BREAK THE WALL BETWEEN HOMOPHILY AND HETEROPHILY FOR GRAPH REPRESENTATION LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Homophily and heterophily are intrinsic properties of graphs that describe whether two linked nodes share similar properties. Although many Graph Neural Network (GNN) models have been proposed, it remains unclear how to design a model so that it can generalize well to the whole spectrum of homophily. This work addresses the challenge by identifying three graph features, including the ego node feature, the aggregated node feature, and the graph structure feature, that are essential for graph representation learning. It further proposes a new GNN model called OGNN (Omnipotent Graph Neural Network) that extracts all three graph features and adaptively fuses them to achieve generalizability across the whole spectrum of homophily. Extensive experiments on both synthetic and real datasets demonstrate the superiority (average rank 1.56) of our OGNN compared with state-of-the-art methods. Our code will be available at `https://`.

1 INTRODUCTION

Graph neural networks (GNNs) have been proven to be a powerful approach to learning graph representations for node classification tasks (Kipf & Welling, 2016). Researchers have proposed different GNN model designs based on the underlying assumption of either graph homophily or heterophily. On the one hand, many works (Kipf & Welling, 2016; Hamilton et al., 2017; Veličković et al., 2017; Gao et al., 2018; Klicpera et al., 2018; Xu et al., 2018b; Wu et al., 2019) assume strong graph homophily, which means linked nodes in the graph tend to share similar properties or labels. These GNNs adopt a message passing paradigm that recursively propagates and aggregates node features through the edges in the graph to produce smoothed node representations (Gilmer et al., 2017; Battaglia et al., 2018). We refer to these GNNs as *homophily-based GNNs*.

On the other hand, some recent works empirically demonstrate the poor performance of homophily-based GNNs on heterophilic graphs, which have the opposite property to homophily. Examples of these graphs include online transaction networks where fraudsters are more likely to connect with customers than their colleagues, and protein networks where different amino acid types tend to be connected. GNNs designed for heterophilic graphs leverage various strategies to generalize better on these graphs, such as embedding and mixing graph topology with node features (Lim et al., 2021a), ego-neighbor embedding separation, and higher-order neighborhood representation combination (Zhu et al., 2020), degree correction and signed messages (Yan et al., 2021), and generalized PageRank weights (Chien et al., 2020). They perform much better on graphs with a strong heterophily property but achieve only comparable, if not worse, accuracy on homophilic graphs than homophily-based GNNs. We refer to these GNNs as *heterophily-based GNNs*.

Neither homophily-based GNNs nor heterophily-based GNNs are ideal, since they invisibly establish a wall between homophily and heterophily for graph representation learning — a GNN can work well on either heterophilic graphs or homophilic graphs, but not both. Real-world graph data, however, could have various levels of homophily. It is a hassle to first classify a graph as homophilic or heterophilic before a suitable GNN model can be identified. What's worse, some graphs cannot be easily classified as homophilic or heterophilic. For example, `twitch-gamer` (Rozemberczki & Sarkar, 2021) is a social network graph in which the users follow each other by their game interests, while the nodes (i.e., users) are labeled by gender. Since the following relationships (i.e., the connections) are not dominated by gender (i.e., the label), the homophily of `twitch-gamer` is 0.55, which falls in the ambiguous intermediate region on the homophily spectrum. One needs

to try both homophily-based GNNs and heterophily-based GNNs to identify the suitable models, significantly increasing data analytic costs.

To address the problems, this paper proposes a model called omnipotent GNN (OGNN) that can generalize well to the whole spectrum of homophily. The design of OGNN is motivated by the observation that each node in a graph can be modeled by different types of features, including the feature of itself, the feature of its neighbors, and its connections with the other nodes. These features are of different importance in determining a node's property depending on the graph's homophily. The basic idea of OGNN is to effectively transform and adaptively fuse these features to derive each node's representation in the graph. Our main contributions are summarized as follows:

- We identify three aspects of the graph features that are essential for graph representation learning: ego node feature, aggregated node feature, and graph structure feature. We analyze the importance of these graph features and propose a general form for extracting them. Our detailed ablation study demonstrates that these features have different importance for the graphs with different homophily.
- We propose the OGNN model that generalizes to graphs over the homophily spectrum. OGNN could effectively learn graph representation by integrating the three graph features with adaptive feature fusion under a bi-level optimization framework.
- We conduct extensive experiments to compare OGNN with state-of-the-art GNNs using synthetic and real graph benchmarks that cover the full homophily spectrum. OGNN outperforms 8 baseline models and achieves an average rank of 1.56 on 9 real datasets. Specifically, OGNN achieves higher node classification accuracy, 4.55% higher than GCN (Kipf & Welling, 2016), 4.39% higher than MIXHOP (Abu-El-Haija et al., 2019), and 3.27% higher than H2GCN (Zhu et al., 2020), on average over the real datasets.

2 Notations and Preliminaries

Notations. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ denotes a graph with N nodes and M edges, where $\mathcal{V}$ and $\mathcal{E}$ are the set of nodes and edges respectively, $|\mathcal{V}| = N$ and $|\mathcal{E}| = M$. We use $\mathbf{A} \in \{0,1\}^{N\times N}$ as the adjacency matrix where $\mathbf{A}[i,j] = 1$ if $(v_i, v_j) \in \mathcal{E}$ otherwise $\mathbf{A}[i,j] = 0$. Each node $v_i \in \mathcal{V}$ has a raw feature vector x_i of size D. The raw feature vectors of all nodes in the graph form a feature matrix $\mathbf{X} \in \mathbb{R}^{N\times D}$. The nodes are categorized into C classes. The label of a node v_i of class k is represented by a one-hot vector y_i of size C, whose k-th dimension is 1 while the other dimensions are 0. All the label vectors form the label matrix $\mathbf{Y} \in \{0,1\}^{N\times C}$. This paper focuses on the node classification task (Kipf & Welling, 2016). The goal of the node classification task is, given a train set $\mathcal{V}_{\text{train}} \in \mathcal{V}$ with label set $\mathbf{Y}_{\text{train}}$, learn a mapping $\mathcal{F} : \mathcal{V} \to \mathbf{Y}$, which maximizes the possibility $P(\arg\max \mathcal{F}(v) = \arg\max y_v)$ for any vertex $v \in \mathcal{V} \setminus \mathcal{V}_{\text{train}}$.

Homophily and Heterophily. Graph homophily $\mathcal{H}$ measures the overall similarity between the nodes connected by an edge in terms of the labels. There are multiple ways for computing graph homophily (Pei et al., 2020; Zhu et al., 2020; Lim et al., 2021b; Apollonio et al., 2022). In this paper, we adopt the most widely-used edge homophily (Zhu et al., 2020):

Definition 1. *Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with labels $\mathbf{Y}$, the edge homophily is defined as $\mathcal{H}_{edge}(\mathcal{G}, \mathbf{Y}) = \frac{1}{|\mathcal{E}|}\sum_{(u,v)\in\mathcal{E}} 1(y_u = y_v)$, which represents the fraction of the edges that connect two nodes with the same class label.*

The edge homophily ranges from 0 to 1. Graphs with edge homophily close to 1 are called homophilic graphs, while the ones with edge homophily close to 0 are called heterophilic graphs.

Graph Neural Networks. Most GNNs follow the message passing framework. In the message passing framework, the hidden state of a node $v \in \mathcal{V}$ depends on the features of its neighbors and its ego feature. A typical message passing layer is

$$h_v^{l+1} = \Psi^l(h_v^l, \underset{u\in\mathcal{N}(v)}{\Gamma^l}(\Phi(h_u^l, h_v^l))), \tag{1}$$

where h^l represents the hidden feature of a node at layer l, Ψ and Φ are transformation functions, which could be any differentiable functions like linear transformations or multi-layer perceptron (MLP). Γ is a permutation invariant function such as sum, mean, and max, while $\mathcal{N}(v)$ is the node set of v's neighborhoods. The message passing layer transforms the features from the neighborhoods with

Φ as the messages and then aggregates them with Γ. After that, Ψ updates the node feature with the ego feature and the aggregated message. By stacking multiple message passing layers in the model, GNNs could iteratively propagate information to the target node from multi-hop neighborhoods.

3 Graph Representation Learning over Different Homophily

Although many different GNNs have been proposed, most of them are designed under the assumption of either strong homophily or heterophily, making them incapable of generalizing well to a whole spectrum of homophily. To illustrate our statement, we conduct experiments to validate the classification accuracy of several GNN models on a synthetic benchmark `syn-cora` (Zhu et al., 2020). `syn-cora` provides homophily levels varying from 0.0 to 1.0 with 0.1 as the interval. The nodes and their raw features of `syn-cora` are from the `Cora` dataset (Sen et al., 2008; Yang et al., 2016), while the edges are randomly generated according to different homophily settings. Details about `syn-cora` are in Section 5.1. Figure 1 shows the result of different GNNs and our proposed OGNN on `syn-cora` with varying homophily levels.

Overall, most existing GNNs cannot generalize well over the spectrum of homophily. Specifically, GCN (Kipf & Welling, 2016), and DAGNN (Liu et al., 2020) achieve high accuracy on graphs with high homophily, but their performance drops severely as the homophily decreases. When the homophily is lower than 0.5, their performance is even worse than Multi-Layer Perceptron (MLP) which relies solely on the node features without any structural information. Meanwhile, LINKX (Lim et al., 2021a) outperforms MLP, DAGNN, and GCN on graphs with low homophily, but its performance on the graphs with strong homophily, i.e., $0.7 \sim 1.0$ is the worst besides MLP.

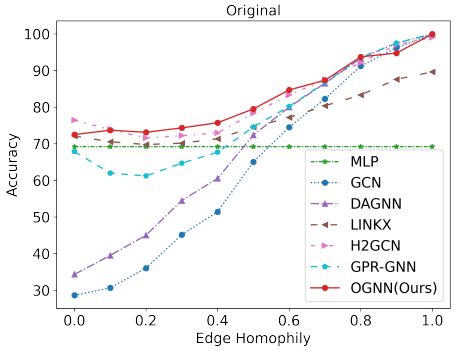


Figure 1: Comparisons of our model and other models on synthetic dataset `syn-cora`.

Both H2GCN (Zhu et al., 2020) and GPR-GNN (Chien et al., 2020) are designed to adapt to both homophilic and heterophilic graphs. However, GPR-GNN cannot compete with MLP on low edge homophily because their model design still follows the message passing paradigm. Although H2GCN is the winning solution from existing GNNs on the `syn-cora` dataset, it is outperformed by our proposed approach OGNN. Moreover, H2GCN shows worse performance on real-world heterophilic graphs compared to heterophily-based GNNs, e.g., LINKX (see Section 5.2). We suspect that the reason is that H2GCN does not leverage graph structure features as in LINKX and our approach OGNN. In Section 5.2, we empirically show that graph structure is an important feature for learning on real-world heterophilic graphs.

To summarize, existing GNNs hardly show consistently good performance on graphs with various homophily settings. Therefore, in this paper, we aim at designing a new GNN that can generalize well across graphs with different homophily.

4 OGNN: A GNN for Graphs with Different Homophily

In this section, we first analyze three types of features from a graph: the ego-node feature, the aggregated neighborhood feature, and the graph structure feature, which cover multi-faceted information for graph representation learning. Then we present our model OGNN which effectively extracts the three types of features and adaptively fuses them with a bi-level optimization training scheme.

4.1 Graph Information Analysis

Ego-Node Feature. The ego-node feature refers to a node's embedding resulting from transforming the node's raw feature without considering its neighborhoods. Formally, the ego-node feature h_{ego} of a node v with its raw feature x_v is:

$$h_v^{ego} = f_{\text{ego}}(x_v), \tag{2}$$

where $f_{\text{ego}}(\cdot)$ can be any transformation functions such as linear transformation or MLPs.

The ego-node feature is the basic and essential information for node classification tasks. This intuition is based on two observations. First, as shown in Figure 1, applying MLP on only raw node features of heterophilic graphs can significantly outperform many homophily-based GNNs (Kipf & Welling, 2016; Veličković et al., 2017; Liu et al., 2020) that rely on both the raw node features and the aggregated neighborhood features. This empirical observation implies that the ego-node feature alone plays an important role in GNN's generalizability. Second, a recent study (Zhu et al., 2020) proves that, under some conditions, a GCN layer that separately embeds ego-node features and neighbor-node features is more capable of generalizing to heterophilic graphs than co-embedding them. It is because, in heterophilic settings, the class labels and the raw features of a node and its neighborhoods may be different. The ego-node feature could be a more robust signal for the node's label than the mixture of features from the node itself and its neighborhood.

We empirically validate this intuition in Section 5.2 and demonstrate that (1) GNNs with ego-node features can achieve $0.25\% \sim 4.30\%$ higher accuracy than those without ego-node features across graphs with different levels of homophily. (2) The importance of ego-node features increases as the homophily of a graph becomes lower. For example, on `syn-cora` graph with 0.8 homophily, the importance of ego-node features is 0.197 out of 1, while with 0.2 homophily, the importance increases to 0.656.

Aggregated Neighborhood Feature. The aggregated neighborhood feature complements ego-node features by capturing the information from the neighborhood to the target node. Formally, the aggregated neighborhood feature h_{agg} of a node v with raw feature x_v is

$$h_v^{agg} = f_{\text{agg}}(x_v, \{x_u, u \in \mathcal{N}^k(v), k = 1, 2, 3, \cdots\}), \tag{3}$$

where $\mathcal{N}^k(v)$ represents the set of k-hop neighbors and $f_{\text{agg}}(\cdot)$ is a function that transforms and aggregates the features from the neighborhood as well as the target node's own feature.

There are many approaches to materialize the function $f_{\text{agg}}(\cdot)$. One approach is to simply stack multiple message passing layers as in many homophily-based GNNs such as GCN and GAT. Each message passing layer first transforms the features and then aggregates them to pass to the next layer, as in Eq. 1. However, this approach allows only a limited size of neighborhoods to be considered since the performance of these GNNs degrades severely when more than three layers are stacked (Li et al., 2018; Zhang et al., 2022). An alternative approach is a decoupled transformation and aggregation strategy where raw features are transformed first and then go through multiple aggregation layers without being transformed again. Recent works (Klicpera et al., 2018; Liu et al., 2020) demonstrate that the alternative approach could achieve much better generalization performance on homophilic graphs as it can effectively capture the information from a large neighborhood region. We will also adopt this strategy in OGNN.

Our empirical study shows that (1) GNNs with aggregated neighborhood features can achieve $0.13\% \sim 13.96\%$ higher accuracy than those without aggregated features across graphs with different homophily, and (2) the importance of the aggregated neighborhood features increases as the homophily of a graph becomes higher. For example, on `syn-cora` graph with 0.2 homophily, the importance of aggregated features is 0.284 out of 1, while with 0.8 homophily, the importance increases to 0.772.

Graph Structure Feature. Although the aggregated neighborhood feature involves the local connections around the target node, it loses a fair amount of the graph structural information because of the permutation invariant aggregation process. Therefore, it is necessary to take the graph structure information as an independent information source for graph representation learning. Formally, the graph structure feature is:

$$h_v^{strc} = f_{\text{strc}}(\{\mathbf{A}^k[v, :], k = 1, 2, 3, \cdots\}), \tag{4}$$

where $\mathbf{A}^k$ is the k-th power of the adjacency matrix $\mathbf{A}$, $\mathbf{A}^k[v, :]$ is the v-th row of $\mathbf{A}^k$, and $f_{\text{strc}}(\cdot)$ could be any transformation functions.

Graph structure feature has been shown to be very effective in node classification tasks on heterophilic graphs. Both LINK and LINKX (Zheleva & Getoor, 2009; Lim et al., 2021a) leverage the simplest form of graph structure feature, the adjacency matrix, to learn node embedding and show better performance than many GNNs on heterophilic graphs (Abu-El-Haija et al., 2019; Chien et al., 2020;

Zhu et al., 2020). However, these works consider only $k = 1$, i.e., the adjacency matrix with only 1-hop neighbors. In this work, We propose to use the combination of different powers of the adjacency matrix as a more general form of graph structure feature. The intuition behind it is that $k = 1$ captures local structure information while $k > 1$ captures regional/global structure information.

Our empirical study shows that (1) on all the 9 real-world graphs we experiment with, the best performance is achieved when k is larger than one, and (2) graph structure features can improve classification accuracy by $0.78\% \sim 20.49\%$ across the graphs with different homophily.

To sum up, the three types of features summarize distinct graph information: the ego-node feature and the aggregated neighborhood feature model the information from a node's feature and its neighborhoods respectively; the graph structure feature models both the local and global structure information regardless of the node features.

4.2 OGNN Model design

OGNN effectively extracts all three types of graph features and adaptively fuses them together to achieve good generalizability across the whole spectrum of homophily. We now explain its three main components, *feature extractors*, *adaptive feature fusion*, and *bi-level optimization* in details.

Feature Extractors. The *Ego-Node Feature Extractor* is a simple linear transformation of the raw node feature matrix $\mathbf{X} \in \mathbb{R}^{N\times D}$: $\mathbf{H}_{\text{ego}} = \mathbf{X}\mathbf{W}_{\text{ego}}$, where $\mathbf{W}_{\text{ego}} \in \mathbb{R}^{D\times d}$ is the transformation matrix and $\mathbf{H}_{\text{ego}} \in \mathbb{R}^{N\times d}$ is the extracted ego-node features. We use linear transformation instead of MLP because linear transformation consistently achieves the best accuracy in our empirical evaluation.

The *Aggregated Neighborhood Feature Extractor* adopts the state-of-the-art design in the message passing framework, where feature transformation and propagation are decoupled (Zeng et al., 2021). It propagates the ego-node features $\mathbf{H}_{\text{ego}}$ and combines the features of different propagation steps to enlarge the receptive field: $\mathbf{H}_{\text{agg}} = \sum_{i=1}^{s_1} \hat{\mathbf{A}}^i \mathbf{H}_{\text{ego}}$, where $\hat{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$ is the normalized adjacency matrix, and $\mathbf{D}$ is the diagonalized node degrees. s_1 is the maximum propagation step.

The *Graph Structure Feature Extractor* computes the combination of different powers of the adjacency matrix after a linear transformation: $\mathbf{H}_{\text{strc}} = \sum_{j=1}^{s_2} \mathbf{A}^j \mathbf{W}_{\text{strc}}$, where $\mathbf{W}_{\text{strc}} \in \mathbb{R}^{N\times d}$ is the transformation matrix, and s_2 is the maximum power of the adjacency matrix. The use of a simple linear transformation is for its efficiency, as we do not observe performance gains from an MLP.

Adaptive Feature Fusion. The three types of features play different roles in modeling graph information and thus their importance in different homophily settings varies. The adaptive weighted feature fusion module assigns a trainable scalar importance score for each feature so that it can *automatically* learn the features' importance from the input graph:

$$\mathbf{H} = \sigma(\pi_1\mathbf{H}_{\text{ego}} + \pi_2\mathbf{H}_{\text{agg}} + \pi_3\mathbf{H}_{\text{strc}}), \quad \pi_i \quad = \frac{\exp p_i}{\sum_{j=1}^{3} \exp p_j}, \tag{5}$$

where $\mathbf{H}$ is the fused feature and σ is a non-linear activation function ReLU. $\mathcal{P} = \{p_i | i = 1, 2, 3\}$ are trainable parameters, and $\{\pi_i | i = 1, 2, 3\}$ are the weights for each feature.

After obtaining the fused feature $\mathbf{H}$, we predict the labels for each node with a linear classifier, $\mathbf{Y}_{\text{pred}} = \text{Softmax}(\mathbf{H}\mathbf{W}_{\text{pred}})$, where $\mathbf{Y}_{\text{pred}} \in \mathbb{R}^{N\times C}$ is the predictions and $\mathbf{W}_{\text{pred}} \in \mathbb{R}^{d\times C}$ is the predictor's parameters.

Bi-level Optimization. To train our model parameters and feature fusion weights jointly, we borrow the idea of bi-level optimization (Liu et al., 2018; Dong & Yang, 2019). Suppose the model parameters is $\mathbf{W}$ and the parameters for feature fusion is $\mathcal{P}$. The loss function of the node classification tasks is:

$$\mathcal{L}(\mathbf{W}, \mathcal{P}, \mathcal{G}, \mathbf{X}, \mathbf{Y}) = -\frac{1}{|\mathbf{Y}|} \sum_{y_i \in \mathbf{Y}} y_i^{\mathrm{T}} log(\hat{y}_i), \tag{6}$$

where $\mathcal{G}$ is the graph, $\mathbf{X}$ and $\mathbf{Y}$ are raw node features and ground-truth labels respectively, and $\hat{y}_i \in \mathbf{Y}_{\text{pred}}$ is the prediction of our model for node i. Then the objective of our bi-level optimization is:

$$\begin{aligned} &\min_{\mathcal{P}} \mathcal{L}_{\text{valid}}(\mathbf{W}^*, \mathcal{P}, \mathcal{G}, \mathbf{X}_{\text{valid}}, \mathbf{Y}_{\text{valid}}), \\ \text{s.t.} \quad &\mathbf{W}^* = \arg\min_{\mathbf{W}} \mathcal{L}_{\text{train}}(\mathbf{W}, \mathcal{P}, \mathcal{G}, \mathbf{X}_{\text{train}}, \mathbf{Y}_{\text{train}}). \end{aligned} \tag{7}$$

In short, we optimize the model parameters $\mathbf{W}$ on the train set, while optimizing the feature fusion parameters $\mathcal{P}$ on the validation set alternatively.

Time Complexity. The time complexity of extracting the ego-node features is $\mathcal{O}(\text{nnz}(\mathbf{X}) \cdot d)$, where $\text{nnz}(\mathbf{X})$ is the number of non-zero values in the raw feature matrix $\mathbf{X}$, d is the hidden feature dimension. Extracting the aggregated neighborhood feature takes $\mathcal{O}(s_1 Md + s_1 Nd)$ to propagate features from s_1-hop neighbors and sum them up, where M and N are the number of edges and nodes in the graph. Similarly, extracting the graph structure feature takes $\mathcal{O}(s_2 Md + s_2 Nd)$ with sparse matrix multiplications. The feature fusion step takes $\mathcal{O}(Nd)$ for reweighting and summing up the features. Finally, the linear classifier takes $\mathcal{O}(NdC)$ to do the predictions.

5 Experiments

We conduct experiments on both synthetic datasets and real datasets with various homophily to examine the efficacy of our model in terms of test accuracy.

5.1 Experimental Settings

Synthetic Dataset. We generate synthetic graphs `syn-cora` with the approach in H2GCN (Zhu et al., 2020). The `syn-cora` dataset provides 11 graphs with homophily ranging from 0.0 to 1.0 with 0.1 as the interval. The raw node features and labels for each graph are sampled from the `cora` dataset (Sen et al., 2008). The edges of the graph are generated gradually according to the given homophily. We evaluate the average test accuracy over five trials for all the methods on these graphs with the same train/validation/test data splits (25%, 25%, 50%, for each class).

Real Dataset. We also evaluate our method and existing GNNs on 9 real-world datasets, whose homophily ranges from $0.222 \sim 0.931$. Table 4 in Appendix Section A summarizes the datasets detailed statistics. `Cora` (Sen et al., 2008), `CiteSeer` (Sen et al., 2008), `PubMed` (Sen et al., 2008), and `Coauthor CS & Physics` (Shchur et al., 2018) are widely-used homophilic graphs, while `penn94` (Traud et al., 2012), `arXiv-year` (Hu et al., 2020), `genius` (Lim & Benson, 2021), and `twitch-gamer` (Rozemberczki & Sarkar, 2021) are graphs with low to medium homophily (Lim et al., 2021b;a). The data splits of these datasets are explained in Appendix Section A. We report the average and the standard deviation of the test accuracy in the following experiments.

Baselines for Comparison. Our baselines include message-passing based GNNs (GCN (Kipf & Welling, 2016), GAT (Veličković et al., 2017), and DAGNN (Liu et al., 2020)), GNNs designed for heterophilic graphs (LINKX (Lim et al., 2021a), H2GCN (Zhu et al., 2020), MIXHOP (Abu-El-Haija et al., 2019), and GPR-GNN (Chien et al., 2020)). Our experiments also include MLP because it is a strong baseline for heterophilic graphs. We did grid-based hyper-parameter search for all baselines and our approach (Details in Appendix Section C).

Hardware Specifications. We run experiments on both synthetic and real world benchmarks with a 12-core CPU, 8 GB Memory, and an NVIDIA GeForce GTX 1080 Ti GPU with 11 GB GPU Memory for all the methods except for H2GCN, because it suffers from the out-of-memory (OOM) problem. For H2GCN, we use a workstation with a 12-core CPU, 32 GB Memory, and an NVIDIA Quadro RTX 8000 GPU with 48 GB GPU Memory.

5.2 Quantitative Results

Results on Synthetic Dataset. Table 1 reports the average test accuracy over five random splits on the graphs in the `syn-cora` dataset. Overall, OGNN outperforms the existing methods on most homophily settings, with seven settings at top-1 and four at top-2. Especially, in the ambiguous homophily region (about $0.4 \sim 0.6$), OGNN achieves state-of-the-art results by pushing the best accuracy up to 2.76%. A full version with standard deviation can be found in Appendix Section D.

The results echo our analysis in Section 3. The performance of homophily-based GNNs such as GCN, GAT and DAGNN are outstanding at high homophily settings. However, they fail to perform well on the heterophilic graphs and are much worse than the baseline MLP, which is graph-agnostic. LINKX is a simple but strong baseline on non-homophilic graphs, but it performs poorly on high homophily settings (≥ 0.7), merely better than MLP. For H2GCN, MIXHOP, and GPR-GNN, which are designed

to generalize well on both homophilic and heterophilic graphs, they show variant performance in our experiments. MIXHOP and GPR-GNN cannot achieve as good accuracy as they claimed on low homophily settings, reflecting their unsatisfying generalization performance. H2GCN shows the most competitive performance among existing GNNs, especially on very low homophily settings (0.0 and 0.1). However, OGNN outperforms it on a larger range of homophily ($0.2 \sim 0.8$), which are also more common cases in real-world datasets.

Table 1: Test accuracy of different methods on the graphs with different homophily in `syn-cora` dataset. **Red** and blue represent top-1 and top-2 ranking in terms of accuracy respectively.

synh	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
MLP	69.20	69.20	69.20	69.20	69.20	69.20	69.20	69.20	69.20	69.20	69.20
GCN	28.61	30.64	36.03	45.15	51.39	65.04	74.48	82.22	91.21	96.19	99.92
GAT	29.41	30.32	34.83	43.86	51.15	64.80	74.34	81.45	90.46	95.79	**100.0**
DAGNN	34.32	39.49	45.01	54.48	60.51	72.36	80.00	86.49	93.32	**97.48**	99.95
LINKX	72.09	70.54	69.76	70.13	71.34	74.53	77.16	80.35	83.30	87.59	89.60
H2GCN	**76.43**	**73.86**	71.58	72.17	72.95	78.31	83.27	**87.43**	92.09	97.00	98.98
MIXHOP	39.44	38.95	41.05	48.93	55.09	64.75	74.45	82.44	91.45	96.25	**100.0**
GPR-GNN	67.86	61.96	61.21	64.69	67.67	74.56	80.19	86.68	93.54	97.45	**100.0**
OGNN	72.49	73.67	**73.11**	**74.32**	**75.71**	**79.49**	**84.67**	87.32	**93.70**	94.75	99.95

Results on Real Dataset. Table 2 reports the results on the 9 real datasets. Overall, OGNN achieves seven top-1 and one top-2 over 9 datasets, with an average rank of 1.56. Compared with homophily-based GNNs including GCN, GAT, and DAGNN, our method outperforms the best of them (i.e., DAGNN) on the homophilic graphs and is far better than any of them on the heterophilic graphs. OGNN is also competitive compared to the heterophily-based GNNs, including LINKX, H2GCN, MIXHOP, and GPR-GNN. On `penn94` and `arXiv-year`, we improve the accuracy by $0.72 \sim 4.31\%$ and $0.5 \sim 11.61\%$ respectively. On `genius` and `twitch-gamer`, we have small gaps (0.86% and 0.13%) compared with the best method LINKX.

Table 2: Average test accuracy $\pm$ standard deviation on the real datasets. **Red** and blue represent top-1 and top-2 ranking in terms of accuracy respectively. OOM means a model runs out of memory on a specific dataset.

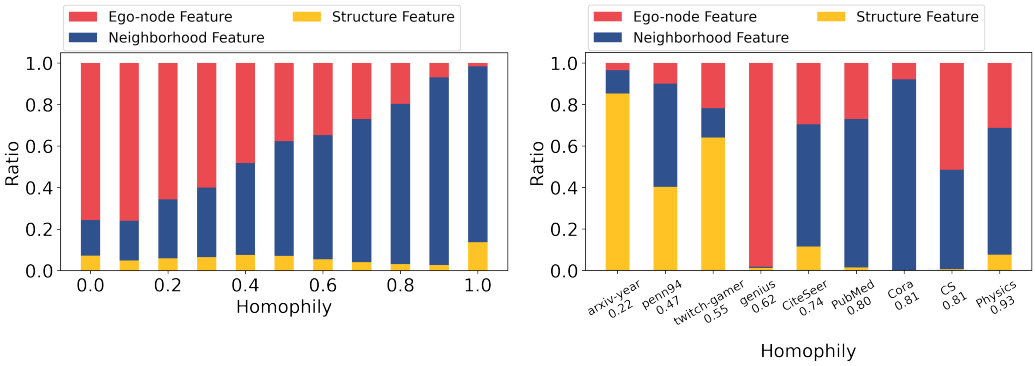
	arXiv-year	penn94	twitch-gamer	genius	CiteSeer	PubMed	Cora	Coauthor CS	Coauthor Physics	Avg. Rank
Homophily	0.22	0.47	0.55	0.62	0.74	0.80	0.81	0.81	0.93	-
#Nodes	169,343	41,554	168,114	421,961	3,327	19,717	2,708	18,333	34,493	-
#Edges	1,166,243	1,362,229	6,797,557	984,979	4,552	44,324	5,278	81,894	247,962	-
#Classes	5	2	2	2	6	3	7	15	5	-
MLP	$36.70_{\pm 0.21}$	$73.6_{\pm 0.40}$	$60.92_{\pm 0.07}$	$86.68_{\pm 0.09}$	$50.94_{\pm 4.20}$	$66.04_{\pm 2.29}$	$52.56_{\pm 2.55}$	$83.08_{\pm 1.00}$	$82.15_{\pm 5.11}$	8.56
GCN	$46.02_{\pm 0.26}$	$82.47_{\pm 0.27}$	$62.18_{\pm 0.26}$	$87.42_{\pm 0.37}$	$63.36_{\pm 2.06}$	$78.12_{\pm 1.60}$	$77.90_{\pm 1.18}$	$90.35_{\pm 0.88}$	$92.39_{\pm 0.89}$	5.33
GAT	$49.37_{\pm 0.20}$	$81.45_{\pm 0.55}$	$62.32_{\pm 0.23}$	$86.59_{\pm 1.06}$	$65.90_{\pm 1.88}$	$76.78_{\pm 2.38}$	$76.98_{\pm 1.75}$	$88.86_{\pm 0.65}$	$92.57_{\pm 0.60}$	5.44
DAGNN	$37.00_{\pm 0.20}$	$74.49_{\pm 0.37}$	$59.80_{\pm 0.13}$	$80.86_{\pm 3.82}$	$68.50_{\pm 0.88}$	$79.94_{\pm 1.30}$	$83.70_{\pm 1.12}$	$91.81_{\pm 0.24}$	$93.79_{\pm 0.68}$	4.89
LINKX	$56.00_{\pm 1.34}$	$84.71_{\pm 0.52}$	$\mathbf{66.06}_{\pm 0.19}$	$\mathbf{90.77}_{\pm 0.27}$	$53.66_{\pm 3.69}$	$67.66_{\pm 4.29}$	$62.66_{\pm 2.12}$	$88.53_{\pm 1.43}$	$89.37_{\pm 1.52}$	5.11
H2GCN	$49.09_{\pm 0.10}$	$81.54_{\pm 0.56}$	OOM	$90.54_{\pm 0.16}$	$64.40_{\pm 1.44}$	$76.30_{\pm 2.80}$	$79.24_{\pm 1.75}$	$91.18_{\pm 0.58}$	$93.56_{\pm 0.48}$	4.89
MIXHOP	$51.78_{\pm 0.26}$	$83.63_{\pm 0.54}$	$65.65_{\pm 0.30}$	$90.61_{\pm 0.24}$	$56.98_{\pm 4.80}$	$76.14_{\pm 2.37}$	$73.80_{\pm 4.02}$	$89.79_{\pm 0.91}$	$93.33_{\pm 0.75}$	4.78
GPR-GNN	$44.89_{\pm 0.20}$	$81.12_{\pm 0.63}$	$62.00_{\pm 0.25}$	$90.02_{\pm 0.13}$	$64.72_{\pm 1.59}$	$79.12_{\pm 0.87}$	$80.44_{\pm 1.53}$	$90.74_{\pm 0.60}$	$93.86_{\pm 0.36}$	4.44
OGNN	$\mathbf{56.50}_{\pm 0.13}$	$\mathbf{85.43}_{\pm 0.82}$	$65.93_{\pm 0.17}$	$89.91_{\pm 0.27}$	$\mathbf{71.50}_{\pm 1.83}$	$\mathbf{81.10}_{\pm 1.29}$	$\mathbf{84.12}_{\pm 1.29}$	$\mathbf{92.83}_{\pm 0.38}$	$\mathbf{93.87}_{\pm 0.43}$	1.56

Ablation Study. We present ablation study to show the effectiveness of our design choices: the three graph features, adaptive feature fusion, and bi-level optimization. Table 3 reports the accuracy results from five variants of our model by removing one design element at a time.

Graph features. The 2nd to the 4th row demonstrate the contribution of the three graph features on the model's accuracy. Overall, models without one of the features suffer from $2.49\% \sim 5.81\%$ accuracy drop on all the datasets on average, indicating the importance of these features on graph representation learning regardless of their homophily. Specifically, models without the aggregated neighborhood feature have a larger accuracy drop on homophilic graphs (i.e., `Cora`, `CiteSeer`, `PubMed`, `Coauthor CS & Physics`). It echoes the high performance of message passing-based neural networks (e.g., GCN and DAGNN) on homophilic graphs. On the contrary, models without the graph structure feature suffer more severely on heterophilic graphs (i.e., `penn94`, `arXiv-year`,

Table 3: Ablation studies on the real datasets.

Homophily	arXiv 0.22	penn94 0.47	twitch 0.55	genius 0.62	CiteSeer 0.74	PubMed 0.80	Cora 0.81	CS 0.81	Physics 0.93	Δ Avg. -
OGNN	**56.50**	**85.43**	**65.93**	**89.91**	**71.50**	**81.10**	**84.12**	**92.83**	**93.87**	-
w/o ego	53.80 $_{2.71\downarrow}$	81.83 $_{3.60\downarrow}$	65.68 $_{0.25\downarrow}$	85.61 $_{4.30\downarrow}$	68.44 $_{3.06\downarrow}$	78.92 $_{2.18\downarrow}$	83.12 $_{1.00\downarrow}$	88.84 $_{3.99\downarrow}$	92.56 $_{1.31\downarrow}$	2.49↓
w/o agg	53.17 $_{3.33\downarrow}$	84.81 $_{0.62\downarrow}$	65.78 $_{0.15\downarrow}$	89.78 $_{0.13\downarrow}$	57.54 $_{13.96\downarrow}$	72.32 $_{8.78\downarrow}$	72.66 $_{11.46\downarrow}$	87.72 $_{5.11\downarrow}$	85.14 $_{8.73\downarrow}$	5.81↓
w/o strc	36.01 $_{20.49\downarrow}$	75.50 $_{9.93\downarrow}$	61.59 $_{4.34\downarrow}$	86.87 $_{3.04\downarrow}$	69.82 $_{1.68\downarrow}$	81.12 $_{0.02\uparrow}$	83.34 $_{0.78\downarrow}$	92.46 $_{0.37\downarrow}$	93.41 $_{0.46\downarrow}$	4.56↓
w/o fusion	53.45 $_{3.05\downarrow}$	84.35 $_{1.08\downarrow}$	65.76 $_{0.17\downarrow}$	87.64 $_{2.27\downarrow}$	67.72 $_{3.78\downarrow}$	76.04 $_{5.06\downarrow}$	82.48 $_{1.64\downarrow}$	91.13 $_{1.70\downarrow}$	92.64 $_{1.23\downarrow}$	2.22↓
w/o bi-level	53.27 $_{3.23\downarrow}$	83.71 $_{1.72\downarrow}$	65.74 $_{0.19\downarrow}$	88.25 $_{1.66\downarrow}$	68.60 $_{2.90\downarrow}$	73.32 $_{7.78\downarrow}$	81.02 $_{3.10\downarrow}$	90.31 $_{2.52\downarrow}$	93.15 $_{0.72\downarrow}$	2.65↓



(a) Results on `syn-cora` dataset. (b) Results on real datasets.

Figure 2: Importance of the features after feature fusion.

`genius`, and `twitch-gamer`). It echoes the high performance of GNNs that rely heavily on graph structures (e.g., LINKX) on heterophilic graphs.

Adaptive feature fusion. Models without adaptive feature fusion treat the three graph features equally and sum them up without re-weighting to produce the final node feature. It suffers from an accuracy drop of 2.22% on average, indicating the importance of adaptive feature fusion. Another widely-used approach for feature fusion is to concatenate the features. Although concatenation could learn separated parameters for different features, the features could not be balanced well by these parameters, leading to similar unsatisfying results as summing up the features (see Table 9 in Appendix Section D.2).

Bi-level optimization. Without bi-level optimization, the feature fusion weights are jointly optimized with the model parameters on the training dataset. Its accuracy drops by 2.65% on average. This phenomenon is consistent with the observation in the NAS domain (Liu et al., 2018): training model parameters and feature fusion weights jointly on the same training set would cause over-fitting and thus poor generalization performance.

Feature Fusion Analysis. We measure the importance of the three graph features by computing the proportion for each of them after feature fusion. Formally, the importance of the features is computed by $\mathcal{I}_s = \frac{\pi_i \langle \mathbf{H}_s \rangle}{\pi_1 \langle \mathbf{H}_{\text{ego}} \rangle + \pi_2 \langle \mathbf{H}_{\text{agg}} \rangle + \pi_3 \langle \mathbf{H}_{\text{strc}} \rangle}$, where $s \in \{ego, agg, strc\}$ and $\langle \cdot \rangle$ computes the averaged absolute value of a specific feature.

Figures 2a show the results on `syn-cora`. We could observe three dominant trends. (1) The importance of the ego-node features increases from close to around 0.05 to 0.8 as the homophily of a graph becomes lower. This indicates that the ego-node feature is a more reliable signal than other features for graphs with low homophily. (2) The importance of the aggregated neighborhood feature increases from 0.2 to 0.8 as the homophily increases, echoing the intuition that a node has similar properties to its neighbors on homophilic graphs. (3) All the features have a non-trivial importance score for graphs with homophily within 0.2 and 0.8 (which is common for real graphs). This indicates the importance of all the features in learning node embeddings, echoing insights from ablation study. One thing worth mentioning is, since the graph links in `syn-cora` are generated randomly, which means that the graph structure features cannot capture meaningful information from the adjacency matrix as we expected, it's hard to figure out the trend of this feature from Figure 2a.

Figure 2b shows the results on real graphs. Since real graphs have different intrinsic graph properties, including the raw node features and the graph links, we cannot compare the changes in feature impor-

tance across graphs like what we did for `syn-cora`. Instead, we focus on comparing the importance of different features given specific graphs. Our observations are summarized as follows. (1) For homophilic graphs (i.e., `CiteSeer`, `PubMed`, `Cora`, `Coauthor-CS`, and `Coauthor-Physics`), the aggregated neighborhood features play the most important role to the node classification accuracy. This phenomenon echoes what we observe in `syn-cora`. (2) For graphs with low to medium homophily (i.e., `arXiv-year`, `penn94`, and `twitch-gamer`), the graph structure features take the biggest proportion compared to the other two features, indicating the strong impact of this feature. It is consistent with our ablation study where removing the graph structure features causes the most severe accuracy drops. (3) The graph `genius` almost totally relies on ego-node features. We suspect that the reason is that its node features are sufficient to serve the node classification task. As shown in Table 2, applying MLP on the raw node feature already forms a strong baseline for this graph.

6 Related Works

Graph neural networks (GNNs) have achieved remarkable success on node classification tasks (Gao et al., 2018; Klicpera et al., 2018; Xu et al., 2018b; Wu et al., 2019). Many GNNs (Niepert et al., 2016; Hamilton et al., 2017; Monti et al., 2017; Veličković et al., 2017; Gao et al., 2018; Xu et al., 2018a; Wang et al., 2019) fall into the message passing framework (Gilmer et al., 2017; Battaglia et al., 2018), which iteratively transforms and propagate the messages from the spatial neighborhoods through the graph topology to update the embedding of a target node. To name a few, GCN (Kipf & Welling, 2016) designs a layer-wise propagation rule based on a first-order approximation of spectral convolutions on graphs. GraphSAGE (Hamilton et al., 2017) extends GCN by introducing a recursive node-wise sampling scheme to improve the scalability. Graph attention networks (GAT) (Veličković et al., 2017) enhances GCN with the attention mechanism (Vaswani et al., 2017). Later works (Gao et al., 2018; Xu et al., 2018a; Wang et al., 2019; Chen et al., 2020; Li et al., 2021; Zeng et al., 2021) try to design more expressive GCN variants by overcoming the over-smoothing problem of GCNs (Oono & Suzuki, 2019; Zhang et al., 2022). DeepGCN (Li et al., 2019) utilizes residual connections, dense connections, and dilated convolutions to build deeper GCNs for point cloud semantic segmentation. APPNP (Klicpera et al., 2018) leverages personalized PageRank to improve the propagation scheme. DAGNN (Liu et al., 2020) proposes to decouple the transformation and the propagation operation to increase receptive fields. However, most of the above-mentioned GNNs fail to achieve good performance on heterophilic graphs (Pei et al., 2020; Lim et al., 2021a) because they assume strong homophily in graphs.

Recent works start to pay attention to heterophilic graphs. MixHop (Abu-El-Haija et al., 2019) proposes a graph convolutional layer that utilizes multiple powers of the adjacency matrix to learn general mixed neighborhood information. H2GCN (Zhu et al., 2020) identifies three key designs, ego- and neighbor-embedding separation, higher-order neighbors, and the combination of intermediate representations to boost the representation learning for heterophilic graphs. Generalized PageRank (GPR) GNN (Chien et al., 2020) adaptively controls the contribution of different propagation steps. LINKX (Lim et al., 2021a) separately embeds the adjacency matrix and the node features and then combines them with MLPs. GGCN (Yan et al., 2021) leverages two strategies, including degree correction for adjusting degree coefficients and signed messages for optionally negating the messages, to overcome the over-smoothing problem. Although many of these methods achieve better performance on heterophilic graphs, they achieve comparable, if not worse, accuracy on homophilic graphs than traditional message passing-based GNNs. In this paper, our goal is to design a GNN that can generalize well to the whole spectrum of homophily.

7 Conclusion

In this paper, we propose a graph neural network OGNN, which integrates three classes of graph features including the ego node feature, the aggregated neighborhood feature, and the graph structure feature. OGNN automatically handles graphs with different homophily via adaptive feature fusion and bi-level optimization. Extensive experiments show that OGNN achieves state-of-the-art accuracy performance compared with strong baselines on both synthetic datasets and real datasets covering the full graph homophily spectrum. Additional ablation studies further illustrate the necessity of the three aspects of the graph features and the proposed adaptive features fusion mechanism.

REFERENCES

Sami Abu-El-Haija, Bryan Perozzi, Amol Kapoor, Nazanin Alipourfard, Kristina Lerman, Hrayr Harutyunyan, Greg Ver Steeg, and Aram Galstyan. Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In *international conference on machine learning*, pp. 21–29. PMLR, 2019.

Nicola Apollonio, Paolo G Franciosa, and Daniele Santoni. A novel method for assessing and measuring homophily in networks through second-order statistics. *Scientific reports*, 12(1):1–18, 2022.

Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. Simple and deep graph convolutional networks. In *International Conference on Machine Learning*, pp. 1725–1735. PMLR, 2020.

Eli Chien, Jianhao Peng, Pan Li, and Olgica Milenkovic. Adaptive universal generalized pagerank graph neural network. *arXiv preprint arXiv:2006.07988*, 2020.

Xuanyi Dong and Yi Yang. Searching for a robust neural architecture in four gpu hours. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1761–1770, 2019.

Hongyang Gao, Zhengyang Wang, and Shuiwang Ji. Large-scale learnable graph convolutional networks. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 1416–1424, 2018.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pp. 1263–1272. PMLR, 2017.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems*, 33:22118–22133, 2020.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

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

Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. *arXiv preprint arXiv:1810.05997*, 2018.

Guohao Li, Matthias Muller, Ali Thabet, and Bernard Ghanem. Deepgcns: Can gcns go as deep as cnns? In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 9267–9276, 2019.

Guohao Li, Matthias Müller, Bernard Ghanem, and Vladlen Koltun. Training graph neural networks with 1000 layers. In *International conference on machine learning*, pp. 6437–6449. PMLR, 2021.

Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *Thirty-Second AAAI conference on artificial intelligence*, 2018.

Derek Lim and Austin R Benson. Expertise and dynamics within crowdsourced musical knowledge curation: A case study of the genius platform. In *ICWSM*, pp. 373–384, 2021.

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

Derek Lim, Xiuyu Li, Felix Hohne, and Ser-Nam Lim. New benchmarks for learning on non-homophilous graphs. *arXiv preprint arXiv:2104.01404*, 2021b.

Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018.

Meng Liu, Hongyang Gao, and Shuiwang Ji. Towards deeper graph neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 338–348, 2020.

Federico Monti, Davide Boscaini, Jonathan Masci, Emanuele Rodola, Jan Svoboda, and Michael M Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5115–5124, 2017.

Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. Learning convolutional neural networks for graphs. In *International conference on machine learning*, pp. 2014–2023. PMLR, 2016.

Kenta Oono and Taiji Suzuki. Graph neural networks exponentially lose expressive power for node classification. *arXiv preprint arXiv:1905.10947*, 2019.

Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. Geom-gcn: Geometric graph convolutional networks. *arXiv preprint arXiv:2002.05287*, 2020.

Benedek Rozemberczki and Rik Sarkar. Twitch gamers: a dataset for evaluating proximity preserving and structural role-based node embeddings. *arXiv preprint arXiv:2101.03091*, 2021.

Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.

Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868*, 2018.

Amanda L Traud, Peter J Mucha, and Mason A Porter. Social structure of facebook networks. *Physica A: Statistical Mechanics and its Applications*, 391(16):4165–4180, 2012.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics (tog)*, 38(5):1–12, 2019.

Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *International conference on machine learning*, pp. 6861–6871. PMLR, 2019.

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

Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. In *International conference on machine learning*, pp. 5453–5462. PMLR, 2018b.

Yujun Yan, Milad Hashemi, Kevin Swersky, Yaoqing Yang, and Danai Koutra. Two sides of the same coin: Heterophily and oversmoothing in graph convolutional neural networks. *arXiv preprint arXiv:2102.06462*, 2021.

Zhilin Yang, William Cohen, and Ruslan Salakhudinov. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*, pp. 40–48. PMLR, 2016.

Hanqing Zeng, Muhan Zhang, Yinglong Xia, Ajitesh Srivastava, Andrey Malevich, Rajgopal Kannan, Viktor Prasanna, Long Jin, and Ren Chen. Decoupling the depth and scope of graph neural networks. *Advances in Neural Information Processing Systems*, 34:19665–19679, 2021.

Wentao Zhang, Zeang Sheng, Ziqi Yin, Yuezihan Jiang, Yikuan Xia, Jun Gao, Zhi Yang, and Bin Cui. Model degradation hinders deep graph neural networks. *arXiv preprint arXiv:2206.04361*, 2022.

Elena Zheleva and Lise Getoor. To join or not to join: the illusion of privacy in social networks with mixed public and private user profiles. In *Proceedings of the 18th international conference on World wide web*, pp. 531–540, 2009.

Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. Beyond homophily in graph neural networks: Current limitations and effective designs. *Advances in Neural Information Processing Systems*, 33:7793–7804, 2020.

A REAL-WORLD DATASETS DETAILS

In our experiments, we use the following real-world datasets to evaluate our method and existing GNNs. The homophily of these datasets ranges from $0.222 \sim 0.931$. Table 4 summarizes the detailed dataset statistics.

`Cora` (Sen et al., 2008), `CiteSeer` (Sen et al., 2008), `PubMed` (Sen et al., 2008), and `Coauthor CS & Physics` (Shchur et al., 2018) have high edge homophily and are usually considered as homophilic graphs. For these graphs, we follow the data split in GCN (Kipf & Welling, 2016) and DAGNN (Liu et al., 2020). For `Cora`, `CiteSeer`, and `PubMed`, we randomly sample 20 nodes from each class as the train set, and sample 500 nodes from the rest as the validation set and 1000 nodes as the test set. For `Coauthor CS & Physics`, we randomly sample 20 nodes per class as the train set, 30 nodes per class as the validation set, and the rest nodes as the test set.

`penn94` (Traud et al., 2012), `arXiv-year` (Hu et al., 2020), `genius` (Lim & Benson, 2021), and `twitch-gamer` (Rozemberczki & Sarkar, 2021) are graphs with lower homophily. For these graphs, we follow the data split in LINKX (Lim et al., 2021b;a), which uses the $50\%/25\%/25\%$ nodes as the train/validation/test set respectively.

For all the datasets, we generate 5 random data splits for computing the average and standard deviation of the models' performance.

Table 4: Statistics for the real-world datasets.

	#Classes	#Nodes	#Edges	#Features	degree	edge homophily	Nodes	Edges	Classes
Cora	7	2,708	5,278	1,433	1.949	0.81	papers	citation	research field
CiteSeer	6	3,327	4,552	3,703	1.368	0.736	papers	citation	research field
PubMed	3	19,717	44,324	500	2.248	0.802	papers	citation	research field
Coauthor CS	15	18,333	81,894	6,805	4.467	0.808	authors	co-authors	research field
Coauthor Physics	5	34,493	247,962	8,415	7.189	0.931	authors	co-authors	research field
penn94	2	41,554	1,362,229	5	32.782	0.47	peoples	friends	Gender
arXiv-year	5	169,343	1,166,243	128	6.887	0.222	papers	citation	year
genius	2	421,961	984,979	12	2.334	0.618	users	followers	Gender
twitch-gamer	2	168,114	6,797,557	7	40.434	0.545	Twitch users	followers	Gender

B IMPLEMENTATION DETAILS

B.1 MODEL IMPLEMENTATION DETAILS

The implementation of OGNN basically follows the model we described in Section 4.2, in which we omit some details for simplicity. The complete implementation details of OGNN are listed in Table 5. Before extracting the ego-node feature, the raw input node features are fed into a Dropout layer. In the aggregated neighborhood feature extraction, we reuse the intermediate results of different hops of neighbors to simplify the computation. For example, when computing $\hat{\mathbf{A}}^i\mathbf{H}_{\text{ego}}$, we use the dot product of $\hat{\mathbf{A}} \cdot (\hat{\mathbf{A}}^{i-1}\mathbf{H}_{\text{ego}})$ instead of computing the power of $\hat{\mathbf{A}}$. Moreover, because $\hat{\mathbf{A}}^{i-1}\mathbf{H}_{\text{ego}}$ has the same shape of $\mathbf{H}_{\text{ego}}$, computing $\hat{\mathbf{A}} \cdot (\hat{\mathbf{A}}^{i-1}\mathbf{H}_{\text{ego}})$ is always a sparse-dense matrix multiplication, which is more efficient than computing the power of $\hat{\mathbf{A}}$. Similarly, we use the same strategy to compute $\mathbf{H}_{\text{strc}}$. Because we use the original adjacency matrix instead of a normalized one in structure feature extraction (for accuracy performance purposes), the magnitude of $\mathbf{A}^j$ will grow exponentially with j increasing, which may cause the value out of range problem. Therefore, we utilize Batch Normalization layers to scale down the feature matrix after each adjacency matrix multiplication. In the feature fusion module, we dropout the fused features before activating it with ReLU.

B.2 TRAINING DETAILS

To implement our bi-level optimization training scheme, we utilize two optimizers to train the model parameters $\mathbf{W}$ and the feature fusion parameters $\mathcal{P}$ respectively. The model parameters $\mathbf{W}$ are trained with an Adam optimizer (Kingma & Ba, 2014) $\mathcal{O}_1$ on the training dataset. The learning rate and weight decay rate of $\mathcal{O}_1$ is decided by the hyper-parameter settings, which is described in Section C. On the other hand, the feature fusion parameters $\mathcal{P}$ are trained with another Adam optimizer $\mathcal{O}_2$ on the validation dataset with a fixed learning rate 0.01. We train $\mathcal{P}$ for 10 epochs after training $\mathbf{W}$ for every 20 epochs. When training $\mathcal{P}$, we'll set the Dropout

Table 5: OGNN Model Implementation details

Module	Implementation Details
Input Feature	$\mathbf{X}$
Ego Node Feature	$\mathbf{H}_{\text{ego}} = \text{Linear}(\text{Dropout}(\mathbf{X}))$
Aggregated Neighborhood Feature	$\mathbf{H}_{\text{agg}} = \sum_{i=1}^{s_1} \hat{\mathbf{A}}^i \mathbf{H}_{\text{ego}}$
Graph Structure Feature	$\mathbf{H}_{\text{strc}} = \sum_{j=1}^{s_2} \text{BN}^j(\mathbf{A})\mathbf{W}_{\text{strc}}$, $\text{BN}^j(\mathbf{A}) = \underbrace{\text{BN}(\mathbf{A} \cdot \text{BN}(\ldots \text{BN}(\mathbf{A})))}_{s_2}$
Feature Fusion	$\mathbf{H} = \text{ReLU}(\text{Dropout}(\pi_1\mathbf{H}_{\text{ego}} + \pi_2\mathbf{H}_{\text{agg}} + \pi_3\mathbf{H}_{\text{strc}}))$, $\pi_i = \frac{\exp p_i}{\sum_{j=1}^{3} \exp p_j}$
Prediction Head	$\mathbf{Y}_{\text{pred}} = \text{Softmax}(\mathbf{H}\mathbf{W}_{\text{pred}})$

layers and Batch Normalization layers to evaluation mode. We early stop the model if the validation accuracy does not increase for 100 epochs or the total number of training epochs reaches 3000.

C Hyper-parameter Settings

In our experiments, we use the hidden channels (d), the propagation steps for aggregated neighborhood feature extraction (s_1), the power of the adjacency matrix for graph structure feature extraction (s_2), the learning rate η, the weight decay λ, and the feature normalization (ν) as the hyper-parameters. For all the datasets, we perform grid search over the following hyper-parameter options:

$$
\begin{aligned}
d &\in \{64 \quad 128\} \\
s_1 &\in \{2 \quad 5 \quad 10 \quad 20\} \\
s_2 &\in \{1 \quad 2 \quad 5\} \\
\eta &\in \{0.01 \quad 0.001\} \\
\lambda &\in \{0.001 \quad 0.0005\} \\
\nu &\in \{\text{True} \quad \text{False}\}
\end{aligned}
$$

We also list the best hyper-parameter settings for all the real-world datasets in Table 6

Table 6: Best hyper-parameter settings for the real-world datasets.

	arXiv	penn94	twitch	genius	CiteSeer	PubMed	Cora	CS	Physics
Homophily	0.22	0.47	0.55	0.62	0.74	0.80	0.81	0.81	0.93
d	64	64	128	128	128	128	128	64	128
s_1	2	5	10	2	20	20	20	10	20
s_2	2	2	2	5	2	5	5	5	2
η	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
λ	0.0005	0.0005	0.0005	0.001	0.001	0.0005	0.0005	0.001	0.0005
ν	False	False	False	False	True	True	True	False	False

D Detailed Results

D.1 Detailed Syntheic Dataset Results

We list the detailed results of the experiments on the synthetic datasets in Table 7 and 8, which includes standard deviation of the accuracy performance compared to Table 1. Overall, OGNN outperforms the existing methods on most homophily settings with seven settings at top-1 and four at top-2, and pushes the best accuracy boundary for up to 2.76%.

D.2 Using Concatenation as Feature Fusion

Table 9 shows the experimental results of using concatenation as feature fusion approach instead of summing up the features.

Table 7: Test accuracy of different methods on the graphs with different homophily from 0 to 0.5 in `syn-cora` dataset. **Red** and blue represent top-1 and top-2 ranking in terms of accuracy respectively.

synh	0	0.1	0.2	0.3	0.4	0.5
MLP	$69.20_{\pm 1.92}$	$69.20_{\pm 1.92}$	$69.20_{\pm 1.92}$	$69.20_{\pm 1.92}$	$69.20_{\pm 1.92}$	$69.20_{\pm 1.92}$
GCN	$28.61_{\pm 2.38}$	$30.64_{\pm 1.58}$	$36.03_{\pm 1.15}$	$45.15_{\pm 2.18}$	$51.39_{\pm 1.48}$	$65.04_{\pm 1.60}$
GAT	$29.41_{\pm 2.20}$	$30.32_{\pm 2.52}$	$34.83_{\pm 1.67}$	$43.86_{\pm 1.44}$	$51.15_{\pm 1.25}$	$64.80_{\pm 1.81}$
DAGNN	$34.32_{\pm 1.51}$	$39.49_{\pm 1.18}$	$45.01_{\pm 2.42}$	$54.48_{\pm 3.17}$	$60.51_{\pm 1.40}$	$72.36_{\pm 1.83}$
LINKX	$72.09_{\pm 1.65}$	$70.54_{\pm 1.84}$	$69.76_{\pm 1.07}$	$70.13_{\pm 1.30}$	$71.34_{\pm 1.17}$	$74.53_{\pm 1.83}$
H2GCN	**$76.43_{\pm 1.27}$**	**$73.86_{\pm 3.05}$**	$71.58_{\pm 1.75}$	$72.17_{\pm 1.44}$	$72.95_{\pm 0.76}$	$78.31_{\pm 1.96}$
MIXHOP	$39.44_{\pm 2.98}$	$38.95_{\pm 2.21}$	$41.05_{\pm 2.69}$	$48.93_{\pm 2.72}$	$55.09_{\pm 2.30}$	$64.75_{\pm 1.88}$
GPR-GNN	$67.86_{\pm 2.66}$	$61.96_{\pm 2.53}$	$61.21_{\pm 2.36}$	$64.69_{\pm 2.88}$	$67.67_{\pm 3.41}$	$74.56_{\pm 2.60}$
OGNN	$72.49_{\pm 1.74}$	$73.67_{\pm 1.61}$	**$73.11_{\pm 0.94}$**	**$74.32_{\pm 2.29}$**	**$75.71_{\pm 1.00}$**	**$79.49_{\pm 1.58}$**

Table 8: Test accuracy of different methods on the graphs with different homophily from 0.6 to 1 in `syn-cora` dataset. **Red** and blue represent top-1 and top-2 ranking in terms of accuracy respectively.

synh	0.6	0.7	0.8	0.9	1
MLP	$69.20_{\pm 1.92}$	$69.20_{\pm 1.92}$	$69.20_{\pm 1.92}$	$69.20_{\pm 1.92}$	$69.20_{\pm 1.92}$
GCN	$74.48_{\pm 1.36}$	$82.22_{\pm 2.20}$	$91.21_{\pm 1.15}$	$96.19_{\pm 0.65}$	$99.92_{\pm 0.18}$
GAT	$74.34_{\pm 2.03}$	$81.45_{\pm 1.72}$	$90.46_{\pm 1.12}$	$95.79_{\pm 0.71}$	**$100.0_{\pm 0.00}$**
DAGNN	$80.00_{\pm 1.26}$	$86.49_{\pm 2.13}$	$93.32_{\pm 1.45}$	**$97.48_{\pm 0.31}$**	$99.95_{\pm 0.07}$
LINKX	$77.16_{\pm 1.22}$	$80.35_{\pm 1.47}$	$83.30_{\pm 1.23}$	$87.59_{\pm 1.07}$	$89.60_{\pm 1.86}$
H2GCN	$83.27_{\pm 1.25}$	**$87.43_{\pm 0.88}$**	$92.09_{\pm 0.46}$	$97.00_{\pm 0.28}$	$98.98_{\pm 0.62}$
MIXHOP	$74.45_{\pm 1.44}$	$82.44_{\pm 2.69}$	$91.45_{\pm 1.11}$	$96.25_{\pm 0.89}$	**$100.0_{\pm 0.00}$**
GPR-GNN	$80.19_{\pm 1.57}$	$86.68_{\pm 0.69}$	$93.54_{\pm 1.36}$	$97.45_{\pm 0.28}$	**$100.0_{\pm 0.00}$**
OGNN	**$84.67_{\pm 1.30}$**	$87.32_{\pm 1.25}$	**$93.70_{\pm 1.13}$**	$94.75_{\pm 3.47}$	$99.95_{\pm 0.07}$

Table 9: Using concatenation as feature fusion.

Homophily	arXiv 0.22	penn94 0.47	twitch 0.55	genius 0.62	CiteSeer 0.74	PubMed 0.80	Cora 0.81	CS 0.81	Physics 0.93	Δ Avg. -
OGNN	**56.50**	**85.43**	**65.93**	**89.91**	**71.50**	**81.10**	**84.12**	**92.83**	**93.87**	-
w/ sum	53.45 $_{3.05\downarrow}$	84.35 $_{1.08\downarrow}$	65.76 $_{0.17\downarrow}$	87.64 $_{2.27\downarrow}$	67.72 $_{3.78\downarrow}$	76.04 $_{5.06\downarrow}$	82.48 $_{1.64\downarrow}$	91.13 $_{1.70\downarrow}$	92.64 $_{1.23\downarrow}$	2.22↓
w/ concate	56.55 $_{0.05\uparrow}$	82.14 $_{3.29\downarrow}$	65.87 $_{0.06\downarrow}$	89.44 $_{0.47\downarrow}$	66.70 $_{0.47\downarrow}$	74.70 $_{6.40\downarrow}$	81.10 $_{3.02\downarrow}$	89.73 $_{3.10\downarrow}$	92.74 $_{1.13\downarrow}$	2.47↓