Graph Structure Refinement Neural Networks for Robust Node Classification in API-based Systems

Xuhua Ai[1], Yuan Yin[1], Qi Meng[1], Zijian Lin[1], Zonghui Wei[2], Yiting Huang[1*]
[1]Digital Operation Center, Guangxi Power Grid Co., Ltd., Guangxi, China
[2]Digitalization Department, Guangxi Power Grid Co., Ltd., Guangxi, China
*Corresponding author: Yiting Huang (e-mail: 1014258769@qq.com).

ABSTRACT Graph Neural Networks (GNNs) are widely used for node classification on graph-structured data, including graphs induced from API-based power systems. However, the effectiveness of existing GNNs strongly depends on the quality of the underlying graph structure, which is often noisy or suboptimal in practice. In this paper, we propose Graph Structure Refinement Neural Network (GSR-GNN), a novel framework that learns task-aware edge importance weights to refine graph structures. GSR-GNN employs a differentiable structure refinement module and a refined message passing scheme to enable more robust information aggregation, together with a structure-preserving loss to maintain essential topological properties. Experimental results on benchmark datasets demonstrate that GSR-GNN consistently outperforms state-of-the-art methods.

INDEX TERMS Graph Neural Networks, Structure Refinement, Edge Weighting, Robust Learning, Node Classification, Graph Structure Learning

I. INTRODUCTION

GRAPH Neural Networks (GNNs) have emerged as a powerful framework for learning representations on graph-structured data, achieving state-of-the-art performance in applications such as social network analysis [1], recommendation systems [2], and bioinformatics [3]. Most GNN architectures rely on message passing mechanisms [4], where node representations are iteratively updated by aggregating information from neighbors. Despite their success, existing GNNs implicitly assume that the given graph structure is reliable and well-suited for downstream tasks—an assumption that rarely holds in real-world scenarios.

In practice, graph structures often suffer from various imperfections that can significantly degrade GNN performance. First, graphs may contain noisy edges that connect semantically dissimilar nodes due to data collection errors or automated linking processes, leading to misleading information aggregation. Second, important connections between related nodes may be missing, preventing effective information propagation. Third, even when edges are meaningful, their contributions are typically treated as uniform, ignoring differences in relevance across connections. These issues make standard message passing highly sensitive to graph quality, amplifying noise and resulting in corrupted node representations.

This structural sensitivity has motivated extensive research on robust GNNs and graph structure learning. Early spectral and spatial GNN models, including GCN [5], GraphSAGE [6], and GAT [7], demonstrated strong performance but remain vulnerable to structural imperfections and over-smoothing [8]. To address these limitations, structure learning methods such as LDS [9], GRCN [10], and Pro-GNN [11] aim to refine graph topology, while robust GNNs mitigate noise through adversarial training or graph purification. Attention-based models further introduce adaptive edge weighting [7], yet many approaches either modify topology discretely or compute dynamic weights that are tightly coupled to specific architectures.

In this paper, we propose Graph Structure Refinement Neural Network (GSR-GNN), a novel framework that refines graph structures by learning continuous, task-aware edge importance weights. Instead of adding or removing edges, GSR-GNN preserves the original topology while enabling fine-grained control over information flow through a differentiable structure refinement module. By jointly optimizing edge weights and GNN parameters, GSR-GNN improves robustness to structural imperfections while maintaining essential topological properties, leading to more reliable node representations.

II. METHODOLOGY

In this section, we present the complete methodology of our Graph Structure Refinement Neural Network (GSR-GNN). We begin with the problem definition, then detail the structure refinement module, refined message passing scheme, and overall training objective.

A. STRUCTURE REFINEMENT MODULE

The core innovation of GSR-GNN is the structure refinement module, which learns to assign importance weights to edges based on their likelihood of being meaningful connections. This module serves as a learnable preprocessing step that can be integrated with any GNN architecture.

Edge Weight Prediction: For each edge $(v_i, v_j) \in \mathcal{E}$ in the original graph, we compute an importance weight $w_{ij} \in [0, 1]$ that reflects the significance of this connection. The weight is predicted using a neural network that takes the node features as input:

$$w_{ij} = \sigma(\text{MLP}_\theta(\phi(\mathbf{x}_i, \mathbf{x}_j))) \tag{1}$$

where MLP_θ is a multi-layer perceptron with parameters θ, $\phi(\cdot, \cdot)$ is a feature combination function, and σ is the sigmoid activation function that ensures the output weights are in $[0, 1]$.

Feature Combination Strategies: We consider two strategies for combining node features to compute edge weights:

B. CONCATENATION-BASED COMBINATION

The concatenation-based approach directly combines the features of connected nodes by concatenating their feature vectors:

$$\phi_{\text{concat}}(\mathbf{x}_i, \mathbf{x}_j) = [\mathbf{x}_i \parallel \mathbf{x}_j] \tag{2}$$

where $\parallel$ denotes vector concatenation. This approach provides the MLP with complete information about both nodes, allowing it to learn arbitrary functions of the node features. However, it doubles the input dimension, which may increase the number of parameters.

C. BILINEAR-BASED COMBINATION

The bilinear-based approach captures multiplicative interactions between node features through element-wise multiplication:

$$\phi_{\text{bilinear}}(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{W} \cdot (\mathbf{x}_i \odot \mathbf{x}_j) + \mathbf{b} \tag{3}$$

where $\odot$ denotes element-wise multiplication, $\mathbf{W} \in \mathbb{R}^{d \times F}$ is a learnable weight matrix, $\mathbf{b} \in \mathbb{R}^d$ is a bias vector, and d is the output dimension. This approach captures feature interactions more efficiently than concatenation and can be more parameter-efficient when $d < 2F$.

Refined Adjacency Matrix: After computing edge weights for all edges in the original graph, we construct the refined adjacency matrix $\widetilde{\mathbf{A}} \in \mathbb{R}^{N \times N}$:

$$\widetilde{\mathbf{A}}_{ij} = \begin{cases} w_{ij}, & \text{if } A_{ij} = 1, \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

Note that we only refine the weights of existing edges and do not add new edges to the graph. This design choice preserves the sparsity of the original graph and reduces the computational overhead.

D. REFINED MESSAGE PASSING

Using the learned edge weights, we perform refined message passing where the importance of each connection determines the amount of information propagated across it.

(1) **Weighted Message Passing:** The message from node v_j to node v_i is scaled by the learned edge weight w_{ij}:

$$\mathbf{m}_{j \to i} = w_{ij} \cdot \mathbf{h}_j \tag{5}$$

where $\mathbf{h}_j \in \mathbb{R}^d$ is the hidden representation of node v_j from the previous layer. This weighting scheme ensures that messages from more important edges contribute more to the aggregation;

(2) **Neighborhood Aggregation:** The aggregated message at node v_i is computed by summing the weighted messages from all neighbors:

$$\begin{aligned} \mathbf{m}_i &= \sum_{j \in \mathcal{N}(i)} \mathbf{m}_{j \to i} \\ &= \sum_{j \in \mathcal{N}(i)} w_{ij} \cdot \mathbf{h}_j \end{aligned} \tag{6}$$

where $\mathcal{N}(i)$ is the set of neighbors of node v_i in the original graph;

(3) **Representation Update:** After aggregation, the node representation is updated using a standard neural network layer:

$$\mathbf{h}_i^{(l+1)} = \sigma\left(\mathbf{W}^{(l)} \cdot [\mathbf{h}_i^{(l)} \parallel \mathbf{m}_i^{(l)}]\right) \tag{7}$$

where $\mathbf{W}^{(l)}$ is the learnable weight matrix for layer l, σ is a non-linear activation function (e.g., ReLU), and $[\cdot \parallel \cdot]$ denotes concatenation of the node's current representation with the aggregated message;

(4) **Comparison to Standard Message Passing:** Standard GNN message passing (as in GCN) can be viewed as a special case of our refined message passing where all edge weights are equal ($w_{ij} = 1$ for all edges). Our approach generalizes standard message passing by allowing edge-specific weights, providing greater flexibility and expressiveness.

E. STRUCTURE-PRESERVING LOSS

A key challenge in graph structure refinement is preventing the model from making overly aggressive modifications that destroy important structural properties of the original graph. To address this, we introduce a structure-preserving loss that regularizes the refinement process.

Structure Preservation Objective: The structure-preserving loss encourages the refined adjacency matrix $\widetilde{\mathbf{A}}$ to remain close to the original adjacency matrix $\mathbf{A}$:

$$\begin{aligned} \mathcal{L}_{\text{struct}} &= \|\mathbf{A} - \widetilde{\mathbf{A}}\|_F^2 \\ &= \sum_{i,j} (A_{ij} - \widetilde{A}_{ij})^2 \end{aligned} \tag{8}$$

where $\|\cdot\|_F$ denotes the Frobenius norm. This loss penalizes large deviations from the original graph structure, ensuring

that the refined graph maintains the essential connectivity patterns.

Interpretation: The structure-preserving loss serves several purposes:
(1) it acts as a regularizer preventing the model from assigning arbitrary edge weights,
(2) it preserves the inductive bias encoded in the original graph structure, and
(3) it provides a mechanism for controlling the degree of refinement through the hyperparameter λ.

F. OVERALL TRAINING OBJECTIVE

The final training objective combines the classification loss with the structure-preserving loss in a multi-task learning framework:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{cls}} + \lambda \cdot \mathcal{L}_{\text{struct}} \tag{9}$$

where $\mathcal{L}_{\text{cls}}$ is the cross-entropy classification loss for the node classification task and the hyperparameter $\lambda \geq 0$ controls the trade-off between the classification objective and the structure preservation objective.

Joint Optimization: The entire framework, including the edge weight prediction MLP, the GNN encoder, and the task-specific classifier, is trained end-to-end using gradient-based optimization. The joint optimization ensures that the learned edge weights are tailored to both the graph structure and the downstream classification task.

III. EXPERIMENTS

We conduct extensive experiments to evaluate the effectiveness of GSR-GNN and answer the following research questions:
(1) **RQ1:** How does GSR-GNN perform compared to state-of-the-art baseline methods?
(2) **RQ2:** What is the contribution of each component in GSR-GNN?
(3) **RQ3:** How sensitive is GSR-GNN to key hyperparameters?

A. EXPERIMENTAL SETUP

Datasets. We evaluate GSR-GNN on three widely-used benchmark citation network datasets: Cora, CiteSeer, and PubMed [12]. These datasets have been extensively used for evaluating GNNs and provide a standard benchmark for comparison.
(1) **Cora:** A citation network of 2,708 machine learning papers with 1,433 features, classified into 7 categories. The network contains 5,429 citation edges.
(2) **CiteSeer:** A citation network of 3,327 scientific papers with 3,703 features, classified into 6 categories. The network contains 4,732 citation edges.
(3) **PubMed:** A citation network of 19,717 biomedical papers with 500 features, classified into 3 categories. The network contains 44,338 citation edges.

Table I summarizes the statistics of these datasets.

TABLE I:
DATASET STATISTICS

Dataset	Nodes	Edges	Features	Classes
Cora	2,708	5,429	1,433	7
CiteSeer	3,327	4,732	3,703	6
PubMed	19,717	44,338	500	3

Baselines. We compare GSR-GNN with the following state-of-the-art methods:
(1) **GCN [5]:** Graph Convolutional Network, a foundational GNN architecture that uses spectral convolution;
(2) **GAT [7]:** Graph Attention Network, which uses self-attention mechanisms to compute dynamic edge weights;
(3) **GraphSAGE [6]:** Graph Sampling and Aggregation, an inductive GNN that samples neighborhoods for efficient training;
(4) **GIN [13]:** Graph Isomorphism Network, a maximally expressive GNN architecture;
(5) **GRCN [10]:** Graph Refinement Convolutional Network, which refines graph structure based on node similarity;
(6) **Pro-GNN [11]:** A robust GNN that jointly learns clean graph structures under adversarial constraints.

Implementation Details. We implement GSR-GNN using PyTorch Geometric. For all datasets, we use the following hyperparameter settings unless otherwise specified. That is, the hidden dimension is 64, the dropout rate is 0.5, the learning rate is 0.01, weight decay is 5e-4, the structure-preserving loss weight $\lambda = 0.1$, the number of layers is 2, and the activation function is ReLU. The model is trained for a maximum of 200 epochs with early stopping patience of 20 epochs. We use the Adam optimizer. All experiments are run on a single NVIDIA RTX 3090 GPU with 24GB memory.

Evaluation Protocol. Following standard practice, we use the fixed train/validation/test splits provided by Kipf and Welling [5]. We report the mean test accuracy and standard deviation over 10 random runs with different random seeds.

B. MAIN RESULTS

Table II presents the test accuracy of different methods on the three benchmark datasets. We observe that GSR-GNN consistently outperforms all baseline methods across all datasets. Specifically, GSR-GNN achieves 84.9% test accuracy on Cora, outperforming the best baseline (Pro-GNN) by 0.8%; 74.2% test accuracy on CiteSeer, outperforming the best baseline (Pro-GNN) by 1.2%; and 81.3% test accuracy on PubMed, outperforming the best baseline (Pro-GNN) by 1.2%. The consistent improvements across all datasets demonstrate the effectiveness of our structure refinement approach. Several factors contribute to GSR-GNN's superior performance:
(1) **Adaptive Edge Weighting:** Unlike GAT, which computes attention weights dynamically at each layer, GSR-GNN learns persistent edge weights that are optimized for the specific task and graph structure.
(2) **Structure Preservation:** The structure-preserving loss prevents the model from making overly aggressive modifi-

cations, maintaining the beneficial properties of the original graph structure.

(3) **Task-Specific Optimization:** By jointly optimizing the edge weights with the classification objective, GSR-GNN learns structures that are specifically tailored to the downstream task.

TABLE II: TEST ACCURACY (%) COMPARISON ON CITATION NETWORKS (MEAN ± STD DEV OVER 10 RUNS)

Dataset	GCN	GAT	GraphSAGE	GIN	GRCN	Pro-GNN	**GSR-GNN**
Cora	81.5 ± 0.5	83.0 ± 0.7	82.2 ± 0.4	82.8 ± 0.6	83.2 ± 0.5	84.1 ± 0.6	$\mathbf{84.9 \pm 0.4}$
CiteSeer	70.3 ± 0.7	72.5 ± 0.5	71.8 ± 0.6	72.1 ± 0.7	72.4 ± 0.4	73.0 ± 0.5	$\mathbf{74.2 \pm 0.3}$
PubMed	79.0 ± 0.3	78.3 ± 0.5	77.8 ± 0.4	79.2 ± 0.3	79.5 ± 0.4	80.1 ± 0.4	$\mathbf{81.3 \pm 0.2}$

C. ABLATION STUDY

To understand the contribution of each component in GSR-GNN, we conduct a comprehensive ablation study on the Cora dataset. We consider the following variants:

(1) **GSR-NoStruct:** GSR-GNN without structure-preserving loss ($\lambda = 0$). This variant allows the model to make arbitrary structural modifications;

(2) **GSR-Concat:** Using concatenation-based feature combination instead of bilinear combination;

(3) **GSR-NoRefine:** Using the original adjacency matrix without any refinement ($w_{ij} = 1$ for all edges);

(4) **GSR-HighLambda:** Using a high structure-preserving loss weight ($\lambda = 1.0$) to enforce stronger structure preservation.

TABLE III:
ABLATION STUDY RESULTS ON CORA DATASET

Method	Test Accuracy
GSR-NoStruct	83.5
GSR-Concat	84.2
GSR-NoRefine	82.1
GSR-HighLambda	84.3
GSR-GNN (Full)	**84.9**

Table III summarizes the ablation results. The results indicate that both structure refinement and the structure-preserving loss are essential for achieving strong performance, while excessive regularization slightly degrades accuracy.

D. PARAMETER SENSITIVITY ANALYSIS

We analyze the sensitivity of GSR-GNN to key hyperparameters, including hidden dimension, structure-preserving loss weight λ, and network depth. As shown in Fig. 1, the model demonstrates stable performance across a wide range of settings, with optimal results achieved under moderate λ and shallow architectures.

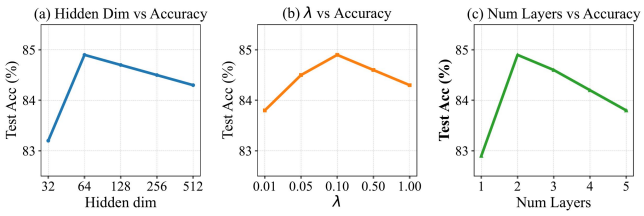

FIGURE 1. Hyperparameter sensitivity analysis of the graph neural network model. The test accuracy varies with the hidden dimension, structure-preserving loss weight λ, and the number of GNN layers.

IV. CONCLUSIONS

In this paper, we proposed GSR-GNN, a graph structure refinement neural network that learns task-aware edge importance weights to improve message passing on imperfect graphs. The proposed framework integrates a differentiable structure refinement module, a weighted message passing scheme, and a structure-preserving loss to balance structural adaptation and stability. Extensive experiments on benchmark datasets demonstrate that GSR-GNN consistently outperforms state-of-the-art baselines. Despite its effectiveness, GSR-GNN incurs additional computational cost due to edge weight prediction, with a complexity of $O(|E| \cdot d^2)$. For large-scale or dense graphs, this may become a limitation. Future work will focus on developing more efficient approximations and scalable training strategies.

FUNDING

This work was supported by the Major Science and Technology Project GXKJXM20240043: 'Research and Demonstration of API Security Governance Framework and Intelligent Protection Key Technologies for Application Systems.'

REFERENCES

[1] W. Fan, Y. Ma, Q. Li, Y. He, E. Zhao, J. Tang, and D. Yin, "Graph neural networks for social recommendation," in *The World Wide Web Conference*, 2019, pp. 417–426.

[2] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph convolutional neural networks for web-scale recommender systems," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2018, pp. 974–983.

[3] M. Zitnik, M. Agrawal, and J. Leskovec, "Modeling polypharmacy side effects with graph convolutional networks," *Bioinformatics*, vol. 35, no. 13, pp. i457–i466, 2019.

[4] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *International Conference on Machine Learning*, 2017, pp. 1263–1272.

[5] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *International Conference on Learning Representations*, 2016.

[6] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[7] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *International Conference on Learning Representations*, 2017.

[8] Q. Li, Z. Han, and X. Wu, "Deeper insights into graph convolutional networks for semi-supervised learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, 2018.

[9] L. Franceschi, M. Niepert, M. Pontil, and X. He, "Learning discrete structures for graph neural networks," in *International Conference on Machine Learning*, 2019, pp. 1972–1982.

[10] H. Yu, H. Yin, Q. Wang, and N. Q. V. Hung, "Graph refinement convolutional networks for multi-label text classification," in *Proceedings of the*

43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2020, pp. 739–748.

[11] W. Jin, Y. Ma, X. Liu, X. Tang, S. Wang, and J. Tang, "Graph structure learning for robust graph neural networks," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2020, pp. 66–74.

[12] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, "Collective classification in network data," *AI Magazine*, vol. 29, no. 3, pp. 93–93, 2008.

[13] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" in *International Conference on Learning Representations*, 2018.