Graph Relearn Network: Reducing Performance Variance and Improving Prediction Accuracy of Graph Neural Networks

Zhenhua Huang[a,b,c,1], Kunhao Li[a,1], Yihang Jiang[a], Zhaohong Jia[a,*], Linyuan Lv[b,c,*] and Yunjie Ma[c,d]

[a] *Anhui University, Hefei, 230039, China*
[b] *University of Science and Technology of China, Hefei, 230026, China*
[c] *Institute of Dataspace, Hefei Comprehensive National Science Center, Hefei, 230088, China*
[d] *Hefei University of Technology, Hefei, 230009, China*

ARTICLE INFO

Keywords:
Graph Neural Networks
Prediction Stability
Rewrite Mechanism
Node Classification

Abstract

Recent studies have shown that the predictive performance of graph neural networks (GNNs) is inconsistent and varies across different experimental runs, even with identical parameters. The prediction variability limits GNNs' applicability, and the underlying reasons remain unclear. We identified a key factor contributing to this issue: the oscillation of the predicted classes of some nodes during GNN training. To address this problem, we propose a novel framework, known as Graph Relearn Network (GRN), designed to reduce prediction variance by iteratively refining the predictions of unstable nodes. The GRN framework operates in two phases: pre-predict and relearn. During the pre-predict phase, a graph-dense encoder is trained to pre-predict the node categories. In the relearn phase, the model intensively focuses on the unstable nodes to optimize the predictions. Extensive experiments on ten graph datasets demonstrate that the GRN significantly enhances the performance stability of GNNs (with std. reduced by up to 75%), and achieves state-of-the-art performance in prediction accuracy (increased by up to 11.97%). GRN improves the performance stability of GNNs by mitigating the disruptions caused by unstable nodes and enhances the prediction accuracy in node classification tasks.

1. Introduction

Graph neural networks (GNNs) [1] have been extensively applied in numerous domains such as knowledge representation [2], text classification [3, 4], traffic prediction [5], recommendation systems [6, 7], and anomaly detection [8]. Classic and advanced GNNs include GCN [9], GAT [10], GIN [11], LightGCN [12], UniMP [13], FusedGAT [14], ASDGN [15], and RAHG [16], etc. These models predominantly follow the message-passing framework [17, 18]. Current research primarily focuses on developing new methods to enhance node or graph representations [13, 14, 19], explain GNN predictions [20, 21, 22, 23, 24], and improve the robustness of GNNs [25, 26, 27]. However, recent studies have revealed that GNNs are susceptible to attacks [28, 26], leading to significant performance degradation when graph structures are manipulated. Several approaches have been proposed for enhancing the robustness of GNNs [29, 30, 25, 27]. Despite these studies, the current literature lacks a comprehensive understanding of the underlying factors that contribute to GNN performance instability. In addition, previous studies have primarily reported the average performance and standard variance of GNNs [31, 10, 11, 14], present considerable performance variances or instability compared to traditional machine learning approaches [32, 33]. This phenomenon is evident even under consistent parameter settings. For example, in an experiment involving 100 runs on the Airports (USA) dataset as illustrated in Fig. 1, traditional machine-learning methods such as SVM showed consistent accuracy. The trivial deep neural network (DNN), a multilayer perceptron, exhibited slight fluctuations in prediction performance. However, the graph neural network, specifically GCN, demonstrated significant variance in performance despite utilizing graph structure to achieve the highest prediction accuracy. The accuracy fluctuated between 48.12% and 54.6%. Similarly, on the Cora dataset, GCN's accuracy varied from 85.84% to 88.66%. This performance instability in node classification has not been adequately addressed in previous studies and the underlying causes remain unclear [34].

By visualizing and analyzing the training process and predictions of GNNs, we identified a crucial factor that affects the stability of GNN representations. As depicted in Fig. 2, during the GCN training on the Cora dataset [9], the predicted label of node 2186 fluctuated between the blue and green classes. Similarly, the central node 2276 was classified as green in the 167th epoch, changed to orange in the 172nd epoch, and finally transitioned to blue in the 223rd epoch. These shifts in the predicted classes (referred to as unstable nodes) during the training process contribute to the performance instability of GNN predictions. Our statistical analysis further indicated that the majority of these unstable nodes were typically located at the periphery and junctions of clusters or communities within the graph.

Our analysis shows that unstable nodes account for approximately 5% to 15% of the total nodes in models such as ChebNet, GCN, GAT, FusedGAT, and ASDGN. Current message-passing-based GNNs are affected by unstable

*Corresponding Authors

zhhuangscut@gmail.com (Z. Huang); kunhomlihf@gmail.com (K. Li); yihangjiangahu@gmail.com (Y. Jiang); zhjia@mail.ustc.edu.cn (Z. Jia); linyuan.lv@ustc.edu.cn (L. Lv); ma.yunjie@foxmail.com (Y. Ma)

ORCID(s): 0000-0003-3178-9721 (Z. Huang); 0000-0002-4952-0969 (K. Li); 0000-0001-6607-7025 (Z. Jia)

[1] Equal Contributions

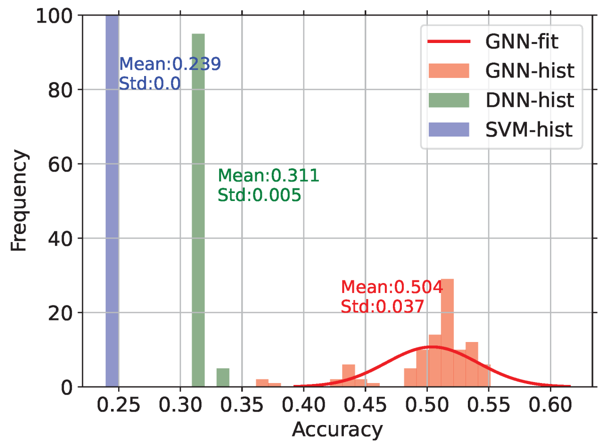


Fig. 1: Accuracy distribution of different models after 100 trials on the Airports (USA) dataset.

nodes, which can compromise their prediction accuracy. However, these models do not effectively utilize historical prediction data to address the issue of unstable nodes during each training epoch. To overcome this challenge, we developed a relearn mechanism inspired by the rewrite mechanism [35]. Originally designed to eliminate inconsistent words from a generated response prototype and to rewrite them to align with a specific personality, we adapt this concept for node classification to prioritize inconsistent nodes and enhance their representations for consistent predictions over time. This approach is similar to the sample reweighting strategies in machine learning, in which less reliable samples are assigned more weight to enhance their prediction accuracy [36].

Building on this concept, we introduce the Graph Relearn Network (GRN), a novel framework consisting of two alternating training phases: pre-predict and relearn. In the pre-predict phase, a graph-dense encoder (GDE) is trained to predict the node classes. We specifically design a graph-dense block (GDB) to learn node representations and mitigate the over-smoothing issue [1] in GNNs, inspired by dense network architectures [37]. However, some predictions from the pre-predict phase may still be faulty and require correction during the relearn phase.

In the relearn phase, an unstable node detector based on spectral clustering is employed to identify unstable nodes in the pre-predict phase. We incorporate a probability walk based on the nodes' degree to sample stable nodes and balance classes. To improve the predictive consistency of central nodes during training, we concentrate on stabilizing the predictions of volatile nodes within their local structure and introduce a neighbor supporter. The primary objective of this supporter is to improve the learning of the local node structure and representation.

By decreasing the proportion and influence of unstable nodes, our model significantly enhances the prediction stability of GNNs. Furthermore, the prediction accuracy of the GRN is markedly improved. Extensive experiments demonstrate that the instability of GNNs notably affects their prediction accuracy.

The main contributions of this study are as follows:

- We have identified a primary cause of prediction instability in GNNs: the oscillation of predicted node classes during training.
- We propose a novel framework, the graph relearn network (GRN), to improve the performance instability of GNNs. In the GRN, the relearn phase is designed to correct the pre-predictions of the graph neural network. Moreover, GRN alleviates the skipping phenomenon of the predicted node classes.
- The GRN considerably enhances the performance stability of GNNs by up to 75% on node classification tasks, and the prediction accuracy is also increased by up to 11.97% compared to strong baselines in extensive experiments on ten real-world graph datasets [2].

The remainder of this paper is organized as follows. In Section 2, we review related studies on the design of GNN, GNN stability, and rewrite mechanism. Section 3 introduces the node classification symbols that appear in this paper and explains the reason for the unstable prediction phenomenon. The framework of the GRN is introduced in detail in Section 4. Extensive experiments in Section 5 verify the performance of GRN and baselines. Finally, Section 6 summarizes this study.

2. Related Works

2.1. Designing of Graph Neural Networks

The concept of graph neural networks, which dates back to 2009, was designed to process data represented in graph formats [38]. Research in this field has increased recently [39, 40, 1]. Broadly, GNN research can be categorized into two broad streams. The first focuses on applying GNNs to solve diverse problems, such as knowledge representation, text generation, traffic forecasting, recommendation systems, molecular generation, and action recognition [2, 3, 5, 12, 41, 42, 43, 44], etc. The second stream focuses on the fundamental theories and architectures of GNNs [45, 9, 10, 46, 13, 47] etc. ChebNet [45] uses graph spectral and Chebyshev polynomials to learn features of graph-format data. GCN [9] simplifies the ChebNet approach by using feature propagation to aggregate adjacency information of nodes, enhancing computational efficiency. GAT [10] dynamically weights the significance of each node's neighbors, significantly improving node classification performance. GraphSage [48] extends GCN into an inductive framework that can predict unseen nodes by leveraging sampled neighborhood functions. Shi et al. [13] proposed a novel unified message-passing model (UniMP) that incorporates feature and label propagation in both the training

[2]The code link has released in https://github.com/PreckLi/Graph-Relearn-Network

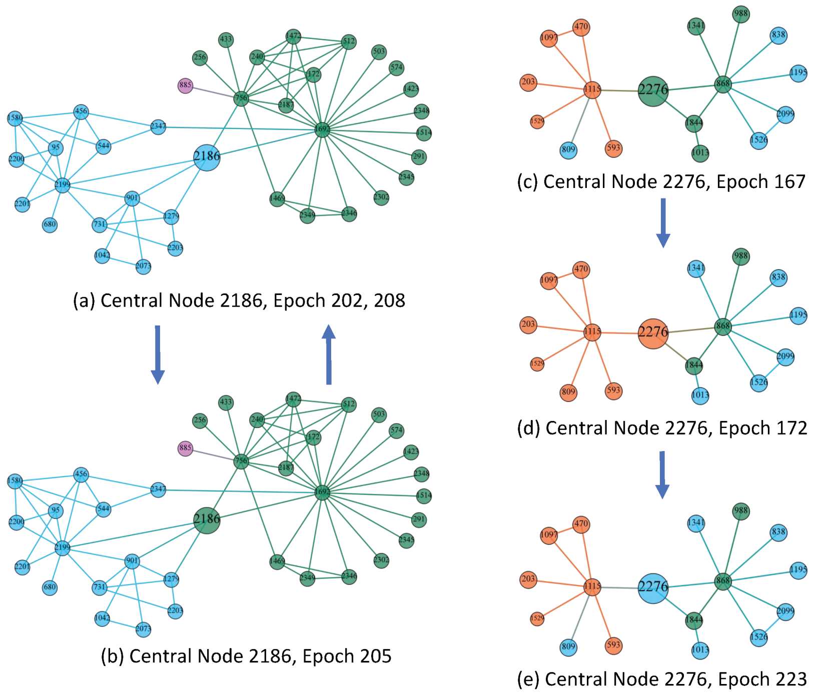

(a) Central Node 2186, Epoch 202, 208
(b) Central Node 2186, Epoch 205
(c) Central Node 2276, Epoch 167
(d) Central Node 2276, Epoch 172
(e) Central Node 2276, Epoch 223

Fig. 2: Phenomenon of the predicted labels skip between classes in Cora. Nodes with the same color indicate the same class.

and inference phases. GPS [49] represents a scalable graph Transformer with linear complexity that supports multiple types of encodings and enhances model adaptability. The anti-symmetric deep graph network (ASDGN) [15] offers a framework for stable and non-dissipative GNN design, inspired by ordinary differential equations to preserve the long-range information between nodes.

However, an analysis of the average accuracy and standard deviation reported in these previous studies revealed that the performance of GNNs varies significantly across different experimental runs. Few studies have addressed or discussed the phenomenon of performance instability in GNNs, and they have not attempted to mitigate its effects. Ideally, a more effective GNN should consistently deliver stable performance and achieve higher accuracy with reduced variability.

2.2. Stability of Graph Neural Networks

Most studies on stability have focused on the robustness of deep neural networks to the disturbance of external noise. Zheng et al. [50] presented a general stability training method designed to fortify deep networks against minor input distortions resulting from common types of image processing. For graph robustness, Wang et al. [51] analyzed the stability of convolutional neural networks on the relative perturbation of Laplace-Beltrami operators to understand the stability of a GNN on large graphs. Li et al. [25] introduced an unsupervised framework designed to refine the graph structure, substantially boosting the robustness of the vanilla GCN while maintaining the same level of computational complexity. Kenlay et al. [30] established a clear upper limit, demonstrating that graph neural networks maintain their stability when rewiring occurs among high-degree nodes. Song et al. [27] proposed a general framework for graph neural networks, which lays the foundation for a new class of robust GNNs. Arghal et al. [26] limited the frequency response of GNN filter banks, offering PAC-style guarantees on GNN stability based on scenario optimization results. Zhao et al. [52] explored GNNs based on different neural flows, focusing on how these relate to several stability concepts, including BIBO stability, Lyapunov stability, structural stability, and conservative stabilities. Dong et al. [53] proposed a novel approach to enhance the expressivity of GNNs through graph canonization, demonstrating a trade-off between expressivity and stability, and introducing universal graph canonization as a potential solution.

However, the type of instability in prediction accuracy discussed in this study differs from the issues addressed in the above works. It is crucial to examine the performance stability because deploying an unstable model is challenging in real-world applications, and addressing this instability could significantly impact the prediction accuracy of GNNs.

2.3. Rewrite Mechanism

Ren et al. [36] introduced a new meta-learning algorithm that adjusts the weights of the training samples according to their gradient directions. This approach significantly improves the performance in scenarios with class imbalances and corrupted labels, particularly when only a limited quantity of clean validation data is accessible. Elgohary et al. [54] presented the concept of rewriting for the question-in-context task, where a context-dependent question is transformed into a self-contained one with the same answer, based on the historical context of a conversation.

TABLE I
NOTATIONS IN GRN.

Symbols	Definitions or descriptions
G	A general graph
V	Node set
P	Node predictions during E epochs
F_{in}, F_{hid}, F_c	Feature dimension of input, hidden, classes
V_u, V_{st}	Unstable node set, sampled stable neighbors
V_{sp}	Sampled node set
r	Sample ratio in sampler
N, N_s	Number of nodes and sampled nodes
Z, $\tilde{Z}$	Output embedding of GDE and the supporter
Y	Label of nodes
$\mathcal{L}_p$, $\mathcal{L}_r$	Loss functions in pre-predict and relearn phases

Song et al. [35] proposed a powerful framework to avoid the generation of inconsistent persona words during text generation. The generate-delete-rewrite mechanism deletes inconsistent words from a generated response prototype and further rewrites it to a personality-consistent one. Ponnusamy et al. [55] proposed a self-learning system for large-scale conversational AI agents. The method employs an absorbing Markov chain model for collaborative filtering and utilizes a guardrail rewrite selection mechanism that assesses these corrections based on feedback friction data. AFR [56] is a rapid approach for modeling updating, aimed at minimizing the dependence on non-essential features. It modifies the final layer of a conventionally trained ERM base model using a weighted loss. This adjustment prioritizes instances where the ERM model underperforms, effectively increasing focus on the minority group without requiring explicit group labels. However, the rewrite mechanism is unsuitable for GNNs; thus, we design a relearn mechanism to support and correct the predictions of GNNs and prevent nodes from skipping between classes.

3. Node Classification Task

A general graph is represented by $G = (V, A, X)$, where $V = \{v_1, v_2, \dots, v_N\}$ denotes the node set, $A \in \mathbb{R}^{N\times N}$ is the adjacency matrix, and $X \in \mathbb{R}^{N\times F_{in}}$ is the original features of the nodes. $P \in \mathbb{R}^{N\times E}$ captures node predictions over the most recent E epochs. $V_u = \{v_{u1}, v_{u2}, \dots, v_{un}\}$ and $V_{sp} = \{v_{sp1}, v_{sp2}, \dots, v_{spn}\}$ denote the unstable nodes set and sampled nodes set, respectively.

To predict the labeling task of a node (i.e., the node classification task), the set of node labels is represented as Y_L. The labeled and unlabeled node sets from V can be expressed as $V_L = \{v_{l1}, \cdots, v_{ln}\}$ and $V_{UL} = \{v_{ul1}, \cdots, v_{uln}\}$.The goal of the node classification task is to leverage the graph G and the set of labeled nodes V_L to train the GNN, which facilitates the prediction of classes for the unlabeled nodes in V_{UL}. The symbols used in this paper along with their definitions and descriptions are summarized in Table I.

4. Graph Relearn Network

The framework of GRN is shown in Fig. 3. In the pre-predict phase, the graph-dense encoder (GDE) is trained on the graph data to generate node pre-predictions. We record the node classifications from the last E epochs after GDE convergence and store them in P, which is utilized by a detector in the relearn phase.

During the relearn phase, the detector encodes the node predictions into a binary pulse matrix P' using the latest E predictions. P' captures the changes in the node prediction, and a spectral clustering method is then employed to identify the unstable nodes V_u. To ensure class balance and enhance the representativeness of negative samples, sampling strategies are implemented. We adopt a degree probability mechanism for negative sampling to produce stable neighbors V_{st} around V_u. The sampled set V_{sp} comprises V_u and V_{st}. In the relearning phase, V_{sp} is fed into a neighbor supporter that leverages the local neighbor information to reinforce the evidence of nodes belonging to the same class.

The pre-predict and relearn phases are trained alternately. The framework of the GRN is presented in Algorithm 2.

4.1. Pre-Predict Phase

The pre-predict phase includes the GDE and its training process. The predictions in this phase are intermediate results of GRN.

4.1.1. Graph Dense Encoder

In the pre-predict phase, a graph-dense encoder (GDE) is used to train node representations motivated by DenseNet [37] and ResNet [57]. A GDE includes multiple graph-dense blocks (GDBs). A GDB has one or multiple graph-dense modules (GDMs). Following GAT [10], a GDM in GDB is referred to as a head. We stack multi-head GDM in a GDB. The main structure of GDB is shown in Fig. 4. To learn structure features and prevent the node representations of the model from decaying effectively, we employ a dense-net structure to connect features between the GDB layers and input node features. This combination allows the integrated nodes to retain more information and preserve their differences, thereby mitigating the issue of over-smoothing. The design of the GDB amplifies feature differentiation between nodes while enabling the GRN to learn with minimal information loss, facilitating subsequent detection and sampling of unstable nodes.

The convolution operations in the GDM are formulated as follows:

$$\hat{X}_k^{l+1} = \sigma(Lap\hat{X}^l\mathcal{W}_k + X^t Res_k + \hat{X}^l\mathcal{D}_k) \tag{1}$$

where $Lap = D - A$ is the Laplacian matrix, and D and A are the degree and adjacency matrix of the nodes, respectively. $X^t \in \mathbb{R}^{N\times F_{in}}$ is the input feature of the t^{th} block, where F_{in} is the dimension size. $\hat{X}^l \in \mathbb{R}^{N\times F_{out}}$ is the input feature of l^{th} the GDM, where F_{out} is the output dimension size. $\hat{X}^1$ is set as X^t. $Res_k \in \mathbb{R}^{F_{in}\times F_{out}}$, $\mathcal{D}_k \in \mathbb{R}^{F_{out}\times F_{out}}$, and $\mathcal{W}_k \in \mathbb{R}^{F_{out}\times F_{out}}$ are the residual weight,

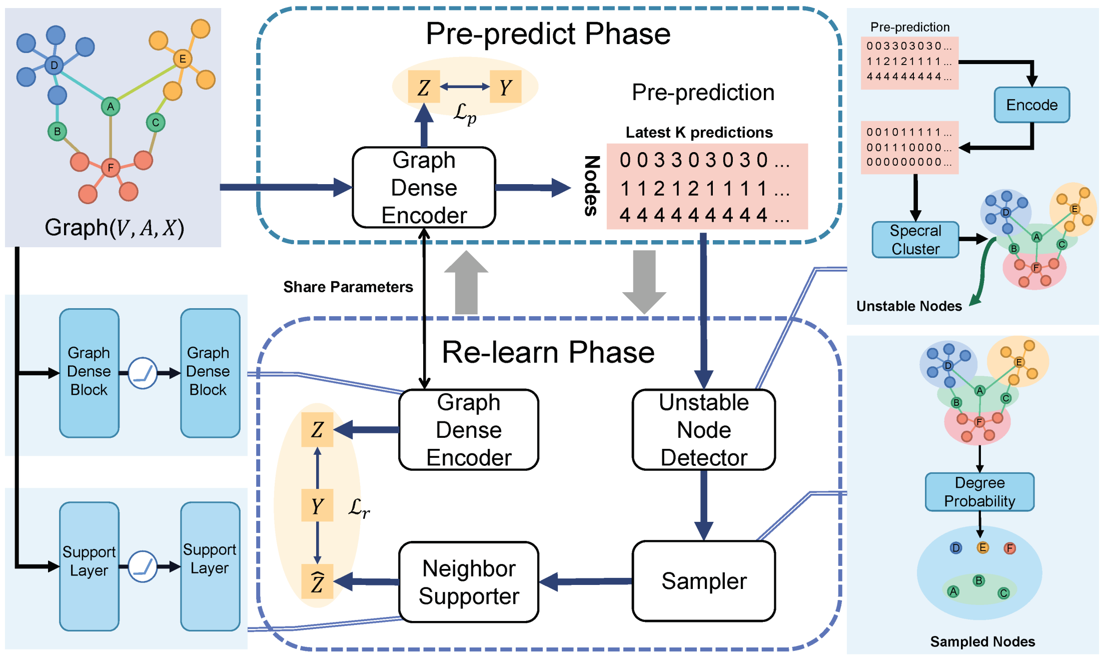


Fig. 3: Framework of the GRN. The parameters of GDE are shared in the pre-predict and relearn phases.

dense weight, and shared weight matrix of $Head_k$ (a GDM), respectively. Before the first graph convolution layer, a linear layer is applied to transform the dimension of X^t from F_{in} to F_{out}.

The output of a GDB is:

$$X^{t+1} = \sum_{k=1}^{\mathcal{K}} \hat{X}_k^L \tag{2}$$

We adopt the sum pool to stack the outputs of several heads and obtain the output of the GDB, where $\mathcal{K}$ is the number of heads and L is the depth of a graph-dense layer.

4.1.2. *Training of Pre-Predict Phase*

In the pre-predict phase, we train a graph-dense encoder (GDE) with two graph-dense blocks (GDBs). The output features $Z \in \mathbb{R}^{N\times C}$ of the GDE are as follows:

$$Z = softmax(GDB_2(ReLU(GDB_1(X, A)), A)) \tag{3}$$

where C is the number of node classes.

The layer of GDB can also be adjusted to four or six layers, etc. The GDE can also be replaced by other graph neural networks such as GCN, GAT, etc. The loss function is cross-entropy for the pre-predict phase, which is computed as:

$$\mathcal{L}_p = -\sum_{i\in V_L}\sum_{c=1}^{C} Y_{ic} \ln Z_{ic} \tag{4}$$

4.2. Relearn Phase

The relearn phase includes unstable node detection, negative sampling, and classification optimization using a neighbor supporter. The phase corrects the error pre-predictions from the pre-predict phase.

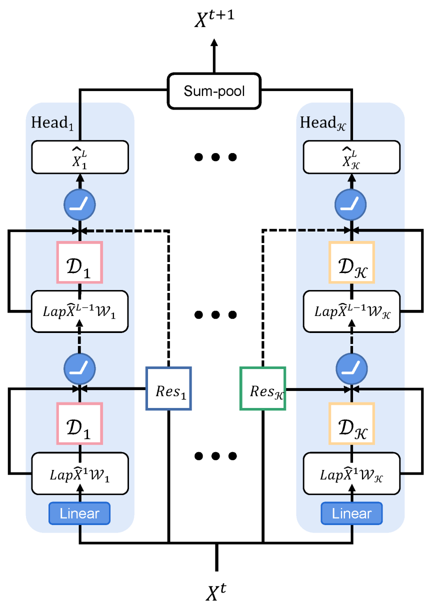


Fig. 4: Structure of a GDB.

4.2.1. *Unstable Node Detector*

The graph dense encoder (GDE) generates a matrix $P \in \mathbb{R}^{N\times E}$ that captures the predictions of the latest E epochs from the pre-predict phase. The matrix P is then utilized as the input for an unstable node detector, specifically designed to identify nodes that exhibit fluctuations in class predictions. In the detector, spectral clustering is employed to categorize nodes based on the variability of their class assignments. The detector's internal encoder converts class variation into a binary signal, where a change in prediction is encoded as 1, and no change is encoded as 0. Consequently, the pulse matrix $P' \in \mathbb{R}^{N\times E}$ is formalized as follows:

$$p'_{i,j} = \begin{cases} 1 & \text{if } p_{i,j} \neq p_{i,j+1}, \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

This binary encoding streamlines the detection of class skips by converting the prediction history into a sequence of pulses, where 1 indicates a skip. Subsequently, the pulse matrix P' is used as a feature matrix in the spectral clustering algorithm. Specifically, we derive the similarity matrix $W_s \in \mathbb{R}^{N\times N}$ from P', which plays a crucial role in delineating the proximities between the encoded prediction behaviors of the nodes. The similarity matrix W_s is essential for the spectral clustering process because it helps to categorize nodes into clusters that reflect stable and unstable predictions. The element $w_{i,j}$ of the similarity matrix is calculated as follows:

$$w_{i,j} = \sum_{k=1}^{N} \exp\left(-\frac{||\mathcal{P}'_i - \mathcal{P}'_k||^2}{2\sigma^2}\right), \tag{6}$$

where $P' = \{\mathcal{P}'_1, \cdots, \mathcal{P}'_N\}^T$. σ^2 represents the variance of the features in the pulse matrix P', and a summation is performed over all nodes to establish the full set of pairwise similarities.

The degree matrix D_s corresponding to P' is a diagonal matrix, where each diagonal element $d_{i,i}$ is the sum of similarities for node i:

$$d_{i,i} = \sum_{j=1}^{N} w_{i,j}. \tag{7}$$

The Laplacian matrix of the feature space is defined as $Lap_s = D_s - W_s$. By analyzing the eigenvectors $\{u_1, u_2\}$ associated with the two largest eigenvalues of Lap_s, we construct a feature matrix $\mathcal{U} \in \mathbb{R}^{N\times 2}$. Using the K-Means algorithm [58], we partition the nodes into two distinct groups based on their representations in the feature matrix $\mathcal{U}$: the stable node set V_{st} and the unstable node set V_u. This clustering helps to identify nodes with stable and unstable prediction patterns, facilitating targeted interventions to improve model accuracy and stability.

4.2.2. Sampler

The distribution of stable and unstable nodes identified by the detector is typically unbalanced, and focusing exclusively on unstable nodes during the relearning phase can introduce bias and inequity. To address this issue, the relearn phase incorporates a sampling strategy that broadens the scope of the samples based on the unstable nodes identified by the detector. Specifically, we employ negative sampling to select stable nodes from the k-hop neighbors of the unstable nodes. The approach enables the GRN to develop more comprehensive stable and unstable characteristics present in the graph.

Algorithm 1: Degree probability walk sampling.

```
Input: General graph $G = (V, A)$, unstable nodes $V_u$,
    sample ratio $r$
Output: Sampled nodes $V_{sp}$.
  $neg\_nodes = [\,]$
  $V_{sp} = [\,]$
  for $v_i$ in $V_u$ do
    $subset = \mathbf{subgraph}(v_i, A)$
    $num\_sample = r \times \mathbf{len}(subset)$
    while $\mathbf{len}(neg\_nodes) < num\_sample$ do
      $degrees = \mathbf{neighs\_d}(v_i)$
      $neighs\_prob = \frac{degrees}{\mathbf{sum}(degrees)}$
      $v_p = \mathbf{Choose}(subset, neighs\_prob)$
      $neg\_nodes.append(v_p)$
      if $v_i \neq v_p$ then
        Set $v_i = v_p$
      end if
    end while
  end for
  $V_{sp} = \mathbf{Union}(V_{sp}, neg\_nodes)$
  return $V_{sp}$
```

To sample representative nodes effectively, we found that nodes with higher degrees harbor more structural information regarding their surroundings or neighbors [59]. The sampler selects stable neighbors based on the degree probability within a two-hop neighborhood of the central unstable nodes. The output of the sampler is denoted by $V_{sp}(v_{sp1}, v_{sp2}, \cdots, v_{spn})$, which comprises V_u and their stable neighbors V_{st}. The process is detailed in Algorithm 1, where neg_nodes is the set of nodes sampled around the unstable nodes. The **subgraph** (v_i, A) is the function that obtains the two-hop neighbors of v_i, and **neighs_d** (v_i) is the function that computes the neighbors' degree of v_i, and **Choose** $(subset, neighs\_prob)$ is the function that selects nodes from the $subset$ based on the probability of $neighs\_degree$. This method ensures a balanced and informative selection of nodes for relearning and enhances the overall stability and accuracy of the model.

4.2.3. Neighbor Supporter

To enhance the prediction consistency of the central nodes during training and to target the predictions of unstable nodes within the local structure specifically, we introduce a component known as the neighbor supporter (supporter) during the relearn phase. The supporter consists of multiple support layers (SLs) that leverage k-hop neighbor structures and features to enhance the prediction accuracy of the central node. In addition, the supporter incorporates a mask matrix M, which is crucial for controlling the scope of aggregation within the sampled sets. This matrix ensures that the aggregation process remains confined to relevant neighborhood structures, thereby optimizing the learning process. The propagation process of a support layer (SL) is as follows:

$$H_c^{l+1} = D_s^{-1} M A^k M^T D_s^{-1} H_s^l \mathcal{W}_s \tag{8}$$

where D_s is the degree matrix of the sampled set V_{sp}, A^k is the adjacency matrix of the k power, $\mathcal{W}_s \in \mathbb{R}^{F_{in}\times F_{out}}$ denotes the learnable weights. $H_s^l \in \mathbb{R}^{N_c\times F_{in}}$ is the input matrix in the l^{th} supporter layer (SL) and $M \in \mathbb{R}^{N_s\times N_s}$ is the mask matrix of V_{sp}, which ensures that the source or destination nodes are included in V_{sp}. The $m_{i,j}$ is expressed as:

$$m_{i,j} = \begin{cases} 1 & \text{if} \quad i \in V_{sp} \ or \ j \in V_{sp} \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

Algorithm 2: Framework of GRN.

```
    Data: General graph $G = (A, V, X)$.
    Result: Representation of nodes $Z$.
1   Initialize the parameters $\mathcal{W}_k$, $R_k$, $\mathcal{D}_k$ of GDE ;
2   while not converged do
3       while epoch in pre-predict do
4           Compute the node representation $Z$ using GDE (Eq.(3));
5           Update $\mathcal{W}_k$, $R_k$, $\mathcal{D}_k$ using cross-entropy (Eq.(4));
6       end
7       Obtain the node predictions $P$;
8       Identify the unstable node set $V_u$ by the Detector (Sec.(4.2.1));
9       Determine the sampled node set $V_{sp}$ by the Sampler (Sec.(4.2.2));
10      Initialize the parameters $\mathcal{W}_s$ of the Supporter;
11      while epoch in relearn do
12          Recalculate the node representation $Z$ by the GDE (Eq.(3));
13          Generate the sampled node representations $\tilde{Z}$ by Supporter (Eq.(10));
14          Update $\mathcal{W}_k$, $R_k$, $\mathcal{D}_k$, $\mathcal{W}_s$ by the relearn loss (Eq.(11));
15      end
16  end
17  Return the final node representation $Z$ from the GDE;
```

4.2.4. *Training of Relearn Phase*

In the relearn phase, the GDE and the supporter are trained simultaneously. The supporter applies two support layers (SLs) to predict the categories of the sampled nodes. The output $\tilde{Z}$ is:

$$\tilde{Z} = softmax(SL_2(ReLU(SL_1(H_s, A)), A)) \tag{10}$$

where H_s denotes the features of V_{sp}.

The relearn loss equation is:

$$\mathcal{L}_r = -\sum_{i\in V_L}\sum_{c=1}^{C} Y_{ic} \ln Z_{ic} - \sum_{i\in V_{sp}}\sum_{c=1}^{C} Y_{ic} \ln \tilde{Z}_{ic} \tag{11}$$

where Y_c is the label of the sample set V_{sp}.

Overall, the training of the GRN involves two separate and alternating processes: training in pre-predict and training in relearn.

4.3. Time Complexity

During the training, the time expenditure associated with the explainable training phase of the Graph Relearn Networks (GRN) is primarily bifurcated into two distinct components: the pre-predict phase and relearn phases. The pre-predict phase predominantly encompasses the Graph Dense Encoder (GDE), which is meticulously constructed using $\mathcal{K}$ Graph Dense Blocks (GDBs). Consequently, the temporal complexity of the GDE can be formulated as $\mathcal{K} \times \{O(|E|\times F_{in}\times F_{hid})+O(F_{in}\times F_{hid})+O(F_{in}\times F_{hid})\}$, which is simplified to approximately $O(\mathcal{K} \times |E| \times F_{in} \times F_{hid})$.

The relearn phase integrates the GDE with a bilayer neighbor supporter. The time complexity attributed to the neighbor supporter is expressed as $O(|E| \times F_{in} \times F_{hid})$. In summary, the overall time complexity of the GRN's training phase is established as $O(\mathcal{K} \times |E| \times F_{in} \times F_{hid})$, which is $\mathcal{K}$ times the cost of GCN [9].

5. Experimental Analysis

5.1. Data Description

We verified the performance of the GRN on ten benchmark datasets as follows:

Cora [60]: Cora includes 2708 scientific publications on machine learning, and nodes are divided into seven categories according to the topics of the paper. Edges are the citations between papers.

Citeseer [60]: This dataset includes 3327 scientific publications, and nodes are divided into six categories.

Pubmed [60]: PubMed includes 19717 scientific publications on diabetes from the Pubmed database, and nodes are divided into three categories.

Photo: The Amazon Photo network from [61]. Nodes represent goods and edges represent that two goods are frequently bought together.

Physics: The Coauthor Physics network from [61]. Nodes represent authors that are connected by an edge if they co-authored a paper.

CS: The Coauthor CS network from [61]. Nodes represent authors that are connected by an edge if they co-authored a paper.

Terrorist: A public dataset collected from the PIT repository [62]. This dataset contains information about terrorists and their relationships. A vector with a value 0/1 describes each relationship.

Airports [63]: The Airports network, where nodes denote airports and labels correspond to activity levels. It includes three graph datasets **Brazil**, **Europe**, and **USA**.

TABLE II
PERFORMANCES (%) ON NODE CLASSIFICATION TASK.

Dataset	Cora	CiteSeer	PubMed	Photo	CS	Physics	Terrorist	Brazil	Europe	USA
$\lvert V\rvert$	2,708	3,327	19,717	7,650	18,333	34,493	881	131	399	1,190
$\lvert E\rvert$	5,429	4,732	44,338	238,162	163,788	495,924	8,592	1,038	5,995	13,599
# classes	7	6	3	8	15	5	2	4	4	4
Homophily	0.8100	0.7355	0.8023	0.8272	0.8081	0.9314	0.9235	0.4683	0.4048	0.6978
ChebNet	86.83±1.66	72.58±1.44	87.47±0.46	92.24±0.49	90.36±0.56	95.72±0.50	78.47±3.02	42.59±8.32	45.43±4.78	47.18±5.41
GCN	87.25±1.41	72.63±1.68	87.98±0.53	94.18±0.39	93.03±0.22	96.50±0.20	78.14±2.60	40.37±7.49	36.42±4.56	51.36±3.24
GAT	86.81±1.36	73.45±1.32	86.67±0.65	94.57±0.76	92.37±0.30	96.44±0.23	78.24±2.88	38.89±11.02	38.02±6.34	52.18±3.44
GIN	66.19±8.10	58.56±2.29	83.62±1.39	78.51±16.64	73.65±3.97	94.75±0.54	74.24±12.57	31.11±9.69	32.22±6.18	38.57±3.11
LightGCN	86.67±1.60	75.27±1.35	83.92±0.55	92.71±0.57	92.99±0.55	95.91±0.13	77.97±2.91	45.44±7.48	48.15±4.35	53.28±3.48
UniMP	87.16±1.36	75.33±1.34	88.67±0.44	90.56±1.01	93.65±0.34	96.11±0.24	78.53±2.96	45.19±6.79	48.40±4.93	52.82±3.67
GPS	82.14±1.47	71.77±1.16	88.59±0.48	93.66±0.61	91.88±0.71	96.38±0.34	79.10±3.66	42.22±8.31	44.20±5.02	45.71±7.14
ASDGN	83.28±0.82	74.19±1.73	88.83±0.55	91.21±1.32	93.07±0.77	96.76±0.18	78.81±2.95	44.44±11.11	41.73±5.66	46.22±3.63
GRN(GRN-Deg_SC)	**89.76±0.45**	**78.77±0.38**	**90.42±0.11**	**95.84±0.26**	**95.97±0.12**	**97.06±0.08**	**82.66±2.10**	**57.41±5.04**	**54.20±2.89**	**56.51±2.15**
Std. reduction	0.37	0.78	0.33	0.13	0.10	0.10	0.50	1.75	1.46	0.96
Reduction Ratio	45.12%	67.24%	75.00%	33.33%	45.45%	55.55%	19.23%	25.77%	33.56%	30.87%
Least Imp.	2.51	3.44	1.59	1.27	2.32	0.30	3.56	11.97	5.80	3.23
Average Imp.	6.70	7.34	3.98	5.12	6.21	0.99	4.74	15.62	12.73	8.09

TABLE III
ABLATION STUDIES.

Dataset	Cora	Citeseer	Pubmed	Photo	CS	Physics	Terrorist	Brazil	Europe	USA
GRN(GRN-Deg_SC)	89.76±0.45	78.77±0.38	90.42±0.11	95.84±0.26	95.97±0.12	97.06±0.08	82.66±2.10	57.41±5.04	54.20±2.89	56.51±2.15
GRN-$Rand\_SC$	88.53±0.59	77.42±0.96	89.31±0.22	95.35±0.35	95.42±0.16	96.83±0.09	80.45±2.70	55.93±8.22	52.96±2.02	52.27±3.45
GRN-Deg_Rule	88.34±0.53	77.78±0.68	89.72±0.45	95.25±0.33	95.51±0.16	96.92±0.05	80.56±1.91	54.07±5.79	52.47±4.21	52.31±5.98
GRN-$Rand\_Rule$	88.14±0.49	77.37±0.58	89.52±0.13	94.87±0.23	95.49±0.27	96.94±0.03	80.11±2.47	51.85±6.21	51.23±2.99	50.25±6.26
GDE	87.42±1.29	75.50±1.20	89.60±0.53	95.31±0.61	95.43±0.38	96.95±0.23	80.62±3.71	45.93±8.95	47.19±5.17	51.72±3.00

The statistics of the graphs are summarized at the top of Table II, including the edge homophily ratio, to reflect the node homogeneity relations in a graph [64].

5.2. Baselines

We consider the following strong baselines:

ChebNet [45]: ChebNet generalizes convolutional operation to graph networks and is simplified based on Chebyshev polynomials.

GCN [9]: A classic graph convolution network that updates the nodes' features by averaging the neighboring nodes' features.

GAT [10]: Based on the self-attention mechanism [65], the model aggregates neighbor features via multi-attention heads. GAT achieved the SOTA performance on many datasets in the node classification task.

LightGCN [12]: LightGCN learns user and item embeddings by linearly propagating features across the user-item interaction graph.

UniMP [13]: A unified message passing model that effectively combines GNNs and label propagation algorithm (LPA) to incorporate feature and label propagation at both training and inference time. It is named as **TransformerConv** in PyG [18].

GPS [49]: A general, powerful, and scalable graph Transformer with linear complexity that supports multiple types of encodings. We used GCN as the backbone in the experiments.

ASDGN [15]: A framework for stable and non-dissipative DGN design, conceived through the lens of ordinary differential equations.

5.3. Implementations

Following the previous framework and ensuring fair comparisons across various graph neural network (GNN) models, we standardized the experimental setup as follows:

The hidden size f_{hid} was set to 128 for all GNNs. The Adam optimizer, with a weight decay of 5e-4 was employed during both training phases. The learning rate was set to 0.003. The datasets were randomly divided into three segments: 60% for training, 20% for validation, and 20% for testing. Both L and $\mathcal{K}$ in GDB (Subsection 4.1) were set to 2 The sample ratio in Algorithm 1 was set to 0.8. The initialization of all parameters in the GRN followed that of Xavier [66]. Since the implementation of models is mostly based on PyG [18], the results may differ from those in the original papers. However, the performance was compared fairly.

5.4. Node Classification

We run each approach on the datasets ten times, reporting the Micro-F1 score as prediction accuracy with standard deviation, the results are shown in Table II. The Airports (Brazil, Europe, and USA) and TerroristRel do not contain data of nodes' features and are replaced by one-hot encoding and one's matrix, respectively. The GRN is specifically

TABLE IV
PERFORMANCES (%) OF GRN VARIANTS.

Dataset	Cora	CiteSeer	PubMed	Photo	CS	Physics	Terrorist	Brazil	Europe	USA
GCN	87.25±1.41	72.63±1.68	87.98±0.53	94.18±0.39	93.03±0.22	96.50±0.20	78.14±2.60	40.37±7.49	36.42±4.56	51.36±3.24
GRN-GCN	88.54±0.40	76.35±0.09	89.16±0.10	93.28±0.14	93.46±0.22	96.52±0.02	79.27±3.16	48.15±6.63	50.12±7.33	52.94±2.46
Improve(Accuracy/std.)	1.29 / 1.01	3.72 / 1.59	1.18 / 0.43	-0.90 / 0.25	0.43 / 0.00	0.02 / 0.18	1.13 / -0.56	7.78 / 0.86	13.70 / -2.77	1.58 / 0.78
GRN-4L	86.34±1.61	74.62±1.15	89.62±0.56	95.05±0.53	95.14±0.30	96.97±0.24	79.89±1.86	52.59±6.58	46.91±5.41	50.71±3.76
GRN-6L	86.94±0.86	73.57±1.39	89.81±0.25	94.93±0.74	94.93±0.27	96.89±0.20	78.93±2.40	43.70±8.57	47.65±6.35	45.38±2.13
GCN-4L	78.69±7.52	68.71±3.42	85.77±0.40	82.79±8.80	79.61±6.80	92.10±5.19	77.74±2.82	41.48±4.91	30.74±4.53	48.03±3.80
GCN-6L	73.90±7.63	64.40±5.46	80.89±9.87	69.97±23.29	72.29±20.36	86.06±9.34	24.92±18.81	29.26±7.67	27.41±5.79	44.16±7.28

TABLE V
COEFFICIENT OF VARIATION ON DIFFERENT DATASETS.

Dataset	ChebNet	GCN	GAT	GIN	LightGCN	UniMP	GPS	ASDGN	GRN
Cora	0.0191	0.0162	0.0157	0.1224	0.0185	0.0156	0.0179	0.0098	**0.0050**
CiteSeer	0.0198	0.0231	0.0180	0.0391	0.0179	0.0178	0.0162	0.0233	**0.0048**
PubMed	0.0053	0.0060	0.0075	0.0166	0.0066	0.0050	0.0054	0.0062	**0.0012**
Physics	0.0053	0.0041	0.0080	0.2120	0.0061	0.0112	0.0065	0.0145	**0.0027**
Photo	0.0062	0.0024	0.0032	0.0539	0.0059	0.0036	0.0077	0.0083	**0.0012**
CS	0.0052	0.0021	0.0024	0.0057	0.0014	0.0025	0.0035	0.0019	**0.0008**
Terrorist	0.0385	0.0333	0.0368	0.1693	0.0373	0.0377	0.0463	0.0374	**0.0254**
Brazil	0.1953	0.1856	0.2833	0.3114	0.1646	0.1502	0.1969	0.2500	**0.0878**
Europe	0.1052	0.1253	0.1668	0.1918	0.0903	0.1019	0.1136	0.1357	**0.0533**
USA	0.1147	0.0631	0.0659	0.0806	0.0653	0.0695	0.1563	0.0785	**0.0380**

denoted as GRN-Deg_SC, which incorporates the degree probability walk-based sampler and the spectral clustering-based unstable nodes detector.

The results clearly show that GRN achieves superior predictive accuracy and the lowest standard deviation compared to other competitive baseline methods. In well-known citation datasets such as Cora, Citeseer, and Pubmed, the GRN has shown performance improvements of at least 1.59%, with a maximum enhancement of up to 7.34%. Notably, in smaller graphs with lower homophily, such as Brazil, Europe, and USA, the GRN exhibited remarkable effectiveness, consistently reducing the volatility that is often associated with smaller datasets. However, the enhancements in accuracy and standard deviation observed with GRN on large-scale datasets are relatively modest, which can be attributed to the inherent stability of these larger datasets against significant fluctuations.

5.5. Ablation Study and Variants Exploration

In addition, we explored GRN variants by using different strategies for the detector and the sampler. The detector variants included the $Rule$ and SC methods. The $Rule$ is a rule-based method to identify unstable nodes within 50 epochs. The SC method uses spectral clustering for the detector. The sampler variants include $Rand$ for random sampling and Deg for degree-based sampling.

As shown in TABLE II and TABLE III, the degree-based sampling variant of the GRN (GRN-Deg_SC) outperforms other methods, achieving either the highest accuracy or the lowest standard deviation across multiple datasets. For instance, on the PubMed dataset, the GRN-Deg_SC and GRN-Deg_Rule variants exhibited robust performances of 90.42% ± 0.11% and 89.72% ± 0.45%, respectively. This suggests that leveraging the inherent centrality degree within graph structures effectively aligns with the model's ability to learn and predict neighborhood structures.

In contrast, the rule-based detection approach ($Rule$) yields lower performance metrics than the spectral clustering-based detector (SC), highlighting the superior ability of spectral clustering to identify unstable nodes accurately.

The ablation studies underscore the importance of the relearn phase in the GRN framework. The variant without this phase, denoted as GDE, shows a noticeable increase in standard deviation compared to GRN, indicating reduced result consistency. For example, on the Brazil dataset, the standard deviation for GDE is 8.95, which is significantly higher than the 5.04 in the GRN-Deg_SC variant, indicating the stabilizing effect of the relearn phase.

The performances of GRN variants are detailed in TABLE IV. When the GRN-GCN variant shows a significant performance improvement compared to the original GCN. The GRN-GCN variant achieves an impressive accuracy improvement of approximately 14% on the Europe dataset, demonstrating the efficacy of our proposed framework in enhancing the predictive power of GNNs.

To verify the effectiveness of the GRN in addressing the over-smooth problem, we tested the GRN model with deeper network architectures, specifically, GRN-4L and GRN-6L denoted for four and six layers of depth, respectively. From TABLE IV, GRN exhibites only minor performance reductions with increased depth. Notably, in the Cora and PubMed datasets, GRN-6L surpassed the GRN-4L, achieving accuracies of 86.94% and 89.81%, respectively. This indicates that the GRN can effectively utilize deeper network structures to extract more complex features. In contrast,

TABLE VI
PROPORTION (%) OF STABLE NODES (NODES WITH UNCHANGED PREDICTIONS).

Dataset	ChebNet	GCN	GAT	GIN	LightGCN	UniMP	GPS	ASDGN	GRN
Cora	94.86	95.10	95.53	51.87	94.29	93.35	91.87	76.89	**96.89**
CiteSeer	91.02	92.42	83.31	56.35	91.40	88.77	92.48	72.78	**94.96**
PubMed	97.66	98.15	92.97	71.21	97.58	96.44	91.40	87.24	**98.50**
Physics	98.30	98.47	95.93	71.93	98.36	97.54	99.02	98.89	**99.27**
Photo	96.01	94.70	94.62	38.27	95.78	90.50	86.21	82.48	**98.50**
CS	97.20	97.03	91.84	59.00	96.81	92.72	82.00	77.00	**98.42**
Terrorist	82.95	97.73	92.61	68.18	97.16	97.73	96.59	92.05	**98.30**
Brazil	92.31	82.69	76.15	76.15	83.46	81.92	92.31	88.46	**96.15**
Europe	96.84	91.52	72.15	74.68	95.44	83.16	93.67	77.22	**98.86**
USA	92.52	90.80	67.56	57.61	93.61	84.24	88.36	77.31	**97.35**

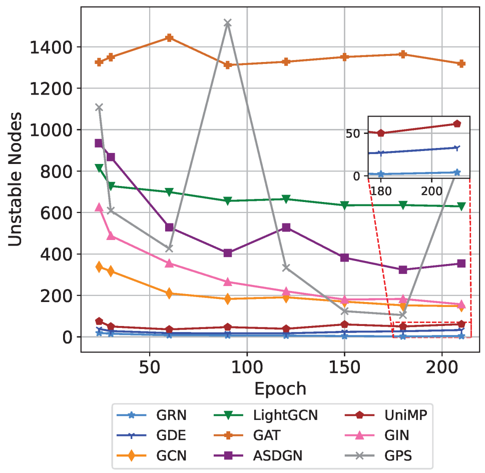
increasing the number of layers in traditional GCNs often leads to diminished performance or even a significant decline. This is typically owing to issues such as overfitting or vanishing gradients, which hinder the model's ability to generalize effectively. The ability of GRN to either maintain or enhance performance with additional layers highlights its robustness and stability, making it well-suited for managing the complexities associated with deeper GNN architectures.

5.6. Quantification

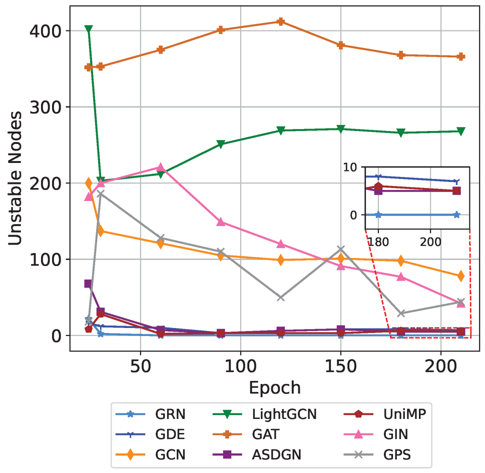
To assess the performance of the GRN model in terms of representation stability compared to other strong baselines, we employ the Coefficient of Variation (CV) as the metric for stability assessment. This metric, defined by the equation ($CV = \frac{\sigma}{\mu}$), where σ is the standard deviation and μ is the mean, effectively neutralizes the impact of scale in accuracy measurements (with a lower CV value indicating enhanced model stability). The CV values for all models across various datasets are presented in Table V, where GRN achieves the best performance.

To assess the effectiveness of the Graph Relearn Network (GRN) in addressing the phenomenon of node prediction class oscillations, we also quantify the instances of node skipping. Table VI presents the average proportions of stable nodes over the last 50 epochs under identical experimental conditions. Stable nodes are defined as those with unchanged predictions across epochs. The GRN consistently yields more consistent and stable predictions across all datasets than the baseline models.

We conducted experiments with the GRN on the CiteSeer and USA datasets, alternating between the pre-predict phase (15 epochs per iteration) and the relearn phase (15 epochs per iteration) for seven cycles, totaling 210 epochs. We also trained all baseline models and monitored the counts of stable nodes throughout the process. As illustrated in Fig. 5, during the initial epochs, the GRN, along with GDE and UniMP, exhibited significantly fewer unstable nodes compared to the other models. A consistent decrease in the number of unstable nodes was observed, which eventually stabilized as the training progressed. The GPS model exhibited notable oscillations in node stability, particularly in the Citeseer dataset. Although UniMP and GDE also performed well, maintaining a low proportion of unstable nodes, the GRN consistently exhibited the lowest proportion of unstable nodes and achieved the smallest variance in prediction accuracy among the evaluated models.



(a) Citeseer



(b) USA

Fig. 5: Number of unstable nodes of GRN and other baselines during training.

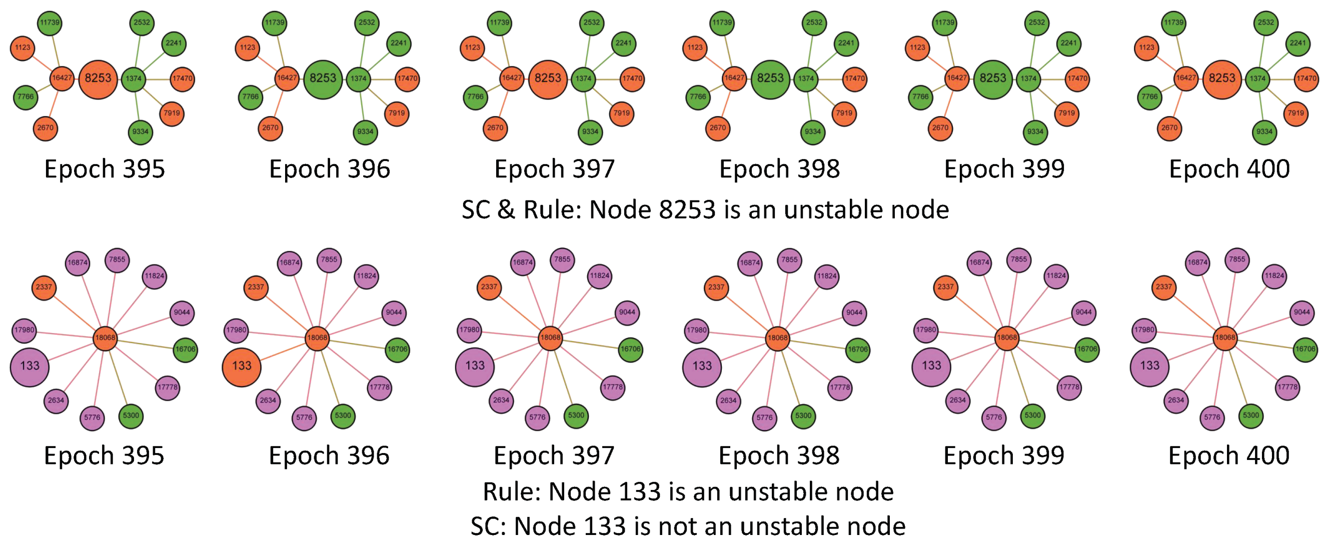

Epoch 395
Epoch 396
Epoch 397
Epoch 398
Epoch 399
Epoch 400
SC & Rule: Node 8253 is an unstable node
Epoch 395
Epoch 396
Epoch 397
Epoch 398
Epoch 399
Epoch 400
Rule: Node 133 is an unstable node
SC: Node 133 is not an unstable node

Fig. 6: Comparison of different unstable node detectors. SC and Rule indicate the unstable node detectors based on the spectral clustering and rule methods, respectively (Section 5.5). Nodes with the same color are in the same class.

In Appendix A, we delve into the characteristics of unstable nodes and demonstrate that the majority are either peripheral nodes or those overlapping between communities, which provides further insight into the structural challenges addressed by the GRN in enhancing prediction stability.

5.7. Cases by Different Detectors

In this section, we investigate different methods for detecting unstable nodes within the GRN, specifically focusing on rule-based and spectral clustering approaches as detailed in Section 5.5. Figure 6 presents a visual representation of the predictions made by both detectors. Node 8253, which was characterized by its continuous fluctuations in the predicted category, is deemed unstable by both the rule-based and spectral clustering methods. In contrast, node 133, which exhibited a category change only in the early training phase (epoch 396) before stabilization, is flagged as unstable by the rule-based method but not by the spectral clustering detector. This indicates that the spectral clustering approach is more attuned to the prediction stability, making it a preferable option for identifying truly unstable nodes in the GRN.

5.8. Visualization

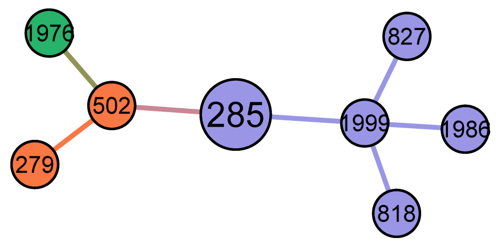
We draw unstable nodes on the Cora dataset to visualize the effects of the GRN on the node-skipping phenomenon. As shown in Fig. 7, node 2186 is situated at the intersection of two major clusters or communities. GRN focuses more on the distribution of neighboring nodes and consistently predicts the correct blue class in subsequent epochs. The neighbors of node 2276 have a relatively balanced distribution of categories and are located at a junction between clusters or communities. In addition, the neighbors of nodes 285 and 2274 are illustrated; both are located at the pivotal connecting points among node clusters. During the relearn phase, the GRN effectively corrects the predicted category of this node.

Additionally, we assessed the representational capabilities of various methods by visualizing the node representations from GRN, GCN, GAT, and LightGCN, all trained on the Cora dataset. We employed T-SNE [67] to reduce the dimensions of the node representations from the final layer to two dimensions. The visualization results, presented in Fig. 8, show that node representations learned by the GRN are clearly distributed around the center, with seven well-delineated, color-coded categories indicating distinct inter-cluster separations. Although GCN and LightGCN also perform well, their intra-cluster node aggregation appears comparatively less defined than that of GRN. The performance of GAT is moderately satisfactory, with some classes of nodes appearing more dispersed. Overall, these results highlight GRN's superior ability to capture and learn distinct node characteristics fundamentally.

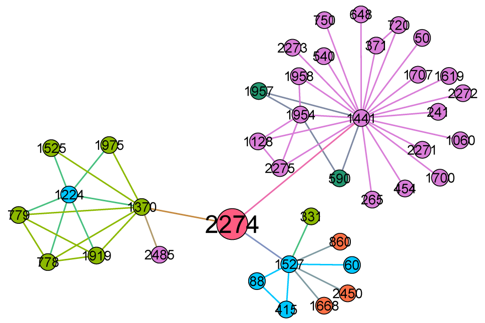
5.9. Model Convergence

We evaluate the convergence characteristics of the GRN on CiteSeer and USA datasets. The procedure of the GRN includes phases: the pre-prediction phase (20 epochs), and the subsequent relearn phase (10 epochs), which repeats seven times (for a total of 210 epochs). The training processes for GDE and GCN are also recorded. The curves depicting the test accuracy, training, validation, and test loss are shown in Fig. 9.

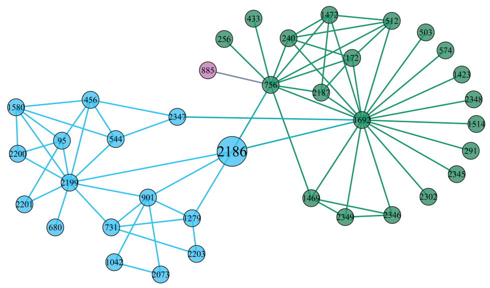
The experimental results reveal that both GRN and GDE initially exhibit a slight increase in validation and test loss curves during the early stages of training, which then gradually decline as they approach convergence. Correspondingly, the accuracy curves exhibit a steady increase before stabilizing at a plateau. Notably, the absence of the relearn phase in the GDE framework leads to poorer accuracy compared to GRN. In contrast, GCN shows signs of overfitting on both datasets, particularly on the CiteSeer dataset. This



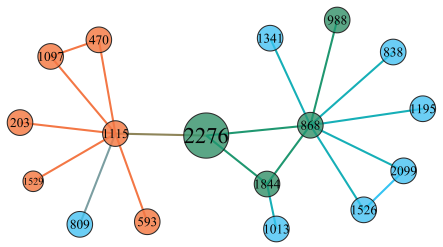
(a) Central Node 285, Epoch 236, 238, 240



(b) Central Node 2274, Epoch 295, 297, 300

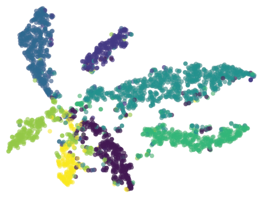


(c) Central Node 2186, Epoch 202, 205, 208

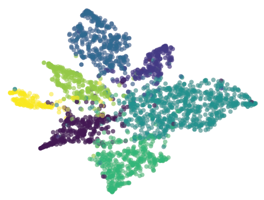


(d) Central Node 2276, Epoch 167, 172, 223

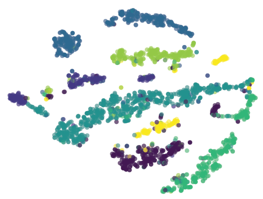
Fig. 7: Cases of GRN training on the Cora dataset. Nodes with the same color indicate the same class.



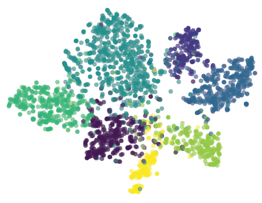
(a) GRN



(b) GCN

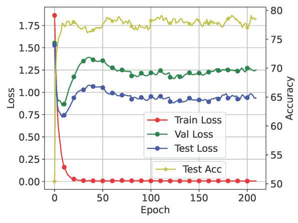


(c) GAT

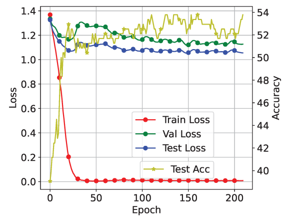


(d) LightGCN

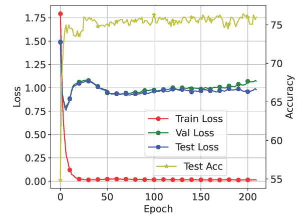
Fig. 8: Visualization of GRN and baselines on the Cora dataset by TSNE[67].



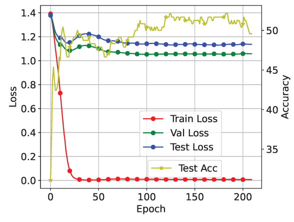
(a) CiteSeer-GRN



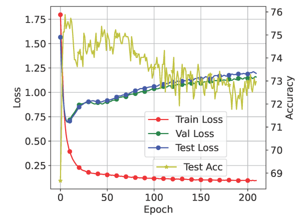
(b) USA-GRN



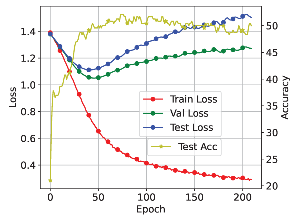
(c) CiteSeer-GDE



(d) USA-GDE



(e) CiteSeer-GCN



(f) USA-GCN

Fig. 9: Loss and accuracy curves of GRN, GDE, and GCN on the CiteSeer and USA.

suggests that the GDE architecture offers certain benefits in mitigating overfitting.

Further details on quantifying the over-smoothing effect of the different methods using the Dirichlet energy are provided in Appendix B [68].

6. Conclusion

GNNs based on message passing often exhibit instability because the predicted node classes oscillate between categories. This issue, which is the focus of this study, contributes significantly to the performance instability in terms of prediction accuracy. To address this issue, we introduce the Graph Relearn Network (GRN), which is a novel framework designed to correct and stabilize predictions by relearning unstable nodes during training. The GRN achieves the lowest standard deviation and highest prediction accuracy on ten benchmark graph datasets.

However, the GRN framework exhibits certain limitations, notably that the enhancement in representational performance for the underlying GNN backbone is not markedly pronounced, and there is an increase in computational overhead by approximately 2 to 4 times compared to traditional GNNs. Despite these drawbacks, the framework remains highly versatile and can be adapted to any graph neural network model that addresses node classification tasks, thereby ensuring the representational stability of GNNs. This adaptability extends to a variety of applications, including graph structure anomaly detection, rumor detection, molecular classification, and drug generation, among others. Our experiments demonstrate that the GRN framework significantly mitigates instability and thereby affects the prediction accuracy of GNNs.

CRediT authorship contribution statement

Zhenhua Huang: Conceptualization, Methodology, Software. **Kunhao Li:** Data, Writing, Implementation. **Yihang Jiang:** Implementation, Revising. **Zhaohong Jia:** Advising, Conceptualization. **Linyuan Lv:** Revising, Supervision. **Yunjie Ma:** Advising, Revising.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that may influence the work.

Data availability

Data will be made available on request.

Acknowledgement

This work was supported by the National Natural Science Foundation of China (Number 71971002). We also thank the support of PyG [18] for our experiments.

Appendix

A. Statistics of Unstable Nodes

We selected a subset of representative unstable nodes generated by GCN on the Airports (USA) dataset and visualized their two-hop neighborhoods in Fig. 10. Due to their fluctuating predicted categories, these unstable nodes are colored white, while other nodes are colored consistently to denote category membership. These unstable nodes frequently occur at the peripheries of clusters, and at the junctions between different clusters or communities, indicating that their categorization is highly susceptible to the influence of neighboring nodes from various categories.

To further understand the distribution of unstable nodes in traditional graph neural networks, we conducted a statistical analysis of their locations. We utilized a dataset processed by GCN (noting that GRN has effectively eliminated many unstable nodes, making it unsuitable for this analysis) and employed a spectral clustering (SC) detector to identify the unstable nodes. We used KM_config [69] to determine the proportion of nodes at the peripheries of clusters, and Demon [70] to identify nodes at the junctures between communities. As shown in TABLE VII, the statistical results indicate that the unstable nodes generated by traditional graph neural networks predominantly occur at the periphery of communities and the junctions between communities. This positioning exposes periphery nodes to frequent updates during neighbor aggregations within the graph convolutional layers, making them particularly vulnerable to influences from connected nodes, as illustrated in Fig. 10 (a)(b)(c)(d). The merging of information from different communities in the representations of nodes situated between communities leads to oscillations in predicted classes, as seen in Fig. 10 (i). Unstable nodes can also be found in more complex positions, such as those shown in Fig. 10 (g)(f)(h). This detailed analysis underscores the challenges in managing node stability in graph neural networks and highlights the areas where improvements are necessary.

B. Over Smoothing Relief

The over-smoothing is a prevalent issue in graph neural networks, where node features become increasingly indistinguishable after several layers of convolution. Dense connections in GRN are applied to alleviate it. We apply Dirichlet Energy [68] on graph data to evaluate the advantage of the GRN on the over-smoothing relief. Higher Dirichlet Energy indicates better performance in the perspective of the over-smoothing problem. The Dirichlet Energy normalized by the node degrees is:

$$\varepsilon(X^n) = \frac{1}{N} \sum_{i \in V} \sum_{j \in N_i} \left\| \frac{X_i^n}{\sqrt{1+d_i}} - \frac{X_j^n}{\sqrt{1+d_j}} \right\|_2^2 \tag{12}$$

where X^n is the feature of the n^{th} layer, $\mathcal{N}_i$ is the neighbors set of node i, d_i and d_j are the degree of node i and node j. We set $\mathcal{N}_i$ as a two-hop subgraph. The Dirichlet Energy of baselines and GRN is reported in Table VIII, and the best and second-best performances are highlighted by the underline.

After undergoing two-layer convolutions, GRN outperforms all baseline models on most datasets, particularly in the Cora, Citeseer, and USA with the highest Dirichlet Energies. This performance suggests that GRN is capable of producing a greater degree of feature distinction after convolution, effectively mitigating the over-smoothing issue.

Although GRN does not always secure the absolute highest scores on datasets like Pubmed and Europe, it consistently ranks among the top performers, underscoring its robustness across various types of graph data. While models such as GPS and ASDGN occasionally match or exceed GRN's performance on specific datasets, they lack the overall consistent efficacy demonstrated by GRN.

TABLE VII
DISTRIBUTION PROPORTION (%) OF UNSTABLE NODES IN GRAPHS BASED ON GCN.

Dataset	Cora	Citeseer	Pubmed	Photo	CS	Physics	Terrorist	Brazil	Europe	USA
Edge nodes	51.05	49.40	69.71	30.83	38.15	28.12	9.67	28.98	24.55	47.44
Overlapping nodes	18.12	6.42	11.46	65.14	54.62	68.81	54.83	55.07	72.45	48.57

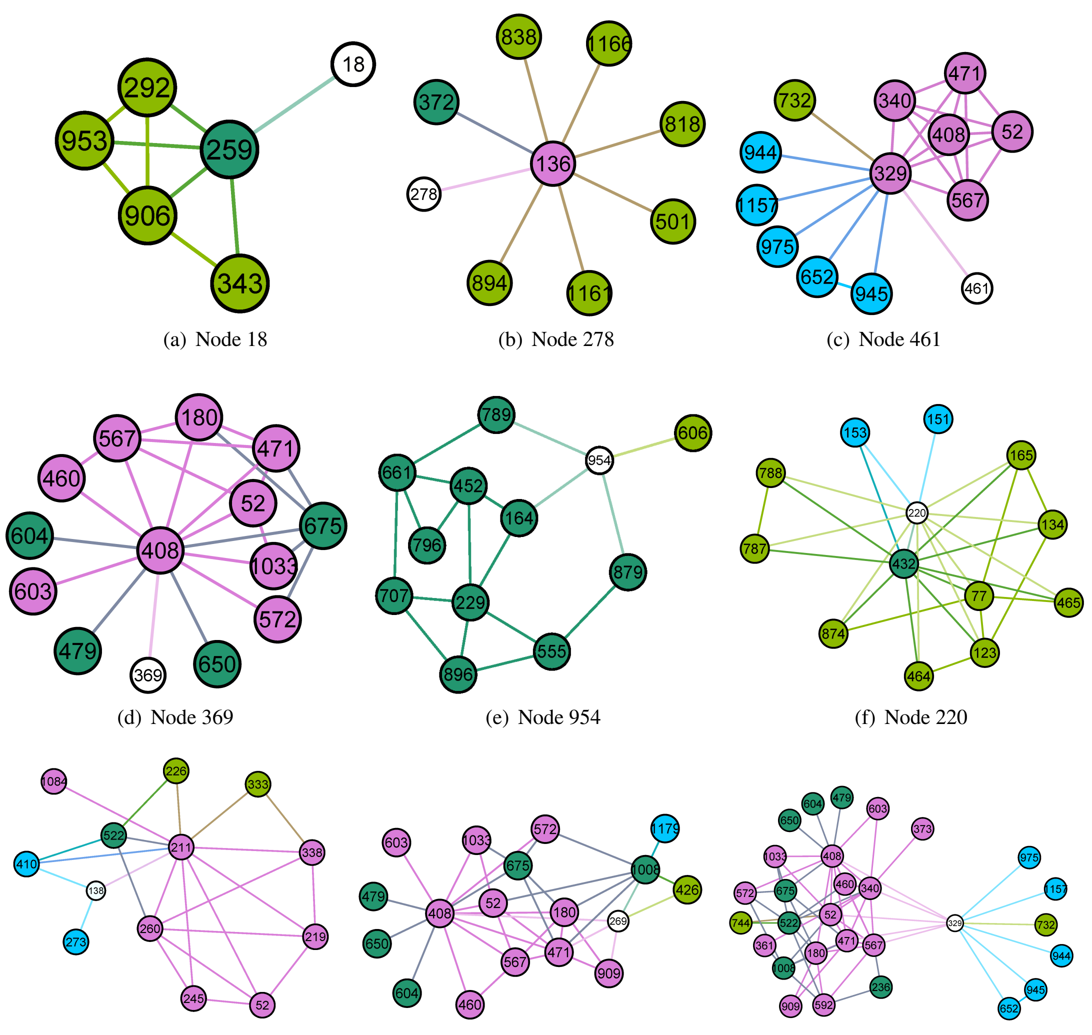

(a) Node 18 (b) Node 278 (c) Node 461
(d) Node 369 (e) Node 954 (f) Node 220
(g) Node 138 (h) Node 269 (i) Node 329

Fig. 10: Cases of the detected unstable nodes on the USA dataset based on GCN. Nodes with the same color indicate the same class. White nodes indicate the unstable nodes.

TABLE VIII
DIRICHLET ENERGY OF DIFFERENT METHODS.

Dataset	Cora	Citeseer	Pubmed	Physics	Photo	CS	Terrorist	Brazil	Europe	USA
Chebnet	18.84	15.37	3.14	21.25	36.31	46.70	7.43	37.70	**213.79**	113.69
GCN	8.48	4.50	0.97	5.37	12.76	17.66	2.81	9.50	44.44	19.19
GAT	14.69	5.45	2.48	22.60	43.76	53.45	0.46	9.56	7.60	5.57
GIN	4.99	3.68	3.56	2.27	4.91	9.32	8.13	12.09	4.58	10.06
LightGCN	10.56	9.79	1.33	6.07	13.94	18.70	0.42	3.49	25.19	9.32
UniMP	30.03	23.18	8.09	28.21	73.14	59.73	0.38	34.81	45.48	31.23
GPS	21.00	20.22	**18.15**	12.38	36.18	50.94	0.07	47.05	79.89	56.98
ASDGN	15.92	13.44	3.86	22.83	**88.25**	62.21	0.22	21.15	34.28	21.59
GRN	**41.33**	**35.45**	13.76	**28.59**	49.90	**71.34**	**18.31**	**53.80**	134.68	**137.21**

References

[1] Y. Zhou, H. Zheng, X. Huang, S. Hao, D. Li, J. Zhao, Graph neural networks: Taxonomy, advances, and trends, ACM Transactions on Intelligent Systems and Technology (TIST) 13 (2022) 1–54.

[2] Y. Zhao, H. Zhou, R. Xie, F. Zhuang, Q. Li, J. Liu, Incorporating global information in local attention for knowledge representation learning, in: Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, 2021, pp. 1341–1351.

[3] L. Yao, C. Mao, Y. Luo, Graph convolutional networks for text classification, in: Proceedings of the AAAI conference on artificial intelligence, volume 33, 2019, pp. 7370–7377.

[4] D. Wang, P. Liu, Y. Zheng, X. Qiu, X.-J. Huang, Heterogeneous graph neural networks for extractive document summarization, in: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, 2020, pp. 6209–6219.

[5] M. Li, Z. Zhu, Spatial-temporal fusion graph neural networks for traffic flow forecasting, in: Proceedings of the AAAI conference on artificial intelligence, volume 35, 2021, pp. 4189–4196.

[6] C. Xu, Y. Zhang, H. Chen, L. Dong, W. Wang, A fairness-aware graph contrastive learning recommender framework for social tagging systems, Information Sciences (2023) 119064.

[7] J. Huang, R. Xie, Q. Cao, H. Shen, S. Zhang, F. Xia, X. Cheng, Negative can be positive: Signed graph neural networks for recommendation, Information Processing & Management 60 (2023) 103403.

[8] X. Li, C. Xiao, Z. Feng, S. Pang, W. Tai, F. Zhou, Controlled graph neural networks with denoising diffusion for anomaly detection, Expert Systems with Applications 237 (2024) 121533.

[9] T. N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks (2017).

[10] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, Y. Bengio, Graph attention networks (2018).

[11] K. Xu, W. Hu, J. Leskovec, S. Jegelka, How powerful are graph neural networks?, in: Proceedings of International Conference on Learning Represent, 2018.

[12] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, M. Wang, Lightgcn: Simplifying and powering graph convolution network for recommendation, in: Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, 2020, pp. 639–648.

[13] Y. Shi, Z. Huang, S. Feng, H. Zhong, W. Wang, Y. Sun, Masked label prediction: Unified message passing model for semi-supervised classification, Proceedings of International Joint Conference on Artificial Intelligence (2021).

[14] H. Zhang, Z. Yu, G. Dai, G. Huang, Y. Ding, Y. Xie, Y. Wang, Understanding gnn computational graph: A coordinated computation, io, and memory perspective, in: Proceedings of Machine Learning and Systems, volume 4, 2022, pp. 467–484.

[15] A. Gravina, D. Bacciu, C. Gallicchio, Anti-symmetric DGN: a stable architecture for deep graph networks, in: The Eleventh International Conference on Learning Representations, 2023.

[16] K. Li, Z. Huang, Z. Jia, Rahg: A role-aware hypergraph neural network for node classification in graphs, IEEE Transactions on Network Science and Engineering (2023).

[17] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, G. E. Dahl, Neural message passing for quantum chemistry, in: International conference on machine learning, PMLR, 2017, pp. 1263–1272.

[18] M. Fey, J. E. Lenssen, Fast graph representation learning with PyTorch Geometric, in: Proceedings of ICLR Workshop on Representation Learning on Graphs and Manifolds, 2019.

[19] W. Zhang, Y. Shen, Z. Lin, Y. Li, X. Li, W. Ouyang, Y. Tao, Z. Yang, B. Cui, Pasca: A graph neural architecture search system under the scalable paradigm, in: Proceedings of the ACM Web Conference 2022, 2022, pp. 1817–1828.

[20] R. Ying, D. Bourgeois, J. You, M. Zitnik, J. Leskovec, Gnn explainer: A tool for post-hoc explanation of graph neural networks (2019).

[21] D. Luo, W. Cheng, D. Xu, W. Yu, B. Zong, H. Chen, X. Zhang, Parameterized explainer for graph neural network, Advances in neural information processing systems 33 (2020) 19620–19631.

[22] X. Wang, Y. Wu, A. Zhang, F. Feng, X. He, T.-S. Chua, Reinforced causal explainer for graph neural networks, IEEE Transactions on Pattern Analysis and Machine Intelligence (2022).

[23] Z. Zhang, Q. Liu, H. Wang, C. Lu, C. Lee, Protgnn: Towards self-explaining graph neural networks, in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 36, 2022, pp. 9127–9135.

[24] H. Zhenhua, L. Kunhao, W. Shaojie, J. Zhaohong, Z. Wentao, M. Sharad, Ses: Bridging the gap between explainability and prediction of graph neural networks, in: 40th International Conference on Data Engineering, 2024.

[25] K. Li, Y. Liu, X. Ao, J. Chi, J. Feng, H. Yang, Q. He, Reliable representations make a stronger defender: Unsupervised structure refinement for robust gnn, in: Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2022, pp. 925–935.

[26] R. Arghal, E. Lei, S. S. Bidokhti, Robust graph neural networks via probabilistic lipschitz constraints, in: Learning for Dynamics and Control Conference, PMLR, 2022, pp. 1073–1085.

[27] Y. Song, Q. Kang, S. Wang, K. Zhao, W. P. Tay, On the robustness of graph neural diffusion to topology perturbations, Advances in Neural Information Processing Systems 35 (2022) 6384–6396.

[28] L. Ruiz, Z. Wang, A. Ribeiro, Graphon and graph neural network stability, in: ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2021, pp. 5255–5259.

[29] F. Gama, J. Bruna, A. Ribeiro, Stability properties of graph neural networks, IEEE Transactions on Signal Processing 68 (2020) 5680–5695.

[30] H. Kenlay, D. Thano, X. Dong, On the stability of graph convolutional neural networks under edge rewiring, in: ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2021, pp. 8513–8517.

[31] J. Qiu, Q. Chen, Y. Dong, J. Zhang, H. Yang, M. Ding, K. Wang, J. Tang, Gcc: Graph contrastive coding for graph neural network pre-training, in: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2020, pp. 1150–1160.

[32] M. I. Jordan, T. M. Mitchell, Machine learning: Trends, perspectives, and prospects, Science 349 (2015) 255–260.

[33] O. A. Alimi, K. Ouahada, A. M. Abu-Mahfouz, A review of machine learning approaches to power system security and stability, IEEE Access 8 (2020) 113512–113531.

[34] L. Ruiz, F. Gama, A. Ribeiro, Graph neural networks: architectures, stability, and transferability, Proceedings of the IEEE 109 (2021) 660–682.

[35] H. Song, Y. Wang, W. Zhang, X. Liu, T. Liu, Generate, delete and rewrite: A three-stage framework for improving persona consistency of dialogue generation, in: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, 2020, pp. 5821–5831.

[36] M. Ren, W. Zeng, B. Yang, R. Urtasun, Learning to reweight examples for robust deep learning, in: International conference on machine learning, PMLR, 2018, pp. 4334–4343.

[37] G. Huang, Z. Liu, L. Van Der Maaten, K. Q. Weinberger, Densely connected convolutional networks, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 4700–4708.

[38] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, G. Monfardini, The graph neural network model, IEEE transactions on neural networks 20 (2008) 61–80.

[39] S. Abadal, A. Jain, R. Guirado, J. López-Alonso, E. Alarcón, Computing graph neural networks: A survey from algorithms to accelerators, ACM Computing Surveys (CSUR) 54 (2021) 1–38.

[40] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, S. Y. Philip, A comprehensive survey on graph neural networks, IEEE transactions on neural networks and learning systems 32 (2020) 4–24.

[41] J. You, B. Liu, Z. Ying, V. Pande, J. Leskovec, Graph convolutional policy network for goal-directed molecular graph generation, in: Proceedings of Advances in neural information processing systems, volume 31, 2018.

[42] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, J. Leskovec, Graph convolutional neural networks for web-scale recommender systems, in: Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining, 2018, pp. 974–983.

[43] X. Chen, F. Zhou, K. Zhang, G. Trajcevski, T. Zhong, F. Zhang, Information diffusion prediction via recurrent cascades convolution, in: Proceedings of the IEEE 35th international conference on data engineering (ICDE), IEEE, 2019, pp. 770–781.

[44] Z. Chen, S. Li, B. Yang, Q. Li, H. Liu, Multi-scale spatial temporal graph convolutional network for skeleton-based action recognition, in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 35, 2021, pp. 1113–1122.

[45] M. Defferrard, X. Bresson, P. Vandergheynst, Convolutional neural networks on graphs with fast localized spectral filtering, in: Proceedings of the Advances in Neural Information Processing Systems, 2016, pp. 3844–3852.

[46] W. Hamilton, Z. Ying, J. Leskovec, Inductive representation learning on large graphs, Proceedings of Advances in neural information processing systems 30 (2017).

[47] F. M. Bianchi, D. Grattarola, L. Livi, C. Alippi, Graph neural networks with convolutional arma filters, IEEE Transactions on Pattern Analysis and Machine Intelligence (2021).

[48] W. Hamilton, Z. Ying, J. Leskovec, Inductive representation learning on large graphs, Advances in neural information processing systems 30 (2017).

[49] L. Rampášek, M. Galkin, V. P. Dwivedi, A. T. Luu, G. Wolf, D. Beaini, Recipe for a general, powerful, scalable graph transformer, Advances in Neural Information Processing Systems 35 (2022) 14501–14515.

[50] S. Zheng, Y. Song, T. Leung, I. Goodfellow, Improving the robustness of deep neural networks via stability training, in: Proceedings of the ieee conference on computer vision and pattern recognition, 2016, pp. 4480–4488.

[51] Z. Wang, L. Ruiz, A. Ribeiro, Stability of neural networks on manifolds to relative perturbations, in: ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, 2022, pp. 5473–5477.

[52] K. Zhao, Q. Kang, Y. Song, R. She, S. Wang, W. P. Tay, Adversarial robustness in graph neural networks: A hamiltonian approach, in: A. Oh, T. Neumann, A. Globerson, K. Saenko, M. Hardt, S. Levine (Eds.), Advances in Neural Information Processing Systems, volume 36, Curran Associates, Inc., 2023, pp. 3338–3361.

[53] Z. Dong, M. Zhang, P. Payne, M. A. Province, C. Cruchaga, T. Zhao, F. Li, Y. Chen, Rethinking the power of graph canonization in graph representation learning with stability, in: The Twelfth International Conference on Learning Representations, 2024.

[54] A. Elgohary, D. Peskov, J. Boyd-Graber, Can you unpack that? learning to rewrite questions-in-context (2019) 5918–5924.

[55] P. Ponnusamy, A. Ghias, Y. Yi, B. Yao, C. Guo, R. Sarikaya, Feedback-based self-learning in large-scale conversational ai agents, AI magazine 42 (2022) 43–56.

[56] S. Qiu, A. Potapczynski, P. Izmailov, A. G. Wilson, Simple and fast group robustness by automatic feature reweighting, in: A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, J. Scarlett (Eds.), Proceedings of the 40th International Conference on Machine Learning, volume 202 of *Proceedings of Machine Learning Research*, PMLR, 2023, pp. 28448–28467.

[57] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

[58] M. Ahmed, R. Seraj, S. M. S. Islam, The k-means algorithm: A comprehensive survey and performance evaluation, Electronics 9 (2020) 1295.

[59] R. Mondragón, Estimating degree–degree correlation and network cores from the connectivity of high–degree nodes in complex networks, Scientific reports 10 (2020) 5668.

[60] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, T. Eliassi-Rad, Collective classification in network data, AI magazine 29 (2008) 93–93.

[61] O. Shchur, M. Mumme, A. Bojchevski, S. Günnemann, Pitfalls of graph neural network evaluation, in: Relational Representation Learning Workshop, NeurIPS 2018, 2018.

[62] B. Zhao, P. Sen, L. Getoor, Event classification and relationship labeling in affiliation networks, in: Proceedings of the workshop on statistical network analysis (SNA) at the 23rd international conference on machine learning (ICML), 2006, pp. 271–280.

[63] L. F. Ribeiro, P. H. Saverese, D. R. Figueiredo, struc2vec: Learning node representations from structural identity, in: Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining, 2017, pp. 385–394.

[64] J. Zhu, Y. Yan, L. Zhao, M. Heimann, L. Akoglu, D. Koutra, Beyond homophily in graph neural networks: Current limitations and effective designs, Advances in neural information processing systems 33 (2020) 7793–7804.

[65] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, I. Polosukhin, Attention is all you need, in: Proceedings of the 31st Advances in Neural Information Processing Systems, 2017, pp. 5998–6008.

[66] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in: AISTATS, JMLR Workshop and Conference Proceedings, 2010, pp. 249–256.

[67] L. Van der Maaten, G. Hinton, Visualizing data using t-sne., Journal of machine learning research 9 (2008).

[68] T. K. Rusch, M. Bronstein, S. Mishra, A survey on oversmoothing in graph neural networks, SAM Research Report 2023 (2023).

[69] S. Kojaku, N. Masuda, Core-periphery structure requires something else in the network, New Journal of physics 20 (2018) 043012.

[70] M. Coscia, G. Rossetti, F. Giannotti, D. Pedreschi, Demon: a local-first discovery method for overlapping communities, in: Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining, 2012, pp. 615–623.