Enhancing Heterophilic Graph Neural Network Performance through Label Propagation in K-Nearest Neighbor Graphs

Hyun Seok Park
Kookmin University
20151741@kookmin.ac.kr

Ha-Myung Park
Kookmin University
hmpark@kookmin.ac.kr

***Abstract*—How can we exploit Label Propagation (LP) to improve the performance of GNN models on heterophilic graphs? Graph Neural Network (GNN) models have received a lot of attention as a powerful deep learning technology that uses graph structure and features, and has achieved an archived state-of-the-art performance for graph-related tasks. LP has been applied in various studies to improve performance of GNN models. However, LP does not perform well on heterophilic graphs, where nodes of different types are linked with each other, since LP assumes that the graphs inherently exhibits homophily, where similar nodes tend to be linked. Such heterophilic graphs are increasingly common nowadays.**

In this paper, we propose LPkG (Label Propagation on k-Nearest Neighbor Graphs of Graph Autoencoder), a simple but effective method to engage LP to improve the performance of GNN models even on heterophilic graphs. LPkG constructs a supplementary homophilic graph, peforms LP on this graph, and uses the results together with the results of GNN models. The supplementary graph is a k-Nearest Neighbor (k-NN) graph genereated from a latent space computed by Graph Autoencoder (GAE). Experimental results demonstrate that LPkG consistently achieves performance improvement on various heterophilic graph datasets: 2.75% on the Wisconsin dataset, 2.23% on the Texas dataset, and 2.55% on the Cornell dataset.

***Index Terms*—Label Propagation, Graph Neural Network, Heterophilic graph, k-NN graph, Graph Autoencoder**

I. Introduction

How can we leverage Label Propagation (LP) to improve the performance of Graph Neural Network (GNN) models on heterophilic graphs? The myriad of information surrounding us is often represented as heterophilic graphs; a graph is considered heterophilic if different kinds of nodes are typically connected to each other by edges. Fig. 1b is an example of a heterophilic graph. For example, in an online transaction network, fraudsters may have more connections to regular customers than other fraudsters [1]; in a dating network, individuals may prefer connections with individuals of the opposite gender [2]; in a protein-protein interaction network, different types of amino acids are connected [3].

To analyze such heterophilic graphs, several GNN models have been proposed over the past few years. While the heterophily of graphs poses challenges for traditional GNN models as they learns by aggregating features of each node's direct neighbors, GNN models for heterophilic graphs address these challenges by either aggregating features from neighbors that are more than 2 hops away [4], [5], or from potential neighbors identified based on node feature similarity [6]–[10]. Although these models have exhibited superior performance over traditional GNN models in the context of heterophilic graphs, it is important to consider their relatively short history, which suggests substantial unexplored potential.

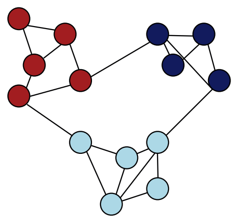
Meanwhile, several studies have found that LP complements GNN models especially when observable data are scarce [11], [12]; LP propagates labels directly along the edges while GNN models infer labels from node features. Nevertheless, the application of LP to heterophilic graphs has rarely been investigated, primarily due to LP's inherent assumption of homophily, which refers to the tendency of nodes to connect with nodes having the same class label.

In this paper, we propose LPkG (Label Propagation on k-Nearest Neighbor Graphs of Graph Autoencoder), a simple but effective method that improves the performance of GNN models by making supplementary homophilic graph, then using the label distribution obtained by LP. To reduce the dependence on features and create more meaningful supplementary homophilic graph, we use a Graph Autoencoder (GAE) [13] to generate latent representations that include both graph structure and features. We then construct a k-NN graph from the latent representations and use the graph as the supplementary homophilic graph. We demonstrate the effectiveness of our proposed method through various datasets and experiments. Significantly, we observe notable enhancements of 0.68%, 2.75%, 2.23% and 2.55% on the Chameleon, Wisconsin, Texas, and Cornell datasets, respectively, when LPkG is applied to FSGNN, which is one of the state-of-the-art models.

The main contributions of this paper are summarized as follows:

- We propose LPkG, a method to engage Label Propagation into learning GNN models to improve the performance. LPkG constructs a supplementary homophilic graph by constructing k-NN graph from the latent space computed

This work was funded by the Korea Meteorological Administration Research and Development Program "Developing Intelligent Assistant Technology and Its Application for Weather Forecasting Process" under Grant (KMA2021-00123). Ha-Myung Park is the corresponding author.


(a) Homophilic Graph

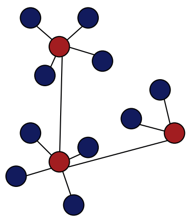

(b) Heterophilic Graph

Fig. 1: A illustration of homophilic and heterophilic graph. The color represents the class of the node.

by Graph Autoencoder, and performs LP on the supplementary graph.

- We have configured GAE's loss function to feature reconstruction error to ensure that the features produced by GAE exhibit homophily.
- To demonstrate the effectiveness of LPkG in improving the performance on heterophilic graphs, we apply it to various datasets and a GNN model and evaluate its performance in node classification tasks. Our method achieved up to 2.75% performance improvement compared to one of the state-of-the-art GNN model on heterophilic graphs.

II. Related Work

A. Label Propagation

Numerous methods incorporating LP into GNN models have been proposed for tasks on graphs. LP assumes that the graph is homophilic and estimates the labels of unlabeled nodes using the label distribution of the neighbors [14]–[17]. Wang and Leskovec [18] explore the relationship between LP and Graph Convolutional Network (GCN) where LP is employed to learn edge weights and improve performance when applied to GCN. Dong et al. [11] use LP to generate pseudo-labels for unlabeled nodes by propagating given labels, and incorporate them into GCN for joint learning. Bellei et al. [12] improve GCN performance by incorporating neighboring node labels in the first layer.

Meanwhile, Zhong et al. [19] focus on improving the effectiveness of LP in heterophilic graphs. They employ a multi-layer perceptron (MLP) trained on raw features to estimate the probabilities of inter-class connections. By leveraging these probabilities during the LP process, they achieve improvement in the performance of LP for heterophilic graphs. However, this approach has several challenges; it heavily relies on the features, making its performance sensitive to variations in feature distribution. Moreover, it does not exploit any advanced GNN models that have demonstrated superior performance across various graph-related tasks. On the other hand, LPkG uses GAE as a module, taking into account both the graph structure and node features. Since GAE can be replaced with newer GNN models that may be developed in the future, there is potential for LPkG to evolve alongside advancements in GNN models.

B. GNN models for heterophilic graphs

Various methods for improving the performance of GNN models in heterophilic graphs have been studied. We classify these methods into two categories: one for finding potential neighbors and the other for aggregating from higher order neighbors. Several GNN models consider the heterophily of graphs by finding potential neighbors and using their features during aggregation. To identify the potential neighbors, Geom-GCN [6] maps a graph to a Euclidean latent space; NL-GNN [9] and GPNN [10] use attention mechanisms; and UGCN [7] and SimP-GCN [8] use the k-NN algorithm. Meanwhile, other GNN models incorporate higher order neighbors during aggregation. MixHop [4] incorporates messages from multiple hops of neighbors, and mixes using distinct linear transformations. H2GCN [5] uses the representation obtained from neighboring nodes at lower order to make predictions about nodes at higher order. TDGNN [20] employs tree decomposition to transform neighbors from different hops into subgraphs, and aggregates in a parallel during learning. However, these models do not take into account the label distribution of both direct and potential neighbors, which can be significantly helpful in inferring node labels. In contrast, we emphasize that our LPkG exploits LP to incorporate the label distribution of potential neighbors into the inference.

III. Preliminaries

A. Graph

The input graph $G = (V, E)$ is an undirected graph where V represents the set of nodes $v_1, v_2, \cdots, v_n$, and E represents the set of edges $e_1, e_2, \cdots, e_m$. The $n \times n$ adjacency matrix A is used to represent connections between nodes, with entries being 1 if a connection exists and 0 if not. The $n \times F$ feature matrix X captures node features, where F corresponds to the feature dimension. The $n \times c$ label matrix Y signifies class labels, with c denoting the number of distinct classes.

B. Homophily and Heterophily

The homophily ratio $\mathcal{H}(G)$ of a graph G is a metric that measures how much the neighbors of nodes in a graph share the same label [21], and defined as follows:

$$\mathcal{H}(G) = \frac{1}{|\mathrm{V}|} \sum_{v \in V} \frac{|\{u \in N(v) : y(v) = y(u)\}|}{|N(v)|}$$

where $N(v)$ is the neighbor set of node v, $y(v)$ is the label of node v, and $|S|$ represents the cardinality of any set S. The range of $\mathcal{H}(G)$ is $[0, 1]$. A graph G is regarded as homophilic if $\mathcal{H}(G)$ is close to 1; conversely, it is regarded as heterophilic if it is close to 0.

C. Label Propagation

Label Propagation (LP) [14] is one of the most commonly used techniques in graph-based analysis. LP iteratively propagates labels of some nodes to unlabeled nodes according to their relationships in order to predict labels for the entire set of nodes. Let $Y^{(0)}$ be the initial label matrix of size $n \times c$ that is generated from the node labels, where c is the number of labels. That is, $Y_{ij}^{(0)} = 1$ if node v_i's label is j, otherwise 0. Then, for $t \geq 0$, where t represents the number of iterations, LP recursively calculates $Y^{(t+1)}$ as follows:

$$Y^{(t+1)} = (I - S)(\alpha \tilde{A}\tilde{Y}^{(t)} + (1-\alpha)\tilde{A}Y^{(0)}) + SY^{(0)}$$

where α is a weight parameter for balancing between updated labels and initial labels, $\tilde{A}$ is the symmetrically normalized adjacency matrix with self-connections, $\tilde{Y}^{(t)}$ is the row-normalized version of $Y^{(t)}$, S is the diagonal indicator matrix where $S_{ii} = \sum_j Y_{ij}^{(0)}$, and I is the identity matrix.

D. Graph Autoencoder

Graph Autoencoder (GAE) [13] is an unsupervised neural network model that uses graph convolutional layers [22] to encode both the graph structure and features into a latent space efficiently. The graph convolution layer used in GAE is defined as:

$$Gconv(X, \tilde{A}) = \tilde{A}XW$$

where $\tilde{A}$ represents the symmetrically normalized adjacency matrix with self-connections and W is a weight matrix. GAE consists of an encoder enc and a decoder dec. The encoder enc computes the latent representation X' as:

$$X' = enc(X, \tilde{A}) = Gconv(\sigma(Gconv(X, \tilde{A})))$$

where σ is an activation function. The decoder dec generates the reconstructed adjacency matrix $\bar{A}$ from X' as:

$$\bar{A} = dec(X') = \sigma(X'X'^{\top})$$

where σ is an sigmoid function. The weight parameters of GAE are trained to minimize the difference between $\tilde{A}$ and $\bar{A}$ so that the latent representation involves the graph structural information.

IV. Proposed Method

In this paper, we propose LPkG (Label Propagation on k-Nearest Neighbor Graphs of Graph Autoencoder), a simple but effective method to engage LP to improve the performance of GNN models even on heterophilic graphs. The challenge faced by LPkG lies in the fact that while LP assumes homophily, the graphs we are interested in exhibit heterophily; thus, simply applying LP to these graphs does not aid in performance improvement. LPkG addresses this problem by constructing a supplementary homophilic graph and then applying LP on this graph. We have discovered that when generating latent vectors with GAE, nodes of the same type tend to exist closer to each other in the latent space (see Fig. 6). Following this discovery, LPkG generate the supplementary graph as a k-NN graph on the latent space computed by GAE.

In section IV-A, we demonstrate how LPkG uses a GAE and leverages the latent representations obtained from the trained GAE to create a k-NN graph. Then in section IV-B, we demonstrate how to use LP on the supplementary homophilic graph created from section IV-A and apply the resulting label distribution to train GNN models.

A. Build k-NN graph using GAE

In this section, we introduce how LPkG constructs a supplementary homophilic graph. The main idea is to use GAE to convert features into a latent representation and construct a k-NN graph based on cosine similarity. To more effectively construct a supplementary homophilic graph, LPkG sets the reconstruction target of the decoder as features and proceeds with training. Fig. 2 simplifies the representation of the process, illustrating how LPkG uses GAE to generate a supplementary homophilic graph. The decoder dec used in LPkG is as follows:

$$dec(X') = f(\sigma(f(X')))$$

where f denotes a fully connected layer and σ is denoted an activation function. Therefore, the GAE used in LPkG takes the adjacency matrix A and the feature matrix X, passing them through an encoder enc to obtain latent representations X', and then it reconstructs these representations into features using a decoder dec. Additionally, during the training process, an MSE loss function is employed, and the GAE is trained to minimize this loss. The loss function l is as follows:

$$\hat{X} = dec(enc(X, \tilde{A}))$$

$$l(X, \hat{X}) = \frac{1}{n}\sum_{i=1}^{n}(X_i - \sigma(\hat{X}_i))^2$$

where σ is an sigmoid function, X_i and $\hat{X}_i$ are the values at the i-th row in X and $\hat{X}$, respectively. After training is completed, the encoder enc obtains the latent representation X' from the original adjacency matrix A and feature matrix X.

We build a k-NN graph using the cosine similarities between the latent representations of nodes. Let $sim(i, j)$ be the cosine similarity between nodes i and j, defined as follows:

$$sim(i, j) = \frac{X'_i \cdot X'_j}{\|X'_i\| \cdot \|X'_j\|}$$

where $\|X'_i\|$ and $\|X'_j\|$ denotes the L2 norms of X'_i and X'_j, respectively. Then, the adjacency matrix $\hat{A}$ of the k-NN graph is generated according to the following condition:

$$\hat{A}_{ij} = \begin{cases} 1, & \text{if } j \in top_k(i) \\ 0, & \text{otherwise} \end{cases}$$

where $top_k(i)$ is the set of k nodes having the highest cosine similarities.

B. LP & GNN models training

The generated supplementary homophilic graph provides new connections and a distribution of labels among neighbors that were not present in the original connectivity. After applying LP using the training label, we combine the results of LP with the learning results of the GNN models by performing a weighted average. Fig. 3 demonstrates the process of performing LP on the generated supplementary homophilic graph and subsequently incorporating it into the

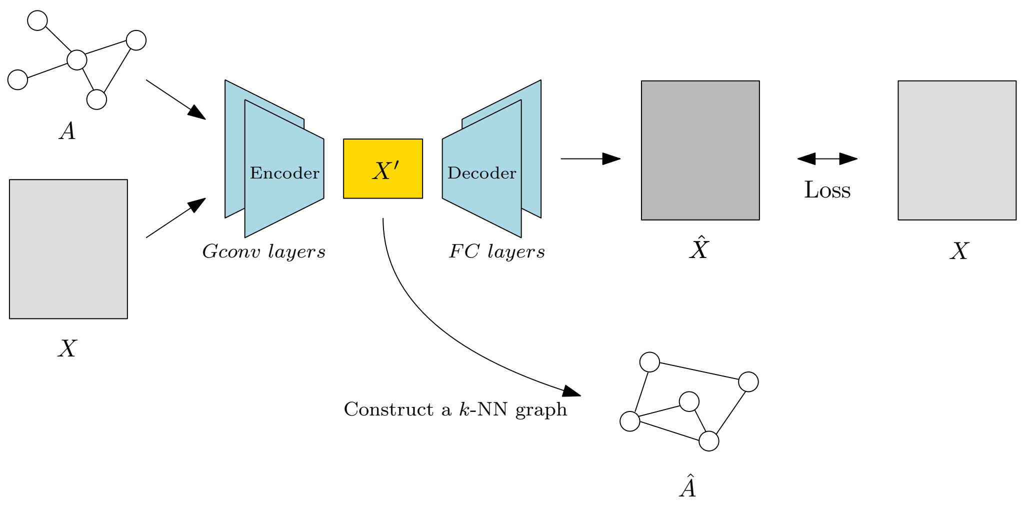


Fig. 2: This illustration shows how LPkG uses GAE to create a supplementary homophilic graph.

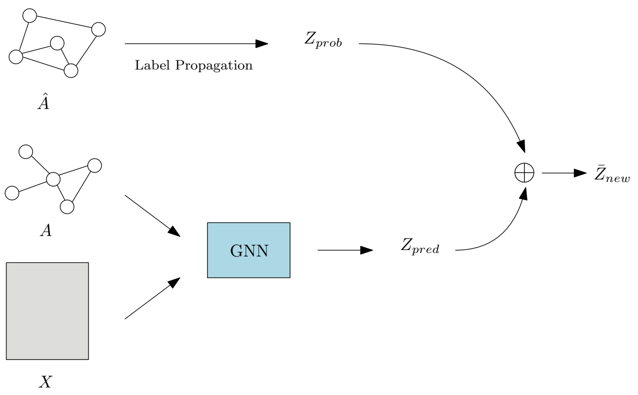


Fig. 3: This illustration shows the process in which LPkG performs label propagation on the generated supplementary homophilic graph, using the results for the training of a GNN model.

training process of a GNN models. The final label distribution, $Y^{(t+1)}$ obtained after $t+1$ iterations of LP is defined by Section III-C. Furthermore, to use the results of this process in the training procedure of a GNN models, a new predicted value $\bar{Z}_{new}$ is defined by taking a weighted average, as follows:

$$\bar{Z}_{new} = (1-\beta)Z_{pred} + \beta Z_{prob}$$

where Z_{pred} represents the predictions of existing GNN models, β is the ratio and learning parameter of the weighted average, and Z_{prob} is the label distribution $Y^{(t+1)}$ obtained through LP. Algorithm 1 lists a pseudocode of LPkG. Normalized adjacency matrix $\tilde{A}$ and feature matrix learn GAE (line 1). Subsequently, the reconstructed feature X' is obtained through the trained GAE enc (line 2). Next, create a k-NN graph $\hat{A}$ with X', and perform LP on $\hat{A}$ (line 3-9). Finally, the label distribution obtained through LP is obtained (line 10).

Algorithm 1 LPkG

Input: Normalized adjacency matrix $\tilde{A}$, encoder layers enc, decoder layers dec, initial label matrix $Y^{(0)}$, weight parameter α, diagonal indicator matrix S where $S_{ii} = \sum_j Y_{ij}^{(0)}$, identity matrix I, iteration number t_1, label distribution Z_{prob}.

```
1: Train $enc$ and $dec$ to minimize $l(X, dec(enc(\tilde{A}, X)))$ ▷ Graph Autoencoder
2: $X' \leftarrow enc(\tilde{A}, X)$
3: Build $k$-NN graph $\hat{A}$ using $X'$
4: $Y \leftarrow \hat{A}Y^{(0)}$
5: $\tilde{Y} \leftarrow$ row-normalized version of $Y$
6: for iteration $\leftarrow 1, 2, \ldots, T-1$ do
7:     $Y = (I - S)(\alpha\hat{A}\tilde{Y} + (1-\alpha)\hat{A}Y^{(0)}) + SY^{(0)}$ ▷ Label Propagation
8:     $\tilde{Y} \leftarrow$ row-normalized version of $Y$
9: end for
10: $Z_{prob} \leftarrow \tilde{Y}$
```

Dataset	Nodes	Edges	Features	Classes	Homo ratio
Chameleon	2,277	36,101	2,325	5	23%
Squirrel	5,201	198,535	2,089	5	22%
Wisconsin	251	499	1,703	5	21%
Texas	183	309	1,703	5	11%
Cornell	183	295	1,703	5	30%
Actor	7,600	26,659	932	5	22%

TABLE I: A summary of the node classification datasets.

V. Experiments

In this section, we demonstrate through experiments how effectively LPkG makes supplementary homophilic graphs to enhance the performance of LP. Additionally, we experimentally show that LPkG is effective in improving the performance of GNN models when training heterophilic graphs. We aim to

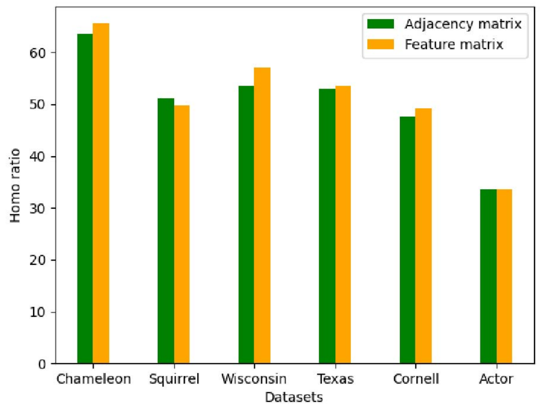


Fig. 4: The impact of changing the GAE decoder's reconstruct target on the homophily ratio.

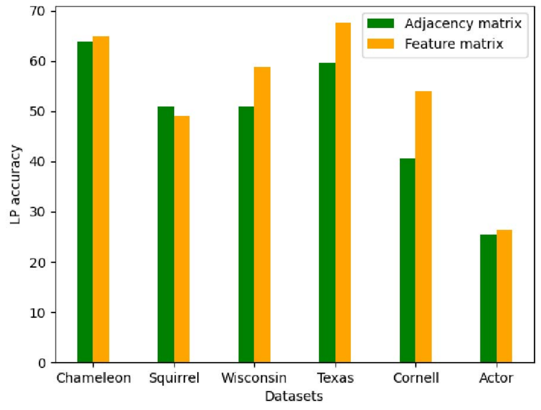


Fig. 5: The impact of changing the GAE decoder's reconstruct target on the LP node classification accuracy.

address the following questions through our experiments:

- $Q1$. **Effect of GAE decoder on reconstruct target. (Section V-B)** What is the impact of using features instead of the adjacency matrix as the target of the GAE decoder
- $Q2$. **Effect of GAE. (Section V-C)** What is the effectiveness of using GAE when converting a heterophilic graph to a supplementary homophilic graph?
- $Q3$. **Node classification accuracy. (Section V-D)** What could be the impact of performing LP on the generated supplementary homophilic graph and using the resulting label distribution in a GNN models?
- $Q4$. **Effect of hyperparameters. (Section V-E)** How do hyperparameters affect the performance of LP and GNN models?

Dataset	On raw graph	On k-NN graph (Proposed)	Change (%)
Chameleon	23.11	65.69	184.3
Squirrel	22.27	50.64	127.4
Wisconsin	20.60	56.28	173.2
Texas	11.19	53.09	374.4
Cornell	30.36	48.66	60.3
Actor	21.95	33.42	52.3

TABLE II: The result of homophily ratio on the raw graph and the k-NN graph with the latent representation of the GAE (k = 5). Red highlights indicate an increase.

Dataset	On raw graph	Using AE	Using GAE (Proposed)
Chameleon	44.08	31.58	**66.23**
Squirrel	33.05	27.19	**49.66**
Wisconsin	27.45	**60.78**	56.86
Texas	29.73	**67.57**	59.46
Cornell	51.35	48.65	**54.05**
Actor	22.50	**35.20**	27.57

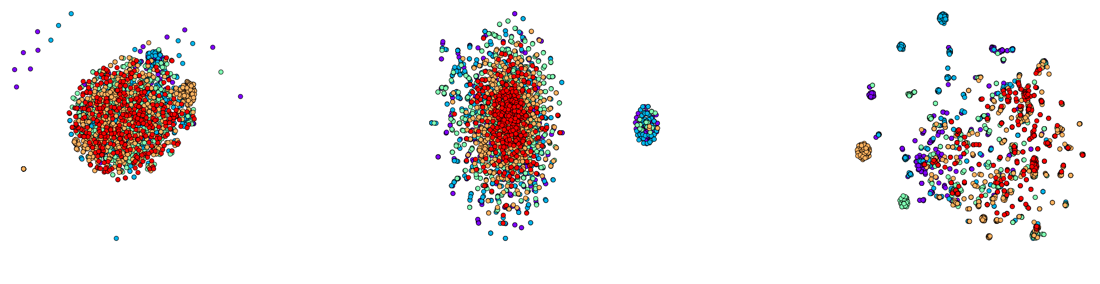
TABLE III: The node classification accuracy of LP on the raw graph, the k-NN graph with the latent representations from the AE and GAE (k = 5). Bold highlights represents the highest performance.

A. *Experiment Setting*

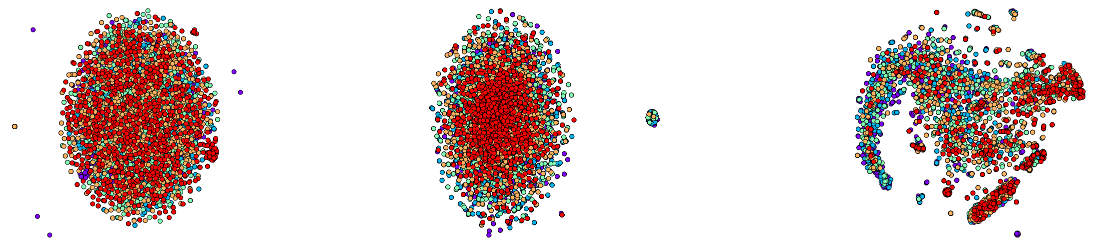
We introduce our experimental setup, which consists of datasets, and baseline model.

1) Datasets: The nodes in the Texas, Cornell, and Wisconsin datasets [6] represent web pages, the edges represent hyperlinks, and the node features represent web pages as bag-of-words. The nodes in the Chameleon and Squirrel datasets [23] represent Wikipedia articles, the edges represent connections between articles, and the node features represent the presence of certain nouns in articles. The nodes in the Actor dataset [24] represent actors, the edges represent co-occurrences on a Wikipedia page, and the node features represent some keywords on the page. These datasets are classified as heterophilic graphs, and they are split into 60% for training, 20% for validation, and 20% for testing for the fully-supervised node classification task, as described in [6]. The datasets description is given in TABLE I. Additionally, in TABLE I, the Homo ratio is derived through the formula explained in Section III-B.

2) Baseline: We apply the framework LPkG to existing GNN models and evaluate their performance. To verify the performance improvement through LPkG, we use FS-GNN [25], which aggregates features from multiple hops to learn heterophilic graphs. FSGNN is chosen due to its respectable node classification performance on most heterophilic graphs and its achievement of state-of-the-art results on several datasets. We use the settings and hyperparameters proposed in paper [25]. In addition, we report the performence as mean node classification accuracy over 10 random splits. All experiments are conducted using AMD Ryzen 7 3800XT 8-Core and RTX 3070 workstations with 8GB memory.



(a) Raw feature (b) Latent representation through AE (c) Latent representation through GAE



(d) Raw feature (e) Latent representation through AE (f) Latent representation through GAE

Fig. 6: Visualization using t-SNE of the raw feature (left), the latent representation generated through MLP-based autoencoder (middle), and the latent representation generated through GAE (right) of the Chameleon (top) and Squirrel (bottom) datasets.

	Chameleon	Wisconsin	Texas	Squirrel	Cornell	Actor
GEOM-GCN	60.90	64.12	67.57	38.14	60.81	31.63
GCNII	62.48	81.57	77.84	N/A	76.49	N/A
H2GCN-1	57.11	86.67	84.86	36.42	82.16	35.86
FSGNN	78.15	85.88	85.13	**74.34**	84.60	35.13
FSGNN + LPkG (Proposed)	**78.68**	**88.24**	**87.03**	74.23	**86.76**	**35.97**

TABLE IV: The table shows the accuracy of the fully-supervised node classification task across various datasets. Bold highlights represents the highest performance.

Hyperparameters	Serach ranges
β learning rate	{0.0001, 0.0005, 0.001, 0.005}
k	{5, 10, 15, 20, 25, 30}
GAE dimension	{128/64, 256/128, 512/256}

TABLE V: The table shows the search ranges for finding the optimal hyperparameters of LPkG.

B. *Effect of GAE decoder on reconstruct target*

We investigate the effect of using features as the reconstruct target in GAE decoder instead of the adjacency matrix. Fig. 4 shows the homophily ratio when the reconstruct target is the adjacency matrix and when it is the feature. Additionally, Fig. 5 demonstrates the node classification accuracy after constructing a k-NN graph using the trained GAE latent representations and performing LP. The results of both experiments indicate that, for the majority of datasets, using features as the reconstruct target is more effective in generating supplementary homophilic graphs than using the adjacency matrix.

C. *Effect of GAE on generating supplementary homophilic graph*

In this section, we approach the effects of using GAE to makes supplementary homophilic graph. TABLE II shows the homophily ratio of the raw graph and the homophily ratio of k-NN graphs created by considering both the graph and features using GAE. The hyperparameters used for the experiments were k = 5 and GAE dimensions of 256/128. The results show an increase in homophily ratio in all graphs compared

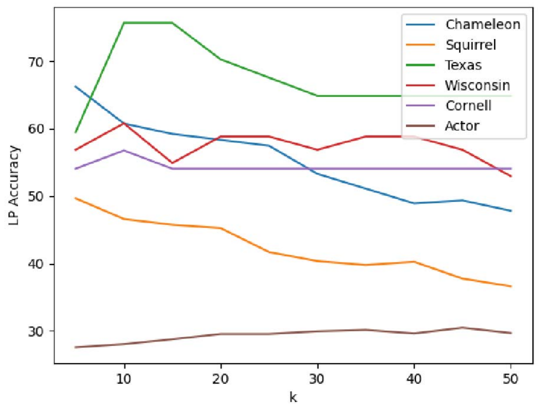


Fig. 7: The effect of k on LP performance when creating a k-NN graph.

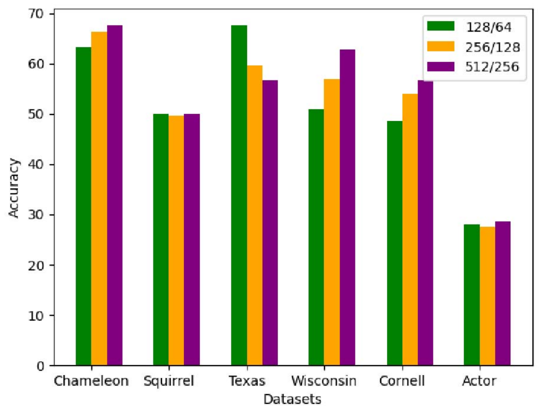


Fig. 8: The effect of dimension of GAE on LP performance.

to the raw graph. Particularly noteworthy is the significant increase of 184.3% and 374.4% in the Chameleon and Texas datasets, respectively. This indicates that creating a k-NN graph successfully identifies potential neighbors that were not present in the raw graph.

Furthermore, TABLE III compares the node classification accuracy of LP in various graphs, raw graph, the k-NN graph with the latent representations from the Autoencoder (AE) composed of MLP and GAE. Importantly, significant performance improvements are observed, especially in datasets with complex connectivity patterns like Chameleon and Squirrel datasets. These results provide evidence that using GAE to create k-NN graph is effective in transforming complex heterophilic graphs into more homophilic graphs.

In addition, Fig. 6 shows the results of t-SNE [26] visualization for raw features, latent representations obtained from a trained AE composed of MLP, and latent representations

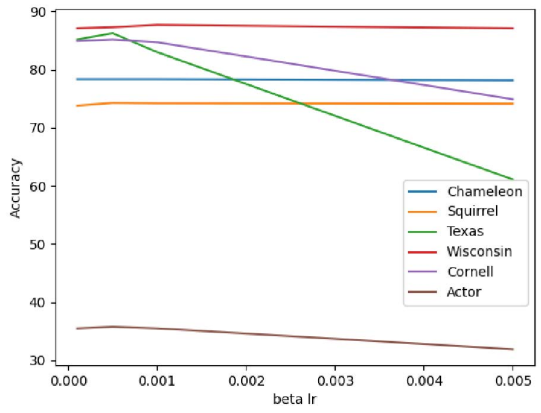


Fig. 9: The effect of learning rate of β on FSGNN accuracy.

obtained using a trained GAE on both the Chameleon and Squirrel datasets. The visualization demonstrates that leveraging graph structure with GAE is more effective in capturing complex feature distributions compared to AE or raw features. Moreover, this demonstrates that LPkG can effectively construct a supplementary homophilic graph, enabling us to obtain label distributions that were previously inaccessible.

D. Node Classfication Experiment

In this section, we conduct experiments on various datasets to verify that LPkG improves the performance of GNN models on heterophilic graphs. TABLE IV shows the accuracy of different models on heterophilic graphs and the accuracy when LPkG is applied to FSGNN. It shows an overall improvement when applied to the base FSGNN. Particularly, we observe improvements of 2.75%, 2.23%, and 2.55% on the Wisconsin, Texas, and Cornell datasets, respectively, and a 0.68% improvement on the Chameleon dataset. This result implies that through LPkG, it is possible to effectively enhance the performance of a GNN models on heterophilic graphs.

E. Hyperparameters

The hyperparameter settings of LPkG are shown in TABLE V. To investigate the effect of LP iterations, we conducted experiments from 0 to 500 iterations. We found that all datasets converged sufficiently from 30 iterations, so our experiments were conducted at 30 iterations. Furthermore, the investigation into the impact of the learning rate for GAE on GNN model accuracy ranged from 0.0001 to 0.005, revealing that it does not exert significant influence and converges. Consequently, for all experiments, the learning rate for GAE is set at 0.0001. In addition, we investigated the effect of α in LP from 0.1 to 0.99. We found that the performance of LP node classification did not differ significantly, and that most of the datasets showed better performance with higher α values. Therefore, we fixed α to 0.99 and conducted the experiments.

1) The effect of k on LP accuracy.: Fig. 7 shows the influence of the hyperparameter k of the k-NN graph on the performance of LP. The experiment is conducted with GAE dimensions of 256/128. The results show that the accuracy decreases when the value of k is too high for most datasets. This shows that a suitable value of k must be guaranteed.

2) The effect of GAE dimension on LP accuracy.: Fig. 8 demonstrates the influence of GAE dimensions on the accuracy of LP. The experiment is conducted with k of 5. The results show that performance is not guaranteed with too small dimensions. A certain size must be guaranteed to generate a latent representation that preserves the graph and features.

3) The effect of β learning rate on GNN model.: Fig. 9 illustrates the performance variation of FSGNN with respect to the learning rate of β. The experiment is conducted with GAE dimensions of 256/128 and k of 5. In most datasets, it can be observed that as the learning rate of β increases, the accuracy of FSGNN decreases, indicating the necessity for proper learning rate settings.

VI. Conclusion

We introduce a method, LPkG, that exploits Label Propagation (LP) to improve the performance of Graph Neural Network (GNN) models. LPkG uses Graph Autoencoder (GAE) to generate latent representations that incorporate both the graph structure and features. Moreover, in this process, to create a more meaningful supplementary homophilic graph, the reconstruction target of the decoder is set as features during training, leading to the construction of a k-NN graph. LPkG also use the label distribution obtained by performing Label Propagation on the generated supplementary homophilic graph for training GNN models. In our experiments, we demonstrate that specifying the reconstruction target of the GAE as features proves to be more effective in constructing the supplementary homophilic graph. Furthermore, using GAE to capture both feature and graph structure proves more effective than using features alone. Moreover, we compare GNN model node classification performance on various heterophilic graph datasets, and we observe significant improvements of 2.75%, 2.23%, and 2.55% on the Wisconsin, Texas, and Cornell datasets, respectively.

References

[1] S. Pandit, D. H. Chau, S. Wang, and C. Faloutsos, "Netprobe: a fast and scalable system for fraud detection in online auction networks," in *Proceedings of the 16th international conference on World Wide Web*, 2007, pp. 201–210.

[2] J. Zhu, R. A. Rossi, A. Rao, T. Mai, N. Lipka, N. K. Ahmed, and D. Koutra, "Graph neural networks with heterophily," 2021.

[3] J. Zhu, Y. Yan, L. Zhao, M. Heimann, L. Akoglu, and D. Koutra, "Beyond homophily in graph neural networks: Current limitations and effective designs," 2020.

[4] S. Abu-El-Haija, B. Perozzi, A. Kapoor, N. Alipourfard, K. Lerman, H. Harutyunyan, G. V. Steeg, and A. Galstyan, "Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing," 2019.

[5] J. Zhu, Y. Yan, L. Zhao, M. Heimann, L. Akoglu, and D. Koutra, "Generalizing graph neural networks beyond homophily," *CoRR*, vol. abs/2006.11468, 2020. [Online]. Available: https://arxiv.org/abs/2006.11468

[6] H. Pei, B. Wei, K. C. Chang, Y. Lei, and B. Yang, "Geom-gcn: Geometric graph convolutional networks," *CoRR*, vol. abs/2002.05287, 2020. [Online]. Available: https://arxiv.org/abs/2002.05287

[7] D. Jin, Z. Yu, C. Huo, R. Wang, X. Wang, D. He, and J. Han, "Universal graph convolutional networks," *Advances in Neural Information Processing Systems*, vol. 34, pp. 10 654–10 664, 2021.

[8] W. Jin, T. Derr, Y. Wang, Y. Ma, Z. Liu, and J. Tang, "Node similarity preserving graph convolutional networks," 2021.

[9] M. Liu, Z. Wang, and S. Ji, "Non-local graph neural networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 12, pp. 10 270–10 276, dec 2022. [Online]. Available: https://doi.org/10.1109%2Ftpami.2021.3134200

[10] S. Qi, W. Wang, B. Jia, J. Shen, and S.-C. Zhu, "Learning human-object interactions by graph parsing neural networks," 2018.

[11] H. Dong, J. Chen, F. Feng, X. He, S. Bi, Z. Ding, and P. Cui, "On the equivalence of decoupled graph convolution network and label propagation," in *Proceedings of the Web Conference 2021*, 2021, pp. 3651–3662.

[12] C. Bellei, H. Alattas, and N. Kaaniche, "Label-gcn: An effective method for adding label propagation to graph convolutional networks," *arXiv preprint arXiv:2104.02153*, 2021.

[13] T. N. Kipf and M. Welling, "Variational graph auto-encoders," 2016.

[14] X. Zhu and Z. Ghahramani, "Learning from labeled and unlabeled data with label propagation," 2002.

[15] X. Zhu, Z. Ghahramani, and J. D. Lafferty, "Semi-supervised learning using gaussian fields and harmonic functions," in *Proceedings of the 20th International conference on Machine learning (ICML-03)*, 2003, pp. 912–919.

[16] D. Zhou, O. Bousquet, T. Lal, J. Weston, and B. Schölkopf, "Learning with local and global consistency," *Advances in neural information processing systems*, vol. 16, 2003.

[17] X. Zhu, *Semi-supervised learning with graphs.* Carnegie Mellon University, 2005.

[18] H. Wang and J. Leskovec, "Unifying graph convolutional neural networks and label propagation," *CoRR*, vol. abs/2002.06755, 2020. [Online]. Available: https://arxiv.org/abs/2002.06755

[19] Z. Zhong, S. Ivanov, and J. Pang, "Simplifying node classification on heterophilous graphs with compatible label propagation," *arXiv preprint arXiv:2205.09389*, 2022.

[20] Y. Wang and T. Derr, "Tree decomposed graph neural network," 2021.

[21] X. Zheng, Y. Liu, S. Pan, M. Zhang, D. Jin, and P. S. Yu, "Graph neural networks for graphs with heterophily: A survey," *arXiv preprint arXiv:2202.07082*, 2022.

[22] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[23] B. Rozemberczki, C. Allen, and R. Sarkar, "Multi-scale attributed node embedding," 2021.

[24] J. Tang, J. Sun, C. Wang, and Z. Yang, "Social influence analysis in large-scale networks," in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2009, pp. 807–816.

[25] S. K. Maurya, X. Liu, and T. Murata, "Improving graph neural networks with simple architecture design," 2021.

[26] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne." *Journal of machine learning research*, vol. 9, no. 11, 2008.