IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 23 November 2024, accepted 15 December 2024, date of publication 20 December 2024,
date of current version 30 December 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3520903

RESEARCH ARTICLE

Information-Enhanced Graph Neural Network for Transcending Homophily Barriers

XIAO LIU, LIJUN ZHANG, AND HUI GUAN
College of Information and Computer Sciences, University of Massachusetts Amherst, Amherst, MA 01002, USA

Corresponding author: Xiao Liu (xiaoliu1990@cs.umass.edu)

This work was supported in part by the National Science Foundation under Grant CNS-2312396, Grant CNS-2338512, Grant CNS-2224054, and Grant DMS-2220211.

ABSTRACT Homophily and heterophily are intrinsic properties of graphs that describe whether linked nodes share similar properties. While Message Passing Neural Networks (MPNNs) have shown remarkable success in node classification tasks, their performance often deteriorates within specific homophily ranges, which we term **the gray area**. In this work, we identify and theoretically demonstrate the challenges faced by MPNNs in this gray area, highlighting the limitations of existing approaches in addressing it. To overcome these limitations, we propose the **INformation-enhanced Graph Neural Network (INGNN)**, which introduces a novel framework that integrates three complementary features–ego-node features, graph structure features, and aggregated neighborhood features–through an adaptive feature fusion mechanism based on bi-level optimization. This design enables INGNN to transcend the homophily barriers and generalize effectively across the entire homophily spectrum. We validate the effectiveness of INGNN through extensive experiments on both synthetic and real-world datasets with different graph homophily. Specifically, INGNN outperforms 12 state-of-the-art MPNNs with an average rank of 1.78 on 9 real node classification datasets. Our ablation studies further show experimental evidence of how the integrated features contribute to the model's performance under different homophily settings. INGNN is open-sourced and available at https://github.com/xl1990/ingnn.

INDEX TERMS Graph neural networks (GNNs), graph homophily, graph node classification.

I. INTRODUCTION

Graph neural networks (GNNs) have been proven to be a powerful approach to learning graph representations for node classification tasks [1]. Researchers have proposed different GNN model designs based on the underlying assumption of graph homophily or heterophily. On the one hand, many works [1], [2], [3], [4], [5], [6], [7] assume strong graph homophily, which means that 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 [8], [9]. On the other hand, some works empirically demonstrate the poor performance of homophily-based GNNs on heterophilic graphs and design specific GNNs for heterophilic graphs. For instance, LINKX [10] proposes to separately embed the adjacency matrix and the node features, and GloGNN [11] performs node neighborhood aggregation from the entire set of nodes in the graph to find Global homophily. They perform much better on graphs with strong heterophily but achieve a worse accuracy on homophilic graphs than homophily-based GNNs.

Neither GNNs designed for homophily nor those designed for heterophily are optimal, since they invisibly establish a barrier between homophily and heterophily for graph representation learning — a GNN can work well on heterophilic graphs or homophilic graphs, but not both. However, the real-world graph data 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` [12] is a social network graph in which users follow each

The associate editor coordinating the review of this manuscript and approving it for publication was Giacomo Fiumara.

other by their interests in the game, 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, which we call the "gray area".

Recent studies [13], [14], [15], [16], [17], [18], [19] employ various strategies to tackle both homophilic and heterophilic graphs. They are empirically demonstrated to achieve good generalization performance across the entire spectrum of graph homophily. For instance, H2GCN [13] identifies a set of key designs that boost learning from the graph structure under heterophily without compromising performance for homophilic graphs. GPR-GNN [15] adaptively learns the generalized pagerank weights, irrespective of the level of homophily or heterophily exhibited by the node labels. However, these methods do not adequately address the critical question: what specific conditions of homophily under which traditional MPNNs exhibit underperformance. Furthermore, they do not provide a systematic approach to improve performance under such conditions.

In this work, we first demonstrate that MPNNs would underperform in some specific homophily ranges, revealing why homophily-assumption–based methods cannot perform well in what we call gray area. Then we exploit the original graph information to systematically solve the gray area challenge, namely, transcending the homophily barriers formed by the gray area. The two types of original graph information, graph structures, and node features, can be used together and separately, leading to three features including *ego-node features*, *graph structure features*, and *aggregated neighborhood features*. MPNNs, however, take advantage of only the last feature for downstream tasks. We find that all three features are necessary for MPNNs to generalize on the whole spectrum of graph homophily and prove it both theoretically and empirically. Based on the insight, we propose an INformation-enhanced MPNN (INGNN), which adaptively fuses these three features to derive nodes' representation and achieves outstanding accuracy compared to state-of-the-art baselines. We summarize our main contributions as follows:

- **Theoretical Contribution.** We show theoretically that MPNNs would underperform within specific homophily ranges that could not be simply categorized as homophily or heterophily, which we refer to as gray area in this paper.
- **The INGNN Model.** We propose INformation-enhanced Graph Neural Network (INGNN), which uplifts MPNNs' accuracy performance over the whole homophily spectrum. INGNN automatically integrates graph structure features and ego-node features into the MPNN framework using adaptive feature fusion under a bi-level optimization framework.
- **Empirical Contribution.** We conduct extensive experiments to compare INGNN with state-of-the-art GNNs using synthetic and real graph benchmarks that cover the full homophily spectrum. INGNN outperforms 12 baseline models and achieves an average rank of 1.78 on 9 real datasets. INGNN achieves higher node classification accuracy, 4.17% higher than GCN [1], 6.48% higher than LINKX [10], and 3.81% higher than H2GCN [13], on average over the real datasets.
- **Ablation Study.** Our ablation study demonstrates that all three features, the fusing mechanism, and the optimization procedure play indispensable roles in different types of graphs. Moreover, we empirically verify the necessity of the three features by enhancing GCN [1], H2GCN [13], and LINKX [10] with the three features they did not originally have, which improves their classification accuracy by 1.50%, 0.40%, and 4.97%, respectively.

II. RELATED WORKS

A. GNNs FOR HOMOPHIC GRAPHS.

Graph neural networks (GNNs) have achieved remarkable success on node classification tasks [4], [5], [6], [7]. Many GNNs [2], [3], [4], [20], [21], [22], [23] fall into the message passing framework [8], [9], which iteratively transforms and propagates the messages from the spatial neighborhoods through the graph topology to update the embedding of a target node. To name a few, GCN [1] designs a layer-wise propagation rule based on a first-order approximation of spectral convolutions on graphs. GraphSAGE [2] extends GCN by introducing a recursive node-wise sampling scheme to improve scalability. Graph attention networks (GAT) [3] enhance GCN with the attention mechanism [24]. Later work [4], [22], [23], [25], [26], [27] tried to design more expressive GCN variants by overcoming the over-smoothing problem of GCNs [28], [29]. DeepGCN [30] utilizes residual connections, dense connections, and dilated convolutions to build deeper GCNs for point cloud semantic segmentation. APPNP [5] leverages personalized PageRank to improve the propagation scheme. DAGNN [31] proposes to decouple the transformation and the propagation operation to increase receptive fields. However, most of the GNNs mentioned above fail to achieve good performance on heterophilic graphs [10], [32] because they assume strong homophily in graphs.

B. GNNs FOR HETEROPHILIC GRAPHS

Recent works have started to pay attention to heterophilic graphs. LINKX [10] increases performance on heterophilic graphs by combining the embedded adjacency matrix and node features with MLPs. GloGNN [11] performs node neighborhood aggregation from the whole set of nodes in the graph to find Global homophily, thereby tackling the challenge posed by heterophilic graphs. Although these approaches enhance performance on heterophilic graphs, their effectiveness on homophilic graphs does not match that of homophily-based MPNNs.

C. GNNs FOR BOTH HOMOPHIC AND HETEROPHILIC GRAPHS

Some other works explore diverse strategies to achieve good performance on graphs over the whole homophily spectrum as we focus on in this paper. MIXHOP [14] proposes a graph convolutional layer that utilizes multiple powers of the adjacency matrix to learn general mixed neighborhood information, which demonstrates effective generalization capabilities on synthetic graphs with different homophily. H2GCN [13] identifies three key designs, ego- and neighbor-embedding separation, higher-order neighbors, and the combination of intermediate representations to boost representation learning for heterophilic graphs without sacrificing performance on homophilic graphs. Generalized PageRank (GPR) GNN [15] adaptively learns the generalized pagerank weights for node embeddings in different propagation steps to better adapt to the homophily or heterophily structure of a graph. GGCN [16] leverages two strategies, including degree correction to adjust degree coefficients and signed messages to optionally negate messages, to overcome the over-smoothing problem. ACM-GCN [17] proposes a multi-channel mixing mechanism, enabling adaptive filtering at nodes with different homophily. BernNet [18] learns an arbitrary spectral filter such as those in GCN, DAGNN, and APPNP via the Bernstein polynomial approximation. Similarly, AdaGNN [19] introduces trainable filters to capture the varying importance of different frequency components. Both BernNet and AdaGNN exhibit compelling empirical results on homophilic and heterophilic graphs.

While these methods introduce varied strategies to bolster performance on heterophilic graphs without compromising efficacy on homophilic ones, they do not thoroughly investigate the specific homophily conditions under which MPNNs are inherently predisposed to underperform in node classification tasks. In this paper, we aim to address this research gap by elucidating the 'gray area' challenge inherent to MPNNs and advocating for a comprehensive exploitation of the graph information to surmount this challenge.

III. MPNNs UNDERPERFORMANCE IN GRAY AREA

This section starts with the notations and the definitions of graph homophily and the MPNNs, then states the gray area challenge for MPNNs with theoretical proof.

A. NOTATIONS AND PRELIMINARIES

1) NOTATIONS

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$ 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 and the raw feature vectors of all nodes form a feature matrix $\mathbf{X} \in \mathbb{R}^{N\times D}$. We refer to the adjacency matrix $\mathbf{A}$ and the feature matrix $\mathbf{X}$ together as *original graph information*.

2) 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 to calculate graph homophily [13], [32], [33], [34]. In this paper, we adopt the most widely used edge homophily [13]. Edge homophily ranges from 0 to 1. Graphs with edge homophily close to 1 are called homophilic graphs, while those with edge homophily close to 0 are called heterophilic graphs.

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 edges that connect two nodes with the same class label.*

3) MESSAGE PASSING NEURAL NETWORKS (MPNNs)

Most GNNs follow the message-passing framework where the hidden state of a node $v \in \mathcal{V}$ depends on the features of its neighbors and itself. 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 and h^0 is the raw feature vector of the node (i.e., $h^0 = x$). 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 of the neighborhoods with Φ into messages and then aggregates them with Γ. After that, Ψ updates the node's feature with the aggregated message. By stacking multiple message-passing layers, GNNs iteratively propagate information to the target node from multi-hop neighborhoods. We call this process as *the node features aggregation* and the extracted features (i.e., $\{h_v^l, l = 1, 2, \cdots\}$) as *aggregated neighborhood features*.

B. GRAY AREA CHALLENGE

We prove that in a specific homophily range, which we refer to as gray area, the node features aggregation process may hurt MPNN's performance.

Theorem 1: There exists a graph homophily range in which the misclassification rate of a classification model will increase after node features aggregation.

The key of proving Theorem 1 is to identify a homophily range and show that if the homophily of a graph falls into this range, classifying with aggregated neighborhood features only will increase the misclassification rate compared with using raw node features $\mathbf{X}$. To prove that there must exist such a homophily range, we identify three specific homophily cases in which the misclassification rate changes (increases or decreases) when using aggregated neighborhood features. Then based on the intermediate value theorem, we can prove the existence of the homophily range with an increased misclassification rate. Without loss of generality, we prove the theorem using a binary node classification problem.

Proof of Theorem 1: We first introduce the necessary definitions related to the problem. Suppose that there is a regular graph $\mathcal{G}$ with homophily $\mathcal{H}$ and degree d. The nodes $\mathcal{V}$ in $\mathcal{G}$ can be categorized into two classes, and the feature of a node v from each class is sampled from two normal distribution $N(\mu_1, \sigma_1^2)$ and $N(\mu_2, \sigma_2^2)$ respectively. We call the probability density function of the two distributions $f_1(x)$ and $f_2(x)$. An optimal classifier for a binary node classification problem will categorize a node into a class in which the node feature has a higher probability of being sampled from its distribution. The misclassification rate of such an optimal classifier ϵ is the integral over the overlapping area of $f_1(x)$ and $f_2(x)$,

$$\epsilon = 1 - \phi_1(z) + \phi_2(z), \tag{2}$$

where ϕ_1 and ϕ_2 are the two cumulative distribution functions and z is the solution of $f_1(x) = f_2(x)$. In particular, we denote the misclassification rate of using the raw node features as ϵ_{raw}.

Next, we discuss how the two node feature distributions change after aggregation, which will affect the misclassification rate. We consider the case where node features are aggregated by an averaging function (i.e., Γ is the mean function in Eq. 1). Since the graph homophily is $\mathcal{H}$, for an arbitrary node, there are $\mathcal{H}d$ neighbors from the same class and $(1-\mathcal{H})d$ from the other class in average. The aggregated features of the nodes from class 1 follow the distribution of

$$N(\mathcal{H}\mu_1 + (1-\mathcal{H})\mu_2, \frac{\mathcal{H}\sigma_1^2 + (1-\mathcal{H})\sigma_2^2}{d}). \tag{3}$$

Similarly, those from class 2 follow

$$N(\mathcal{H}\mu_2 + (1-\mathcal{H})\mu_1, \frac{\mathcal{H}\sigma_2^2 + (1-\mathcal{H})\sigma_1^2}{d}). \tag{4}$$

Although it is challenging to obtain the analytical solution for z and ϵ with the distributions of the two classes after aggregation, we could easily identify three special cases $\mathcal{H} \in \{1.0, 0.0, 0.5\}$ whose misclassification rate based on aggregated neighborhood features becomes higher or lower compared to ϵ_{raw}.

- **Case 1:** If $\mathcal{H} = 1.0$, the distribution of aggregated neighborhood features are

$$N(\mu_1, \frac{\sigma_1^2}{d}) \text{ and } N(\mu_2, \frac{\sigma_2^2}{d}), \tag{5}$$

for class 1 and 2 respectively. Compared to the original feature distributions, the mean values are the same but the variances are reduced, leading to a smaller overlapping area of the distributions. In other words, the misclassification rate is reduced, i.e., $\epsilon_{\mathcal{H}=1.0} < \epsilon_{raw}$.

- **Case 2:** If $\mathcal{H} = 0.0$, the distribution of aggregated neighborhood features are

$$N(\mu_2, \frac{\sigma_2^2}{d}) \text{ and } N(\mu_1, \frac{\sigma_1^2}{d}), \tag{6}$$

for class 1 and 2 respectively. Compared to Case 1, the distributions are exchanged, which means the misclassification rate should be the same as $\mathcal{H} = 1.0$, namely $\epsilon_{\mathcal{H}=0.0} = \epsilon_{\mathcal{H}=1.0} < \epsilon_{raw}$.

- **Case 3:** If $\mathcal{H} = 0.5$, the distribution of aggregated neighborhood features are

$$N(\frac{\mu_1 + \mu_2}{2}, \frac{\sigma_1^2 + \sigma_2^2}{2d}), \tag{7}$$

for both class 1 and 2. In this case, the two distributions are indistinguishable for any classifier, i.e., $\epsilon_{\mathcal{H}=0.5} = 1.0 > \epsilon_{raw}$, which is the worst case for binary classification.

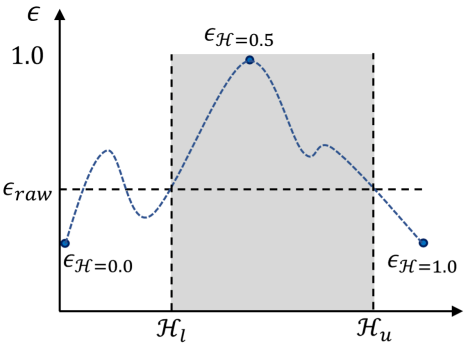


FIGURE 1. The misclassification rate when using the aggregated neighborhood features for graphs with different homophily.

Figure 1 shows the misclassification rate when using aggregated features for graphs with different $\mathcal{H}$. Since the misclassification rate (Eq. 2) is a continuous function and $\epsilon_{\mathcal{H}=0.0} = \epsilon_{\mathcal{H}=1.0} < \epsilon_{raw} < \epsilon_{\mathcal{H}=0.5}$, according to the intermediate value theorem, there exist $0.0 < \mathcal{H}_l < 0.5$ and $0.5 < \mathcal{H}_u < 1.0$, such that for all $\mathcal{G}$ with homophily $\mathcal{H} \in (\mathcal{H}_l, \mathcal{H}_u)$, i.e. the gray area in Figure 1, the misclassification rate of the aggregated features is higher than that of the raw features. In other words, there exists a graph homophily range in which the misclassification rate increases after aggregation. We refer to the range as **gray area** in this paper. Therefore the formal definition of gray area is,

*Definition 2: Given that edge homophily ranges from 0 to 1, the **gray area** is defined as a homophily range $\mathcal{H} \in (\mathcal{H}_l, \mathcal{H}_u)$, where $0.0 < \mathcal{H}_l < 0.5 < \mathcal{H}_u < 1.0$. Within this range, the misclassification rate of the aggregated neighborhood features is higher than that of the raw node features. Formally,*

$$\begin{aligned} &Gray\ Area = \{\mathcal{H} \mid \mathcal{H} \in (\mathcal{H}_l, \mathcal{H}_u)\} \\ &0.0 < \mathcal{H}_l < 0.5 < \mathcal{H}_u < 1.0, \epsilon(\mathcal{H}) > \epsilon_{raw}, \end{aligned} \tag{8}$$

where $\epsilon(\mathcal{H})$ represents the misclassification rate after aggregation, and ϵ_{raw} denotes the misclassification rate using raw node features.

We further extend the proof for Theorem 1 to multiple classes. Assume that the nodes in the graph could be categorized into K classes, while the node feature distribution of class k is $N(\mu_k, \sigma_k^2)$. Then given a graph homophily $\mathcal{H}$, the node feature distribution of class k after one step aggregation

will be,

$$N(\mathcal{H}\mu_k + \frac{1-\mathcal{H}}{K-1}\sum_{i\neq k}\mu_i, \frac{\mathcal{H}\sigma_k^2 + \frac{1-\mathcal{H}}{K-1}\sum_{i\neq k}\sigma_i^2}{d}). \quad (9)$$

Then similar to our discussion for binary classification, there are two special cases when $\mathcal{H} \in \{1.0, \frac{1}{K}\}$.

- **Case 1:** If $\mathcal{H} = 1.0$, the distribution of aggregated neighborhood features of class k is

$$N(\mu_k, \frac{\sigma_1^2}{d}) \text{ and } N(\mu_k, \frac{\sigma_2^2}{d}). \quad (10)$$

  For the same reason for binary classification case, $\epsilon_{\mathcal{H}=1.0} < \epsilon_{raw}$.
- **Case 2:** If $\mathcal{H} = \frac{1}{K}$, the distribution of aggregated neighborhood features are

$$N(\frac{\sum_{i=1}^{K}\mu_i}{K}, \frac{\sum_{i=1}^{K}\sigma_i^2}{Kd}), \quad (11)$$

  for all classes. In this case, the distributions are indistinguishable for any classifier, i.e., $\epsilon_{\mathcal{H}=\frac{1}{K}} = 1.0 > \epsilon_{raw}$, which is the worst case for the multi-class classification.

Then according to the intermediate value theorem, there exists $\frac{1}{K} < \hat{\mathcal{H}} < 1.0$ such that for all $\frac{1}{K} \leq \mathcal{H} < \hat{\mathcal{H}}$, $\epsilon_{\mathcal{H}} > \epsilon_{raw}$. Hence we prove that Theorem 1 holds for multiple classes.

■

C. EXTENSION TO NON-GAUSSIAN FEATURES

Our primary analysis above assumes that node features are sampled from Gaussian distributions for mathematical convenience. However, real-world graph data often exhibit non-Gaussian characteristics. To evaluate the generality of the gray area hypothesis, we extend our analysis to include other distributions, such as uniform and exponential distributions, which are representative of commonly observed cases.

For uniform distributions, where features of nodes from different classes are sampled from distinct intervals, and for exponential distributions, where features are sampled from distributions with varying rates, the aggregation process leads to significant overlaps in feature distributions within the gray area. This overlap arises because aggregated neighborhood features tend to converge towards similar values as they mix features from multiple classes if the graph homophily nears 0.5.

The increased overlap between feature distributions reduces their separability, thereby impairing the ability of a classifier to distinguish between classes. As a result, the misclassification rate of aggregated features exceeds that of raw features, consistent with the gray area hypothesis. Although the exact boundaries of the gray area may vary depending on the feature distribution, our findings confirm that the phenomenon persists across different types of distributions.

IV. INFORMATION-ENHANCED GNN (INGNN)

We propose to solve the gray area challenge by fully exploiting the graph information. Specifically, given a graph $\mathcal{G}$, there are three features $\mathbf{H}_{\text{agg}} = f_{\text{agg}}(\mathbf{X}, \mathbf{A})$, $\mathbf{H}_{\text{ego}} = f_{\text{ego}}(\mathbf{X})$, and $\mathbf{H}_{\text{strc}} = f_{\text{strc}}(\mathbf{A})$, which result from the use of the two original graph information, the feature matrix $\mathbf{X}$ and the adjacency matrix $\mathbf{A}$, together and separately. Our basic idea is to integrate $\mathbf{H}_{\text{ego}}$ and $\mathbf{H}_{\text{strc}}$ into MPNNs to address the gray area challenge.

The first feature $\mathbf{H}_{\text{agg}}$, namely the aggregated neighborhood feature, is exactly what MPNNs are computing. To extract this feature, we follow the state-of-the-art design where feature transformation and propagation are decoupled [31]. It transforms and propagates the raw node features $\mathbf{X}$ and combines the features of different propagation steps to enlarge the receptive field:

$$\mathbf{H}_{\text{agg}} = \sum_{i=1}^{s_1}\hat{\mathbf{A}}^i\mathbf{X}\mathbf{W}_{\text{agg}}, \quad \hat{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}, \quad (12)$$

where $\hat{\mathbf{A}}$ is the normalized adjacency matrix, and $\mathbf{D}$ is the diagonalized node degrees. s_1 is the maximum propagation step.

The other two features $\mathbf{H}_{\text{ego}}$ and $\mathbf{H}_{\text{strc}}$, refer as *ego-node features* and *graph structure features*, can be integrated to alleviate the MPNNs' underperformance in the gray area. We further propose to automatically combine the three features for graphs with different homophily through *adaptive feature fusion*. The three newly introduced components together with the aggregated neighborhood feature form the proposed Information-Enhanced GNN (INGNN) and we will elaborate on these components in the following subsections.

A. EGO-NODE FEATURE EXTRACTOR

We first introduce a separate feature extractor for ego-node features. To extract the ego-node features, we use a simple linear transformation of the node feature matrix $\mathbf{X} \in \mathbb{R}^{N\times D}$:

$$\mathbf{H}_{\text{ego}} = \mathbf{X}\mathbf{W}_{\text{ego}}, \quad \mathbf{W}_{\text{ego}} \in \mathbb{R}^{D\times d}, \quad (13)$$

where $\mathbf{W}_{\text{ego}}$ is the transformation matrix. We use linear transformation instead of MLP because linear transformation consistently achieves the best accuracy in our empirical evaluation.

Rationale. In Theorem 1, we show that using aggregated neighborhood features only could increase misclassification rates under some graph homophily settings. To alleviate the underperformance phenomenon, we prove that ego-node features can depress the node misclassification rate in these settings. This is also empirically supported by our ablation study that the ego node feature can improve the accuracy of INGNN by 2.11% (see Section V-B).

Corollary 1: Combining ego-node features with aggregated neighborhood features, i.e., $[f_{agg}(\mathbf{X}, \mathbf{A}), f_{ego}(\mathbf{X})]$ can depress the misclassification rate in the gray area of Theorem 1.

Proof of Corollary 1: Recall that the misclassification rate of using ego-node features $f_{ego}(\mathbf{X})$ is ϵ_{raw}, and that of using aggregated neighborhood features $f_{agg}(\mathbf{X}, \mathbf{A})$ is larger than ϵ_{raw} when the graph homophily falls in the gray area. Therefore, combining $f_{ego}(\mathbf{X})$ could reconcile the negative effects of $f_{agg}(\mathbf{X}, \mathbf{A})$ and thus reduce the misclassification rate in the gray area.

B. GRAPH STRUCTURE FEATURE EXTRACTOR

We also propose to extract *the graph structure feature* by computing the combination of different powers of the adjacency matrix after a linear transformation:

$$\mathbf{H}_{\text{strc}} = \sum_{j=1}^{s_2} \mathbf{A}^j \mathbf{W}_{\text{strc}}, \quad \mathbf{W}_{\text{strc}} \in \mathbb{R}^{N \times d}, \tag{14}$$

where $\mathbf{W}_{\text{strc}}$ is the transformation matrix, and s_2 is the maximum power of the adjacency matrix. We use a simple linear transformation for its efficiency and we do not observe performance gains from using an MLP.

Rationale. The root cause of the elevated misclassification rate within gray area is the conflation of node feature distributions after aggregation, which renders the nodes indistinguishable from each other. To address this, we posit that introducing a unique identifier for each node – an identifier that remains invariant under node aggregation – would enable our model to accurately classify nodes, even when the aggregated neighborhood features converge towards identical representations. Drawing inspiration from recent works that consider the adjacency matrix as a separate discriminative feature [10], [11], we propose to leverage both the adjacency matrix and its s_2-hop variants as such unique identifiers, and to extract graph structure features from the graph. We hypothesize that these extracted features will serve as a stabilizing anchor as well as a spatial embedding, mitigating the performance degradation observed in these challenging homophily scenarios.

Empirically, our ablation study shows that the graph structure feature can improve the accuracy of INGNN by 4.35% on average (see Section V-B). Furthermore, by incorporating the graph structure feature into an existing MPNN, H2GCN [13], we observe an average accuracy improvement of 0.40% (see Section V-D). These results demonstrate the effectiveness of the graph structure feature in improving the accuracy of MPNNs.

C. ADAPTIVE FEATURE FUSION

We further introduce an adaptive feature fusion module, which assigns a trainable scalar importance score for each feature so that it can *automatically* learn the features' importance from the input graph:

$$\mathbf{H} = \sigma(\pi_1 \mathbf{H}_{\text{ego}} + \pi_2 \mathbf{H}_{\text{agg}} + \pi_3 \mathbf{H}_{\text{strc}}), \quad \pi_i = \frac{\exp p_i}{\sum_{j=1}^{3} \exp p_j}, \tag{15}$$

where $\mathbf{H}$ is the fused feature and σ is the 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.

Rationale. The rationale behind adopting this feature fusion method is two-fold. First, the three features reflect distinct aspects of the graph information, which will have different importance for the representation learning on the graphs with diverse homophily. Therefore, we propose to assign weights to the features instead of treating them equally. Later, Table 4 will show that adaptive feature fusion yields improved accuracy compared to simple summation or concatenation, with performance gains of 1.84% and 2.09% respectively. We also verified the importance of the features for different graphs and the superiority of weighted feature fusion in the ablation study detailed in Section V-B. Second, a common practice for combining features with weights is to treat the assigned weights as hyperparameters, which requires significant searching costs for different graphs. To avoid the time-consuming hyperparameter search, we borrow the idea of *bi-level optimization*, from the Neural Architecture Search domain (NAS) [35], [36], to adaptively learn the feature weights together with the model parameters, which is more efficient and could be automatically generalized to the graphs with different homophily.

Suppose that the model parameters are $\mathbf{W}$ and the parameters for feature fusion are $\mathcal{P}$. The loss function of the node classification tasks is:

$$\mathcal{L}(\mathbf{W}, \mathcal{P}, \mathcal{G}, \mathbf{X}, \mathbf{Y}) = -\frac{1}{|\mathbf{Y}|} \sum_{y_i \in \mathbf{Y}} y_i^{\mathrm{T}} log(\hat{y}_i), \tag{16}$$

where $\mathbf{Y}$ is ground-truth labels and $\hat{y}_i \in \mathbf{Y}_{\text{pred}}$ is the prediction of our model for node i. The objective of our bi-level optimization is:

$$\begin{aligned} &\min_{\mathcal{P}} \ \mathcal{L}_{\text{valid}}(\mathbf{W}^*, \mathcal{P}, \mathcal{G}, \mathbf{X}_{\text{valid}}, \mathbf{Y}_{\text{valid}}), \\ &\text{s.t. } \mathbf{W}^* = \arg\min_{\mathbf{W}} \mathcal{L}_{\text{train}}(\mathbf{W}, \mathcal{P}, \mathcal{G}, \mathbf{X}_{\text{train}}, \mathbf{Y}_{\text{train}}). \end{aligned} \tag{17}$$

In short, we optimize the model parameters $\mathbf{W}$ on the train set, while alternatively optimizing the feature fusion parameters $\mathcal{P}$ on the validation set. This training strategy outperforms training the feature fusion parameters together with the model parameters using the same optimizer by an average of 2.27% across 9 datasets, as shown in the ablation study (see Table 6).

D. IMPLEMENTATION DETAILS

The complete implementation details of INGNN are listed in Table 1. We highlight two implementation considerations. First, to improve the efficiency of computing the power of the adjacency matrix $\hat{\mathbf{A}}^i$, we reuse the intermediate results of different hops of neighbors to simplify the computation. Specifically, we compute $\hat{\mathbf{A}}^i \mathbf{H}_{\text{ego}}$ recursively

TABLE 1. INGNN model implementation details.

Module	Implementation Details
Input Feature	$\mathbf{X}$
Ego Node features	$\mathbf{H}_{\text{ego}} = \text{Linear}(\text{Dropout}(\mathbf{X}))$
Aggregated Neighborhood features	$\mathbf{H}_{\text{agg}} = \sum_{i=1}^{s_1} \hat{\mathbf{A}}^i \mathbf{H}_{\text{ego}}$
Graph Structure features	$\mathbf{H}_{\text{strc}} = \sum_{j=1}^{s_2} \underbrace{\text{BN}(\mathbf{A} \cdot \text{BN}(\ldots \text{BN}(\mathbf{A}\mathbf{W}_{\text{strc}})))}_{j}$
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	$\mathbf{Y}_{\text{pred}} = \text{Softmax}(\mathbf{H}\mathbf{W}_{\text{pred}})$

as $\hat{\mathbf{A}} \cdot (\hat{\mathbf{A}}^{i-1}\mathbf{H}_{\text{ego}})$ instead of $(\hat{\mathbf{A}}^i)\mathbf{H}_{\text{ego}}$ in the aggregated node feature $\mathbf{H}_{\text{agg}}$. In each recursion, the only operation employed is a sparse-dense matrix multiplication, which is more efficient than directly computing the power of the adjacency matrix. We adopt the same strategy when computing the graph structure feature $\mathbf{H}_{\text{strc}}$. Second, to ensure numerical stability when computing $\mathbf{H}_{\text{strc}}$, we incorporate batch normalization to normalize the scale of the feature matrix after each adjacency matrix multiplication.

1) BI-LEVEL OPTIMIZATION IMPLEMENTATION

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 [37] $\mathcal{O}_1$ on the training dataset. The learning rate and weight decay rate of $\mathcal{O}_1$ is decided by the hyper-parameter settings. 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 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.

2) TIME COMPLEXITY ANALYSIS

We show that our model has linear scalability with respect to the number of nodes N. The time complexity for extracting the ego-node features is $\mathcal{O}(N \cdot d^2)$, where N represents the number of nodes in the graph, and d denotes the dimension of the hidden feature. Extracting the aggregated neighborhood features requires $\mathcal{O}(s_1 Md)$ for aggregating features from s_1-hop neighbors using sparse matrix multiplications, where M stands for the number of edges in the graph. Similarly, the time complexity for extracting the graph structure features is $\mathcal{O}(s_2 Md)$. The feature fusion step necessitates $\mathcal{O}(Nd)$ for reweighting and summing up the features. Lastly, the linear classifier takes $\mathcal{O}(NdC)$ to perform predictions. Assuming that the average degree of the graph is constant D, we have $M = ND$. Consequently, the total time complexity of our proposed model INGNN is $\mathcal{O}(N \cdot d^2) + \mathcal{O}(s_1 NDd) + \mathcal{O}(s_2 NDd) + \mathcal{O}(Nd) + \mathcal{O}(NdC)$, which increases linearly with the number of nodes N.

V. EXPERIMENTS

We conduct experiments on both synthetic and real datasets to examine the efficacy of our model in terms of test accuracy compared to baselines in Section V-A. We then present ablation studies to show the effectiveness of the three feature extractors, adaptive feature fusion, and bi-level optimization in Section V-B. We also examine the insights from feature fusion in Section V-C and show the benefits of adding the proposed graph structure features and ego-node features to existing methods in Section V-D.

Datasets: We generate synthetic graphs `syn-cora` with the approach in H2GCN [13]. 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 [38]. The edges of the graph are generated gradually according to the given homophily. We evaluate the average test accuracy over five trials on these graphs with the official data splits (25%, 25%, 50%, for each class).

We also evaluate our method and existing GNNs on 9 real-world datasets. The statistics of the datasets are summarized in Table 2 and the homophily of these datasets ranges from $0.222 \sim 0.931$. `Cora` [38], `CiteSeer` [38], `PubMed` [38], and `Coauthor CS & Physics` [39] have high edge homophily and are usually considered as homophilic graphs. For these graphs, we follow the data split in GCN [1] and DAGNN [31]. 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` [40], `arXiv-year` [41], `genius` [42], and `twitch-gamer` [12] are graphs with lower homophily. For these graphs, we follow the data split in LINKX [10], [33], which uses the 50%/25%/25% nodes as the train/validation/test set respectively. For all the datasets, we generate five random data splits for computing the average and standard deviation of the models' performance.

TABLE 2. Statistics for the real-world datasets.

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

Baselines for Comparison: Our baselines include pure MLP applied on the node feature, methods based on the homophily assumption (GCN [1], GAT [3], and DAGNN [31]), methods designed for heterophilic graphs (LINKX [10] and GloGNN [11]), and those that can generalize on the whole spectrum of graph homophily (MIXHOP [14], GPR-GNN [15], H2GCN [13], AdaGNN [19], BernNet [18], and ACM-GCN [17]). We did a grid-based hyperparameter search for all baselines and our approach. Specifically, we use the hidden channels (d), the propagation steps for aggregated neighborhood features extraction (s_1), the power of the adjacency matrix for graph structure features extraction (s_2), the learning rate η, the weight decay λ, and the feature normalization (ν) as hyperparameters. For all the datasets, we perform a grid search over the following hyperparameter options:

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

We also list the best hyper-parameter settings for all the real-world datasets in Table 3.

TABLE 3. Best hyper-parameter settings for the real-world 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
d	128	64	64	128	128	64	64	64	128
s_1	2	10	5	2	10	20	20	10	10
s_2	5	1	5	5	1	2	1	1	2
η	0.001	0.01	0.001	0.01	0.001	0.01	0.01	0.001	0.001
λ	0.001	0.001	0.0005	0.001	0.0005	0.001	0.001	0.001	0.0005
ν	False	False	False	False	True	True	True	False	False

A. ACCURACY PERFORMANCE OF INGNN

Table 4 reports the average test accuracy over five random splits with the standard deviation of the accuracy performance on the graphs in the `syn-cora` dataset. Overall, INGNN outperforms the existing methods on most datasets, with six in the top-1 and four in the top-2. In particular, INGNN outperforms the baselines on a larger range of homophily ($0.2 \sim 0.8$), which covers the gray area and also more common cases in real-world datasets. It pushes the best accuracy up to 2.76%. Furthermore, considering that the number of classes in `syn-cora` is seven, the worst-case misclassification rate, as discussed in Theorem 1, is expected to occur around $1/7 \approx 0.14$. As evident from Table 4, within the range of $0 \sim 0.4$, conventional MPNN-based methods, including GCN, GAT, and DAGNN, exhibit lower accuracy compared to a pure MLP applied directly on node features. This empirical observation substantiates the gray area challenge of these algorithms.

Table 5 reports the results on the 9 real datasets. Overall, INGNN achieves five top-1 and three top-2 over 9 datasets, with an average rank of 1.78. INGNN achieves an accuracy improvement of $3.05\% \sim 30.94\%$ compared to MLP, $1.17\% \sim 10.51\%$ compared to GCN, $0.99\% \sim 7.16\%$ compared to GAT, and $0.34\% \sim 18.62\%$ compared to DAGNN regardless of the graph homophily. This reveals the better generalization capability of INGNN on the whole spectrum of graph homophily compared to pure MLP and methods based on the homophily assumption. Compared with methods designed for heterophilous graphs, INGNN obtains better accuracy than LINKX in 8 out of 9 datasets and achieves competitive results as GloGNN in the heterophilic graphs (i.e., `arXiv-year`, `penn94`, `twitch-gamer`, and `genius`) while surpassing it by $2.65\% \sim 14.1\%$ in homophilic graphs. Compared to methods that work on the whole spectrum of graph homophily, i.e, MIXHOP, GPR-GNN, H2GNN, AdaGNN, BernNet, and ACM-GCN, INGNN has the best generalization capability with higher accuracy of 4.01%, 3.43%, 3.81%, 2.89%, 6.57%, 4.46% on average respectively.

B. ABLATION STUDIES

We present ablation studies to show the effectiveness of our design choices: the three graph features, adaptive feature fusion, and bi-level optimization. Table 6 reports the accuracy results from five variants of our model by removing one design element at a time.

1) GRAPH FEATURES

The rows *w/o egg*, *w/o agg*, and *w/o strc* demonstrate the contribution of the three features extracted from the original graph information to the accuracy of the model. Overall, models without one of the three features suffer from $2.11\% \sim 5.43\%$ average accuracy drop on all the datasets, indicating the importance of these features on graph representation learning. Specifically, models without the aggregated neighborhood features (*w/o agg*) have a larger accuracy drop on homophilic graphs (i.e., `Cora`, `CiteSeer`, `PubMed`, `CS`, and `Physics`). It echoes the high performance of MPNNs (e.g., GCN and DAGNN) on homophilic graphs. In contrast, models without the graph structure features (*w/o strc*) suffer from severe accuracy drops, up to 20.52%, especially on heterophilic graphs. While models without the ego-node features (*w/o egg*) have accuracy drop from $0.38\% \sim 4.31\%$. This phenomenon echoes its depression effect on the misclassification rate as stated in Theorem 1.

2) ADAPTIVE FEATURE FUSION

We also investigate the effectiveness of adaptive feature fusion by comparing it to alternative methods such as summing up and concatenating the three graph features in Table 6. When fusing with summation, we observe an average accuracy drop of 1.84%, indicating the significance of the proposed adaptive feature fusion. Another widely used approach for feature fusion is concatenating features. Although concatenation allows one to learn separate parameters for different features, it fails to achieve a well-balanced integration of features, as it leads to an average accuracy drop of 2.09%.

TABLE 4. Test accuracy of different methods on the graphs with different homophily from 0 to 0.5 in `syn-cora` dataset. Bolded red and normal 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	$69.20_{\pm 1.92}$	$69.20_{\pm 1.92}$	$69.20_{\pm 1.92}$	$69.20_{\pm 1.92}$	$69.20_{\pm 1.92}$	$69.20_{\pm 1.92}$	$69.20_{\pm 1.92}$	$69.20_{\pm 1.92}$	$69.20_{\pm 1.92}$	$69.20_{\pm 1.92}$	$69.20_{\pm 1.92}$
GCN	$28.61_{\pm 2.38}$	$30.64_{\pm 1.58}$	$36.03_{\pm 1.15}$	$45.15_{\pm 2.18}$	$51.39_{\pm 1.48}$	$65.04_{\pm 1.60}$	$74.48_{\pm 1.36}$	$82.22_{\pm 2.20}$	$91.21_{\pm 1.15}$	$96.19_{\pm 0.65}$	$99.92_{\pm 0.18}$
GAT	$29.41_{\pm 2.20}$	$30.32_{\pm 2.52}$	$34.83_{\pm 1.67}$	$43.86_{\pm 1.44}$	$51.15_{\pm 1.25}$	$64.80_{\pm 1.81}$	$74.34_{\pm 2.03}$	$81.45_{\pm 1.72}$	$90.46_{\pm 1.12}$	$95.79_{\pm 0.71}$	**$100.0_{\pm 0.00}$**
DAGNN	$34.32_{\pm 1.51}$	$39.49_{\pm 1.18}$	$45.01_{\pm 2.42}$	$54.48_{\pm 3.17}$	$60.51_{\pm 1.40}$	$72.36_{\pm 1.83}$	$80.00_{\pm 1.26}$	$86.49_{\pm 2.13}$	$93.32_{\pm 1.45}$	**$97.48_{\pm 0.31}$**	$99.95_{\pm 0.07}$
LINKX	$72.09_{\pm 1.65}$	$70.54_{\pm 1.84}$	$69.76_{\pm 1.07}$	$70.13_{\pm 1.30}$	$71.34_{\pm 1.17}$	$74.53_{\pm 1.83}$	$77.16_{\pm 1.22}$	$80.35_{\pm 1.47}$	$83.30_{\pm 1.23}$	$87.59_{\pm 1.07}$	$89.60_{\pm 1.86}$
H2GCN	**$76.43_{\pm 1.27}$**	**$73.86_{\pm 3.05}$**	$71.58_{\pm 1.75}$	$72.17_{\pm 1.44}$	$72.95_{\pm 0.76}$	$78.31_{\pm 1.96}$	$83.27_{\pm 1.25}$	**$87.43_{\pm 0.88}$**	$92.09_{\pm 0.46}$	$97.00_{\pm 0.28}$	$98.98_{\pm 0.62}$
MIXHOP	$39.44_{\pm 2.98}$	$38.95_{\pm 2.21}$	$41.05_{\pm 2.69}$	$48.93_{\pm 2.72}$	$55.09_{\pm 2.30}$	$64.75_{\pm 1.88}$	$74.45_{\pm 1.44}$	$82.44_{\pm 2.69}$	$91.45_{\pm 1.11}$	$96.25_{\pm 0.89}$	**$100.0_{\pm 0.00}$**
GPR-GNN	$67.86_{\pm 2.66}$	$61.96_{\pm 2.53}$	$61.21_{\pm 2.36}$	$64.69_{\pm 2.88}$	$67.67_{\pm 3.41}$	$74.56_{\pm 2.60}$	$80.19_{\pm 1.57}$	$86.68_{\pm 0.69}$	$93.54_{\pm 1.36}$	$97.45_{\pm 0.28}$	**$100.0_{\pm 0.00}$**
INGNN	$72.49_{\pm 1.74}$	$73.67_{\pm 1.61}$	**$73.11_{\pm 0.94}$**	**$74.32_{\pm 2.29}$**	**$75.71_{\pm 1.00}$**	**$79.49_{\pm 1.58}$**	**$84.67_{\pm 1.30}$**	$87.32_{\pm 1.25}$	**$93.70_{\pm 1.13}$**	$94.75_{\pm 3.47}$	$99.95_{\pm 0.07}$

TABLE 5. Average test accuracy ± standard deviation on the real datasets. Bolded red and normal 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. Rank represents the average rank over all the datasets.

	arXiv-year	penn94	twitch-gamer	genius	CiteSeer	PubMed	Cora	CS	Physics	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	$36.70_{\pm 0.21}$	$73.60_{\pm 0.40}$	$60.92_{\pm 0.07}$	$86.68_{\pm 0.09}$	$50.94_{\pm 4.20}$	$66.04_{\pm 2.29}$	$52.56_{\pm 2.55}$	$83.08_{\pm 1.00}$	$82.15_{\pm 5.11}$	12.22
GCN	$46.02_{\pm 0.26}$	$82.47_{\pm 0.27}$	$62.18_{\pm 0.26}$	$87.42_{\pm 0.37}$	$63.36_{\pm 2.06}$	$78.12_{\pm 1.60}$	$77.90_{\pm 1.18}$	$90.35_{\pm 0.88}$	$92.39_{\pm 0.89}$	7.78
GAT	$49.37_{\pm 0.20}$	$81.45_{\pm 0.55}$	$62.32_{\pm 0.23}$	$86.59_{\pm 1.06}$	$65.90_{\pm 1.88}$	$76.78_{\pm 2.38}$	$76.98_{\pm 1.75}$	$88.86_{\pm 0.65}$	$92.57_{\pm 0.60}$	7.67
DAGNN	$38.49_{\pm 0.28}$	$74.84_{\pm 0.52}$	$60.36_{\pm 0.14}$	$71.11_{\pm 9.11}$	$67.12_{\pm 1.71}$	$78.28_{\pm 1.58}$	$82.34_{\pm 1.42}$	$91.83_{\pm 0.72}$	$93.22_{\pm 0.77}$	6.78
LINKX	$56.00_{\pm 1.34}$	$84.71_{\pm 0.52}$	$66.06_{\pm 0.19}$	$90.77_{\pm 0.27}$	$53.66_{\pm 3.69}$	$67.66_{\pm 4.29}$	$62.66_{\pm 2.12}$	$88.53_{\pm 1.43}$	$89.37_{\pm 1.52}$	7.56
GloGNN	$54.52_{\pm 0.39}$	**$85.60_{\pm 0.27}$**	**$66.34_{\pm 0.29}$**	**$90.91_{\pm 0.13}$**	$55.72_{\pm 3.09}$	$72.72_{\pm 1.16}$	$74.70_{\pm 1.62}$	$90.50_{\pm 1.29}$	$89.16_{\pm 2.61}$	6.33
MIXHOP	$51.78_{\pm 0.26}$	$83.63_{\pm 0.54}$	$65.65_{\pm 0.30}$	$90.61_{\pm 0.24}$	$56.98_{\pm 4.80}$	$76.14_{\pm 2.37}$	$73.80_{\pm 4.02}$	$89.79_{\pm 0.91}$	$93.33_{\pm 0.75}$	6.44
GPR-GNN	$44.89_{\pm 0.20}$	$81.12_{\pm 0.63}$	$62.00_{\pm 0.25}$	$90.02_{\pm 0.13}$	$64.72_{\pm 1.59}$	$79.12_{\pm 0.87}$	$80.44_{\pm 1.53}$	$90.74_{\pm 0.60}$	**$93.86_{\pm 0.36}$**	5.78
H2GCN	$49.09_{\pm 0.10}$	$81.54_{\pm 0.56}$	OOM	OOM	$64.40_{\pm 1.44}$	$76.30_{\pm 2.80}$	$79.24_{\pm 1.75}$	$91.18_{\pm 0.58}$	$93.56_{\pm 0.48}$	5.86
AdaGNN	$49.49_{\pm 0.16}$	$83.55_{\pm 0.31}$	$64.64_{\pm 0.27}$	$89.68_{\pm 0.81}$	$63.44_{\pm 1.94}$	$76.80_{\pm 1.45}$	$80.66_{\pm 1.07}$	$90.80_{\pm 0.84}$	$92.66_{\pm 0.89}$	5.33
BernNet	$36.49_{\pm 0.18}$	$82.81_{\pm 0.51}$	$62.37_{\pm 0.21}$	$88.83_{\pm 0.68}$	$53.52_{\pm 4.70}$	$75.32_{\pm 1.55}$	$76.52_{\pm 2.97}$	$91.43_{\pm 0.91}$	$91.30_{\pm 0.86}$	8.44
ACM-GCN	$48.41_{\pm 0.30}$	$82.68_{\pm 0.60}$	$61.48_{\pm 0.61}$	$81.19_{\pm 6.15}$	$65.08_{\pm 1.91}$	$76.70_{\pm 1.32}$	$78.90_{\pm 1.66}$	$90.50_{\pm 0.54}$	$92.70_{\pm 0.74}$	7.33
INGNN	**$56.53_{\pm 0.15}$**	$85.44_{\pm 0.60}$	$66.07_{\pm 0.11}$	$89.73_{\pm 0.51}$	**$69.82_{\pm 1.08}$**	**$79.98_{\pm 1.57}$**	**$83.50_{\pm 0.93}$**	**$93.15_{\pm 0.36}$**	$93.56_{\pm 0.65}$	**1.78**

3) 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 2.27% on average. This phenomenon is consistent with the observation in the NAS domain [35]: training model parameters and feature fusion weights jointly on the same training set would cause overfitting, and thus poor generalization performance.

C. INSIGHTS FROM FEATURE FUSION

This section studies the importance of the three graph features in graphs with different homophily. We measure feature importance by computing their proportion after adaptive feature fusion. Formally, the importance score is computed as,

$$\mathcal{I}_s = \frac{\pi_i \langle \mathbf{H}_s \rangle}{\pi_1 \langle \mathbf{H}_{\text{ego}} \rangle + \pi_2 \langle \mathbf{H}_{\text{agg}} \rangle + \pi_3 \langle \mathbf{H}_{\text{strc}} \rangle}, \tag{18}$$

where $s \in \{ego, agg, strc\}$ and $\langle \cdot \rangle$ computes the averaged absolute value of a specific feature.

Figures 2(a) show the results on `syn-cora`. We observe three dominant trends. (1) The importance of the aggregated neighborhood features 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. It is also consistent with our gray area challenge which reveals that the most severe misclassification rate associated with the aggregated neighborhood features materializes at a homophily level of approximately 0.14 (1/7) for the `syn-cora` dataset. (2) 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 features are a more reliable signal than other features in some cases, which is consistent with Theorem 1. (3) All the features have a nontrivial 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 the ablation study.

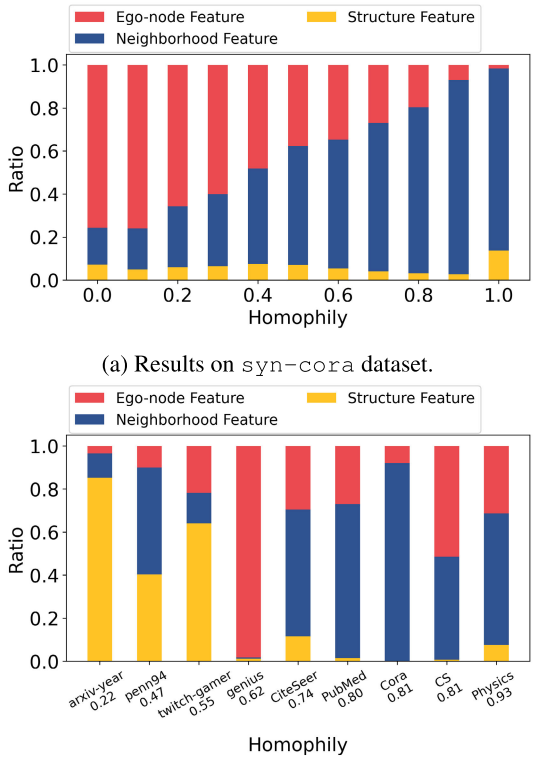
Figure 2(b) shows the results on real graphs. Since real graphs have different intrinsic graph properties, including the raw node features and the graph links, we cannot compare the changes in feature importance across graphs like what we did for `syn-cora`. Instead, we focus

TABLE 6. 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. -
INGNN	**56.53**	**85.44**	**66.07**	**89.73**	**69.82**	**79.98**	**83.50**	**93.15**	**93.56**	-
w/o ego	53.80 2.74↓	81.83 3.62↓	65.68 0.40↓	85.61 4.12↓	68.44 1.38↓	78.92 1.06↓	83.12 0.38↓	88.84 4.31↓	92.56 1.00↓	2.11↓
w/o agg	53.17 3.36↓	84.81 0.63↓	65.78 0.30↓	89.78 0.05↑	57.54 12.28↓	72.32 7.66↓	72.66 10.84↓	87.72 5.43↓	85.14 8.42↓	5.43↓
w/o strc	36.01 20.52↓	75.50 9.94↓	61.59 4.49↓	86.87 2.86↓	69.82 0.00↓	79.62 0.36↓	83.34 0.16↓	92.46 0.69↓	93.41 0.15↓	4.35↓
w/o fusion (sum up)	53.45 3.08↓	84.35 1.09↓	65.76 0.32↓	87.64 2.09↓	67.72 2.10↓	76.04 3.94↓	82.48 1.02↓	91.13 2.02↓	92.64 0.92↓	1.84↓
w/o fusion (concate)	56.55 0.02↑	82.14 3.30↓	65.87 0.20↓	89.44 0.28↓	66.70 3.12↓	74.70 5.28↓	81.10 2.40↓	89.73 3.42↓	92.74 0.82↓	2.09↓
w/o bi-level	53.27 3.26↓	83.71 1.74↓	65.74 0.34↓	88.25 1.48↓	68.60 1.22↓	73.32 6.66↓	81.02 2.48↓	90.31 2.84↓	93.15 0.41↓	2.27↓

TABLE 7. Average test accuracy and accuracy improvement when adding features on GCN, LINKX, and H2GCN. Subscripts report the accuracy improvement from the introduced features compared to Table 5. GCN_+: GCN $+\mathbf{H}_{ego}+\mathbf{H}_{strc}$, $LINKX_+$: LINKX $+\mathbf{H}_{aggr}$, $H2GCN_+$: H2GCN $+\mathbf{H}_{strc}$.

Homophily	arXiv-year 0.22	penn94 0.47	twitch-gamer 0.55	genius 0.62	CiteSeer 0.74	PubMed 0.80	Cora 0.81	CS 0.81	Physics 0.93	Avg. Imp. -
GCN_+	54.60 8.57↑	82.85 0.38↑	65.43 3.25↑	89.01 1.59↑	63.80 0.44↑	77.32 0.80↓	78.52 0.62↑	90.41 0.06↑	91.75 0.63↓	1.50
$LINKX_+$	55.13 0.87↓	84.79 0.08↑	66.08 0.03↑	91.17 0.40↑	64.88 11.22↑	75.72 8.06↑	81.26 18.60↑	91.99 3.46↑	93.11 3.74↑	4.97
$H2GCN_+$	50.71 1.62↑	83.96 2.42↑	OOM	OOM	65.18 0.78↑	75.24 1.06↓	78.04 1.20↓	92.03 0.85↑	92.97 0.59↓	0.40



(a) Results on `syn-cora` dataset.

(b) Results on real datasets.

FIGURE 2. Importance of the features after feature fusion.

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 homophily (i.e., `arXiv-year`), 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) For graphs that exhibit homophily within the gray area (i.e., `penn94`, `twitch-gamer`, and `genius`), the graph structure features and the ego-node features take the most importance proportion. This empirical observation is in agreement with our theoretical evidence, which posits that the integration of these two additional features is instrumental in mitigating the misclassification rate.

D. BENEFITS OF ADDING FEATURES TO EXISTING METHODS

We add the three identified features into existing methods, GCN, LINKX, and H2GCN, to show the benefits of these features. Since GCN only has aggregated neighborhood features, we add ego node features and graph structure features to it as separate branches. For LINKX, we introduce the aggregated neighborhood features since it is an MLP-based method with the other two features already. Regarding H2GCN, graph structure features are introduced as an additional branch. We fuse the introduced new features with the original features of GCN and H2GCN using the bi-level optimization framework as in INGNN. Table 7 shows the average accuracy for the three improved methods, as well as the improvement in accuracy compared to its original performance in Table 5. The introduced features can significantly improve the accuracy of GCN, LINKX, and H2GCN by 1.50%, 0.40%, and 4.97% respectively.

VI. CONCLUSION

This paper studied how to fully exploit the original graph information to improve the performance of MPNNs throughout the homophily spectrum. We showed theoretically that MPNNs would underperform in a homophily range that we call the gray area. To address the gray area challenge, we proposed a graph neural network called INGNN that fully exploits graph information by integrating ego-node features and graph structure features with MPNNs using the adaptive feature fusion technique. Extensive experiments show that INGNN achieves state-of-the-art accuracy compared to strong baselines on synthetic and real datasets. Additional ablation studies further illustrate the need for the three graph features and the adaptive feature fusion mechanism.

REFERENCES

[1] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2016, *arXiv:1609.02907*.
[2] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, Jan. 2017, pp. 1025–1035.
[3] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lió, and Y. Bengio, "Graph attention networks," 2017, *arXiv:1710.10903*.
[4] H. Gao, Z. Wang, and S. Ji, "Large-scale learnable graph convolutional networks," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2018, pp. 1416–1424.
[5] J. Gasteiger, A. Bojchevski, and S. Günnemann, "Predict then propagate: Graph neural networks meet personalized PageRank," 2018, *arXiv:1810.05997*.
[6] K. Xu, C. Li, Y. Tian, T. Sonobe, K. Kawarabayashi, and S. Jegelka, "Representation learning on graphs with jumping knowledge networks," in *Proc. Int. Conf. Mach. Learn.*, Jan. 2018, pp. 5453–5462.
[7] F. Wu, T. Zhang, A. H. Souza, C. Fifty, T. Yu, and K. Q. Weinberger, "Simplifying graph convolutional networks," in *Proc. Int. Conf. Mach. Learn.*, Jan. 2019, pp. 6861–6871.
[8] J. Gilmer, S. S. Schoenholz, P. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *Proc. Int. Conf. Mach. Learn.*, Jan. 2017, pp. 1263–1272.
[9] P. W. Battaglia et al., "Relational inductive biases, deep learning, and graph networks," 2018, *arXiv:1806.01261*.
[10] D. Lim, F. Höhne, X. Li, S. L. Huang, V. Gupta, O. Bhalerao, and S.-N. Lim, "Large scale learning on non-homophilous graphs: New benchmarks and strong simple methods," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, Jan. 2021, pp. 20887–20902.
[11] X. Li, R. Zhu, Y. Cheng, C. Shan, S. Luo, D. Li, and W. Qian, "Finding global homophily in graph neural networks when meeting heterophily," in *Proc. Int. Conf. Mach. Learn.*, Jan. 2022, pp. 13242–13256.
[12] B. Rozemberczki and R. Sarkar, "Twitch gamers: A dataset for evaluating proximity preserving and structural role-based node embeddings," 2021, *arXiv:2101.03091*.
[13] J. Zhu, Y. Yan, 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.*, Jan. 2020, pp. 7793–7804.
[14] S. Abu-El-Haija, B. Perozzi, A. Kapoor, H. Harutyunyan, N. Alipourfard, K. Lerman, G. V. Steeg, and A. Galstyan, "MixHop: Higher-order graph convolutional architectures via sparsified neighborhood mixing," in *Proc. Int. Conf. Mach. Learn.*, May 2019, pp. 21–29.
[15] E. Chien, J. Peng, P. Li, and O. Milenkovic, "Adaptive universal generalized PageRank graph neural network," 2020, *arXiv:2006.07988*.
[16] Y. Yan, M. Hashemi, K. Swersky, Y. Yang, and D. Koutra, "Two sides of the same coin: Heterophily and oversmoothing in graph convolutional neural networks," 2021, *arXiv:2102.06462*.
[17] S. Luan, C. Hua, Q. Lu, J. Zhu, M. Zhao, S. Zhang, X.-W. Chang, and D. Precup, "Revisiting heterophily for graph neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, Jan. 2022, pp. 1362–1375.
[18] 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, Jan. 2021, pp. 14239–14251.
[19] Y. Dong, K. Ding, B. Jalaian, S. Ji, and J. Li, "AdaGNN: Graph neural networks with adaptive frequency response filter," in *Proc. 30th ACM Int. Conf. Inf. Knowl. Manage.*, Oct. 2021, pp. 392–401.
[20] M. Niepert, M. M. Ahmed, and K. Kutzkov, "Learning convolutional neural networks for graphs," in *Proc. Int. Conf. Mach. Learn.*, Jun. 2016, pp. 2014–2023.
[21] F. Monti, D. Boscaini, J. Masci, E. Rodola, J. Svoboda, and M. M. Bronstein, "Geometric deep learning on graphs and manifolds using mixture model CNNs," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 5425–5434.
[22] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" 2018, *arXiv:1810.00826*.
[23] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph CNN for learning on point clouds," *ACM Trans. Graph.*, vol. 38, no. 5, pp. 1–12, Oct. 2019.
[24] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, Jun. 2017, pp. 5998–6008.
[25] M. Chen, Z. Wei, Z. Huang, B. Ding, and Y. Li, "Simple and deep graph convolutional networks," in *Proc. Int. Conf. Mach. Learn.*, vol. 1, Jul. 2020, pp. 1725–1735.
[26] G. Li, M. Müller, B. Ghanem, and V. Koltun, "Training graph neural networks with 1000 layers," in *Proc. Int. Conf. Mach. Learn.*, Jan. 2021, pp. 6437–6449.
[27] H. Zeng, M. Zhang, Y. Xia, A. Srivastava, A. Malevich, R. Kannan, V. Prasanna, L. Jin, and R. Chen, "Decoupling the depth and scope of graph neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, Jan. 2022, pp. 19665–19679.
[28] K. Oono and T. Suzuki, "Graph neural networks exponentially lose expressive power for node classification," 2019, *arXiv:1905.10947*.
[29] W. Zhang, Z. Sheng, Z. Yin, Y. Jiang, Y. Xia, J. Gao, Z. Yang, and B. Cui, "Model degradation hinders deep graph neural networks," 2022, *arXiv:2206.04361*.
[30] G. Li, M. Müller, A. Thabet, and B. Ghanem, "DeepGCNs: Can GCNs go as deep as CNNs?" in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 9266–9275.
[31] M. Liu, H. Gao, and S. Ji, "Towards deeper graph neural networks," in *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2020, pp. 338–348.
[32] H. Pei, B. Wei, K. Chen-Chuan Chang, Y. Lei, and B. Yang, "Geom-GCN: Geometric graph convolutional networks," 2020, *arXiv:2002.05287*.
[33] D. Lim, X. Li, F. Hohne, and S.-N. Lim, "New benchmarks for learning on non-homophilous graphs," 2021, *arXiv:2104.01404*.
[34] N. Apollonio, P. G. Franciosa, and D. Santoni, "A novel method for assessing and measuring homophily in networks through second-order statistics," *Sci. Rep.*, vol. 12, no. 1, pp. 1–18, Jun. 2022.
[35] H. Liu, K. Simonyan, and Y. Yang, "DARTS: Differentiable architecture search," 2018, *arXiv:1806.09055*.
[36] X. Dong and Y. Yang, "Searching for a robust neural architecture in four GPU hours," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 1761–1770.
[37] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.
[38] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Gallagher, and T. Eliassi-Rad, "Collective classification in network data," *AI Mag.*, vol. 29, no. 3, pp. 93–106, Sep. 2008.
[39] O. Shchur, M. Mumme, A. Bojchevski, and S. Günnemann, "Pitfalls of graph neural network evaluation," 2018, *arXiv:1811.05868*.
[40] A. L. Traud, P. J. Mucha, and M. A. Porter, "Social structure of Facebook networks," *Phys. A, Stat. Mech. Appl.*, vol. 391, no. 16, pp. 4165–4180, 2012.
[41] W. Hu, M. Fey, M. Žitnik, Y. Dong, H. Ren, B. Liu, M. Catasta, and J. Leskovec, "Open graph benchmark: Datasets for machine learning on graphs," in *Proc. Adv. Neural Inf. Process. Syst.*, Jan. 2020, pp. 22118–22133.
[42] D. Lim and A. R. Benson, "Expertise and dynamics within crowdsourced musical knowledge curation: A case study of the genius platform," in *Proc. ICWSM*, vol. 15, May 2021, pp. 373–384.

XIAO LIU received the M.S. degree in software engineering from Tongji University, in 2018, and the second M.S. degree in computer science from the University of Massachusetts Amherst (UMass Amherst), Amherst, MA, USA, in 2021, where he is currently pursuing the Ph.D. degree in computer science.

His research interests include machine learning, graph neural networks, and computer vision.

LIJUN ZHANG received the B.S. and M.S. degrees in software engineering from Tongji University, Shanghai, China, in 2016 and 2019, respectively. She is currently pursuing the Ph.D. degree in computer science with the University of Massachusetts Amherst (UMass Amherst), Amherst, MA, USA.

Her research interests include machine learning, multi-task learning, and computer vision.

HUI GUAN received the Ph.D. degree in electrical engineering from North Carolina State University, in 2020. She is currently an Assistant Professor with the College of Information and Computer Sciences, University of Massachusetts Amherst, the flagship campus of the UMass system. Her research interests include machine learning and programming systems. Her current research focuses on improving machine learning (e.g., speed, scalability, and reliability) through innovations in algorithms and programming systems (e.g., compilers and runtime). She is also interested in leveraging machine learning to improve high-performance computing.