Neural Networks 164 (2023) 719–730

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

LoyalDE: Improving the performance of Graph Neural Networks with loyal node discovery and emphasis☆

Haotong Wei, Yinlin Zhu, Xunkai Li, Bin Jiang *

Shandong University, School of Mechanical, Electrical and Information Engineering, Weihai, 264209, China

ARTICLE INFO

Article history:
Received 10 January 2023
Received in revised form 8 April 2023
Accepted 13 May 2023
Available online 19 May 2023

Keywords:
Graph Neural Networks
Semi-supervised Node Classification
Graph Supervision Loyalty

ABSTRACT

Recent years have witnessed an increasing focus on graph-based semi-supervised learning with Graph Neural Networks (GNNs). Despite existing GNNs having achieved remarkable accuracy, research on the quality of graph supervision information has inadvertently been ignored. In fact, there are significant differences in the quality of supervision information provided by different labeled nodes, and treating supervision information with different qualities equally may lead to sub-optimal performance of GNNs. We refer to this as the graph supervision loyalty problem, which is a new perspective for improving the performance of GNNs. In this paper, we devise FT-Score to quantify node loyalty by considering both the local feature similarity and the local topology similarity, and nodes with higher loyalty are more likely to provide higher-quality supervision. Based on this, we propose LoyalDE (**Loyal** Node **D**iscovery and **E**mphasis), a model-agnostic hot-plugging training strategy, which can discover potential nodes with high loyalty to expand the training set, and then emphasize nodes with high loyalty during model training to improve performance. Experiments demonstrate that the graph supervision loyalty problem will fail most existing GNNs. In contrast, LoyalDE brings about at most 9.1% performance improvement to vanilla GNNs and consistently outperforms several state-of-the-art training strategies for semi-supervised node classification.

1. Introduction

Graph-structured data widely exists in the real world and is used to characterize the potential relationships between entities in multiple fields, including social recommendation (Guo & Wang, 2020), drug discovery (Li et al., 2017), and financial risk control (Yang et al., 2020). Due to the high cost of the labeling process, real-world graph-structured data usually has a few labeled and many unlabeled samples. Therefore, simultaneously learning labeled and unlabeled samples in the graph is an essential and natural task, which is called graph-based semi-supervised learning.

As the standard paradigm for graph machine learning, Graph Neural Networks (GNNs) establish an efficient graph representation mechanism based on spectral theory and deep learning. Various of GNNs can be applied for graph-based semi-supervised learning tasks, such as ChebNet (Defferrard et al., 2016), CayleyNet (Levie et al., 2018), GCN (Kipf & Welling, 2016), GAT (Veličković et al., 2017), and GraphSAGE (Hamilton et al., 2017). However, due to the low proportion of labeled nodes, information cannot effectively propagate from labeled nodes to unlabeled nodes, which limits the performance of GNNs (Li et al., 2018). To this end, some effective training strategies seek to expand the training set to promote the model training (Coskun et al., 2019; Li et al., 2018; Tan et al., 2020).

Despite their effectiveness, these methods focus too much on the quantity of labeled nodes, inadvertently ignoring the quality of labeled nodes. For many real-world graph-structured data, there are significant differences in the quality of supervision information provided by different labeled nodes. Treating these labeled nodes with different qualities equally during learning may result in the sub-optimal performance of these above-mentioned GNNs. We refer to this as *the graph supervision loyalty problem*, and the quality of supervision information of labeled nodes in graphs is called *node loyalty*.

Considering the complexity of graph data and limited research on the topic, how to measure node loyalty is a significant challenge. However, most GNNs perform better when handling graphs with higher homophily (Zhu et al., 2020), leading us to intuitively assume that nodes with stronger local homophily can provide better supervision information for GNNs (i.e., have higher loyalty). To verify this assumption, we introduce a novel measure called FT-Score, which is based on local feature similarity and topology similarity. We randomly sample training nodes from

☆ The code will be available at https://github.com/Haotong-Wei/LoyalDE after paper acceptance for publication.

* Corresponding author.
E-mail addresses: weihaotong@mail.sdu.edu.cn (H. Wei), jiangbin@sdu.edu.cn (B. Jiang).

https://doi.org/10.1016/j.neunet.2023.05.023
0893-6080/

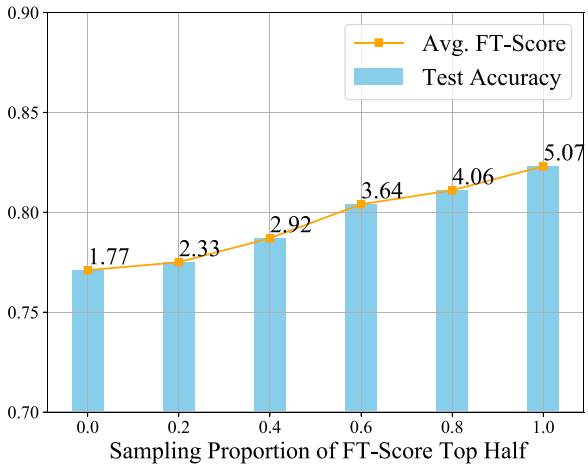


Fig. 1. Impact of the graph supervision loyalty problem on the performance of GNNs. The *x*-axis indicates the proportion of nodes sampled from the top half of the FT-Score distribution for each class.

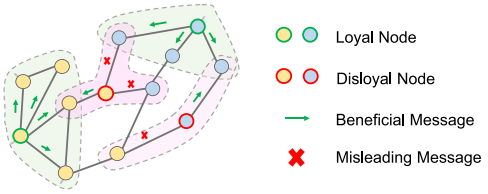


Fig. 2. The message-passing process of loyal and disloyal nodes. Yellow and blue correspond to two different classes.

the top and bottom halves of the FT-Score distribution according to certain proportions and evaluate the performance of GCN on the Cora dataset for each proportion. The experimental result is presented in Fig. 1. As observed, the performance of GNNs is poor when the proportion of nodes from the top half of the FT-Score distribution in the training set is low. Conversely, as the proportion of nodes from the top half of the FT-Score distribution in the training set increases, the performance of GNNs also increases. This finding demonstrates that nodes with higher FT-Score tend to provide higher-quality supervision information, i.e., higher loyalty. Furthermore, this result reveals that the graph supervision loyalty problem significantly affects the ability of GNNs to learn from labeled nodes. We contend that addressing the graph supervision loyalty problem is a novel perspective for improving the performance of GNNs.

To obtain a deeper understanding of the graph supervision loyalty problem, we qualitatively divide the nodes in the network into two classes: (1) *Loyal node*, which tends to have strong local homophily (i.e., similar nodes are more likely to be proximal, and vice versa) and have positive effects on model training; (2) *Disloyal node*, which tends to have poor local homophily and have limited or even negative effects on model training. Fig. 2 illustrates the message-passing process of loyal nodes and disloyal nodes. The positive impact of loyal nodes on the model can be divided into two perspectives. On the one hand, a loyal node with strong local homophily tends to have more typical characteristics of its class, and the provided supervision can help GNNs capture the pattern of its class more effectively; On the other hand, during the message-passing process, plenty of unlabeled neighbors in the same class can obtain latent representations similar to the loyal node, which helps GNNs identify them more accurately. In contrast, due to its poor local homophily, the supervision provided by a disloyal node cannot represent the pattern of its class and may mislead the latent representations of neighbors in different classes during the message-passing process.

Based on this, we propose a novel model-agnostic hot-plugging training strategy LoyalDE, which can effectively discover and emphasize potential loyal nodes, thus providing high-quality supervision for training GNNs and improving their performance. The overall process of LoyalDE can be divided into two steps: (1) Loyal node discovery: We first train a vanilla GNN using the original training nodes and obtain the predicted soft label. Then, we use a learnable adaptive scaler to obtain a personalized scaling temperature for each node, thereby boosting the confidence of loyal nodes and reducing the confidence of disloyal nodes. Finally, we expand the training set by selecting high-confidence nodes from the scaled soft label; (2) Loyal node emphasis: We calculate the FT-Score for each node in the expanded training set, and nodes with higher FT-Score will be assigned larger training weights, promoting the model to obtain high-quality supervision from loyal nodes.

In summary, the contributions of this work are as follows:

- We discover and define the graph supervision loyalty problem, which is a new perspective for improving the performance of GNNs.
- We devise FT-Score to measure the quality of graph supervision information. Based on this, we propose LoyalDE, a model-agnostic hot-plugging GNN training strategy.
- Experiments demonstrate the effectiveness of LoyalDE, which significantly improves the performance of GNNs and consistently outperforms several state-of-the-art GNN training strategies.

The remainder of this paper is organized as follows. Section 2 explains the notations used in this paper and briefly reviews related research about GNNs and semi-supervised node classification. Section 3 presents the details of our proposed node loyalty measure FT-Score and the hot-plugging training strategy LoyalDE. Then, Section 4 reports the experimental results of the proposed LoyalDE. Finally, we conclude our work in Section 5.

2. Preliminaries and related works

We first explain the meaning of the notations used in this paper in Section 2.1. Then, we briefly review GNNs in Section 2.2 and semi-supervised node classification in Section 2.3.

2.1. Notation definitions

We use bold upper case letters to represent the matrix (e.g., $\boldsymbol{W}$), bold lower case letters to represent the vector (e.g., $\boldsymbol{f}$), and $\boldsymbol{A}^T$ to represent the transpose of matrix $\boldsymbol{A}$. Moreover, the ith element of vector $\boldsymbol{a}$ is denoted as $\boldsymbol{a}_i$, and $\boldsymbol{A}_{ij}$ represents the element of matrix $\boldsymbol{A}$ at the ith row and jth column. Further definitions of notations commonly used in the graph learning field are provided in Table 1.

2.2. Graph Neural Networks (GNNs)

Earlier research on GNNs proposes spatial convolution based on hierarchical clustering and spectral convolution based on the spectrum of graph Laplacian, both of which generalize CNNs to graph-structured data (Bruna et al., 2013). However, the proposed convolution operations have a considerable amount of parameters and high computation costs. To this end, ChebNet (Defferrard et al., 2016) employs Chebyshev polynomials to approximate the K-order localized spectral filter. GCN (Kipf & Welling, 2016) further simplifies the graph convolution by employing the 1-order Chebyshev filter to capture local neighborhood information.

Table 1
Description of notations commonly used in this article.

Notations	Definitions
N	Number of nodes
C	Number of classes
$\mathcal{V}$	Node set
$\mathcal{V}_{labeled}$	Node set with labels
$\mathcal{V}_{unlabeled}$	Node set without labels
$\mathcal{E}$	Edge set
F	Dimension of node features
$\boldsymbol{X}$	Node feature matrix
$\boldsymbol{A}$	Adjacency matrix
$\boldsymbol{Y}$	Node label matrix
$\hat{\boldsymbol{Y}}$	Predicted soft label matrix
$\mathcal{N}_i$	Neighbors of ith node, including itself

However, as the aggregation weights of each central node to different neighbor nodes are fixed and equal, the modeling capability of GCN is limited. To this end, GAT (Veličković et al., 2017) employs the graph attention mechanism to adjust the aggregation weights of nodes to their neighbors adaptively, therefore having a stronger modeling capability. Moreover, GCN is difficult to deal with the semi-supervised graph task of inductive learning. GraphSAGE (Hamilton et al., 2017) addresses this limitation by introducing various learnable aggregation functions to perform convolution on each sampled subgraph.

The node representations learned by GNNs can be applied to various graph machine-learning tasks, such as node classification (Dornaika et al., 2023), link prediction (Dai et al., 2022), node clustering (Xia et al., 2022), graph classification (Ju et al., 2022), all of which achieve remarkable performance. More works on GNNs can be found in surveys Wu et al. (2020) and Zhou et al. (2020).

2.3. Semi-supervised node classification

Given an attributed graph $G = (\mathcal{V}, \mathcal{E}, \boldsymbol{X})$, where $\mathcal{V}$ denotes the node set, $\mathcal{E}$ denotes the edge set, and $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N]^T \in \mathbb{R}^{N\times F}$ is the node feature matrix, $\boldsymbol{x}_i \in \mathbb{R}^F$ is the F-dimensional feature vector of node v_i, and $N = |\mathcal{V}|$ holds. Let $\mathcal{V}_{labeled}$ and $\mathcal{V}_{unlabeled}$ be the set of labeled nodes and unlabeled nodes, where $\mathcal{V}_{labeled} \cap \mathcal{V}_{unlabeled} = \emptyset$ and $\mathcal{V}_{labeled} \cup \mathcal{V}_{unlabeled} = \mathcal{V}$. The node label matrix $Y \in \mathbb{R}^{N\times C}$ consists of one-hot encoding vectors for labeled nodes and zero vectors for unlabeled nodes, where C is the number of classes. The goal of the semi-supervised node classification task is to predict the labels of unlabeled nodes $\mathcal{V}_{unlabeled}$. Compared with the supervised setting, where only the feature of labeled nodes could be used, semi-supervised node classification allows the models to use the feature of both labeled and unlabeled nodes to classify unlabeled nodes.

Most of the early research about graph-based semi-supervised learning is based on the cluster assumption (Chapelle & Zien, 2005), which assumes that two topologically close nodes tend to belong to the same class. As a result, these methods mainly exploit graph topology to guide semi-supervised learning, including min-cuts (Blum & Chawla, 2001), randomized min-cuts (Blum et al., 2004), spectral graph transducer (Joachims, 2003), label propagation (Zhu et al., 2003), etc.

However, these above-mentioned methods have unintentionally neglected to utilize node features, which also have great value for identifying class patterns. To this end, many graph-based semi-supervised learning methods aim to jointly model the topological structure and the node features, such as deep semi-supervised embedding (Weston et al., 2008) and Planetoid (Yang et al., 2016).

Research from the same perspective also includes various representative GNNs mentioned in Section 2.2, all of which can be effectively applied for the semi-supervised node classification tasks. Moreover, many advanced training strategies are proposed to address the issue that GNNs' performance degrades in the face of a few labeled nodes. Self-Training and Co-Training (Li et al., 2018) seek to expand the labeled node set and supplement additional supervision information for GNNs training by leveraging the model prediction and the topological structure. LExiCoL (Coskun et al., 2019) identifies latent community structures to discover unlabeled nodes similar to labeled nodes based on clustering. Other effective training strategies also augment the training set by exploring graph properties (Dong et al., 2020; Tan et al., 2020; Wang, Shao, et al., 2021).

Despite their effectiveness, these methods inadvertently ignore the impact of supervision quality on the performance of GNNs. How to design an advanced training method for GNNs that consider the impact of supervision information quality is an important challenge. In this paper, we address this challenge by thoroughly analyzing the effect of supervision quality on GNN performance and propose a novel training strategy, LoyalDE, that enhances GNN performance by incorporating supervision quality considerations.

3. Proposed method

In this section, we present the proposed node loyalty measure FT-Score and the hot-plugging training strategy LoyalDE, which can discover and emphasize potential nodes with high loyalty during model training to improve the model's performance. Specifically, Section 3.1 presents the detail of FT-Score, and LoyalDE is discussed in Section 3.2.

3.1. FT-score: the measure of node loyalty

As introduced earlier, FT-Score considers both local feature similarity and topology similarity, thus effectively measuring the loyalty of nodes. Specifically, we employ the cosine similarity to compute the feature similarity. The feature similarity $S_{feat}(v_i, v_j)$ between v_i and each neighbor $v_j \in \mathcal{N}_i$, as follows:

$$S_{feat}(v_i, v_j) = \frac{\boldsymbol{x}_i \boldsymbol{x}_j}{\|\boldsymbol{x}_i\| \ \|\boldsymbol{x}_j\|}, \tag{1}$$

where x_i and x_j is the feature of node v_i and v_j, respectively.

According to related studies (Liben-Nowell & Kleinberg, 2003), nodes with stronger homophily tend to have more common neighbors with their neighbors. Therefore, we measure the topology similarity as the Jaccard coefficient, which considers the proportion of common neighbors. Specifically, the topology similarity $S_{topo}(v_i, v_j)$ between v_i and each neighbor $v_j \in \mathcal{N}_i$ is calculated as follows:

$$S_{topo}(v_i, v_j) = \frac{|\mathcal{N}_i \cap \mathcal{N}_j|}{|\mathcal{N}_i \cup \mathcal{N}_j|}. \tag{2}$$

Finally, we use a trade-off parameter α to perform linear weighting on local feature similarity and local topology similarity. Formally, the FT-Score of node v_i is calculated as follows:

$$FT(v_i) = \sum_{j\in\mathcal{N}_i} \alpha \frac{\boldsymbol{x}_i \boldsymbol{x}_j}{\|\boldsymbol{x}_i\| \ \|\boldsymbol{x}_j\|} + (1-\alpha)\frac{|\mathcal{N}_i \cap \mathcal{N}_j|}{|\mathcal{N}_i \cup \mathcal{N}_j|}. \tag{3}$$

We claim that the proposed FT-Score is intuitive and explainable. We can employ the FT-Score to effectively measure the loyalty of nodes in the network, which in turn provides guidance for optimizing the quality of supervision in graph-based semi-supervised learning.

3.2. LoyalDE: the proposed training strategy

The overall workflow of the proposed training strategy LoyalDE is shown in Fig. 3. As mentioned earlier, the pipeline of LoyalDE can be divided into two processes: (1) loyal node discovery and (2) loyal node emphasis.

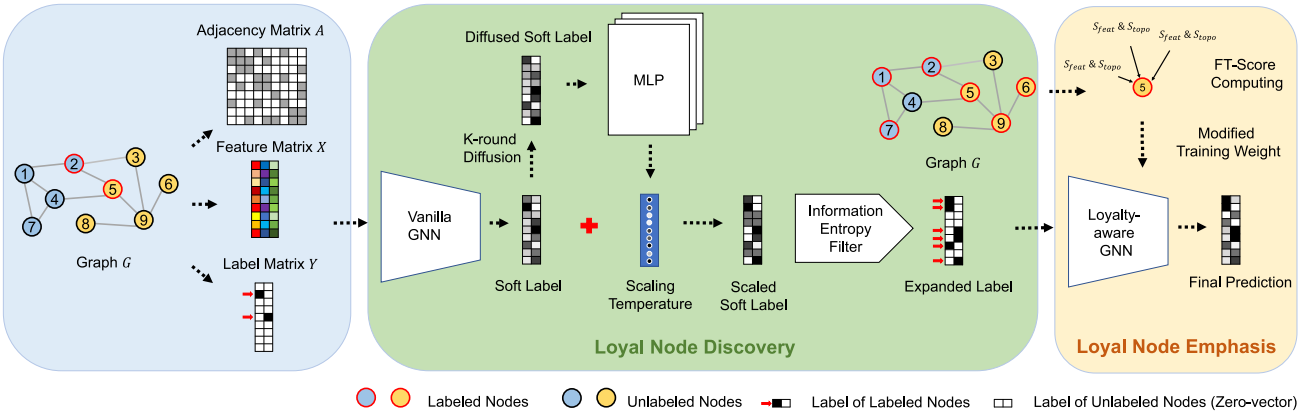


Fig. 3. The overall workflow of LoyalDE for semi-supervised node classification with GNNs. LoyalDE can be divided into two phases: (1) Loyal Node Discovery: discover potential unlabeled loyal nodes through personalized confidence adjustment, thereby expanding the training set (2) Loyal Node Emphasis: use the expanded training set to train a loyalty-aware GNN, which adaptively assigns the training weights of nodes according to FT-Score.

Loyal node discovery. The loyal node discovery phase aims to discover potential high-loyalty unlabeled nodes from the prediction of vanilla GNNs and use them to expand the training set, therefore improving the quality of supervision information.

Specifically, for a given graph $G = (\mathcal{V}, \mathcal{E}, \boldsymbol{X})$ with original labeled and unlabeled node sets $\mathcal{V}_{labeled}$ and $\mathcal{V}_{unlabeled}$, where $\mathcal{V} = \mathcal{V}_{labeled} \cup \mathcal{V}_{unlabeled}$ holds, we first train a vanilla GNN model by optimizing the vanilla cross-entropy loss function for semi-supervised node classification, which is computed as follows:

$$\mathcal{L}_{vanilla} = -\sum_{i \in \mathcal{V}_{labeled}} \sum_{k=1}^{C} \boldsymbol{Y}_{i,k} \log(\hat{\boldsymbol{Y}}_{i,k}), \tag{4}$$

where C denotes the number of classes, $\boldsymbol{Y}$ denotes the ground-truth, and $\hat{\boldsymbol{Y}}$ represents the predicted soft label of the vanilla GNN.

For the obtained predicted soft labels $\hat{\boldsymbol{Y}}$, both loyal nodes and disloyal nodes may have high confidence, therefore directly employing it to expand the training set will introduce low-quality supervision from disloyal nodes, resulting in suboptimal performance. In other words, we should reduce the confidence of disloyal nodes.

As disloyal nodes tend to have poor local homophily, their predicted soft label should be significantly different from their neighborhoods. Based on this, we first conduct K rounds of soft label diffusion, which can be regarded as a Laplacian smoothing process of soft labels. If the central node is a disloyal node, which has poor local homophily, its diffused soft label should come to be smooth, and its confidence is also reduced; The diffused soft label $\hat{\boldsymbol{Y}}_D$ is calculated as follows:

$$\hat{\boldsymbol{Y}}_D = (\tilde{\boldsymbol{D}}^{-\frac{1}{2}} \tilde{\boldsymbol{A}} \tilde{\boldsymbol{D}}^{-\frac{1}{2}})^K \hat{\boldsymbol{Y}}, \tag{5}$$

where $\tilde{\boldsymbol{A}}$ denotes the adjacency matrix with self-loop (i.e., $\tilde{\boldsymbol{A}} = \boldsymbol{A} + \boldsymbol{I}$, $\boldsymbol{I}$ represents the identity matrix), $\tilde{\boldsymbol{D}}$ denotes the degree matrix with self-loop (i.e., $\tilde{\boldsymbol{D}} = \boldsymbol{D} + \boldsymbol{I}$, $\boldsymbol{D}_{ii} = \sum_{j=1}^{N} \boldsymbol{A}_{ij}$), and K denotes the number of diffusion rounds.

However, the diffusion process may change the predicted node labels. Since our goal is to adjust the confidence without changing the predicted classes, we feed the diffused soft label $\hat{\boldsymbol{Y}}_D$ to an adaptive scaler based on multi-layer perceptron to learn a personalized scaling temperature $\boldsymbol{t}_i$ for each node v_i. The scaling temperature $\boldsymbol{T}$ is computed as:

$$\boldsymbol{T} = \log(1 + \exp(\hat{\boldsymbol{Y}}_D \boldsymbol{W})), \tag{6}$$

where $\boldsymbol{W} \in \mathbb{R}^C$ is the learnable parameter, and $\log(1 + exp(\cdot))$ conducts an element-wise soft plus activation for the adaptive scaler.

Then, we employ the scaling temperature $\boldsymbol{T}$ to perform personalized confidence adjustment on the predicted soft label $\hat{\boldsymbol{Y}}$, therefore obtaining a scaled soft label, i.e., $\hat{\boldsymbol{Y}}_S = \boldsymbol{T}\hat{\boldsymbol{Y}}$. The scaled soft label $\hat{\boldsymbol{Y}}_S$ has high confidence for loyal nodes and low confidence for disloyal nodes.

Since minimizing NLL loss (Friedman et al., 2001) can help improve the confidence of correctly predicted samples (Murphy, 1973), and the confidence of wrongly predicted samples can be reduced by minimizing the gap between the largest and second largest in scaled soft label (Wang, Liu, et al., 2021), we design the objective function of the adaptive scaler as follows:

$$\begin{aligned}\mathcal{L}_{scaler} = &-\sum_{i \in \mathcal{V}_{labeled}} \sum_{k=1}^{C} \boldsymbol{Y}_{i,k} \log(\hat{\boldsymbol{Y}}_{S_{i,k}}) + \\ &\frac{1}{|\mathcal{V}_{labeled}|}\Big(\sum_{i=1}^{|cor|} 1 - \hat{\boldsymbol{Y}}^{(cor)}_{S_{i,l}} + \hat{\boldsymbol{Y}}^{(cor)}_{S_{i,s}} + \\ &\sum_{i=1}^{|wro|} \hat{\boldsymbol{Y}}^{(wro)}_{S_{i,l}} - \hat{\boldsymbol{Y}}^{(wro)}_{S_{i,s}}\Big),\end{aligned} \tag{7}$$

where (cor) and (wro) represent the correctly and wrongly predicted node, respectively. $\hat{\boldsymbol{Y}}_{S_{i,l}}$ and $\hat{\boldsymbol{Y}}_{S_{i,s}}$ denote the largest and second-largest components of the scaled soft label of node v_i, respectively.

Afterward, we use a mapping function based on information entropy (Shannon, 1948) to map the scaled soft label $\hat{\boldsymbol{Y}}_{S_i}$ of each node v_i to its acceptance factor $m_i \in [0, C\exp(-1)]$, which is calculated as follows:

$$m_i = C\exp(-1) - H(\sigma(\hat{\boldsymbol{Y}}_{S_i})), \tag{8}$$

where $H(\cdot)$ is the information entropy function, i.e., $H(\boldsymbol{x}) = -\sum_i \boldsymbol{x}_i \log(\boldsymbol{x}_i)$, $\sigma(\cdot)$ is the softmax function. Since $\sigma(\hat{\boldsymbol{Y}}_{S_i})$ is always sum to 1, it can be proved that for a node v_i with higher confidence, the difference of each component value of $\sigma(\hat{\boldsymbol{Y}}_{S_i})$ is greater, its information entropy $H(\sigma(\hat{\boldsymbol{Y}}_{S_i}))$ is smaller, and its acceptance factor m_i is larger.

Finally, we select the unlabeled nodes with high acceptance factor (i.e, high confidence, essentially) to expand the labeled node set, whose pseudo labels are set according to the scaled soft label (i.e., $\text{argmax}(\hat{\boldsymbol{Y}}_S)$). The expanded label node set $\mathcal{V}^*_{labeled}$ is computed as follows:

$$\begin{aligned}\mathcal{V}^*_{labeled} = \mathcal{V}_{labeled} \cup \{v_i | v_i \in \mathcal{V}_{unlabeled}, \\ m_i \geq \frac{1}{|\mathcal{V}_{labeled}|} \sum_{j \in \mathcal{V}_{labeled}} m_j\}.\end{aligned} \tag{9}$$

Compared with the original training set, the expanded training set has more training nodes, which can provide more supervision information for GNNs training. Moreover, the expanded training set is holistically more loyal due to the enhanced confidence of unlabeled loyal nodes during the loyal node discovery phase.

Loyal node emphasis. After the loyal node discovery phase, we get the expanded labeled node set $\mathcal{V}^*_{labeled}$, which will be used to train the GNN model. To make the trained loyalty-aware, we use FT-Score to measure the loyalty of all labeled nodes and increase the training weight of nodes with high loyalty. Specifically, we propose a novel annealing mechanism for the training weights of labeled nodes based on their FT-Score. The modified training weight w_i for node v_i is computed as follows:

$$w_i = w_{min} + \frac{1}{2}(w_{max} - w_{min})(1 - \cos(\frac{Rank(FT(v_i))}{|\mathcal{V}^*_{labeled}|}\pi)), \tag{10}$$

where w_{min} and w_{max} are hyperparameters used to adjust the lower and upper bounds of the training weights, respectively, and $Rank(FT(v_i)) \in [1, |\mathcal{V}^*_{labeled}|]$ is the FT-Score non-decreasing rank of node v_i among $\mathcal{V}^*_{labeled}$. It can be proved that in Eq. (10), a labeled node with a lower FT-Score will be assigned a lower training weight, and the modified training weight of each node will be in the range of $[w_{min}, w_{max}]$.

After obtaining the modified training weights, we define the training loss of the loyalty-aware GNN, which is calculated as follows:

$$\mathcal{L}_{loyalty} = -\sum_{i \in \mathcal{V}^*_{labeled}} w_i \sum_{k=1}^{C} \boldsymbol{Y}_{L_{i,k}} \log(\hat{\boldsymbol{Y}}_{L_{i,k}}), \tag{11}$$

where $\boldsymbol{Y}_L$ is the label of the expanded labeled node set $\mathcal{V}^*_{labeled}$ (ground-truth for the original labeled nodes, and pseudo labels for the expanded nodes), and $\hat{\boldsymbol{Y}}_L$ is the predicted soft label of the loyalty-aware GNN. The difference between the loyalty-aware GNN objective function (i.e., Eq. (11)) and vanilla GNN objective function (i.e., Eq. (4)) is that for loyalty-aware GNN, we assign personalized training weights to each training node according to its FT-Score, which encourages loyalty-aware GNN to accept high-quality supervision.

By employing the LoyalDE strategies, we discover and emphasize loyal nodes to obtain a loyalty-aware GNN, therefore achieving superior performance under low-quality supervision. The overall process of the proposed LoyalDE training strategy is summarized as Algorithm 1.

Algorithm 1 Training GNNs for Semi-supervised Node Classification using LoyalDE Strategy.

Input: attributed graph $G = (\mathcal{V}, \mathcal{E}, \boldsymbol{X})$, labeled node set $\mathcal{V}_{labeled}$, unlabeled node set $\mathcal{V}_{unlabeled}$, node label matrix $\boldsymbol{Y}$, FT-Score trade-off parameter α, soft label diffusion times K, the boundary of modified training weights w_{min} and w_{max}.

Train vanilla GNN model via optimizing Eq. (4);
Employ vanilla GNN to obtain the predicted soft label $\hat{\boldsymbol{Y}}$;
Conduct K-rounds soft label diffusion to obtain $\hat{\boldsymbol{Y}}_D$ via Eq. (5);

Train adaptive scaler via optimizing Eq. (7);
Obtain personalized scaling temperature $\boldsymbol{T}$ via Eq. (6)
Employ personalized scaling temperature $\boldsymbol{T}$ and predicted soft label $\hat{\boldsymbol{Y}}$ to compute the scaled soft label $\hat{\boldsymbol{Y}}_S$
Obtain the expanded training set $\mathcal{V}^*_{labeled}$ via Eq. (8) and Eq. (9).

Compute the FT-Score for each node in the expanded training set $\mathcal{V}^*_{labeled}$ via Eq. (3);
Obtain modified weight for each node according to their FT-Score rank via Eq. (10);
Train loyalty-aware GNN model via optimizing Eq. (11);
Employ the loyalty-aware GNN model to obtain the predicted label $\hat{\boldsymbol{Y}}_L$;

Output: predicted label $\hat{\boldsymbol{Y}}_L$.

4. Experiments

In this section, we evaluate the performance of our proposed training strategy LoyalDE. Specifically, in Section 4.1, we introduce five public benchmark datasets for the semi-supervised node classification task. In Section 4.2, we present the used GNNs and some state-of-the-art training strategies for enhancing GNNs. In Section 4.3, detailed experimental settings are provided. Section 4.4 presents the comparative results of LoyalDE and baselines on the semi-supervised node classification task. We also conduct an ablation study and sensitivity analysis in Sections 4.5 and 4.6, respectively. Moreover, we provide in-depth analyses of the loyal node discovery and loyal node emphasis in Sections 4.7 and 4.8, respectively. Furthermore, in Section 4.9, We verify the generalization of LoyalDE by transferring it to the graph classification task. Finally, we analyze the node loyalty distribution of typical graphs in Section 4.10.

Table 2
Statistics of the five public benchmark datasets for node classification.

Dataset	#Nodes	#Features	#Edges	#Classes
Cora	2708	1433	5278	7
CiteSeer	3327	3703	4552	6
PubMed	19,717	500	44,324	3
CoraFull	19,793	8710	63,421	70
OGB-Arxiv	169,343	128	1,166,243	40

4.1. Datasets

We conduct experiments on five public benchmark datasets, including Cora, CiteSeer, PubMed, CoraFull, and OGB-Arxiv, to verify the performance of LoyalDE. These datasets originate from the commonly used citation networks (Hu et al., 2020; Sen et al., 2008). Each dataset owns the complete information of the attributed graph (i.e., topological structure, node attributes, and node labels). Nodes represent documents, and edges represent reference links. It is worth noting that the OGB-Arxiv dataset is a directed graph, and we preprocess it into an undirected graph before GNNs training. Table 2 shows the statistics of the above datasets.

4.2. Baselines

Since our proposed LoyalDE is a model-agnostic hot-plugging training strategy for semi-supervised node classification, we selected three representative GNNs as our vanilla GNN model:

- **GCN** (Kipf & Welling, 2016) This method employs the Chebyshev first-order convolutions based on spectral theory to capture the global graph structure and learn node embeddings.
- **GAT** (Veličković et al., 2017) This method adaptively adjusts the learning weight of the central node to different neighbor information through the learnable attention coefficient to effectively learn the node embeddings.
- **GraphSAGE** (Hamilton et al., 2017) This method samples a fixed number of neighbors for each central node and

Table 3
Experimental results of node classification with baselines on the Cora, CiteSeer, PubMed and CoraFull datasets about test accuracy (%).

Dataset	Cora			CiteSeer			PubMed			CoraFull		
Loyal proportion	0.0	0.5	1.0	0.0	0.5	1.0	0.0	0.5	1.0	0.0	0.5	1.0
GCN	77.1	79.1	82.3	66.9	69.3	71.0	74.7	76.7	79.8	57.0	58.5	60.7
GCN (Self-Training)	78.5$_{(+1.4\%)}$	78.9$_{(-0.2\%)}$	77.8$_{(-4.5\%)}$	63.6$_{(-3.3\%)}$	64.6$_{(-4.7\%)}$	67.9$_{(-3.1\%)}$	75.7$_{(+1.0\%)}$	77.3$_{(+0.6\%)}$	78.6$_{(-1.2\%)}$	57.7$_{(+0.7\%)}$	58.6$_{(+0.1\%)}$	59.4$_{(-1.3\%)}$
GCN (Co-Training)	77.9$_{(+0.8\%)}$	78.3$_{(-0.8\%)}$	81.2$_{(-1.1\%)}$	70.2$_{(+3.3\%)}$	70.8$_{(+1.5\%)}$	72.0$_{(+1.0\%)}$	80.3$_{(+5.6\%)}$	82.2$_{(+5.5\%)}$	80.8$_{(+1.0\%)}$	57.7$_{(+0.7\%)}$	59.0$_{(+0.5\%)}$	57.8$_{(-2.9\%)}$
GCN (LoyalDE)	**84.6**$_{(+7.5\%)}$	**83.5**$_{(+4.4\%)}$	**85.5**$_{(+3.2\%)}$	**71.1**$_{(+4.2\%)}$	**75.0**$_{(+5.7\%)}$	**74.1**$_{(+3.1\%)}$	**83.5**$_{(+8.8\%)}$	**83.2**$_{(+6.5\%)}$	**83.3**$_{(+3.5\%)}$	**59.5**$_{(+2.5\%)}$	**60.4**$_{(+1.9\%)}$	**62.3**$_{(+1.6\%)}$
GAT	72.3	76.9	80.3	65.6	68.8	69.0	74.7	76.9	81.5	55.8	56.5	58.7
GAT (Self-Training)	68.3$_{(-4.0\%)}$	70.2$_{(-6.7\%)}$	75.0$_{(-5.3\%)}$	68.8$_{(+3.2\%)}$	64.8$_{(-4.0\%)}$	69.6$_{(+0.6\%)}$	54.9$_{(-19.8\%)}$	76.2$_{(-0.7\%)}$	77.6$_{(-3.9\%)}$	56.9$_{(+1.1\%)}$	58.7$_{(+2.2\%)}$	59.2$_{(+0.5\%)}$
GAT (Co-Training)	75.9$_{(+3.6\%)}$	77.1$_{(+0.2\%)}$	80.4$_{(+0.1\%)}$	69.8$_{(+4.2\%)}$	70.7$_{(+1.9\%)}$	70.8$_{(+1.8\%)}$	80.8$_{(+6.1\%)}$	81.5$_{(+4.6\%)}$	80.9$_{(-0.6\%)}$	56.3$_{(+0.5\%)}$	55.2$_{(-1.3\%)}$	56.5$_{(-2.2\%)}$
GAT (LoyalDE)	**79.6**$_{(+7.3\%)}$	**81.0**$_{(+4.1\%)}$	**84.0**$_{(+3.7\%)}$	**71.8**$_{(+6.2\%)}$	**72.0**$_{(+3.2\%)}$	**72.1**$_{(+3.1\%)}$	**82.3**$_{(+7.6\%)}$	**82.1**$_{(+5.2\%)}$	**82.6**$_{(+1.1\%)}$	**57.3**$_{(+1.5\%)}$	**59.3**$_{(+2.8\%)}$	**60.1**$_{(+1.4\%)}$
GraphSAGE	71.1	72.3	73.3	61.4	64.4	69.4	67.0	69.5	72.7	51.1	54.1	55.2
GraphSAGE (Self-Training)	72.5$_{(+1.4\%)}$	72.2$_{(-0.1\%)}$	71.3$_{(-2.0\%)}$	66.0$_{(+4.6\%)}$	67.7$_{(+3.3\%)}$	66.8$_{(-2.6\%)}$	62.9$_{(-4.1\%)}$	70.5$_{(+1.0\%)}$	72.0$_{(-0.7\%)}$	57.0$_{(+5.9\%)}$	57.8$_{(+3.7\%)}$	58.3$_{(+3.1\%)}$
GraphSAGE (Co-Training)	73.4$_{(+2.3\%)}$	72.7$_{(+0.4\%)}$	73.3$_{(+0.0\%)}$	67.1$_{(+5.7\%)}$	71.5$_{(+7.1\%)}$	69.8$_{(+0.4\%)}$	72.4$_{(+5.4\%)}$	76.1$_{(+6.6\%)}$	79.4$_{(+6.7\%)}$	54.8$_{(+3.7\%)}$	55.3$_{(+1.2\%)}$	56.1$_{(+0.9\%)}$
GraphSAGE (LoyalDE)	**77.2**$_{(+6.1\%)}$	**75.0**$_{(+2.7\%)}$	**76.3**$_{(+3.0\%)}$	**69.2**$_{(+7.8\%)}$	**73.9**$_{(+9.5\%)}$	**74.5**$_{(+5.1\%)}$	**76.1**$_{(+9.1\%)}$	**78.5**$_{(+9.0\%)}$	**80.7**$_{(+8.0\%)}$	**58.0**$_{(+6.9\%)}$	**60.4**$_{(+6.3\%)}$	**58.5**$_{(+3.3\%)}$

conducts information aggregation. We choose GCN as the aggregator in all experiments.

We further compare the proposed training strategy LoyalDE with state-of-the-art training strategies for enhancing GNNs. Brief descriptions of these methods are presented below:

- **Self-Training** (Li et al., 2018) This approach combines self-training with GNNs and expands the training set by collecting the most confident predictions for each class.
- **Co-Training** (Li et al., 2018) This approach expands the training set by utilizing random walks to explore the topological structure of the graph and find the most confident nodes for each class.

4.3. Experimental settings

For the reproducibility of experimental results, we fixed the random seed as 2022 in all experiments. For the Cora, CiteSeer, and PubMed datasets, the split of validation nodes and test nodes follows the Planetoid setting (Yang et al., 2016). For the CoraFull dataset, we randomly sample 500 validation nodes and 1000 test nodes. For the OGB-Arxiv dataset, the split of validation nodes and test nodes follows the original default setting. In order to reveal the influence of the graph supervision loyalty problem, for all datasets, we sort nodes not in the validation set and test set according to their FT-Score. Then, for each class, 20 training nodes are randomly sampled from the top half (as loyal nodes) and the bottom half (as disloyal nodes) of the FT-Score according to a certain loyal proportion (i.e., 0.0, 0.5, 1.0). For each vanilla GNN model (i.e., GCN, GAT, GraphSAGE), we adopt a 2-layer structure, the dimension of the hidden layer is 64, and the activation function is ReLu (Agarap, 2018). Moreover, for each dataset, the learning rate is set to 1e-2, the maximum number of epochs is 200, and the optimization is realized by the Adam optimizer. For Cora, CiteSeer, and PubMed datasets, the dropout rate is set to 0.5, and the weight decay is set to 5e-4. For the CoraFull dataset, the weight decay is set to 5e-3. In addition, for the proposed LoyalDE training strategy, we introduced four hyper-parameters, FT-Score trade-off parameter α, soft label diffusion times K, and boundary of modified training weights w_{min} and w_{max}, in all experiments, we set $\alpha = 0.5$, and tune K between 2 and 10, w_{min} between 0.2 and 0.5, and w_{max} between 0.7 and 1.0. In addition, we perform the hyperparameter search for LoyalDE using the Optuna framework (Akiba et al., 2019), which only contains K, w_{min} and w_{max}. The implementation of all methods is based on PyG and Pytorch frameworks. All experiments are conducted on a machine with Intel Xeon Gold 5218R CPU, two GeForce RTX 3090 GPUs, and 250 GB Memory. The operating environment is Ubuntu 22.04, with CUDA 11.8.

Table 4
Experimental results of node classification with baselines on the OGB-Arxiv dataset about test accuracy (%). '–': Run more than 24 h.

Dataset	OGB-Arxiv		
Loyal proportion	0.0	0.5	1.0
GCN	54.6	58.9	62.8
GCN (Self-Training)	55.2$_{(+0.6\%)}$	59.4$_{(+0.5\%)}$	61.1$_{(-1.7\%)}$
GCN (Co-Training)	–	–	–
GCN (LoyalDE)	**62.3**$_{(+7.7\%)}$	**61.9**$_{(+3.0\%)}$	**64.1**$_{(+1.3\%)}$
GAT	54.1	58.6	63.1
GAT (Self-Training)	57.5$_{(+3.4\%)}$	57.1$_{(-1.5\%)}$	62.0$_{(-1.1\%)}$
GAT (Co-Training)	–	–	–
GAT (LoyalDE)	**61.7**$_{(+7.6\%)}$	**60.9**$_{(+2.3\%)}$	**63.4**$_{(+0.3\%)}$
GraphSAGE	53.5	57.3	60.2
GraphSAGE (Self-Training)	52.7$_{(-0.8\%)}$	56.8$_{(-0.5\%)}$	59.8$_{(-0.4\%)}$
GraphSAGE (Co-Training)	–	–	–
GraphSAGE (LoyalDE)	**60.3**$_{(+6.8\%)}$	**60.1**$_{(+2.8\%)}$	**63.7**$_{(+3.5\%)}$

4.4. Experimental results

The comparative results of the proposed LoyalDE and other baseline training strategies with several vanilla GNN models are shown in Tables 3 and 4. Each column in the table represents a data split. For each data split, we compare the performance of three vanilla GNNs (GCN, GAT, GraphSAGE) trained using four different GNN training strategies (vanilla training, Self-Training, Co-Training, and LoyalDE). We indicate the best-performing training strategy for each vanilla GNN model in **bold** and the sub-optimal performance in underline. Additionally, we report the improvement or decrease in performance for each training strategy compared to vanilla GNNs as subscripts. Moreover, We provide the detailed settings corresponding to each result in Appendix A.

It can be observed that for each dataset, the performance of vanilla GNNs gradually improves as the proportion of loyal nodes in the training set increases, which again illustrates the significant impact of the graph supervision loyalty problem on the semi-supervised node classification. After employing the proposed LoyalDE training strategy, the performance of the GNNs outperforms vanilla training consistently. Particularly, LoyalDE brings up to 9.1% performance improvement to vanilla GNNs on the PubMed dataset. Moreover, after performing LoyalDE, the performance of GNNs is no longer as sensitive to the proportion of loyal nodes in the training set as vanilla training. This reveals that through our loyal node discovery and emphasis strategy, potential loyal nodes are injected into the original training set, and the model performance can indeed be improved by adjusting the training weights of loyal nodes.

Compared with the other two training strategies, it can be found that LoyalDE always maintains superior performance. It is worth noting that for extreme cases, when all nodes in the

Table 5
Ablation study on the Cora and CiteSeer datasets about test accuracy (%).

Dayaset	Cora									CiteSeer								
GNN model	GCN			GAT			GraphSAGE			GCN			GAT			GraphSAGE		
Loyal proportion	0.0	0.5	1.0	0.0	0.5	1.0	0.0	0.5	1.0	0.0	0.5	1.0	0.0	0.5	1.0	0.0	0.5	1.0
w/o Discovery	79.6	80.0	82.0	77.3	80.4	82.3	69.5	70.4	69.8	67.7	72.3	73.5	66.0	67.6	69.9	63.0	69.3	67.7
w/o Emphasis	81.2	80.7	82.4	75.4	77.7	81.7	73.7	72.6	72.6	68.7	73.6	71.4	68.5	70.5	71.3	64.3	70.4	68.0
LoyalDE	**84.6**	**83.5**	**85.5**	**79.6**	**81.0**	**84.0**	**77.2**	**75.0**	**76.3**	**71.1**	**75.0**	**74.1**	**71.8**	**72.0**	**72.1**	**69.2**	**73.9**	**74.5**

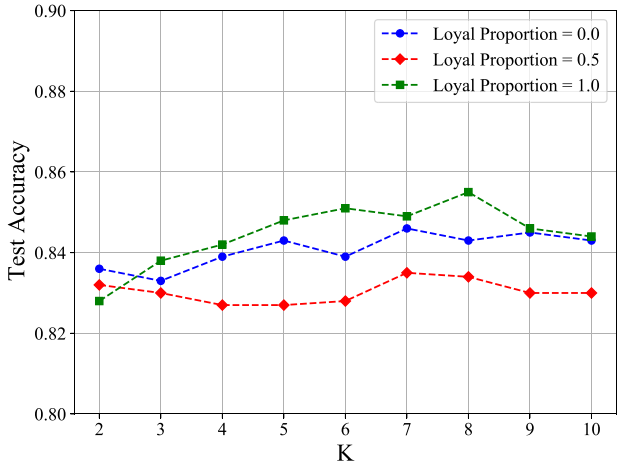


Fig. 4. Sensitivity analysis about the soft label diffusion times K.

training set are disloyal nodes, LoyalDE achieves 6.1%∼7.5% performance improvement on the Cora dataset, while Co-Training only brings 0.8%∼3.6% improvement, which is the suboptimal method in most cases. Moreover, Self-Training and Co-Training have a limited or even negative impact on model training in certain cases (e.g., GAT on the PubMed and CoraFull datasets with 0.0 loyal proportion). Since Self-Training discovers potential high-confidence unlabeled nodes by utilizing the model's prediction, we argue that when the quality of the labeled nodes is poor, the predicted soft label is likely to be wrong, which will bring label noise. Moreover, Co-Training employs random walks to discover potential high-confidence unlabeled nodes near the labeled nodes. We argue that the poor local homophily around disloyal nodes will also lead to the random walks injecting label noise to model training. Furthermore, for large-scale graph datasets like OGB-Arxiv, Co-Training cannot complete the training in the stipulated time limit, as it requires large-scale matrix inversion operation. In contrast, LoyalDE can still achieve efficient performance within a short time, indicating its scalability. Moreover, the primary source of computational overhead in LoyalDE arises from the calculation of the FT-Score for each node, but this can be preprocessed to reduce computation.

4.5. Ablation study

In this section, we conduct an ablation study to investigate each component's importance in LoyalDE. As mentioned above, LoyalDE can be divided into the loyal node discovery phase and the loyal node emphasis phase. Therefore, we remove the loyal node discovery phase (**w/o Discovery** for short) and the loyal node emphasis phase (**w/o Emphasis** for short), respectively, and compare their test accuracy with the completed LoyalDE on Cora and CiteSeer datasets with three vanilla GNNs. Specifically, for **w/o Discovery**, we simply conduct the loyal node emphasis to the original training sets $\mathcal{V}_{labeled}$ instead of the expanded training set $\mathcal{V}^*_{labeled}$ obtained by Eq. (9); For **w/o Emphasis**, we do not modify the training weights of each node in $\mathcal{V}^*_{labeled}$ according to Eq. (10), the objective function is the vanilla cross-entropy loss on $\mathcal{V}^*_{labeled}$, whose labels are ground-truth for the original labeled nodes, and pseudo labels for the expanded nodes. The other experimental settings for the ablation study are the same as the settings introduced in Section 4.3 and Appendix A. The experimental results of the ablation study are shown in Table 5.

From the experimental results, for different vanilla GNNs, both removing the loyal node discovery phase and the loyal node emphasis phase will lead to different degrees of performance degradation of the LoyalDE training strategy. For the Cora dataset with the GCN model, removing the loyal node discovery phase leads to a performance drop of 3.5%∼5.0%; Removing the loyal node emphasis phase leads to a performance drop of 2.8%∼3.4%. For the CiteSeer dataset with the GCN model, removing the loyal node discovery phase leads to a performance drop of 0.6%∼3.4%; Removing the loyal node emphasis phase leads to a performance drop of 1.4%∼2.7%. This reveals that both these two components have a non-negligible impact on the performance of LoyalDE.

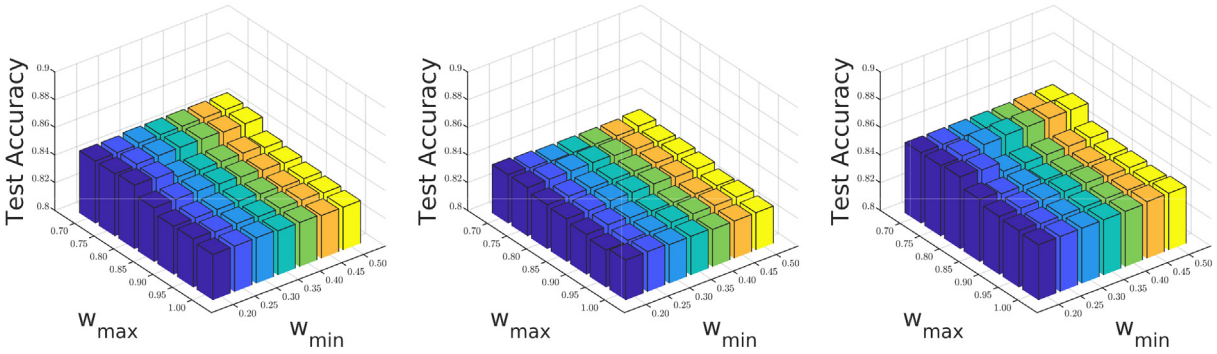
4.6. Sensitivity analysis

In this section, we design sensitivity analysis to verify the effect of different hyperparameter combinations on the performance of the proposed LoyalDE. We employ GCN as the vanilla GNN model and conduct experiments on the Cora datasets with three different loyal proportions of the training set (i.e., 0.0, 0.5, 1.0).

First, to investigate the sensitivity of the loyal node discovery phase, we fix w_{min} and w_{max} to the settings corresponding to the best results, and tune K between 2 and 10. The test accuracy on the Cora dataset with different loyal proportion are shown in Fig. 4. Experiments reveal that although the performance of LoyalDE fluctuates with the variation of soft label diffusion times K, it still tends to be stable between 82.0%∼85.0%, demonstrating that LoyalDE is insensitive to the value of K and has strong robustness.

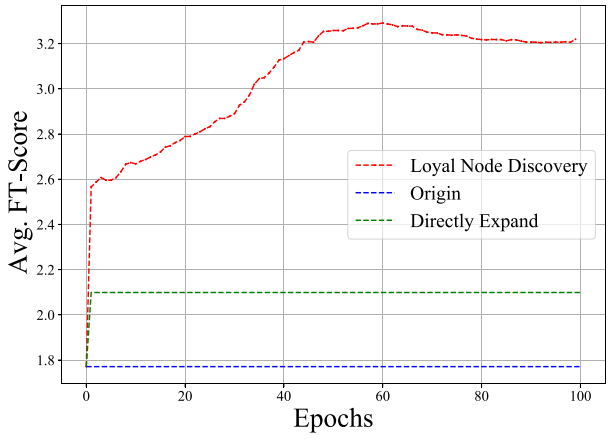
Then, to investigate the sensitivity of the loyal node emphasis phase, we fix the soft label diffusion times K to the settings corresponding to the best results and tune w_{min} between 0.2 and 0.5, and w_{max} between 0.7 and 1.0. The results of different loyal proportions on the Cora dataset are shown in Fig. 5. The experimental results demonstrate that LoyalDE is insensitive to changes in w_{min} and w_{max}. The performance variation of LoyalDE is stable within 3.0% despite varying the combinations of values of w_{min} and w_{max}, demonstrating that our proposed FT-Score can effectively measure the labeled node loyalty, and the proposed annealing mechanism can robustly adjust the training weights of all labeled nodes. At the same time, it also reveals that the loyal node emphasis phase can greatly promote GNNs to learn to a certain extent according to the loyalty of labeled nodes and keep the performance at a fairly high level.

In conclusion, the proposed training strategy LoyalDE is insensitive to all these hyperparameters and holistically robust.

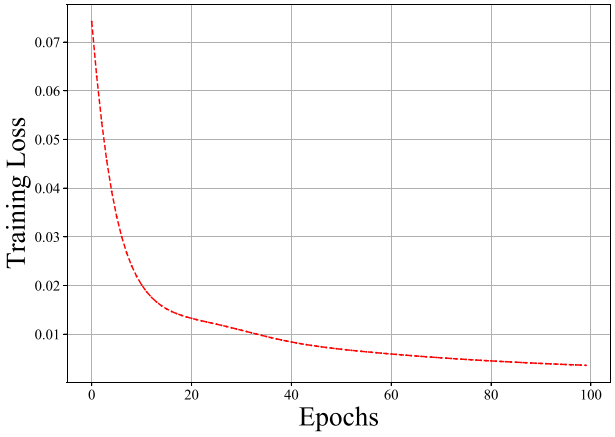


(a) Loyal Proportion = 0.0 (b) Loyal Proportion = 0.5 (c) Loyal Proportion = 1.0

Fig. 5. Sensitivity analysis about the boundary of modified training weights w_{min} and w_{max}.



(a) Average FT-Score of the expanded training set



(b) Training loss of the adaptive scaler

Fig. 6. Effectiveness analysis of loyal node discovery.

4.7. Analysis of loyal node discovery

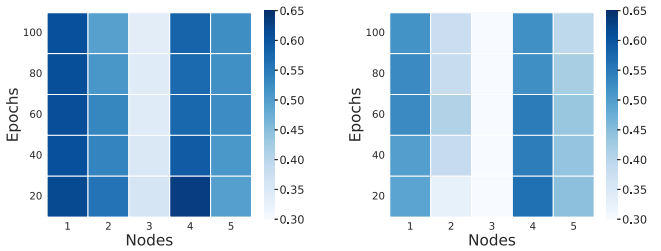
In this section, we utilize experiments to evaluate the effectiveness of the loyal node discovery phase. As mentioned earlier, in the loyal node discovery phase, we train an adaptive scaler to obtain personalized scaling temperature $\boldsymbol{t}_i$ for each node v_i, which will enhance the confidence of loyal nodes, while weakening the confidence of disloyal nodes. Finally, the scaled soft label will be used to expand the training set.

Table 6
The average FT-Score of the original/expanded training sets of Cora and CoraFull datasets.

Dataset	Avg. FT-Score (ori.)	Avg. FT-Score (exp.)
Cora	1.77	3.22
CoraFull	2.28	3.10

We use GCN as the vanilla GNN model and conduct experiments on the Cora dataset with a 0.0 loyal proportion. At each training epoch of the adaptive scaler before the convergence, we use the adaptive scaler to calculate the personalized scaling temperature and expand the training set, and report the average FT-Score of the expanded training set in Fig. 6(a). Meanwhile, the training loss of the adaptive scaler is also presented in Fig. 6(b). As observed, the average FT-Score of the original training set is 1.77. If we simply use the soft label without adaptive scaling to expand the training set, the FT-Score will only increase to 2.09. This is because the original training set contains many disloyal nodes. Therefore, the confidence in vanilla GNN is high for many disloyal nodes. In contrast, our proposed adaptive scaler encourages the confidence of loyal nodes and weakens the confidence of disloyal nodes. After the training converges, the average FT-Score of the expanded training set gradually increases to 3.22. Experimental results demonstrate that the proposed loyal node discovery phase can effectively discover potential loyal nodes and improve the quality of supervision information.

Furthermore, we compare the training set average FT-Score improvements brought by the loyal node discovery phase for the Cora and CoraFull datasets with 0.0 loyal proportion. Specifically, the average FT-Score of the original training set $\mathcal{V}_{labeled}$ is denoted as $Avg.FT-Score(ori)$, and the average FT-Score of the expanded training set $\mathcal{V}^*_{labeled}$ is denoted as $Avg.FT-Score(exp)$. The experimental results are shown in Table 6. As observed, for the Cora dataset (which has 7 classes), the loyal node discovery phase improves the average FT-Score of the training set from 1.77 to 3.22 (almost doubled). In contrast, for the CoraFull dataset (which has 70 classes), the loyal node discovery phase improves the average FT-Score of the training set from 2.28 to 3.10 (slightly improved). Since a higher average FT-Score improvement indicates that the loyal node discovery phase contributes more loyal nodes to the training set, we argue that it is more challenging to discover loyal nodes in graphs with more classes.



(a) Vanilla GCN (b) Loyalty-aware GCN

Fig. 7. Comparison of HS values of vanilla GCN and loyalty-aware GCN on disloyal test nodes.

4.8. Analysis of loyal node emphasis

As mentioned earlier, the loyalty node emphasis phase aims to train a loyalty-aware GNN, which focuses on receiving high-quality supervision from loyal nodes to obtain well-learned node representations. In this section, we investigate the difference between the loyalty-aware GNN and the vanilla GNN in handling disloyal test nodes.

Since disloyal test nodes have poor local homophily, their predicted soft label should be different from their neighbors if the GNN is well-trained to distinguish these nodes. Therefore, for each disloyal test node, we compute the average of the predicted soft label similarity between it and each of its neighbors with different labels. We denote this as the heterophily similarity (HS). Formally, HS of node v_i is computed as follows:

$$\mathrm{HS}(v_i) = \frac{1}{|\mathcal{N}_i|} \sum_{j \in \mathcal{N}_i} \mathbb{1}(\boldsymbol{Y_i} \neq \boldsymbol{Y_j}) \frac{\hat{\boldsymbol{Y}}_{\boldsymbol{i}} \hat{\boldsymbol{Y}}_{\boldsymbol{j}}}{\|\hat{\boldsymbol{Y}}_{\boldsymbol{i}}\| \, \|\hat{\boldsymbol{Y}}_{\boldsymbol{j}}\|}, \tag{12}$$

where $\boldsymbol{Y}_i$ is the ground-truth of node v_i, $\hat{\boldsymbol{Y}}_{\boldsymbol{i}}$ is the predicted soft label of node v_i, and $\mathbb{1}(\cdot)$ is the identity function. Notably, different GNN models will output different predicted soft labels, leading to different HS values for the same central node. Furthermore, a lower HS value indicates that the soft label of the central node can be distinguished from the soft label of its neighbors with different classes. In other words, the GNN model which has lower HS values is well-learned.

We use GCN as the vanilla GNN model and conduct experiments on the Cora dataset with a 0.0 loyal proportion. Specifically, we randomly select five nodes from the FT-Score bottom half of the test node set (i.e., disloyal test nodes). Then, we report the HS values of these nodes every 20 epochs during the training process of the vanilla GNN and the loyalty-aware GNN, respectively. The experimental results are shown in Fig. 7. As observed, for disloyal test nodes, loyalty-aware GNN always achieves lower HS values than vanilla GNN, which demonstrates that loyalty-aware can effectively distinguish disloyal test nodes from their neighbors with different labels.

4.9. Generalization

As graph supervision loyalty is a general problem in graph machine learning, it is natural to consider transferring LoyalDE to the graph classification task with slight modifications. We provide a detailed explanation and algorithm for the transferred LoyalDE in Appendix B.

We conduct experiments on three benchmark graph classification datasets, including PROTEINS, ENZYMES (Borgwardt et al., 2005), and MUTAG (Debnath et al., 1991). All these datasets originate from the biochemical field. Each dataset contains multiple graphs with graph labels. Table 7 shows the statistics of the above datasets. For all datasets, we employ one-layer GCN, GAT, and GraphSAGE as our GNN model, connected with three fully connected layers, all the hidden layer dimensions are set to 64, and we adopt mean pooling as our READOUT function. Moreover, for all datasets, the training, validation, and test set are randomly sampled according to the ratio of 0.2: 0.4: 0.4. The other experimental settings are the same as the node classification task settings presented in Section 4.3.

Table 7
Statistics of the three public benchmark datasets for graph classification. '∼' denotes the average number.

Dataset	#Graphs	#Nodes	#Edges	#Features	#Classes
PROTEINS	1113	∼39.1	∼145.6	3	2
ENZYMES	600	∼32.6	∼124.3	3	6
MUTAG	188	∼17.9	∼39.6	7	2

Table 8
Experimental results of graph classification on the PROTEINS, ENZYMES, and MUTAG datasets about test accuracy (%).

Dataset	PROTEINS	ENZYMES	MUTAG
GCN	69.7	27.5	80.2
GCN (LoyalDE)	$70.6_{(+0.9\%)}$	$32.1_{(+4.6\%)}$	$85.5_{(+5.3\%)}$
GAT	69.9	24.1	82.9
GAT (LoyalDE)	$70.4_{(+0.5\%)}$	$25.0_{(+0.9\%)}$	$89.5_{(+6.6\%)}$
GraphSAGE	71.3	24.5	77.6
GraphSAGE (LoyalDE)	$71.7_{(+0.4\%)}$	$26.3_{+(1.8\%)}$	$81.6_{(+4.0\%)}$

The experimental results are presented in Table 8. We also provide the detailed settings corresponding to each result in Appendix A. It can be found that the transferred LoyalDE training strategy consistently improves the performance of GCN on the graph classification tasks. It is worth noting that LoyalDE has a significant improvement on the MUTAG dataset (up to 6.6%). This further demonstrates that the graph supervision loyalty is a generality problem and the LoyalDE training strategy can significantly improve the performance of GNNs in multiple downstream tasks.

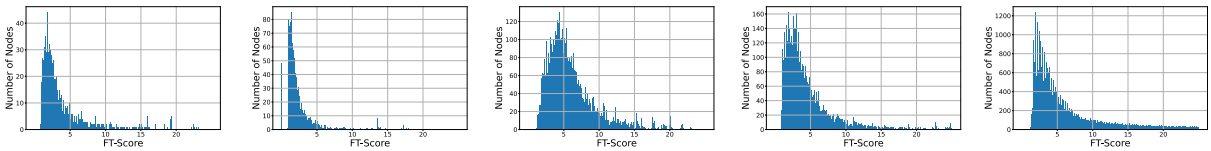
4.10. Node loyalty distribution of typical graphs

In this section, we briefly explore the statistical characteristics of node loyalty of typical graphs, aiming to answer this question: for a typical graph, do most nodes tend to be loyal?

We conduct experiments on the Cora, CiteSeer, PubMed, CoraFull, and OGB-Arxiv datasets. Specifically, for each dataset, we divide the FT-Score of all nodes into 1000 sub-intervals at equal intervals and count the number of nodes in each sub-interval. The experimental results are shown in Fig. 8. The x-axis represents FT-Score, and the y-axis represents the number of nodes in each sub-interval. Experimental results demonstrate that for a typical graph, the FT-Score (loyalty) of most nodes is at a low value, and as FT-Score (loyalty) increases, the number of nodes decreases significantly. In other words, most nodes tend to be disloyal.

5. Conclusion

In this paper, we investigate the graph supervision loyalty problem. To the best of our knowledge, we are the first to define the graph supervision loyalty problem and demonstrate its effect on the performance of GNNs. To this end, we devise a novel node loyalty measure FT-Score, which considers both the topology and feature similarity of a node. Moreover, we propose a novel model-agnostic hot-plugging training strategy LoyalDE, which can discover and emphasize potential nodes with high loyalty during model training to improve the performance of GNNs. The



(a) Cora (b) CiteSeer (c) PubMed (d) CoraFull (e) OGB-Arxiv

Fig. 8. FT-Score distribution of five datasets under $\alpha = 0.5$.

Table 9
The average FT-Score of the training set on the Cora and CiteSeer datasets.

Dataset	Cora			CiteSeer		
Loyal proportion	0.0	0.5	1.0	0.0	0.5	1.0
Avg. FT-Score	1.77	3.23	5.07	1.27	2.46	3.06

Table 10
The average FT-Score of the training set on the PubMed and CoraFull datasets.

Dataset	PubMed			CoraFull		
Loyal proportion	0.0	0.5	1.0	0.0	0.5	1.0
Avg. FT-Score	3.64	5.60	7.85	2.28	4.49	6.43

Table 11
The average FT-Score of the training set on the OGB-Arxiv dataset.

Dataset	OGB-Arxiv		
Loyal proportion	0.0	0.5	1.0
Avg. FT-Score	3.00	5.49	18.07

experimental results on the semi-supervised node classification task demonstrate that LoyalDE significantly improves the performance of GNNs and outperforms several state-of-the-art training strategies for enhancing GNNs' performance. Furthermore, we transfer the LoyalDE to the graph classification task and verify its generalization.

Declaration of competing interest

We declare that we have no financial and personal relationships with other people or organizations that can inappropriately influence our work, there is no professional or other personal interest of any nature or kind in any product, service, or company that could be construed as influencing the position presented in, or the review of, the manuscript entitled.

Data availability

Data will be made available on request.

Acknowledgments

We are grateful to the anonymous referee, who made valuable suggestions to help improve the paper. This paper was supported by the Shandong Provincial Natural Science Foundation, China (ZR2020MA064).

Appendix A. Details about datasets and experiments

For the node classification task, we split the Cora, CiteSeer, PubMed, CoraFull, and OGB-Arxiv datasets according to the loyal proportion of 0.0, 0.5, and 1.0. For the split data, the average FT-Score of the training set is shown in Tables 9, 10, 11. In addition, we verified the effectiveness of LoyalDE on the three vanilla GNN models, including GCN, GAT, and GraphSAGE. The parameters corresponding to the best performance are shown in Tables 12, 13, 14, 15, 16, 17, 18, 19, 20.

For the graph classification task, we split the PROTEINS, ENZYMES, and MUTAG datasets according to the fixed train-valid-test split ratio (0.2:0.4:0.4). We verified the effectiveness of the transferred LoyalDE on three vanilla GNN models, including GCN, GAT, and GraphSAGE. The parameters corresponding to the best performance are shown in Tables 21, 22, 23.

Table 12
Hyperparameter settings that obtain the best results of LoyalDE on the Cora and CiteSeer datasets with GCN model.

Dataset	Cora			CiteSeer		
Loyal proportion	0.0	0.5	1.0	0.0	0.5	1.0
K	7	7	8	10	7	8
w_{min}	0.20	0.25	0.25	0.25	0.35	0.25
w_{max}	0.80	0.85	0.85	0.90	1.00	0.80

Table 13
Hyperparameter settings that obtain the best results of LoyalDE on the PubMed and CoraFull datasets with GCN model.

Dataset	PubMed			CoraFull		
Loyal proportion	0.0	0.5	1.0	0.0	0.5	1.0
K	9	10	4	2	2	2
w_{min}	0.30	0.50	0.45	0.45	0.30	0.45
w_{max}	0.80	0.80	0.85	1.00	0.85	0.85

Table 14
Hyperparameter settings that obtain the best results of LoyalDE on the OGB-Arxiv dataset with GCN model.

Dataset	OGB-Arxiv		
Loyal proportion	0.0	0.5	1.0
K	5	4	6
w_{min}	0.25	0.20	0.35
w_{max}	0.80	0.80	0.85

Table 15
Hyperparameter settings that obtain the best results of LoyalDE on the Cora and CiteSeer datasets with GAT model.

Dataset	Cora			CiteSeer		
Loyal proportion	0.0	0.5	1.0	0.0	0.5	1.0
K	5	2	2	2	10	2
w_{min}	0.30	0.25	0.30	0.25	0.30	0.35
w_{max}	0.85	1.00	0.90	0.90	0.85	0.80

Appendix B. Explanation and algorithm of transferred LoyalDE

For the graph classification task, since each training/test sample is an entire graph instead of a single node, there is no need

Table 16
Hyperparameter settings that obtain the best results of LoyalDE on the PubMed and CoraFull datasets with GAT model.

Dataset	PubMed			CoraFull		
Loyal proportion	0.0	0.5	1.0	0.0	0.5	1.0
K	10	6	3	2	2	2
w_{min}	0.35	0.45	0.40	0.40	0.40	0.35
w_{max}	0.85	0.85	0.95	1.00	0.85	0.80

Table 17
Hyperparameter settings that obtain the best results of LoyalDE on the OGB-Arxiv dataset with GAT model.

Dataset	OGB-Arxiv		
Loyal proportion	0.0	0.5	1.0
K	7	4	5
w_{min}	0.35	0.25	0.35
w_{max}	0.90	0.85	0.80

Table 18
Hyperparameter settings that obtain the best results of LoyalDE on the Cora and CiteSeer datasets with GraphSAGE model.

Dataset	Cora			CiteSeer		
Loyal proportion	0.0	0.5	1.0	0.0	0.5	1.0
K	10	3	10	10	10	3
w_{min}	0.25	0.30	0.20	0.30	0.25	0.20
w_{max}	1.00	1.00	0.85	0.85	0.85	0.90

Table 19
Hyperparameter settings that obtain the best results of LoyalDE on the PubMed and CoraFull datasets with GraphSAGE model.

Dataset	PubMed			CoraFull		
Loyal proportion	0.0	0.5	1.0	0.0	0.5	1.0
K	10	3	6	2	2	3
w_{min}	0.35	0.20	0.35	0.40	0.25	0.20
w_{max}	1.00	0.95	0.80	1.00	0.80	0.80

Table 20
Hyperparameter settings that obtain the best results of LoyalDE on the OGB-Arxiv dataset with GraphSAGE model.

Dataset	OGB-Arxiv		
Loyal proportion	0.0	0.5	1.0
K	6	5	8
w_{min}	0.45	0.20	0.45
w_{max}	0.90	0.80	0.85

Table 21
Hyperparameter settings that obtain the best results of transferred LoyalDE on the PROTEINS, ENZYMES, and MUTAG datasets with GCN model.

Dataset	PROTEINS	ENZYMES	MUTAG
w_{min}	0.35	0.25	0.45
w_{max}	0.90	0.80	0.85

Table 22
Hyperparameter settings that obtain the best results of transferred LoyalDE on the PROTEINS, ENZYMES and MUTAG datasets with GAT model.

Dataset	PROTEINS	ENZYMES	MUTAG
w_{min}	0.25	0.30	0.25
w_{max}	0.90	0.80	0.80

to employ the loyal node discovery phase to expand the training node set. However, we can still encourage GNNs to obtain high-quality supervision from loyal nodes. Thus, we modified our loyal node emphasis phase to adapt to the graph classification task. Specifically, we compute the average FT-score of all nodes in an entire graph as the graph's loyalty. The average FT-Score of graph $G = (\mathcal{V}, \mathcal{E}, \boldsymbol{X})$ is denoted as $\widetilde{FT}(G)$, which is computed as follows:

$$\widetilde{FT}(G) = \frac{1}{|\mathcal{V}|} \sum_{i \in \mathcal{V}} \sum_{j \in \mathcal{N}_i} \alpha \frac{\boldsymbol{x}_i \boldsymbol{x}_j}{\|\boldsymbol{x}_i\| \, \|\boldsymbol{x}_j\|} + (1-\alpha) \frac{|\mathcal{N}_i \cap \mathcal{N}_j|}{|\mathcal{N}_i \cup \mathcal{N}_j|}, \tag{13}$$

where α is the trade-off parameter to perform linear weighting on feature similarity and topology similarity. Then, the graphs with a higher average FT-score (higher average node loyalty) will be given larger training weights. The modified training weight of labeled graph G_i is computed as follows:

$$w_i = w_{min} + \frac{1}{2}(w_{max} - w_{min})\left(1 - \cos\left(\frac{Rank(\widetilde{FT}(G_i))}{|\mathcal{G}_{labeled}|}\pi\right)\right), \tag{14}$$

where w_{min} and w_{max} are hyperparameters used to adjust the lower and upper bounds of the training weights, $Rank(\widetilde{FT}(G_i)) \in [1, |\mathcal{G}^*_{labeled}|]$ is the average FT-Score non-decreasing rank of graph G_i among labeled graph set $\mathcal{G}_{labeled}$.

Finally, the objective function of GNNs for graph classification is calculated as follows:

$$\mathcal{L}_{GC} = -\sum_{i \in \mathcal{G}_{labeled}} w_i \sum_{k=1}^{C} \boldsymbol{Y}_{i,k} \log(\hat{\boldsymbol{Y}}_{i,k}), \tag{15}$$

where C is the number of graph classes, $\boldsymbol{Y}$ is the ground-truth, and $\hat{\boldsymbol{Y}}$ is the predicted soft label of GNNs.

The algorithm of transferred LoyalDE for the graph classification task is presented in Algorithm 2.

Table 23
Hyperparameter settings that obtain the best results of transferred LoyalDE on the PROTEINS, ENZYMES, and MUTAG datasets with GraphSAGE model.

Dataset	PROTEINS	ENZYMES	MUTAG
w_{min}	0.45	0.20	0.35
w_{max}	0.85	0.95	0.80

Algorithm 2 Training GNNs with LoyalDE Strategy on the Graph Classification Task.

Input: attributed graph set $\mathcal{G}$, labeled graph set $\mathcal{G}_{labeled}$, unlabeled graph set $\mathcal{G}_{unlabeled}$, graph label matrix $\boldsymbol{Y}$, FT-Score trade-off parameter α, the boundary of modified training weights w_{min} and w_{max}.

1: Compute the average FT-Score for each labeled graph via Eq. (13);
2: Obtain modified weight for each labeled graph according to their average FT-Score rank via Eq. (14);
3: Train GNN model via optimizing Eq. (15);
4: Employ the GNN model to obtain the predicted graph label $\hat{\boldsymbol{Y}}$;

Output: predicted label $\hat{\boldsymbol{Y}}$.

References

Agarap, A. F. (2018). Deep learning using rectified linear units (relu). arXiv preprint arXiv:1803.08375.

Akiba, T., Sano, S., Yanase, T., Ohta, T., & Koyama, M. (2019). Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 2623–2631).

Blum, A., & Chawla, S. (2001). Learning from labeled and unlabeled data using graph mincuts.

Blum, A., Lafferty, J., Rwebangira, M. R., & Reddy, R. (2004). Semi-supervised learning using randomized mincuts. In *Proceedings of the twenty-first international conference on machine learning* (p. 13).

Borgwardt, K. M., Ong, C. S., Schönauer, S., Vishwanathan, S., Smola, A. J., & Kriegel, H.-P. (2005). Protein function prediction via graph kernels. *Bioinformatics*, *21*(suppl_1), i47–i56.

Bruna, J., Zaremba, W., Szlam, A., & LeCun, Y. (2013). Spectral networks and locally connected networks on graphs. arXiv preprint arXiv:1312.6203.

Chapelle, O., & Zien, A. (2005). Semi-supervised classification by low density separation. In *International workshop on artificial intelligence and statistics* (pp. 57–64). PMLR.

Coskun, M., Gungor, B. B., & Koyuturk, M. (2019). Expanding label sets for graph convolutional networks. arXiv preprint arXiv:1912.09575.

Dai, G., Wang, X., Zou, X., Liu, C., & Cen, S. (2022). MRGAT: Multi-relational graph attention network for knowledge graph completion. *Neural Networks*, *154*, 234–245.

Debnath, A. K., Lopez de Compadre, R. L., Debnath, G., Shusterman, A. J., & Hansch, C. (1991). Structure-activity relationship of mutagenic aromatic and heteroaromatic nitro compounds. Correlation with molecular orbital energies and hydrophobicity. *Journal of Medicinal Chemistry*, *34*(2), 786–797.

Defferrard, M., Bresson, X., & Vandergheynst, P. (2016). Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in Neural Information Processing Systems*, *29*.

Dong, H., Ding, Z., He, X., Feng, F., & Bi, S. (2020). Data augmentation view on graph convolutional network and the proposal of Monte Carlo graph learning. arXiv preprint arXiv:2006.13090.

Dornaika, F., Bi, J., & Zhang, C. (2023). A unified deep semi-supervised graph learning scheme based on nodes re-weighting and manifold regularization. *Neural Networks*, *158*, 188–196.

Friedman, J., Hastie, T., & Tibshirani, R. (2001). *The elements of statistical learning. vol. 1*. New York, NY, USA: 2001: Springer.

Guo, Z., & Wang, H. (2020). A deep graph neural network-based mechanism for social recommendations. *IEEE Transactions on Industrial Informatics*, *17*(4), 2776–2783.

Hamilton, W., Ying, Z., & Leskovec, J. (2017). Inductive representation learning on large graphs. *Advances in Neural Information Processing Systems*, *30*.

Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M., & Leskovec, J. (2020). Open graph benchmark: Datasets for machine learning on graphs. arXiv preprint arXiv:2005.00687.

Joachims, T. (2003). Transductive learning via spectral graph partitioning. In *Proceedings of the 20th international conference on machine learning* (pp. 290–297).

Ju, W., Luo, X., Ma, Z., Yang, J., Deng, M., & Zhang, M. (2022). GHNN: Graph harmonic neural networks for semi-supervised graph-level classification. *Neural Networks*, *151*, 70–79.

Kipf, T. N., & Welling, M. (2016). Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907.

Levie, R., Monti, F., Bresson, X., & Bronstein, M. M. (2018). Cayleynets: Graph convolutional neural networks with complex rational spectral filters. *IEEE Transactions on Signal Processing*, *67*(1), 97–109.

Li, J., Cai, D., & He, X. (2017). Learning graph-level representation for drug discovery. arXiv preprint arXiv:1709.03741.

Li, Q., Han, Z., & Wu, X.-M. (2018). Deeper insights into graph convolutional networks for semi-supervised learning. In *Thirty-second AAAI conference on artificial intelligence*.

Liben-Nowell, D., & Kleinberg, J. (2003). The link prediction problem for social networks. In *Proceedings of the twelfth international conference on information and knowledge management* (pp. 556–559).

Murphy, A. H. (1973). A new vector partition of the probability score. *Journal of Applied Meteorology and Climatology*, *12*(4), 595–600.

Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., & Eliassi-Rad, T. (2008). Collective classification in network data. *AI Magazine*, *29*(3), 93.

Shannon, C. E. (1948). A mathematical theory of communication. *The Bell System Technical Journal*, *27*(3), 379–423.

Tan, L., Yao, W., & Li, X. (2020). Expanding training set for graph-based semi-supervised classification. In *International conference on database and expert systems applications* (pp. 245–258). Springer.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., & Bengio, Y. (2017). Graph attention networks. arXiv preprint arXiv:1710.10903.

Wang, X., Liu, H., Shi, C., & Yang, C. (2021). Be confident! Towards trustworthy graph neural networks via confidence calibration. *Advances in Neural Information Processing Systems*, *34*, 23768–23779.

Wang, Z., Shao, R., Wang, C., Hu, C., Wang, C., & Gong, Z. (2021). Expanding semantic knowledge for zero-shot graph embedding. In *International conference on database systems for advanced applications* (pp. 394–402). Springer.

Weston, J., Ratle, F., & Collobert, R. (2008). Deep learning via semi-supervised embedding. In *Proceedings of the 25th international conference on machine learning* (pp. 1168–1175).

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., & Philip, S. Y. (2020). A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, *32*(1), 4–24.

Xia, W., Wang, S., Yang, M., Gao, Q., Han, J., & Gao, X. (2022). Multi-view graph embedding clustering network: Joint self-supervision and block diagonal representation. *Neural Networks*, *145*, 1–9.

Yang, Z., Cohen, W., & Salakhudinov, R. (2016). Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning* (pp. 40–48). PMLR.

Yang, S., Zhang, Z., Zhou, J., Wang, Y., Sun, W., Zhong, X., Fang, Y., Yu, Q., & Qi, Y. (2020). Financial risk analysis for SMEs with graph-based supply chain mining. In *IJCAI* (pp. 4661–4667).

Zhou, J., Cui, G., Hu, S., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., & Sun, M. (2020). Graph neural networks: A review of methods and applications. *AI Open*, *1*, 57–81.

Zhu, X., Ghahramani, Z., & Lafferty, J. D. (2003). Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the 20th international conference on machine learning* (pp. 912–919).

Zhu, J., Yan, Y., Zhao, L., Heimann, M., Akoglu, L., & Koutra, D. (2020). Beyond homophily in graph neural networks: Current limitations and effective designs. *Advances in Neural Information Processing Systems*, *33*, 7793–7804.