https://doi.org/10.1007/sxxxxx-yyy-zzzz-1

APPENDIX

JAMN-GNN: Jointly-adversarial graph neural network for noisy labels and missing attributes

Guangliang ZHAO[1], Yulin LIU[1], Anchen LI[2], Ping ZHANG[3], Xueyan LIU[2], Yan ZHANG[1], Riting XIA[1]✉

1. College of Computer Science, Inner Mongolia University, Hohhot 010021, China
2. School of Computer Science and Technology, Jilin University, Changchun 130012, China
3. School of Artificial Intelligence, Hebei University of Technology, Tianjin 300401, China

Received month dd, yyyy; accepted month dd, yyyy

E-mail: Riting XIA, xiart@imu.edu.cn.

1 Problem

Despite the extensive attention GNNs have received, two major challenges in most existing approaches are overlooked:

First, the performance of GNNs is highly dependent on accurate node labels, but obtaining precise labels remains challenging in real-world scenarios. There are many graphs with label noise (as shown in Fig. 1(a)). For example, in social networks, labels often rely on unreliable user input. Additionally, graphs are vulnerable to adversarial label-flipping attacks, leading to pervasive label noise in graphs. Many works have demonstrated that label noise significantly degrades the generalization ability of machine learning models in the computer vision and natural language processing domains. In GNNs, the message-passing mechanism further exacerbates this negative effect by propagating incorrect supervisory information from mislabeled nodes throughout the entire graph, resulting in severe consequences.

Second, the performance of GNNs is critically dependent on complete attribute information to achieve optimal results. However, attribute missingness, where certain nodes lack attributes, has become a prevalent issue in graph data due to privacy preservation requirements (as shown in Fig. 1(b)). For example, in social networks, some users may selectively disclose or completely conceal personal information due to privacy concerns, directly resulting in missing attributes of users. The absence of attributes prevents GNN models from obtaining comprehensive data support, thereby compromising their performance. More critically, when GNNs simultaneously encounter both missing attributes and label noise (as shown in Fig. 1(c)), the situation becomes substantially more severe.

The absence of node attributes leaves insufficient discriminative features to suppress the propagation of label noise. This phenomenon creates a vicious cycle, where incorrect labels diffuse along the graph structure, severely undermining the model's robustness. Consequently, GNNs trained under the dual challenges of attribute absence and noisy labels are bound to experience significant performance degradation in real-world scenarios.

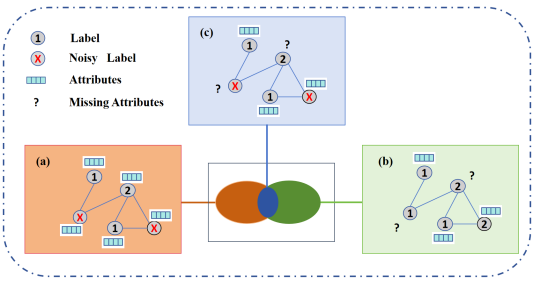


Fig. 1 Sketch maps of graph with (a) incorrect labels with complete attributes, (b) missing attributes with correct labels, and (c) the compound challenge of both defects coexisting.

1.1 Noisy labeled graph with missing attributes.

$\mathcal{G} = (\mathcal{V}, \mathcal{X}^o, \mathcal{A}, \mathcal{Y})$ is a graph with both label noise and missing attributes, where $\mathcal{V} = \{v_1, \ldots, v_N\}$ is the node set with N nodes, $\mathcal{X}^o = \{x_1, \ldots, x_{N^o}\}$ is the node attribute set of the attribute observed nodes, and x_i represents the attributes of node i, $N^0 \subset N$. The adjacency matrix $\mathcal{A} \in \mathbb{R}^{N \times N}$ of graph $\mathcal{G}$ is defined such that $\mathcal{A}_{ij} = 1$ if the nodes v_i and v_j are connected, and $\mathcal{A}_{ij} = 0$ otherwise. $\mathcal{Y} = \mathcal{Y}_N \cup \mathcal{Y}_C$ represents the label set of all nodes, where $\mathcal{Y}_N = \{y_1^N, \ldots, y_{N_L}^N\}$ denotes the noisy labels for a subset of nodes $\mathcal{V}_L = \{v_1, \ldots, v_{N_L}\}$, and $\mathcal{Y}_C = \{y_{N_L+1}^C, \ldots, y_N^C\}$ denotes the clean labels for the subset $\mathcal{V}_C$, $N = N_L \cup N_C$.

2 Preliminaries

2.1 Graph Neural Networks

GNNs have shown remarkable effectiveness in handling graph-structured data across diverse applications. Most prominent GNN variants, such as GCN and GAT, employ a message-passing framework that aggregates information from neighboring nodes guided by the graph structure. The layer-wise propagation of node representations can be formally characterized as follows. For a given node $v \in \mathcal{V}$ in graph $\mathcal{G}$, its node representation $\mathbf{h}_v^{(l)}$ at the l-th layer is computed through two fundamental operations: (1) **Neighbor transformation**. Each neighboring node $u \in \mathcal{N}(v)$ undergoes attribute transformation:

$$\mathbf{h}_v^{(l)} = f_\theta^{(l)}\left(\mathbf{h}_u^{(l-1)}\right), \quad \forall u \in \mathcal{N}(v), \tag{1}$$

where $f_\theta^{(l)}$ denotes a transformation function in the propagation process at layer l, and $\mathcal{N}(v)$ represents the neighborhood of node v. (2) **Feature aggregation**. The target node's representation is updated by combining its previous state with the transformed neighborhood attributes:

$$\mathbf{h}_v^{(l)} = \phi^{(l)}\left(\mathbf{h}_v^{(l-1)}, \bigoplus_{u\in\mathcal{N}(v)} \mathbf{h}_u^{(l)}\right), \tag{2}$$

where $\phi^{(l)}$ is a differentiable aggregation function (e.g., mean, sum, or attention-based pooling), and $\bigoplus$ denotes a permutation-invariant aggregation operator.

2.2 Pseudo-Labeling Methods of Modis

Conventional confidence-based pseudo-labeling approaches exhibit several limitations, including poor calibration of uncertainty estimates and sensitivity to data distribution shifts. To overcome these challenges, Pei et al. introduced Memory Disagreement (MoDis), an innovative framework that measures uncertainty through the prediction dynamics of a node across training epochs.

Specifically, for an unlabeled node v, a memory bank $\mathcal{M}_v = \{\hat{y}_v^{(t)} \mid t \in \mathcal{T}\}$ is first constructed to record its predicted labels at selected epochs $\mathcal{T}$. MoDis is defined as the Shannon entropy of the prediction distribution:

$$\text{MoDis}(v) := \mathbb{H}[P_v] = -\sum_{c=1}^{C} P_v(c)\log P_v(c), \tag{3}$$

where $P_v(c) = \frac{1}{|\mathcal{M}_v|}\sum_{\hat{y}_v\in\mathcal{M}_v} 1[\hat{y}_v = c]$ represents the frequency of class c in the memory bank. A low entropy value indicates consistent predictions (low uncertainty). To enhance discriminability by integrating softmax trajectories, $P_v(c)$ is redefined as the average of softmax distributions:

$$P_v(c) = \frac{1}{|\mathcal{M}_v|}\sum_{\sigma_v\in\mathcal{M}_v} \sigma_v(c), \tag{4}$$

where $\sigma_v^{(t)} = \text{softmax}(z_v^{(t)})$ denotes the softmax probability distribution of node v at epoch t. A sharpening function $\text{Sharpen}(p,\gamma) := p_i^{1/\gamma}/Z$ (with $Z = \sum_l p_l^{1/\gamma}$) is applied to reduce the distribution entropy, emphasizing the impact of high-confidence predictions.

Theoretically, nodes with low MoDis (non-boundary nodes) exhibit label consistency with neighbors in the graph space and are far from decision boundaries in the representation space (low-density region embeddings), thus making them suitable for pseudo-label selection. To mitigate the impact of label noise, MoDis is introduced for pseudo-label acquisition in this study.

3 Experiments

3.1 Datasets

To validate the effectiveness of our method in addressing the challenges of attribute missingness and label noise, we conducted comprehensive experiments on five widely-used benchmark datasets: Cora, CiteSeer, PubMed, DBLP, and Amazon-Computers. The statistical characteristics of these datasets are detailed in Table 1. To ensure consistency, we aligned the training, validation, and test sets with the NoisyGL protocol. For attribute missingness, we randomly select a subset of nodes and remove their attributes. For label noise, following NoisyGL, we introduce Pair noise into the training and validation sets:

Table 1 Statistics of datasets.

Datasets	Nodes	Edges	Features	Classes
Cora	2708	5278	1433	7
CiteSeer	3327	4552	3703	6
PubMed	19717	44338	500	3
DBLP	17716	105734	1639	4
Amazon-Computers	13752	491722	767	10

- **Pair Noise**: Labels are assumed to make mistakes only within the most similar pair classes. More specifically, labels have a probability of p to flip to their pair class.

3.2 Baselines

For our comparative analysis, we evaluate many baselines for handling noisy labels with missing attributes, categorizing them into seven methodological groups: (1) loss correction techniques including Forward and Backward correction [1]; (2) robust loss functions comprising APL [2] and SCE [3]; (3) multi-network learning paradigms (Coteaching [4] and JoCoR [5]); (4) the noise adaptation layer approach (S-model [6]); (5) traditional graph neural network approaches (GCN [7] and GIN [8]); and (6) specialized GNNs for graph with label noise, including the state-of-the-art (NRGNN [9], RTGNN [10], RNCGLN [11], CLNode [12], and CRGNN [13]); (7) specialized GNNs for graphs with missing attributes, including the state-of-the-art (ITR [14], AIAE [15]).

We describe the baselines in detail as follows:

- **Forward** and **Backward** [1]: A loss correction method that revises predictions to obtain unbiased loss on noisy training samples.
- **APL** [2]: The loss function is constructed by combining robust active loss and robust passive loss, improving learning performance while ensuring robustness.
- **SCE** [3]: Enhances model robustness to noisy labels by combining a noise-tolerant term with the standard loss of cross-entropy.
- **Coteaching** [4]: This method maintains two networks to select clean samples for each other. Specifically, small-loss samples with different predictions are selected for training.
- **JoCoR** [5]: Utilizes consistency maximization to handle noisy labels. Instead of hard sampling, it explicitly regularizes two different classifiers to converge in predictions, trained by a joint loss function to minimize discrepancies. It uses pseudo-twin networks for joint training, with the loss function including supervised and contrastive loss.
- **S-model** [6]: Adds a noise-adaptive layer to model the transformation pattern from noisy labels to true labels.
- **GCN** [7]: A popular graph convolutional network based on spectral theory.
- **GIN** [8]: Using multi-layer perceptron to process aggregated information from neighbors.

- **NRGNN** [9] : Employs two GNNs with non-shared parameters for edge prediction and pseudo-label mining, implicitly leveraging latent neighbors to learn node representations.
- **RTGNN** [10]: Based on NRGNN, it governs the label nodes at a fine-grained level.
- **RNCGLN** [11]: Aims to alleviate both graph and label noise issues simultaneously. It first uses graph contrastive loss for local graph learning, employs multi-head self-attention mechanism to learn node representations globally, and then uses pseudo-graph and pseudo-labels to handle graph noise and label noise, respectively.
- **CLNode** [12]: Adopt a curriculum learning strategy to mitigate the impact of label noise. It first uses a multi-view difficulty metric to measure the quality of training nodes, then selects appropriate nodes for GNN training through a training scheduler based on quality.
- **CRGNN** [13]: Adopt a contrastive and dynamic cross-entropy loss are employed to capture local structural information while mitigating the impact of noisy labels and addressing overfitting, respectively. Furthermore, cross-space consistency ensures semantic alignment between embeddings.
- **ITR** [14] : This method combines reliable node attributes with structural graph information to construct effective representations for nodes with missing attributes.
- **AIAE** [15]: AIAE introduced a dual encoder mechanism based on knowledge distillation, aiming to accurately encode both attribute and structural information into the representations of nodes with missing attributes.

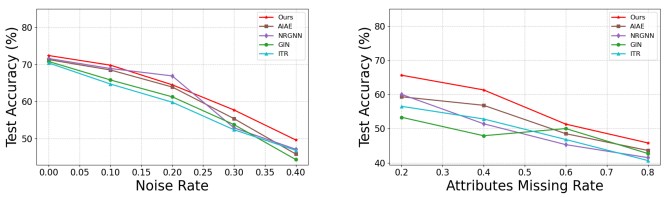
3.3 Implementation Details

All experiments were conducted on a configuration with 90 GB of memory on a NVIDIA Xeon(R) Platinum 8352V. We implement the baselines and proposed JAMN-GNN using PyTorch. For baselines, the same experimental setup as described in the original paper is adopted.

For our model, we use a two-layer GCN as the backbone GNN model, where the hidden layer size is 1024. The topological information encoder and GNN classifier both adopt two-layer GCN architectures. The hidden layers of the projection head and the GNN classifier are 1024 and 256, respectively. The learning rate is 0.02 and the decay rate is 1e-5. During data augmentation, node attributes are randomly masked with a fixed masking rate $r = 0.2$, and the number of multi-head attention heads k ranges within $\{2, 4\}$. In the first training phase, the loss weights α and β are set to 0.5 and 0.6, respectively. In the second phase, these weights are adjusted to $\psi = 3$ and $\phi = 4$. We report the average results from ten runs.

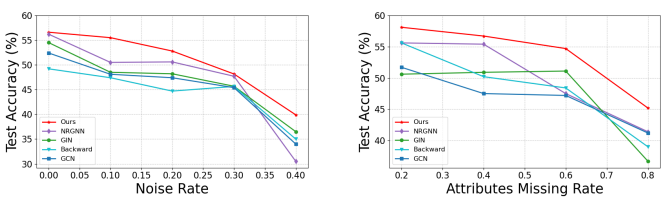
3.4 Impact of Different Noise Rates and Attribute Missing Rates

Robustness is evaluated on all datasets under two configurations. One involved a fixed attribute missing rate of 70% with varying label noise from 0% to 40%, while the other involved a fixed label noise of 30% with varying attribute missing rates from 20% to 80%. The top five results are shown in Figs. 2-6. Based on the experimental results, we observed that as both the label noise rate and attribute missing rate increase, all baselines exhibit a sharp performance decline. The performance gap between JAMN-GNN and the baselines further widens with increasing rates, demonstrating the effectiveness of its integrated strategy that combines attribute completion, dynamic cross-entropy based initial noise filtering, and accurate pseudo-label refinement to simultaneously address label noise and attribute missingness. Even in the absence of label noise, the proposed method still outperforms most baselines, as its pseudo-labeling strategy enhances supervisory signals by identifying reliable samples through cross-phase prediction consistency. Moreover, under high attribute missing rates such as 80%, the proposed approach maintains superior robustness, achieves better performance than the second-best method, and significantly alleviates the issue where missing attributes exacerbate noise propagation through the graph structure.



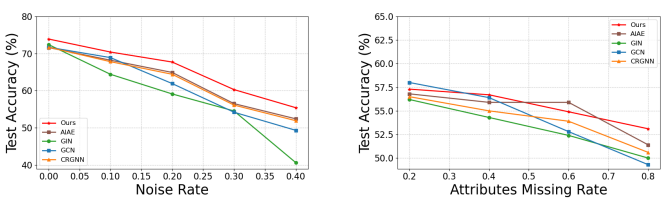
(a) Different levels of label noise **(b)** Different levels of missing attributes

Fig. 2 Accuracy on Cora with various rates of label noise and attribute missing.



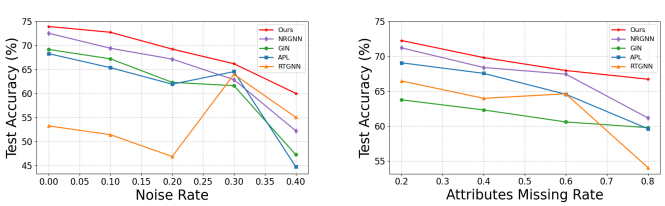
(a) Different levels of label noise **(b)** Different levels of missing attributes

Fig. 3 Accuracy on CiteSeer with various rates of label noise and attribute missing.



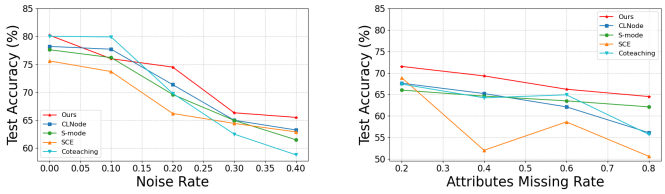
(a) Different levels of label noise **(b)** Different levels of missing attributes

Fig. 4 Accuracy on PubMed with various rates of label noise and attribute missing.



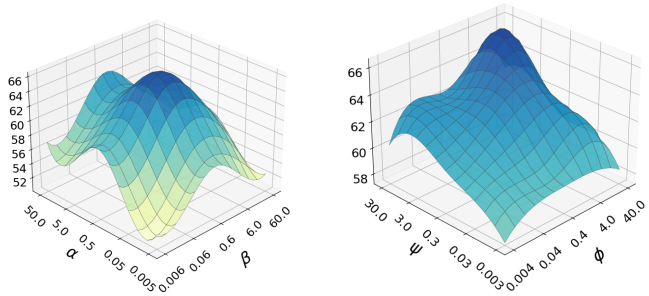
(a) Different levels of label noise **(b)** Different levels of missing attributes

Fig. 5 Accuracy on DBLP with various rates of label noise and attribute missing.



(a) Different levels of label noise **(b)** Different levels of missing attributes

Fig. 6 Accuracy on Amazon-Computers with various rates of label noise and attribute missing.



(a) Parameter Sensitivity for α and β **(b)** Parameter Sensitivity for ψ and ϕ

Fig. 7 Parameter sensitivity analysis on DBLP.

3.5 Ablation Experiment

In this subsection, we conduct an ablation study of each module on the DBLP dataset, with the attribute missing rate fixed at 70% and the label noise rate at 30%. As illustrated in Table 2, we examine two variants of the model: (1) "JAMN-GNN/NF" represents the ablated version without the two-stage noise filtering module, and (2) "JAMN-GNN/AI+NF" denotes the configuration that removes both the topology-guided attribute imputation module and the noise filtering mechanism.

It can be observed that when both the topology-guided attribute imputation module and the noise filtering mechanism are removed, the model achieves the lowest performance. This indicates that alleviating label noise issues is challenging without attribute completion or noise processing. Subsequently, the incorporation of the topology-guided attribute completion module significantly improves the model's performance, demonstrating that attribute completion enhances the model's effectiveness. Furthermore, the introduction of the two-stage noise filtering module enables the model to achieve optimal performance. These observations collectively validate the effectiveness of each module in our proposed method.

Table 2 Comparisons of different modules for JAMN-GNN.

Methods	Accuracy (%)
JAMN-GNN	66.19
JAMN-GNN/NF	63.30
JAMN-GNN/NF+AI	58.98

3.6 Parameter Sensitivity

To analyze the influence of hyperparameters α, β, ψ, and ϕ on the performance of JAMN-GNN, we first define their roles: α and β are the weighting coefficients for the attribute reconstruction loss $\mathcal{L}_{\text{rec}}$ and the preliminary noise-filtering loss $\mathcal{L}_{\text{noise}}$ in the first training stage, respectively; whereas ψ and ϕ are the weighting coefficients for the attribute reconstruction loss $\mathcal{L}_{\text{rec}}$ and the pseudo-label-enhanced supervised loss $\mathcal{L}_{\text{sup}}$ in the second stage, dynamically balancing the contributions of these components.

We conducted a parameter sensitivity analysis as follows: (1) With ψ and ϕ fixed at 0.3 and 0.4, we varied α across $\{0.005, 0.05, 0.5, 5, 50\}$ and β across $\{0.006, 0.06, 0.6, 6, 60\}$; (2) Conversely, with α and β fixed at 0.5 and 0.6, we varied ψ within $\{0.003, 0.03, 0.3, 3, 30\}$ and ϕ across $\{0.004, 0.04, 0.4, 4, 40\}$. The experimental results on the DBLP dataset are reported in Fig. 7. We observe that as α, β, ψ, and ϕ increase, performance first rises and then falls. The model achieves optimal performance when $\alpha = 0.5$, $\beta = 0.6$, $\psi = 3.0$, and $\phi = 4.0$.

3.7 Visualization

To demonstrate the superiority of our method, we visualize the Cora, Citeseer, and Amazon-Computers dataset using t-SNE, where nodes of the same color belong to the same category. This experiment is conducted under the settings where the attribute missing rate is 70% and the label noise rate is 30%. As shown in Figs. 8-10, the boundaries of our method are more distinct. In contrast, only a few categories are relatively clear in other methods, with the remaining nodes mixed together. This highlights the advantages of addressing both attribute missing and label noise in graphs.

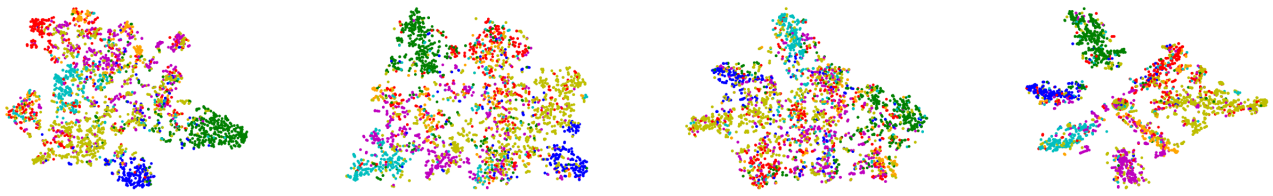
3.8 Computational Efficiency Analysis

To evaluate efficiency, we compared the running time of state-of-the-art methods for handling missing attributes and those for addressing label noise. The running times of all methods were measured under identical hardware configurations to ensure a fair comparison. The results are shown in Table 3.

It can be observed that methods such as ITR and AIAE, which tackle missing attributes, generally require higher computational time. This is because attribute completion is essentially a data generation problem that demands in-depth mining of graph information to reconstruct missing data, resulting in considerable computational overhead. In contrast, label noise handling is a discriminative filtering task that purifies supervisory signals through sample selection, weight adjustment, or label correction at the loss function level, thereby incurring relatively lower additional computational costs. Among the methods that address missing attributes, our approach shows higher efficiency. More importantly, our model achieves significantly superior performance over these methods by a substantial margin.

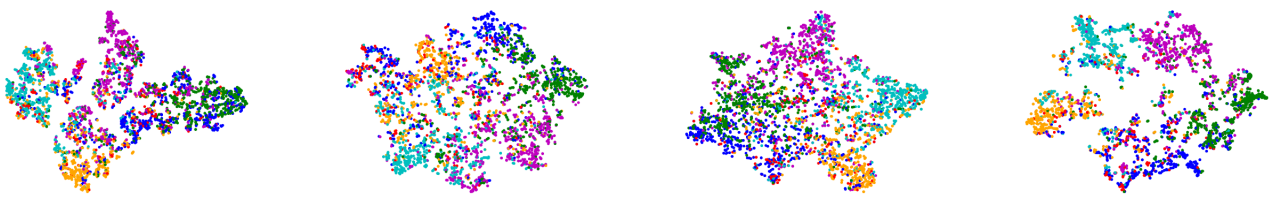
3.9 Analysis of Training Stability and Convergence

To evaluate the training stability and convergence of the model, we performed an experimental analysis of the joint optimization framework. As shown in Figure 11, the experimental results on the DBLP dataset demonstrate that the training accuracy increases steadily with



(a) RTGNN **(b)** NRGNN **(c)** CRGNN **(d)** Ours

Fig. 8 Visualization results on Cora.



(a) RTGNN **(b)** NRGNN **(c)** CRGNN **(d)** Ours

Fig. 9 Visualization results on Citeseer.

Table 3 Comparison of run time (s).

Datasets	NRGNN	RTGNN	CRGNN	ITR	AIAE	Ours
Cora	6	7.2	0.86	21.63	29.6	4
CiteSeer	5	39.6	2.36	36.52	30	8
PubMed	9.45	136	14.7	1220	1180	126
DBLP	46	97	2.36	980	969	138

the number of epochs. The curve exhibits a rapid initial ascent without significant fluctuations, indicating a stable training process. After approximately 150 epochs, the accuracy growth slows and stabilizes at a high level, suggesting that the model has sufficiently converged.

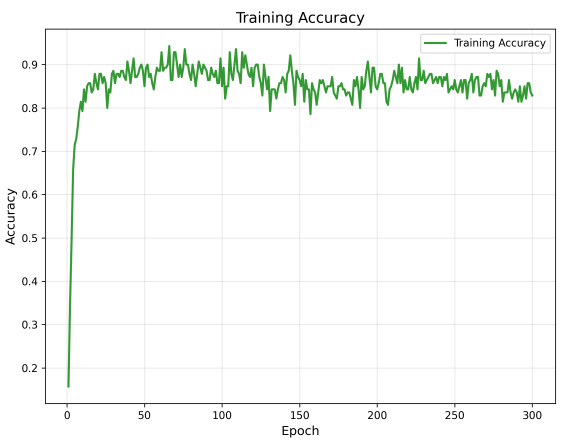


Fig. 11 Training convergence profile on DBLP.

3.10 Case study of JAMN-GNN

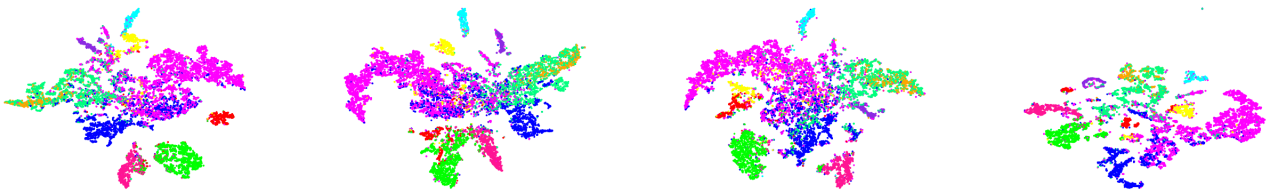
We design a case study to demonstrate the effectiveness of JAMN-GNN and analyze the contribution of each stage in its two-stage noise filtration module. This experiment is conducted under the settings where the attribute missing rate is 70% and the label noise rate is 30%. By visualizing the training set with labels in the embedding space, we demonstrate that the JAMN-GNN can reduce the noise ratio.

In the representation space (as illustrated in Fig. 12), our proposed model effectively reduces the overall noise ratio. The experimental setup is configured as follows: We inject 30% uniform noise into Cora, extract node representations through our model, and perform visualization analysis using t-SNE for dimensionality reduction. The experimental results demonstrate that: (1) In the first stage of the two-stage noise filtration module, the model successfully identifies and removes 9 noisy nodes, verifying the filtering capability of this stage for noisy labels; (2) In the second stage, although the pseudo-labeling process introduces a small amount of new noise, the incorporation of reliable pseudo-labels significantly increases the number of labeled nodes. Consequently, the overall noise ratio decreases by 8%(from 31% to 17%). These results show that through the two-stage noise filtration module, our model effectively reduces the noise ratio.

References

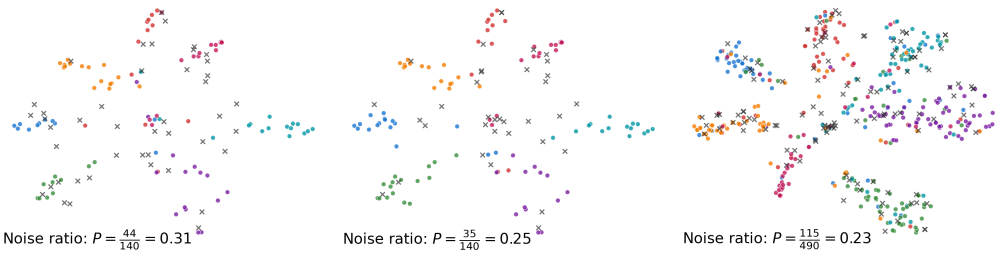
[1] Patrini G, Rozza A, Menon A K, Nock R, Qu L. Making deep neural networks robust to label noise: A loss correction approach. In: CVPR. 2017, 2233–2241

[2] Ma X, Huang H, Wang Y, Romano S, Erfani S M, Bailey J. Normalized loss functions for deep learning with noisy labels. In: ICML. 2020, 6543–6553



(a) RTGNN **(b)** NRGNN **(c)** CRGNN **(d)** Ours

Fig. 10 Visualization results on Amazon-Computers.



(a) Original label noise **(b)** Correct of first stage **(c)** Correct of two stage

Fig. 12 Visualization results on Cora with labels. Color-coded dots represent nodes with truth labels, while the cross markers denote nodes with noisy labels.

[3] Wang Y, Ma X, Chen Z, Luo Y, Yi J, Bailey J. Symmetric cross entropy for robust learning with noisy labels. In: ICCV. 2019, 322–330

[4] Han B, Yao Q, Yu X, Niu G, Xu M, Hu W, Tsang I W, Sugiyama M. Co-teaching: robust training of deep neural networks with extremely noisy labels. In: NeurIPS. 2018, 8536–8546

[5] Wei H, Feng L, Chen X, An B. Combating noisy labels by agreement: A joint training method with co-regularization. In: CVPR. 2020, 13723–13732

[6] Goldberger J, Ben-Reuven E. Training deep neural-networks using a noise adaptation layer. In: ICLR. 2017

[7] Kipf T N, Welling M. Semi-supervised classification with graph convolutional networks. In: ICLR. 2017

[8] Xu K, Hu W, Leskovec J, Jegelka S. How powerful are graph neural networks? In: ICLR. 2019

[9] Dai E, Aggarwal C, Wang S. NRGNN: learning a label noise resistant graph neural network on sparsely and noisily labeled graphs. In: SIGKDD. 2021, 227–236

[10] Qian S, Ying H, Hu R, Zhou J, Chen J, Chen D Z, Wu J. Robust training of graph neural networks via noise novernance. In: WSDM. 2023, 607–615

[11] Zhu Y, Feng L, Deng Z, Chen Y, Amor R, Witbrock M. Robust node classification on graph data with graph and label noise. In: IAAI. 2024, 17220–17227

[12] Wei X, Gong X, Zhan Y, Du B, Luo Y, Hu W. Clnode: Curriculum learning for node classification. In: WSDM. 2023, 670–678

[13] Li X, Li Q, Li D, Qian H, Wang J. Contrastive learning of graphs under label noise. Neural Networks, 2024, 172: 106113

[14] Tu W, Xiao B, Liu X, Zhou S, Cai Z, Cheng J. Revisiting initializing then refining: An incomplete and missing graph imputation network. IEEE Trans, 2025, 3244–3257

[15] Xia R, Zhang C, Li A, Liu X, Yang B. Attribute imputation autoencoders for attribute-missing graphs. Knowl. Based Syst., 2024, 291: 111583