artif. Intell., 2018, vol. 33, no. 1, pp. 3538–3545.
[5] N. T. Hoang and T. Maehara, “Revisiting graph neural networks: All we have is low-pass filters,” 2019, arXiv:1905.09550.
[6] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” in Proc. Int. Conf. Learn. Represent., 2017, pp. 1–14.
[7] J. Zhu, Y. Yao, L. Zhao, M. Heimann, L. Akoglu, and D. Koutra, “Beyond homophily in graph neural networks: Current limitations and effective designs,” in Proc. Adv. Neural Inf. Process. Syst., vol. 33, 2020, pp. 7793–7804.
[8] X. Guo, L. Wu, and L. Zhao, “Deep graph translation,” IEEE Trans. Neural Netw. Learn. Syst., early access, Mar. 17, 2022, doi: 10.1109/TNNLS.2022.314670.
[9] G. Bi, Y. Zheng, W. Wang, Y. Cai, and H. Wu, “Graph polish: A novel graph generation paradigm for molecular optimization,” IEEE Trans. Neural Netw. Learn. Syst., early access, Sep. 14, 2021, doi: 10.1109/TNNLS.2021.3106392.
[10] D. Bacciu and D. Numeroso, “Explaining deep graph networks via input perturbation,” IEEE Trans. Neural Netw. Learn. Syst., early access, Apr. 21, 2022, doi: 10.1109/TNNLS.2022.3165618.
[11] W. Liu et al., “Item relationship graph neural networks for e-commerce,” IEEE Trans. Neural Netw. Learn. Syst., vol. 33, no. 9, pp. 4785–4799, Sep. 2022.
[12] X. Zheng, Y. Liu, S. Pan, M. Zhang, D. Jin, and P. S. Yu, “Graph neural networks for graphs with heterophily: A survey,” 2022, arXiv:2202.07082.

[13] Y. Yan, M. Hashemi, K. Sversky, Y. Yang, and D. Koutra, “Two sides of the same coin: Heterophily and oversmoothing in graph convolutional neural networks,” in Proc. IEEE Int. Conf. Data Mining (ICDM), Nov. 2022, pp. 1287–1292.

[14] D. Bo, X. Wang, C. Shi, and H. Shen, “Beyond low-frequency information in graph convolutional networks,” in Proc. AAAI Conf. Artif. Intell., 2021, vol. 35, no. 5, pp. 3950–3957.

[15] J. Zhu et al., “Graph neural networks with heterophily,” in Proc. AAAI Conf. Artif. Intell., 2021, vol. 35, no. 12, pp. 11168–11176.

[16] J. Chen, W. Liu, and J. Pu, “Memory-based message passing: Decoupling the message for propagation from discrimination,” in Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP), May 2022, pp. 4033–4037.

[17] H. Pei, B. Wei, K. C.-C. Chang, Y. Lei, and B. Yang, “Geom-GCN: Geometric graph convolutional networks,” in Proc. Int. Conf. Learn. Represent., 2020, pp. 1–12.

[18] S. Luan et al., “Revisiting heterophily for graph neural networks,” in Proc. Int. Conf. Learn. Represent., 2020, pp. 1–14.

[19] E. Chien, J. Peng, P. Li, and O. Milenkovic, “Adaptive universal

[20] J. Chen, W. Liu, and J. Pu, “Memory-based message passing: Decoupling the message for propagation from discrimination,” in Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP), May 2022, pp. 4033–4037.

[21] H. Pei, B. Wei, K. C.-C. Chang, Y. Lei, and B. Yang, “Geometric graph convolutional networks,” in Proc. Int. Conf. Learn. Represent., 2020, pp. 1–12.

[22] S. Luan et al., “Revisiting heterophily for graph neural networks,” in Proc. Adv. Neural Inf. Process. Syst., 2022, pp. 1–14.

[23] E. Chien, J. Peng, P. Li, and O. Milenkovic, “Adaptive universal generalized pagerank graph neural network,” in Proc. Int. Conf. Learn. Represent., 2021, pp. 1–24.

[24] Y. Ma, X. Liu, N. Shah, and J. Tang, “Is homophily a necessity for graph neural networks?” in Proc. Int. Conf. Learn. Represent., 2022, pp. 1–20.

[25] I. Bengtsson and K. Zyczkowski, Geometry of Quantum States: An Introduction to Quantum Entanglement. Cambridge, U.K.: Cambridge Univ. Press, 2017.

[26] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, “Spectral networks and locally connected networks on graphs,” 2013, arXiv:1312.6203.

[27] M. Defferrard, X. Bresson, and P. Vandergheynst, “Convolutional neural networks on graphs with fast localized spectral filtering,” in Proc. Adv. Neural Inf. Process. Syst., vol. 29, 2016, pp. 1–9.

[28] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, “How powerful are graph neural networks?” in Proc. Int. Conf. Learn. Represent., 2018, pp. 1–17.

[29] J. Klicpera, A. Bojchevski, and S. Günnemann, “Predict then propagate: Graph neural networks meet personalized pagerank,” in Proc. Int. Conf. Learn. Represent., 2019, pp. 1–15.

[30] F. Errica, M. Podda, D. Bacciu, and A. Micheli, “A fair comparison of graph neural networks for graph classification,” in Proc. Int. Conf. Learn. Represent., 2020, pp. 1–16.

[31] P. W. Battaglia et al., “Relational inductive biases, deep learning, and graph networks,” 2018, arXiv:1806.01261.

[32] G. Wang, R. Ying, J. Huang, and J. Leskovec, “Improving graph attention networks with large margin-based constraints,” 2019, arXiv:1910.11945.

[33] Y. Hou et al., “Measuring and improving the use of graph information in graph neural networks,” in Proc. Int. Conf. Learn. Represent., 2020, pp. 1–16.

[34] J. Chen, S. Chen, M. Bai, J. Pu, J. Zhang, and J. Gao, “Graph decoupling attention Markov networks for semisupervised graph node classification,” IEEE Trans. Neural Netw. Learn. Syst., early access, Mar. 29, 2022, doi: 10.1109/TNNLS.2022.3161453.

[35] M. Chen, Z. Wei, Z. Huang, B. Ding, and Y. Li, “Simple and deep graph convolutional networks,” in Proc. Int. Conf. Mach. Learn., 2020, pp. 1725–1735.

[36] M. He, Z. Wei, Z. Huang, and H. Xu, “BernNet: Learning arbitrary graph spectral filters via Bernstein approximation,” in Proc. Adv. Neural Inf. Process. Syst., vol. 34, 2021, pp. 14239–14251.

[37] M. Liu, Z. Wang, and S. Ji, “Non-local graph neural networks,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 44, no. 12, pp. 10270–10276, Dec. 2022.

[38] B. Fatemi, L. E. Asri, and S. M. Kazemi, “SLAPS: Self-supervision improves structure learning for graph neural networks,” in Proc. Adv. Neural Inf. Process. Syst., vol. 34, 2021, pp. 22667–22681.

[39] S. Abu-El-Haija et al., “MixHop: Higher-order graph convolutional architectures via sparsified neighborhood mixing,” in Proc. Int. Conf. Mach. Learn., 2019, pp. 21–29.

[40] K. Xu, C. Li, Y. Tian, T. Sonobe, K.-i. Kawarabayashi, and S. Jegelka, “Representation learning on graphs with jumping knowledge networks,” in Proc. Int. Conf. Mach. Learn., 2018, pp. 5453–5462.
Junbin Gao received the B.Sc. degree in computational mathematics from the Huazhong University of Science and Technology (HUST), Wuhan, China, in 1982, and the Ph.D. degree from the Dalian University of Technology, Dalian, China, in 1991. From 1982 to 2001, he was an Associate Lecturer, a Lecturer, an Associate Professor, and a Professor of mathematics with the HUST. He was a Senior Lecturer and a Lecturer of computer science with the University of New England, Armidale, NSW, Australia, from 2001 to 2005. He was a Professor of computer science with the School of Computing and Mathematics, Charles Sturt University, Bathurst, NSW, Australia. He is a Professor of big data analytics with The University of Sydney Business School, The University of Sydney, Camperdown, NSW, Australia. His main research interests include machine learning, data analytics, Bayesian learning and inference, and image analysis.

Zengfeng Huang received the B.S. degree in computer science from Zhejiang University, Hangzhou, China, in 2008, and the Ph.D. degree in computer science and engineering with The Hong Kong University of Science and Technology, Hong Kong, in 2013. He is currently an Associate Professor with the School of Data Science, Fudan University, Shanghai, China. Before that he was a Research Fellow of computer science and engineering, UNSW Sydney (UNSW), Kensington, NSW, Australia, and a Post-Doctoral with the Center for Massive Data Algorithmics (MADALGO), Aarhus University, Aarhus, Denmark. His research interests include algorithmic aspects of data science.

Junping Zhang (Senior Member, IEEE) received the B.S. degree in automation from Xiangtan University, Xiangtan, China, in 1992, the M.S. degree in control theory and control engineering from Hunan University, Changsha, China, in 2000, and the Ph.D. degree in intelligent systems and pattern recognition from the Institution of Automation, Chinese Academy of Sciences, Beijing, China, in 2003. He has been a Professor with the School of Computer Science, Fudan University, Shanghai, China, since 2006. His research interests include machine learning, image processing, biometric authentication, and intelligent transportation systems.

Dr. Zhang was an Associate Editor of IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS from 2010 to 2018 and has been an Associate Editor of IEEE Intelligent Systems since 2009.

Jian Pu (Member, IEEE) received the Ph.D. degree from Fudan University, Shanghai, China, in 2014. He was a Post-Doctoral Researcher with the Institute of Neuroscience, Chinese Academy of Sciences, Beijing, China, from 2014 to 2016, and an Associate Professor with the School of Computer Science and Software Engineering, East China Normal University, Shanghai, from 2016 to 2019. He is currently a Young Principal Investigator with the Institute of Science and Technology for Brain-Inspired Intelligence (ISTBI), Fudan University. His current research interests include developing machine learning and computer vision methods for autonomous driving.