Neural Networks 168 (2023) 652–664

Contents lists available at ScienceDirect

Neural Networks

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

LaenNet: Learning robust GCNs by propagating labels

Chunxu Zhang [a,b], Ximing Li [a,b], Hongbin Pei [c], Zijian Zhang [a,b], Bing Liu [a,b], Bo Yang [a,b,*]

[a] College of Computer Science and Technology, Jilin University, Changchun 130012, China
[b] Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Jilin University, Changchun 130012, China
[c] Xi'an JiaoTong University, China

ARTICLE INFO

Keywords:
Graph Convolutional Networks
Robustness
Label

ABSTRACT

Graph Convolutional Networks (GCNs) can be acknowledged as one of the most significant methodologies for graph representation learning, and the family of GCNs has recently achieved great success in the community. However, in real-world scenarios, the graph data may be imperfect, *e.g.,* with noisy and sparse features or labels, which poses a great challenge to the robustness of GCNs. To meet this challenge, we propose a simple-yet-effective LAbel-ENhanced Networks (LaenNet) architecture for GCNs, where the basic spirit is to propagate labels together with features. Specifically, we add an extra LaenNet module at one hidden layer of GCNs, which propagates labels along the graph and then integrates them with the hidden representations as the inputs to the deeper layer. The proposed LaenNet can be directly generalized to the variants of GCNs. We conduct extensive experiments to verify LaenNet on semi-supervised node classification tasks under four noisy and sparse graph data scenarios, including the graphs with noisy features, sparse features, noisy labels, and sparse labels. Empirical results indicate the superiority and robustness of LaenNet compared to the state-of-the-art baseline models. The implementation code is available to ease reproducibility[1].

1. Introduction

Graph Convolutional Networks (GCNs) have drawn increasing attention for excellent performance on graph-structured data (Chen, Wei, Huang, Ding, & Li, 2020; Kipf & Welling, 2017; Pei, Wei, Chang, Lei, & Yang, 2020; Wu et al., 2020; Zhang, Cui, & Zhu, 2020). Generally, the basic idea is to learn the node representations by iterative message passing through neighbors. Numerous variants of GCNs were proposed and successfully applied to varied graph-based tasks, such as node classification (Petar et al., 2018; Rong, Huang, Xu, & Huang, 2019; Wu, Zhang et al., 2019), link prediction (Monti, Bronstein, & Bresson, 2017; Ying, He et al., 2018; Zhang & Chen, 2018), and graph classification (Errica, Podda, Bacciu, & Micheli, 2020; Xu, Hu, Leskovec, & Jegelka, 2019; Ying, You et al., 2018).

Existing GCNs tend to be trained on well-curated datasets such as citation networks (Namata, London, Getoor, Huang, & EDU, 2012; Sen et al., 2008), NELL (Carlson et al., 2010; Yang, Cohen, & Salakhutdinov, 2016) and Coauthor networks (Shchur, Mumme, Bojchevski, & Günnemann, 2018), where the vast majority of labels and features are well manually verified. Unfortunately, such datasets are generally unavailable in real-world applications, where we only have imperfect datasets with noisy and sparse labels as well as features (Hu, Tang, Tang, & Liu, 2013; Li, Wang, & Chen-Chuan, 2012). For instance, in terms of the location prediction task with social networks (*e.g.,* Facebook and Twitter), many users may provide imprecise or even blank location information due to privacy concerns (Li et al., 2012); and in terms of sentiment analysis also with social networks, the texts may contain many noisy words such as spelling mistakes, hot acronyms and informal utterances (Hu et al., 2013).

To make a preliminary analysis of GCNs on such imperfect datasets, we present some early empirical results of node classification accuracy across Cora with four noisy and sparse graph data scenarios. Particularly, we run the representative Graph Convolutional Networks (GCN) (Kipf & Welling, 2017) on the graphs with noisy features, sparse features, noisy labels, and sparse labels, respectively. As shown in Fig. 1, GCN is very sensitive to these noisy and sparse graph data scenarios, where the performance falls sharply as the graph data corruption increases. As a result, this phenomenon poses great challenges to GCNs in real-world applications, which motivates us to design robust GCNs for noisy and sparse graph learning setting.

* Corresponding author at: College of Computer Science and Technology, Jilin University, Changchun 130012, China.
E-mail addresses: cxzhang19@mails.jlu.edu.cn (C. Zhang), liximing86@gmail.com (X. Li), peihongbin@xjtu.edu.cn (H. Pei), zhangzj2114@mails.jlu.edu.cn (Z. Zhang), liubing20@mails.jlu.edu.cn (B. Liu), ybo@jlu.edu.cn (B. Yang).

[1] https://github.com/Zhangcx19/LAENNet.

https://doi.org/10.1016/j.neunet.2023.09.035
Received 31 July 2022; Received in revised form 3 March 2023; Accepted 20 September 2023
Available online 27 September 2023
0893-6080/© 2023 Elsevier Ltd. All rights reserved.

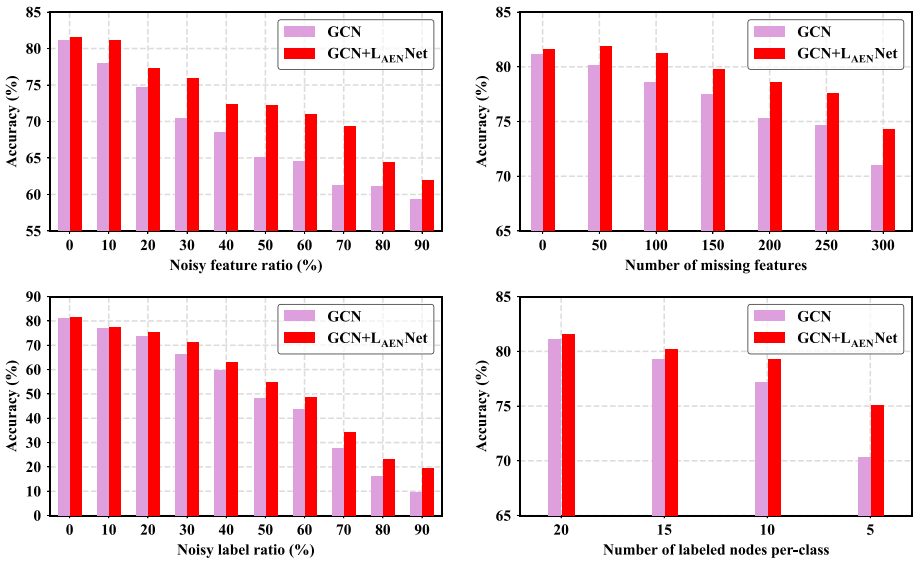


Fig. 1. Experimental results on Cora under four noisy and sparse graph data scenarios, including the graphs with noisy features, sparse features, noisy labels, and sparse labels (details are shown in Section 4). We show the node classification accuracy of GCN and our enhanced version GCN+LaenNet. As the graph data corruption increases, the performance of GCN decreases continuously. In contrast, our model can enhance GCN to make it more robust.

Recently, there have been some attempts to improve the robustness of GCNs under specific scenarios. According to the stage where the robust learning technique is used, we divide the existing works into two categories: *pre-processing before GCNs' training* and *applying during GCNs' training*. For the pre-processing-based methods (Entezari, Al-Sayouri, Darvishzadeh, & Papalexakis, 2020; Wu, Wang et al., 2019), the basic idea is to denoise the graph via heuristic methods so that GCNs can be trained on the denoised graph. However, the pre-processing techniques are generally designed based on prior knowledge about network properties, which are designed for specific noise types and make it difficult to achieve satisfactory performance for various noisy graph data settings. For the methods applying robust learning techniques during GCNs' training (Dai, Aggarwal, & Wang, 2021; Jin et al., 2020; Luo et al., 2021; Zhang & Zitnik, 2020), they aim to denoise or augment the graph along with the GCNs model training. However, these methods generally require additional tremendous computational overhead (Dai et al., 2022). On the other hand, several works have been developed to deal with incomplete node features (Chen, Chen et al., 2020; Taguchi, Liu, & Murata, 2021) or sparse node labels (Sun, Lin, & Zhu, 2020), whose basic idea is to recover or enrich the sparse feature and label information. In summary, there is still a lack of effective and efficient GCNs yet available that can deal with both noisy and sparse graph data.

To overcome the sparse data issue, an intuitive solution is to supplement auxiliary information, *e.g.,* collecting more raw features or ground-truth labels, which is time-consuming and difficult. For the various noisy data scenarios, combining multiple denoising techniques will increase the complexity of GCNs and introduce additional computational overhead. All the above challenges bring a question, *how can we design lightweight GCNs that can handle a variety of common sparse and noisy graph scenarios?* Recalling the working mechanism of GCNs, the consistency assumption, *i.e.,* neighboring nodes own similar characteristics, GCNs can predict the label distribution for all nodes. The label indicates the characteristics about node categories and neighborhood information. These two properties make label prediction a valuable data source. Introducing label prediction into GCNs' learning compensates the sparse data, enhances the topology information preservation, and benefits message passing process.

To implement the idea, our basic inspiration is to take the node label distribution predicted by GCNs as auxiliary features which participate in the message passing together with the node raw features. We develop a simple-yet-effective **LA**bel-**EN**hanced **Net**works (**LaenNet**) architecture for GCNs. Specifically, we add an extra LaenNet module at one hidden layer of GCNs, which propagates the label among the graph and then integrates them with the hidden representations as to the inputs to the deeper layer. By introducing the node label distribution information, the neighborhood aggregation schema can be enhanced with effective information capture which alleviates the impact of sparse data. On the other hand, the node label is learned based on the neighboring nodes and describes the local structure of graph data. Re-sending this information to GCNs can constrain the training to maintain the local smoothness implicitly, which improves the model's resistance to noise and learn robust GCNs. Also, this paradigm is supported by the label continuity assumption which is usually adopted in the semi-supervised graph learning methods (Raghavan, Albert, & Kumara, 2007; Zhou, Bousquet, Lal, Weston, & Schölkopf, 2004). LaenNet can be directly generalized to the variants of GCNs. In this work, we instantiate it with three representative GCNs, including GCN, Simple Graph Convolution (SGC) (Wu, Zhang et al., 2019), and Graph Attention Network (GAT) (Petar et al., 2018). We examine those instantiations by semi-supervised node classification, especially under four challenging scenarios, including noisy features, sparse features, noisy labels, and sparse labels. Empirical results indicate the superiority and robustness of LaenNet compared to the state-of-the-art baseline models.

The contributions of this work can be summarized as follows,

- We propose a novel architecture for GCNs named LaenNet, whose basic idea is to take the label as auxiliary features. As far as we know, this is the first attempt to integrate labels into the message passing process to generally enhance the robustness of GCNs.
- As a lightweight framework, our proposed LaenNet can be easily integrated into existing GCNs to enhance their robustness against sparse and noisy scenarios with neglectable computational cost.
- Experimental results show that LaenNet performs well under four challenging graph data scenarios, and also improves GCNs' performance in generic settings. Additionally, thorough model analysis demonstrates the effectiveness and flexibility of our model.

2. Related work

2.1. Graph convolutional networks

Graph data, which describes the complex relationship between

objects, has been an indispensable data representation form in an increasing number of applications, such as recommender systems (Fan et al., 2022; Wang, He, Cao, Liu, & Chua, 2019), traffic forecasting (Li, Yu, Shahabi, & Liu, 2018; Yu, Yin, & Zhu, 2018) and chemical molecule discovery (Hao et al., 2020; Liu, Allamanis, Brockschmidt, & Gaunt, 2018). Due to the complexity of graph data, it has posed great challenges for machine learning algorithms. Recently, there are lots of studies that extend deep learning technique on graph. As the most successful exploration, Graph Convolutional Networks (GCNs) have attracted a lot of attention.

Generally, there are two mainstream branches of GCNs, including spectral-based methods and spatial-based methods. As the early remarkable research on spectral-based GCNs, Bruna, Zaremba, Szlam, and LeCun (2014) develop a GCN based on the spectrum of graph Laplacian, which generalizes Convolutional Neural Networks (CNNs) to graph signal. Later, Defferrard, Bresson, and Vandergheynst (2016) extend the spectral GCN with fast localized convolutions, which improves computational efficiency by avoiding the use of graph Fourier basis. Graph Convolutional Network (GCN) (Kipf & Welling, 2017) improves the spectral graph convolution via the localized first-order approximation and has become one of the most representative GCNs due to its simplicity. Levie, Monti, Bresson, and Bronstein (2018) design an efficient spectral filtering method based on the Cayley polynomials, which specializes in narrow frequency bands with fewer filter parameters while preserving the advanced properties of Chebyshev filters (Defferrard et al., 2016). Zhu and Koniusz (2020) derive a simple spectral GCN via the Markov Diffusion Kernel, which keeps a good balance between low-pass and high-pass filter bands capturing the global and local neighborhood information, respectively. Bianchi, Grattarola, Livi, and Alippi (2022) design an auto-regressive moving average (ARMA) convolutional layer, which models the flexible filter responses and better captures global structure.

Similar to CNNs, spatial-based methods model GCN layer based on the node's spatial relations. Diffusion-Convolutional Neural Networks (DCNNs) (Atwood & Towsley, 2016) learns the node representation by regarding convolution as a diffusion process and keeps high flexibility and efficiency. Message Passing Neural Networks (MPNNs) (Gilmer, Schoenholz, Riley, Vinyals, & Dahl, 2017) develops a common message passing GCN framework, which summarizes the convolution as information propagation along edges. GraphSAGE (Hamilton, Ying, & Leskovec, 2017) generates node embedding by sampling and aggregating the node features of the local neighborhood, which is an inductive framework and is easily generalized to unseen nodes. Graph Attention Networks (GATs) (Petar et al., 2018) introduces the self-attention mechanism into GCN, which assigns different weights for different neighbor nodes. Graph Isomorphism Network (GIN) (Xu et al., 2019) builds a theoretical framework to analyze the representational power of GCNs by connecting it with Weisfeiler-Lehman (WL) graph isomorphism test, and presents a simple architecture whose representational power is equal to the WL test. Elastic GNNs (Liu et al., 2021) presents a message passing algorithm, which enhances the local smoothness via l1-based graph smoothing.

In addition to the above two categories, there are also other representative GCNs variants, such as Graph AutoEncoders (GAEs) (Kipf & Welling, 2016; Kollias, Kalantzis, Idé, Lozano, & Abe, 2022; Pan et al., 2018; Wang et al., 2021) and Spatial-Temporal Graph Neural Networks (STGNNs) (Guo, Lin, Feng, Song, & Wan, 2019; Guo, Lin, Wan, Li, & Cong, 2021; Seo, Defferrard, Vandergheynst, & Bresson, 2018).

2.2. Learning GCNs with disturbed graph data

In the practical applications, it is common that the graph data is noisy or sparse due to privacy constraints and data collection. In this subsection, we summarize the GCNs that are designed for robust graph learning. Particularly, we divide the existing works into two categories according to the graph data scenarios, including noisy graph data and sparse graph data.

For the works designed for noisy graph data, we summarize them into two paradigms. The basic idea of the first paradigm is to denoise the graph by some pre-processing techniques before the graph is used for GCNs' training (Entezari et al., 2020; Wu, Wang et al., 2019). For example, GCN-Jaccard (Wu, Wang et al., 2019) deletes the edges connecting nodes with low node features Jaccard similarity. The basic observation behind this is that graph structures are usually vulnerable to attacks, and the attack method is to add edges between dissimilar nodes. For the second paradigm, they aim to simultaneously learn a denoised (enhanced) graph and the GCN model so that it can make accurate predictions via the denoised (enhanced) graph (Dai et al., 2021; Jin et al., 2020; Luo et al., 2021; Zhang & Zitnik, 2020). Take Noise-Resistant Graph Neural Networks (NRGNN) (Dai et al., 2021) as an instance, it investigates the semi-supervised node classification of GCN on the graph with noisy labels. It first utilizes an edge predictor to generate missing links between unlabeled nodes and labeled nodes, and then enriches the supervised signals with a pseudo label miner. Finally, a robust GCN is trained on the new graph with an extended label set which can alleviate the negative effect of noisy labels.

There are also some works aiming to solve incomplete and sparse graph data problems, the basic idea is to recover or enrich the sparse node features and labels. Structure-Attribute Transformer (SAT) (Chen, Chen et al., 2020) is proposed to explore GCN's learning on the attribute-missing graph, that is, the attributes of some nodes in the graph are completely missing. Particularly, SAT makes an assumption that the heterogeneous attributes and topology are from the shared latent space, then proposes a joint distribution modeling based GCN by matching the latent representation distribution of two data sources. Graph Convolutional Networks for Missing Features (GCNMF) (Taguchi et al., 2021) is another GCN model to handle partially missing features. GCNMF represents the missing data with Gaussian Mixture Model (GMM) and integrates the process and graph learning into a unified framework. Consequently, the GMM parameters and GCN weights can be learned together in an end-to-end manner. Multi-Stage Self-Supervised (M3S) (Sun et al., 2020) concentrates on improving GCNs' generalization performance given limited label information. Particularly, M3S introduces the DeepCluster technique with the help of aligning mechanism to set pseudo labels for unlabeled nodes, then the confident unlabeled nodes will be added to the label set to enhance the supervision information.

All the above methods are designed to enhance GCNs towards a specific disturbed graph data scenario, *e.g.,* noisy graph structure, noisy labels incomplete features or sparse labels. In this paper, we explore to develop a robust GCN which can deal with common noisy and sparse graph learning settings. Our proposed LaenNet is more generic and we empirically show its superiority performance on various imperfect graph scenarios with noisy features, sparse features, noisy labels, and sparse labels.

2.3. Label enhanced GCNs

Recently, several studies try to enhance GCNs by utilizing label information. Approximate Personalized Propagation of Neural Predictions (APPNP) (Klicpera, Bojchevski, & Günnemann, 2019) proposes to separate the neural network from the message passing process. It first predicts the soft labels via GCN and then propagates the predictions with the personalized PageRank algorithm which can aggregate any depth of neighborhood, to solve the limited neighborhood range problem of GCNs. Graph Convolutional Networks-Label Propagation (GCN-LPA) (Wang & Leskovec, 2021) proposes a unified framework which integrates the Label Propagation Algorithm (LPA) and GCN. It studies the node feature/label influence to explore the relationship between GCNs and LPA, and then takes LPA as a regularization term of GCNs to assist the learning of edge weights to improve GCNs. Similar to APPNP, Correct and Smooth (C&S) (Huang, He, Singh, Lim, & Benson, 2021)

decomposes the model prediction process into two steps, including basic model prediction and prediction result refinement. It first performs basic prediction via a simple predictor, and then utilizes LPA to further correct and smooth the predictions. Unified Message Passaging Model (UniMP) (Shi, Huang, Feng, Zhong, Wang, & Sun, 2021) proposes to map the observed labels to feature space and incorporates the transformed labels and features jointly by a graph Transformer. Moreover, UniMP introduces a masked label prediction strategy to alleviate overfitting.

In summary, these methods regard feature and label as two independent data sources and enhance GCNs' learning in general semi-supervised setting. In this paper, we introduce the label into neighborhood aggregation schema to promote effective information capture. The proposed LaenNet can be easily integrated into existing GCNs to achieve robustness under common noisy and sparse graph data scenarios.

3. Methodology

In this section, we first present preliminaries about notations and GCN. Then, we introduce the details of the proposed LaenNet framework. In addition, we conduct comprehensive comparisons between LaenNet and the related work most akin to us, *i.e.,* GCN-LPA. Finally, we provide the time complexity analysis of the proposed method.

3.1. Formulation

Notation.

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an undirected graph, where $\mathcal{V}$ and $\mathcal{E}$ denote the sets of nodes and edges, respectively. Supposing there are totally n nodes, then the adjacent matrix is denoted by $\mathbf{A} \in \{0,1\}^{n\times n}$. The node feature is denoted by $\mathbf{x}_i \in \mathbb{R}^d$ and its label is denoted by $\mathbf{y}_i \in \{0,1\}^k$, *i.e.,* a one-hot vector, where d and k represent the numbers of features and categories, respectively. Let $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n]^\top \in \mathbb{R}^{n\times d}$ represents the node feature matrix. In this work, we concentrate on the scenario of transductive semi-supervised learning. Accordingly, the node set $\mathcal{V}$ consists of m labeled nodes $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^m$, denoted by a node label matrix $\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_m]^\top \in \mathbb{R}^{m\times k}$, and unlabeled ones $\{\mathbf{x}_i\}_{i=m+1}^n$.

Problem.

We discuss GCNs' learning in the noisy and sparse graph data scenarios. Particularly, we denote the disturbed features as $\overline{\mathbf{X}}$, which is a noisy or sparse variant of the original node features $\mathbf{X}$. Similarly, we represent $\overline{\mathbf{Y}}$ as the disturbed labels. In this paper, our goal is to predict the labels of all unlabeled nodes $\{\mathbf{y}_i\}_{i=m+1}^n$ based on $\{\mathbf{A}, \overline{\mathbf{X}}, \overline{\mathbf{Y}}\}$.

3.2. Preliminary about GCNs

GCNs utilize node features and graph topology structure to learn node latent representations for prediction. They update the node representation based on the representations of its neighboring nodes. To be specific, the computation of each convolution layer can be described as follows:

$$\mathbf{Z}_i = \sigma(\widehat{\mathbf{A}}\mathbf{Z}_{i-1}\mathbf{W}_i), \quad i \in 1, \ldots, l, \tag{1}$$

where $\mathbf{Z}_i$ is the output node embeddings of the ith layer and especially $\mathbf{Z}_0 = \mathbf{X}$; $\sigma(\cdot)$ is the nonlinear activation function, and we incorporate *relu* for hidden layers and *softmax* for output layer; $\widehat{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$ is the symmetrically normalized adjacent matrix and $\mathbf{D}$ is a diagonal matrix where $\mathbf{D}_{ii} = \sum_i \mathbf{A}_{ij}$; and $\mathbf{W}_i$ is the neural network weight parameters of the ith layer. For a GCN with l convolution layers, $\mathbf{Z}_l$ denotes its predictions, conventionally denoted by $\widehat{\mathbf{Y}} = [\widehat{\mathbf{y}}_1, \ldots, \widehat{\mathbf{y}}_n] \in \mathbb{R}^{n\times k}$, the optimization objective of semi-supervised learning is commonly formulated as:

$$\mathcal{L}(\mathbf{W}) = -\frac{1}{m}\sum_{i=1}^{m} \mathbf{y}_i ln \widehat{\mathbf{y}}_i, \tag{2}$$

where $\mathbf{W} = \{\mathbf{W}_i\}_{i=1}^l$ and $\mathcal{L}(\cdot)$ denotes the cross-entropy loss.

3.3. LaenNet

To achieve robust GCNs against noisy and sparse graph data, we propose a straightforward framework LaenNet, which propagates label predictions $\widehat{\mathbf{Y}}$ with node features together. As the prediction output, $\widehat{\mathbf{y}}_i$ depicts the ith node's category distribution information. It closely correlates to the learning task and is a valuable auxiliary feature for convolution operation, which can compensate the missing information introduced by sparse data. On the other hand, based on GCNs' neighborhood aggregation update mechanism, each node can learn $\widehat{\mathbf{y}}_i$ which contains the local graph structure information. Faced with the noisy data, the negative impact will be spread in the propagation process of GCNs. We propose to take label prediction as the auxiliary features participating in neighborhood aggregation, which regularizes the local smoothness and resists the noise during GCNs' training.

To implement the idea, there are three issues that needs to be carefully considered: *First*, most nodes of the graph are initially unlabeled and we need to use the zero vector instead of GCNs' label predictions to assist in the convolution. As a result, the prediction result may be disturbed since the zero-vector labels are uninformative, and bring negative impact for neighborhood aggregation. *Second*, due to the existence of sparse and noisy data, the label prediction $\widehat{\mathbf{Y}}$ may be inaccurate and incorporating them directly into the convolution process is likely to jeopardize GCNs' performance. *Third*, the number of dimensions of raw node features and label predictions are quite different, and directly integrating the two as unified features would weaken the category distribution and local structural characteristics contained in label predictions.

To solve the first problem, we design a coarse-grained filtering mechanism for label predictions. Particularly, we filter out the node if all its neighbor node labels are zero vectors, and delete the edges between it and all neighbors in the graph for the next iteration training of GCNs. We denote the set of all filtered nodes as $\mathbf{F}$, and modify the adjacent matrix as,

$$\mathbf{A}_{:j} = \overrightarrow{0}, \quad j \in \mathbf{F} \tag{3}$$

At the beginning of GCNs' training, some nodes would be filtered out to provide a refined graph structure. As the iterations progress, all nodes gradually acquire valid labels, all of which can be used for training.

To improve the quality of label prediction $\widehat{\mathbf{Y}}$, we propose a fine-grained label refinement strategy: For labeled nodes, we replace their label predictions with real labels. For unlabeled nodes, we only preserve the label predictions with high confidence. In each iteration, once GCNs complete the prediction, we apply the above strategy to refine $\widehat{\mathbf{Y}}$. First, we replace it with an augmented prediction matrix $\widetilde{\mathbf{Y}} = [\widetilde{\mathbf{y}}_1, \widetilde{\mathbf{y}}_2, \ldots, \widetilde{\mathbf{y}}_n]^\top \in \mathbb{R}^{n\times k}$ by using the ground-truth labels of labeled nodes:

$$\widetilde{\mathbf{y}}_i = \begin{cases} \mathbf{y}_i & 1 \le i \le m \\ \widehat{\mathbf{y}}_i & m < i \le n \end{cases} \tag{4}$$

Then, we only keep label predictions with high confidence, and filter out unlabeled nodes with low confidence and set them as zero vectors:

$$\widetilde{\mathbf{y}}_i = \overrightarrow{0}, \quad s.t. \quad m < i \le n, \quad \widetilde{\mathbf{y}}_i < \Psi \tag{5}$$

where Ψ is the confidence threshold hyper-parameter.

To solve the third issue, we design an effective and flexible fusion mechanism of raw features and label predictions. In particular, we propose to first spread the label predictions along edges. As a result, the

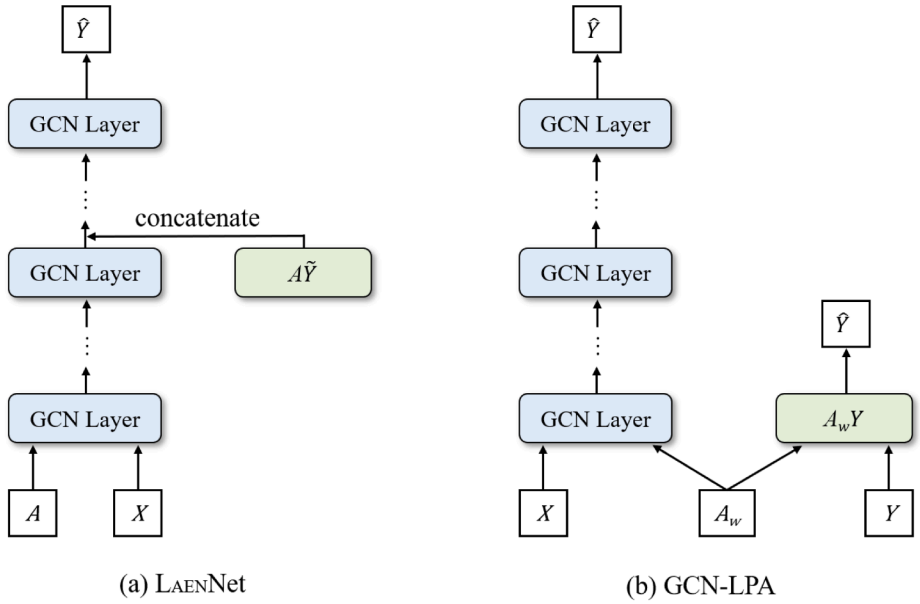


Fig. 2. Frameworks comparison of LaenNet and GCN-LPA. In LaenNet, we propose to take supervision as auxiliary features to participate in convolution. For GCN-LPA, it regards LPA as a regularization to learn the parameterized edge weights.

local structural information contained in the label predictions can be further enhanced through neighbor aggregation. Then, we integrate the enhanced label predictions into the hidden layers of GCNs, where the number of node latent representation dimension is comparable with label predictions. For a GCN with l convolution layers, we can combine the label predictions into any hidden layers through concatenation,

$$\mathbf{Z}_h \leftarrow [\mathbf{Z}_h, \mathbf{A}\widetilde{\mathbf{Y}}], \quad h \in 1, \ldots, l-1 \tag{6}$$

This concatenation result $\mathbf{Z}_h$ continues the typical GCNs operations to deeper layers. The procedure of LaenNet is summarized in Algorithm 1.

First, we review the objective function of GCN-LPA,

$$\mathscr{L}(\mathbf{A}_\mathbf{W}, \theta) = \ell(f_\theta(\mathbf{X}; \mathbf{A}_\mathbf{W}), \mathbf{Y}) + \mathscr{R}(g(\mathbf{Y}; \mathbf{A}_\mathbf{W}), \mathbf{Y}) \tag{7}$$

where $\mathbf{A}_\mathbf{W}$ is a learnable weighted adjacent matrix; f_θ and g are predictive functions of GCNs and LPA, respectively. We can see that GCN-LPA performs feature convolution and label propagation separately and takes LPA as a regularization term to assist the learning of edge weights. While according to Eq. (5), LaenNet takes labels as auxiliary features for prediction. Hence, the overall framework of the two methods is essentially different. We show the differences between the two frameworks in Fig. 2.

Algorithm 1 Training of LaenNet.

Input: Node feature matrix $\mathbf{X}$, adjacent matrix $\mathbf{A}$, node label matrix $\mathbf{Y}$, training epochs T, convolution layers l, integrated hidden layer index h.
Output: Labels of unlabeled nodes $\{\mathbf{y}_i\}_{i=m+1}^n$.

```
Initialize model parameters.
Initialize Ỹ, set labeled nodes {ỹ_i}_{i=1}^m with groundtruth one-hot label vectors and unlabeled nodes {ỹ_i}_{i=m+1}^n with 0⃗.
for t = 1 to T do
    Filter out the nodes whose neighbor node labels are all zero vectors and delete the edges between it and all neighbors in the graph.   ▷ Eq.(3)
    Refine the labels {ỹ_i}_{i=1}^m for labeled nodes.                                                                                    ▷ Eq.(4)
    Refine the labels {ỹ_i}_{i=m+1}^n for unlabeled nodes.                                                                                ▷ Eq.(5)
    for i = 1 to l do
        Compute node embeddings Z_i.                                                                                                      ▷ Eq.(1)
        if i = h then
            Compute combined node embedding Z_h.                                                                                          ▷ Eq.(6)
    Compute model loss ℒ.                                                                                                                 ▷ Eq.(2)
    Update model parameters via gradient descent.
return Labels of unlabeled nodes {y_i}_{i=m+1}^n according to Ŷ.
```

3.4. Differences between GCN-LPA and LaenNet

Among all the related works introducing label information into GCNs, GCN-LPA (Wang & Leskovec, 2021) is the most similar to LaenNet. Here we compare them and figure out the differences between them.

Besides, the label information used by the two frameworks is different. GCN-LPA only utilizes the ground-truth labels to perform label propagation in each iteration. In comparison, LaenNet utilizes both the ground-truth labels and the label predictions together to enhance GCNs.

Compared with GCN-LPA, our LaenNet takes GCNs' predictions as auxiliary features for convolution, which has two advantages: First, the predictions depict the node category distribution, and integrating them with node raw features can compensate the missing information caused by sparse data. Second, the local structural characteristics contained in

Table 1
A summary of dataset statistics.

Dataset	#node	#edge	#feature	#label
Cora	2,708	5,429	1,433	7
Citeseer	3,327	4,732	3,703	6
Pubmed	19,717	44,338	500	3
Coauthor-CS	18,333	81,894	6,805	15
Coauthor-Phy	34,493	247,962	8,415	5

predictions can be used for regularizing the local smoothness and resit noise. Both properties promotes the robust GCNs with our LaenNet. In addition, LaenNet is more flexible, since the learnable edge weight which is the essential contribution of GCN-LPA can be incorporated by using the attention-based GCNs as LaenNet's backbone.

3.5. Time complexity of LaenNet

In this subsection, we analyze the time complexity of LaenNet. For clarity, we restate the notation that, n and k denote the numbers of node and category; d represents feature's dimension; let r be the hidden representation's dimension after a graph convolution operation.

We first consider the time complexity of GCNs. Here, we take Graph Convolutional Network (GCN) (Kipf & Welling, 2017) as an example, and other GCNs are similar. For GCN, the main time cost is the two convolution operations and the time complexity is $O((d+r)n^2)$. For LaenNet instantiated with GCN, the main time cost is the two convolution operations and the label propagation, hence its complexity is $O((d+r)n^2+2kn^2)$. Notably, k is usually much less than d. For example, according to Table 1, k is no more than ten while d is several hundred or thousand. As a result, LaenNet enhances GCNs at a neglectable cost.

4. Experiments

In this section, we conduct experiments to evaluate our method LaenNet and explore to answer the following research questions:

- ***Q1***: Can LaenNet enhance GCNs' robustness under various noisy and sparse graph data scenarios?
- ***Q2***: Can LaenNet improve simple GCNs under generic scenario to gain superior performance compared to strong baselines?
- ***Q3***: How do the key hyper-parameters of LaenNet impact the performance?
- ***Q4***: Is label propagation helpful in graph convolution for learning better node representations?

Datasets.

We validate LaenNet on five benchmark semi-supervised graph learning datasets. Three citation network datasets include Cora, Citeseer, and Pubmed (Namata et al., 2012; Sen et al., 2008). In those datasets, each node represents a paper, and each edge indicates the citation relationship between papers. Two coauthor network datasets include Coauthor-CS and Coauthor-Phy (Shchur et al., 2018). They are based on the Microsoft Academic Graph from the KDD Cup 2016 challenge. In the network, each node indicates an author and the edge between two nodes denotes the two authors coauthor a paper. Node features depict the key words of author's papers and node labels represent the active research fields of the author. We summarize the statistics of five datasets in Table 1.

Implementation details.

In the experiments, we instantiate LaenNet with 3 representative 2-layer GCNs, including GCN (Kipf & Welling, 2017), SGC (Wu, Zhang et al., 2019), and GAT (Petar et al., 2018). Accordingly, we call them as **GCN+LaenNet**, **SGC+LaenNet**, and **GAT+LaenNet**, respectively. Our models are implemented by Pytorch, and run on Ubuntu 16.04.1 with 4 GeForce RTX 2080 GPUs. We tune their hyper-parameters on the validation set and report the average classification accuracy of 10 independent runs.

4.1. Evaluating LaenNet on noisy and sparse graph data (Q1)

To answer **Q1**, we generate dozens of synthetic datasets with various noisy and sparse settings, which are all common scenarios in real word. Specifically, for each benchmark dataset, we generate its splits with the prevalent protocol of semi-supervised node classification, *i.e.,* randomly selecting 20 labeled nodes per-class for training, 500 nodes as the validation set, and 1,000 nodes for testing (Kipf & Welling, 2017; Yang et al., 2016). We then transform them according to the following four scenarios and the implementation details are illustrated in Fig. 3:

- **Noisy features**: For each benchmark dataset, we randomly draw $\alpha(\%)$ features of each node and replace them with noisy features drawn from a uniform distribution over $[0,1]$. The noisy feature ratio α ranges from the set $\{0, 10\%, \ldots, 90\%\}$.
- **Sparse features**: For each benchmark dataset, we first select important features, *i.e.,* non-zero ones, using Lasso regression (Tibshirani, 1996). We then randomly remove a certain number of those features, ranging in the set $\{0, 50, \ldots, 300\}$.

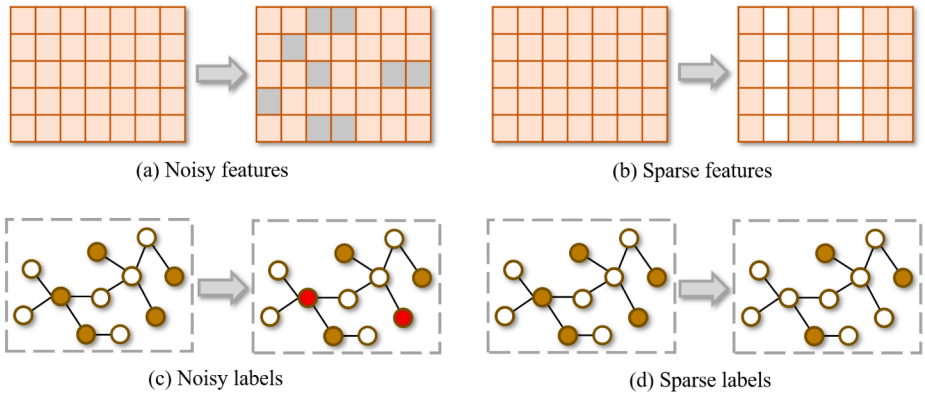
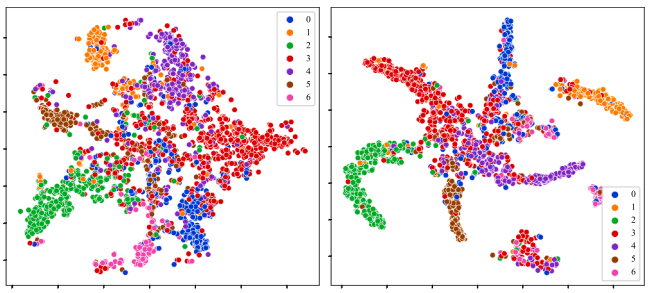


Fig. 3. Implementation details of four sparse and noisy graph data scenarios. (a) Noisy features: the left panel is the original node features and the right panel is the noisy node features where some features are replaced with noisy features drawn from a uniform distribution over [0, 1]. (b) Sparse features: the left panel is the original node features and the right panel is the sparse version where some important feature dimensions are removed. (c) Noisy labels: the left panel is the original graph, with some labeled nodes marked in brown. The right panel is the noisy labels graph where some nodes' labels are replaced with irrelevant labels. (d) Sparse labels: the left panel is the original graph, with some labeled nodes marked in brown. The right panel is used for GCNs' training where some labels are removed.

Table 2
Experimental results under the noisy features setting. We denote the results in bold if LaenNet achieves better performance than backbone GCNs.

	Noisy feature ratio α (%)									
Method	0	10	20	30	40	50	60	70	80	90
Cora										
M3S	80.30	77.30	76.50	69.60	70.50	69.80	69.20	68.50	62.40	60.10
GCN-LPA	80.63	73.80	68.80	64.90	62.30	61.70	61.40	61.10	58.70	57.80
GCN	81.08	77.92	74.68	70.44	68.50	65.14	64.46	61.17	61.12	59.34
GCN + LaenNet	**81.54**	**81.14**	**77.32**	**75.94**	**72.30**	**72.14**	**71.02**	**69.38**	**64.36**	**61.98**
SGC	80.15	77.28	69.38	62.60	59.32	57.20	54.58	53.16	51.98	51.04
SGC + LaenNet	**81.49**	**77.86**	**75.18**	**74.40**	**74.48**	**71.48**	**70.92**	**70.28**	**67.72**	**66.24**
GAT	82.80	79.80	78.20	75.20	75.10	74.60	73.70	70.30	69.00	59.70
GAT + LaenNet	**83.30**	**81.10**	**79.90**	**78.50**	**77.10**	**75.00**	**74.60**	**72.70**	**70.60**	**67.20**
Citeseer										
M3S	72.83	67.00	65.20	62.80	59.40	59.90	58.00	53.30	50.40	50.10
GCN-LPA	72.40	67.60	64.70	56.80	53.70	46.20	45.10	43.80	43.40	41.60
GCN	71.98	66.72	59.72	58.94	58.32	53.78	52.86	52.73	49.80	48.74
GCN + LaenNet	**72.59**	**68.96**	**65.52**	**63.06**	**62.10**	**58.62**	**58.28**	**56.66**	**54.42**	**53.82**
SGC	70.69	**67.64**	57.08	52.64	46.64	45.98	45.54	45.46	43.58	44.22
SGC + LaenNet	**71.44**	66.24	**62.68**	**60.44**	**59.20**	**58.66**	**57.96**	**56.78**	**54.68**	**52.14**
GAT	72.20	71.20	64.60	60.90	59.80	55.90	52.00	50.20	49.40	46.10
GAT + LaenNet	**72.30**	**71.60**	**66.30**	**63.00**	**62.40**	**61.70**	**58.00**	**56.50**	**54.70**	**53.30**
Pubmed										
M3S	80.20	52.20	44.10	43.60	42.00	40.10	40.60	38.90	39.70	38.50
GCN-LPA	77.43	45.60	42.80	41.90	40.30	41.90	40.80	40.10	41.20	41.23
GCN	77.56	75.12	73.70	**70.70**	**68.36**	**64.70**	**59.06**	53.98	47.72	43.74
GCN + LaenNet	**78.92**	**76.50**	**75.30**	68.95	65.80	62.10	57.83	**54.60**	**48.34**	**46.86**
SGC	77.84	74.59	70.62	64.51	58.97	54.08	50.18	43.01	38.02	35.62
SGC + LaenNet	**78.24**	**74.84**	**72.98**	**70.42**	**62.16**	**54.80**	**51.50**	**45.78**	**44.38**	**42.16**
GAT	77.20	73.50	75.60	71.00	68.10	68.40	57.50	51.00	49.10	48.00
GAT + LaenNet	**77.80**	**76.10**	**76.50**	**72.60**	**69.80**	**68.70**	**62.20**	**59.60**	**55.80**	**51.20**
Coauthor-CS										
M3S	90.87	87.40	87.00	86.40	84.30	81.94	80.03	72.70	65.40	55.70
GCN-LPA	91.20	88.00	87.40	87.20	86.10	85.60	84.40	80.10	79.63	79.10
GCN	90.40	88.90	88.70	86.70	85.10	82.10	80.01	77.70	76.10	74.80
GCN + LaenNet	**91.40**	**89.70**	**89.10**	**88.60**	**85.30**	**83.40**	**80.90**	**82.10**	**80.80**	**79.40**
SGC	91.60	89.70	88.70	83.00	79.00	74.70	74.50	66.90	65.60	53.20
SGC + LaenNet	**92.00**	**91.20**	**89.50**	**86.30**	**81.10**	**77.60**	**74.60**	**71.40**	**66.50**	**55.80**
Coauthor-Phy										
M3S	92.70	92.10	90.90	90.80	90.30	90.21	89.40	88.80	88.60	89.10
GCN-LPA	92.70	92.10	90.80	90.30	90.10	89.80	89.70	89.70	89.00	88.30
GCN	92.40	91.90	91.50	90.50	90.30	89.90	88.70	89.30	88.40	88.52
GCN + LaenNet	**92.90**	**92.20**	**91.80**	**92.10**	**91.90**	**93.00**	**90.90**	**92.40**	**90.40**	**91.70**
SGC	93.13	92.13	**90.47**	87.10	82.90	75.20	**74.90**	**67.30**	57.80	55.50
SGC + LaenNet	**94.30**	**92.90**	90.10	**88.40**	**84.00**	**76.90**	72.10	65.40	**61.20**	**57.10**



(a) GCN (b) GCN+L_{AEN}Net

Fig. 4. A 2-D visualization for the node representations from GCN and GCN+LaenNet under noisy features scenario with noise ratio as 50%. Different colors of nodes denote different labels.

- **Noisy labels**: For each benchmark dataset, we randomly select $\gamma(\%)$ labeled nodes and replace their true labels with irrelevant labels. The noisy label ratio γ ranges in the set $\{0, 10\%, \ldots, 90\%\}$.
- **Sparse labels**: For each benchmark dataset, we randomly reduce the number of labeled nodes per class from 20 to 15, 10, and 5, respectively.

We evaluate LaenNet on those synthetic datasets under various scenarios. In addition to validating whether our proposed model can enhance the robustness of the existing GCNs model, we also selected M3S and GCN-LPA as two comparison methods. Particularly, Ms3 is a robust GCN designed for the sparse label setting and GCN-LPA is a label enhanced GCN for improving GCNs' performance. Both methods are general for noisy and sparse settings and details can be found in Section 2. We omit the GAT and GAT+LaenNet on two coauthor datasets due to out of memory. The empirical results are presented in Tables 2, 3, 4, and

Table 3
Experimental results under the sparse features setting. We denote the results in bold if LaenNet achieves better performance than backbone GCNs.

	Number of missing features						
Method	0	50	100	150	200	250	300
Cora							
M3S	80.30	79.45	79.17	79.10	75.77	75.23	71.75
GCN-LPA	80.63	79.80	78.93	78.65	75.80	75.10	71.10
GCN	81.08	80.08	78.56	77.45	75.31	74.60	70.98
GCN + LaenNet	**81.54**	**81.90**	**81.19**	**79.79**	**78.60**	**77.60**	**74.30**
SGC	80.15	81.16	76.83	75.76	74.29	73.61	70.85
SGC + LaenNet	**81.49**	**82.12**	**79.87**	**79.01**	**78.52**	**77.98**	**77.06**
GAT	82.80	**81.10**	79.70	79.00	77.50	71.10	73.60
GAT + LaenNet	**83.30**	81.00	**80.00**	**79.50**	**78.20**	**73.00**	**74.80**
Citeseer							
M3S	72.83	72.10	71.52	71.30	69.40	69.20	68.15
GCN-LPA	72.40	72.10	71.97	70.57	69.23	67.67	65.27
GCN	71.98	71.46	70.28	70.24	69.23	67.70	66.46
GCN + LaenNet	**72.59**	**71.52**	**71.04**	**70.65**	**70.56**	**69.47**	**68.20**
SGC	70.69	69.98	70.10	68.56	67.88	67.27	64.06
SGC + LaenNet	**71.44**	**70.58**	**70.74**	**70.63**	**68.23**	**67.86**	**67.20**
GAT	72.20	73.40	70.20	69.30	71.20	68.40	65.90
GAT + LaenNet	**72.30**	**74.80**	**72.30**	**70.10**	**72.50**	**68.90**	**66.90**
Pubmed							
M3S	80.20	79.30	77.60	75.63	75.00	73.50	71.95
GCN-LPA	77.43	77.21	75.30	74.40	73.30	73.35	71.93
GCN	77.56	77.52	76.20	75.07	75.72	72.82	70.39
GCN + LaenNet	**78.92**	**79.12**	**77.40**	**76.54**	**76.06**	**75.04**	**72.22**
SGC	77.84	77.56	77.74	75.31	75.07	72.32	70.53
SGC + LaenNet	**78.24**	**78.20**	**78.30**	**77.76**	**75.34**	**75.54**	**74.66**
GAT	77.20	**73.60**	77.20	68.90	75.80	71.50	68.10
GAT + LaenNet	**77.80**	71.40	**77.60**	**69.50**	**76.40**	**72.90**	**69.80**
Coauthor-CS							
M3S	90.87	91.13	90.90	89.90	89.80	89.60	89.40
GCN-LPA	91.20	90.70	89.90	90.60	90.40	89.13	88.70
GCN	90.40	90.17	89.90	89.60	89.30	89.00	88.30
GCN + LaenNet	**91.40**	**91.23**	**90.80**	**91.60**	**90.90**	**90.40**	**91.00**
SGC	91.60	**91.50**	**91.40**	91.10	90.50	90.70	90.37
SGC + LaenNet	**92.00**	91.30	91.00	**91.90**	**91.80**	**91.00**	**90.60**
Coauthor-Phy							
M3S	92.70	92.63	92.57	92.20	91.90	91.70	90.40
GCN-LPA	92.70	92.90	92.50	92.30	91.80	91.30	91.10
GCN	92.40	92.27	91.70	91.60	91.47	91.07	89.80
GCN + LaenNet	**92.90**	**93.47**	**92.97**	**93.80**	**93.80**	**92.50**	**93.55**
SGC	93.13	92.27	92.50	92.10	91.93	91.80	91.63
SGC + LaenNet	**94.30**	**94.30**	**94.10**	**93.30**	**93.40**	**92.10**	**93.30**

5. Overall speaking, it can be seen that LaenNet consistently enhances GCNs in all settings and achieve superior performance than other two baselines. Several details of observations are made below.

In Table 2, as the noisy feature ratio α rises, the performances of the three GCNs baselines degrade by up to 20% on Cora and Citeseer, nearly 40% on Pubmed and two coauthor datasets. While the improvement of incorporating LaenNet is consistently remarkable especially when α ranges above 30%. Furthermore, it is notable that LaenNet enhanced versions achieve comparable results under the highest α with the baselines under relatively smaller α. For example, three LaenNet enhanced GCNs on Citeseer when $\alpha = 90\%$ perform better than baselines with $\alpha = 50\%$.

Table 3 illustrates models' performances under important feature dimensions missing. From the experimental results, we can see that LaenNet boosts all the baselines steadily with almost all the numbers of missing features. Among them, the performance boost on Cora, *i.e.,* at most 6.21%, is on averagely higher than the other two datasets, *i.e.,* at most 3.14% and 4.13% on Citeseer and Pubmed respectively.

Among the three GCNs baselines under the imperfect features setting (noisy features and sparse features), the performance of SGC degrades most drastically with the increase of noisy feature ratio (number of missing features). For instance, when the noise feature ratio reaches 40%, SGC decreases by about 20% on three citation datasets and about 10% on two coauthor datasets. The reason is that SGC employs no learnable parameters during feature convolution and relies heavily on the quality of feature. Also, the improvement of incorporating LaenNet into SGC is the highest. Although SGC generally performs worst among the three GCNs facing the highest noise, the performance of LaenNet enhanced SGC is comparable with and even beyond the other two. This demonstrates that taking labels as auxiliary features mitigates the influence of imperfect features.

The models' performance under noisy labels and sparse labels setting are shown in Table 4 and Table 5, respectively. Under the noisy labels setting, as the noisy label ratio γ increases, all GCNs collapse fast and reach nearly random results when noisy label ratio γ reaches 80%. While LaenNet generally and notably boosts GCNs. Specifically, LaenNet improves SGC on Cora by up to 9.62% when noisy label ratio $\gamma = 50\%$; improves GCN on Citeseer by up to 12.72% when $\gamma = 50\%$; improves GAT on Pubmed by up to 15.90% when $\gamma = 80\%$. In terms of sparse labels setting, LaenNet can enhance GCNs in most cases. Especially for Cora, three GCNs baselines are improved significantly on all the numbers of labeled nodes per class. Generally, sparse labels bring a great challenge for GCNs since it is insufficient to train a model with satisfactory discriminative ability. When the number of labeled nodes per class is 5, the label rate is extremely low, LaenNet can also improve GCN by up to 4.76% on Cora, which further illustrates the effectiveness of our method.

Time complexity analysis.

In this subsection, we analyze the time complexity of the proposed LaenNet, which is important for GCNs since real-word graphs are always large. Particularly, we take GCN as a reference, and compare the real running time of GCN+LaenNet, M3S and GCN-LPA on five datasets. Table 6 summarizes the running time of each single epoch for all methods. From the results we can see that, GCN is the fast and integrating with our method would slightly increase the running time. Compared with other two baselines, our method shows advanced scalability, especially on two large Coauthor datasets.

Visualization.

To explore the role of LaenNet in GCNs representation learning, we visualize the learned embedding. We take GCN and LaenNet enhanced GCN as an example and show the visualization results under four imperfect scenarios on Cora dataset. Specifically, to keep the visualization meaningful, we show the results of noisy features scenario with noise ratio as 50%, sparse features scenario with missing dimensions as 300, noisy labels scenario with noise ratio as 30% and sparse labels

Table 4
Experimental results under the noisy labels setting. We denote the results in bold if LaenNet achieves better performance than backbone GCNs.

	Noisy label ratio γ (%)									
Method	0	10	20	30	40	50	60	70	80	90
Cora										
M3S	80.30	78.93	72.90	71.60	60.10	56.20	40.30	29.00	14.00	14.87
GCN-LPA	80.63	76.43	68.80	69.60	55.60	50.67	39.00	42.60	16.10	19.73
GCN	81.08	77.01	73.73	66.22	59.92	48.14	43.71	27.59	16.41	9.81
GCN + LaenNet	**81.54**	**77.52**	**75.54**	**71.15**	**63.07**	**54.87**	**48.92**	**34.13**	**23.37**	**19.40**
SGC	80.15	74.98	70.37	61.08	53.91	45.38	34.86	28.23	18.74	11.97
SGC + LaenNet	**81.49**	**76.20**	**73.77**	**66.06**	**60.20**	**55.00**	**48.43**	**36.58**	**24.90**	**12.77**
GAT	82.80	77.10	74.20	65.60	48.90	54.50	51.90	**38.30**	23.00	18.70
GAT + LaenNet	**83.30**	**77.50**	**77.00**	**68.70**	**50.30**	**58.90**	**52.00**	38.00	**23.20**	**21.80**
Citeseer										
M3S	72.83	71.03	63.00	60.40	54.70	48.90	42.70	24.57	24.40	19.90
GCN-LPA	72.40	67.90	66.10	58.10	50.80	48.43	43.83	33.53	23.70	20.03
GCN	71.98	68.56	65.08	61.51	51.37	39.49	38.43	25.03	23.76	18.61
GCN + LaenNet	72.59	**72.32**	**69.80**	**62.94**	**59.50**	**52.21**	**43.54**	**33.80**	**25.10**	**20.76**
SGC	70.69	**69.26**	63.78	58.04	51.09	45.52	34.11	30.27	21.30	11.53
SGC + LaenNet	**71.44**	68.38	**64.10**	**60.68**	**55.59**	**50.25**	**42.81**	**31.51**	**21.78**	**19.11**
GAT	72.20	70.80	64.90	60.40	59.90	51.90	41.50	36.90	15.80	15.60
GAT + LaenNet	**72.30**	**71.30**	**66.10**	**61.50**	**60.50**	**55.20**	**43.40**	**37.40**	**23.70**	**23.90**
Pubmed										
M3S	80.20	77.27	68.70	65.80	58.90	46.77	41.40	34.67	29.20	30.67
GCN-LPA	77.43	76.10	70.90	65.23	55.63	55.23	39.50	39.20	33.77	24.90
GCN	77.56	73.08	68.41	60.96	56.80	46.57	38.82	35.34	30.79	25.42
GCN + LaenNet	**78.92**	**75.08**	**72.32**	**66.58**	**62.58**	**57.08**	**43.32**	**42.96**	**38.08**	**31.00**
SGC	77.84	72.79	67.26	61.81	54.21	50.63	38.90	27.38	19.34	11.55
SGC + LaenNet	**78.24**	**77.48**	**76.50**	**67.56**	**60.74**	**54.55**	**48.72**	**35.37**	**33.36**	**21.04**
GAT	77.20	72.30	69.90	**67.20**	**59.10**	46.10	40.30	36.40	24.70	27.80
GAT + LaenNet	**77.80**	**73.00**	**70.30**	65.70	58.60	**47.60**	**43.10**	**41.80**	**40.60**	**41.40**
Coauthor-CS										
M3S	90.87	89.97	89.80	88/90	83.50	70.30	63.50	49.00	28.10	10.30
GCN-LPA	91.20	90.90	90.70	88.00	83.40	81.90	72.50	52.83	25.90	13.70
GCN	90.40	90.10	89.93	87.80	79.70	77.10	63.40	56.20	27.90	13.40
GCN + LaenNet	**91.40**	**91.20**	**90.30**	**89.90**	**84.70**	**82.70**	**73.80**	**60.20**	**38.10**	**20.10**
SGC	91.60	**90.80**	**89.80**	**89.80**	82.30	80.00	76.00	59.90	43.30	20.50
SGC + LaenNet	**92.00**	88.40	88.30	88.30	**84.80**	**83.20**	**78.20**	**62.20**	**46.90**	**20.70**
Coauthor-Phy										
M3S	92.70	91.90	88.20	83.50	76.60	64.50	63.50	51.30	26.90	17.40
GCN-LPA	92.70	92.07	91.50	86.30	83.90	69.00	65.60	51.60	31.03	16.90
GCN	92.40	90.80	87.40	85.20	80.90	72.80	62.40	52.70	30.10	19.60
GCN + LaenNet	**92.90**	**93.20**	**91.00**	**87.80**	**82.60**	**75.10**	**65.20**	**56.50**	**36.60**	**21.90**
SGC	93.13	89.20	84.40	82.20	80.50	57.90	43.90	42.90	12.40	11.30
SGC + LaenNet	**94.30**	**92.40**	**90.20**	**89.00**	**83.40**	**76.50**	**61.60**	**48.60**	**32.70**	**29.60**

scenario with labeled nodes per-class as 5, respectively. We visualize the node representations extracted by the last layer of GCN and GCN+LaenNet by mapping them into a 2-D space through t-SNE (van der Maaten & Hinton, 2008). As can be seen from Figs. 4, 5, 6, 7, the node representations learned by GCN mix together, while LaenNet enhanced GCN presents clearer cluster distribution, which demonstrates the better discrimination of representation learned by LaenNet among different classes.

4.2. Comparing LaenNet against strong baselines (Q2)

To answer ***Q2***, we select 5 advanced GCNs for comparison, including Graph Sampling and Aggregation (GraphSAGE) (Hamilton, Ying, & Leskovec, 2017), Geometric Graph Convolutional Networks (Geom-GCN) (Pei et al., 2020), Jumping Knowledge Networks (JKNet) (Xu, Li, Tian, Sonobe, Kawarabayashi, & Jegelka, 2018), DropEdge (Rong et al., 2019) and GCNII (Chen, Wei et al., 2020). Specifically, Geom-GCN has three variants Geom-GCN-I, Geom-GCN-P and Geom-GCN-S. As suggested in Rong et al. (2019), we equip DropEdge with two backbones: JKNet and IncepGCN, and obtain JKNet (Drop) and Incep (Drop), respectively. Besides, we also take four methods which utilize label information to enhance GCNs, including APPNP (Klicpera et al., 2019), M3S (Sun et al., 2020), GCN-LPA (Wang & Leskovec, 2021) and C&S (Huang et al., 2021) as the baselines. For C&S, it has three variants C&S (Base Prediction), C&S (Autoscale) and C&S (FDiff-scale).

We validate the model performance on the original benchmark datasets. For the dataset split, we randomly split nodes of each class into 60%, 20%, and 20% for training, validation and testing following the setting in Pei et al. (2020). Experimental results are shown in Table 7. From the results, we can draw several conclusions as follows:

First, the methods utilizing label information to enhance GCNs

Table 5
Experimental results under the sparse labels setting. We denote the results in bold if LaenNet achieves better performance than backbone GCNs.

	Number of labeled nodes per-class			
Method	20	15	10	5
Cora				
M3S	80.30	79.97	77.60	73.23
GCN-LPA	80.63	77.13	74.94	71.30
GCN	81.08	79.26	77.17	70.31
GCN + LaenNet	**81.54**	**80.17**	**79.33**	**75.07**
SGC	80.15	78.34	75.04	67.96
SGC + LaenNet	**81.49**	**79.54**	**78.86**	**73.37**
GAT	82.80	81.40	80.80	73.40
GAT + LaenNet	**83.30**	**81.80**	**81.00**	**75.70**
Citeseer				
M3S	72.83	72.90	71.53	65.20
GCN-LPA	72.40	72.13	69.87	62.57
GCN	71.98	70.70	69.08	65.65
GCN + LaenNet	**72.59**	**72.51**	**71.75**	**67.16**
SGC	70.69	68.93	68.56	**64.30**
SGC + LaenNet	**71.44**	**69.51**	**69.00**	63.95
GAT	72.20	71.90	**68.00**	70.05
GAT + LaenNet	**72.30**	**72.80**	67.70	**70.50**
Pubmed				
M3S	80.20	75.30	74.36	71.80
GCN-LPA	77.43	74.70	73.50	70.90
GCN	77.56	76.36	73.51	70.28
GCN + LaenNet	**78.92**	**77.36**	**75.44**	**71.98**
SGC	77.84	75.39	73.29	67.36
SGC + LaenNet	**78.24**	**76.08**	**74.64**	**71.32**
GAT	77.20	78.30	**77.90**	71.70
GAT + LaenNet	**77.80**	**78.50**	77.80	**72.10**
Coauthor-CS				
M3S	90.87	90.70	90.40	89.20
GCN-LPA	91.20	90.67	90.70	87.70
GCN	90.40	90.10	89.70	87.60
GCN + LaenNet	**91.40**	**91.10**	**90.60**	**90.00**
SGC	91.60	91.40	90.90	88.00
SGC + LaenNet	**92.00**	**91.60**	**90.90**	**90.20**
Coauthor-Phy				
M3S	92.70	92.20	91.30	91.00
GCN-LPA	92.70	92.67	92.10	89.67
GCN	92.40	91.43	90.60	89.90
GCN + LaenNet	**92.90**	**93.90**	**93.40**	**91.70**
SGC	93.13	92.40	91.70	89.80
SGC + LaenNet	**94.30**	**93.70**	**92.80**	**92.80**

Table 6
Real running time comparison of GCN, GCN + LaenNet, M3S and GCN-LPA. We record the clock time of each single epoch for all methods.

Method	Cora	Citeseer	Pubmed	Coauthor-CS	Coauthor-Phy
GCN	0.03 s	0.05 s	0.12 s	0.35 s	0.65 s
GCN + LaenNet	0.10 s	0.09 s	0.82 s	0.75 s	1.45 s
M3S	0.17 s	0.11 s	1.52 s	1.20 s	2.75 s
GCN-LPA	0.87 s	0.09 s	0.86 s	0.81 s	1.67 s

consistently perform well, which collectively indicates that incorporating label information is helpful for GCNs. Second, compared with baselines, GCNs enhanced with LaenNet reach advanced performance, especially for GAT+LaenNet, which realizes new state-of-the-art results on Cora and Citeseer datasets. Finally, among the three datasets, the performance promotion for LaenNet on Pubmed is inferior to the other two. The possible reason may be that the number of categories of Pubmed is lesser (only 3), resulting in the limited label information gain.

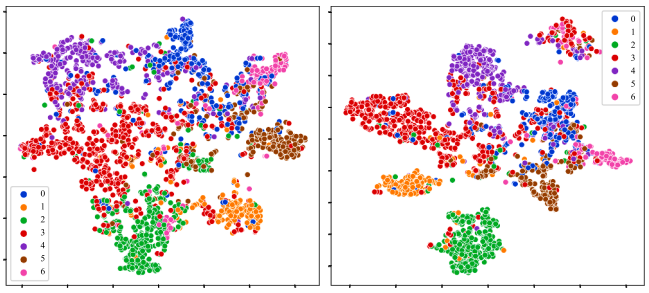


Fig. 5. A 2-D visualization for the node representations from GCN and GCN+LaenNet under sparse features scenario with missing dimensions as 300. Different colors of nodes denote different labels.

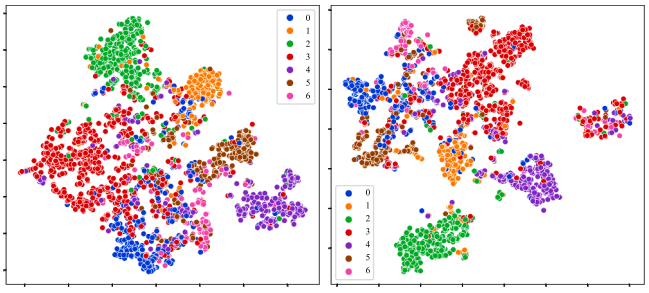


Fig. 6. A 2-D visualization for the node representations from GCN and GCN+LaenNet under noisy labels scenario with noise ratio as 30%. Different colors of nodes denote different labels.

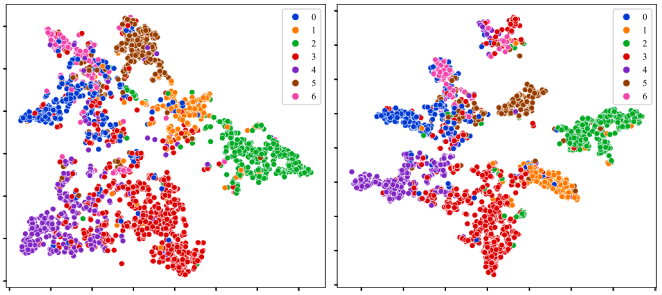


Fig. 7. A 2-D visualization for the node representations from GCN and GCN+LaenNet under sparse labels scenario with labeled nodes per-class as 5. Different colors of nodes denote different labels.

4.3. Impact of key hyper-parameters (Q3)

In this subsection, we study the impact of two key hyper-parameters of LaenNet: the GCNs' layer h that is integrated with our LaenNet and the confidence threshold Ψ that filter out labels. Particularly, we conduct experiments on the three citation network datasets with the generic data split setting used in Section 4.2.

Impact of adding LaenNet in different layer h of GCNs.

Our proposed LaenNet can be integrated into any hidden layer of GCNs, to explore its flexibility, we employ a 6-layer GCN model and add LaenNet into one of each shallow layer $h \in \{1, 2, 3, 4, 5\}$, named

Table 7
Experimental results on the original benchmark datasets. We compare the proposed LaenNet with advanced GCNs and label enhanced GCNs. The best results are indicated in boldface. We omit the results of Geom-GCN and GAT on two Coauthor datasets due to out of memory and denote them as "–".

	Method	Cora	Citeseer	Pubmed	Coauthor-CS	Coauthor-Phy
Advanced GCNs	GraphSAGE	86.80	75.20	84.70	92.60	94.50
	Geom-GCN-I	85.19	77.99	**90.05**	–	–
	Geom-GCN-P	84.93	75.14	88.09	–	–
	Geom-GCN-S	85.27	74.71	84.75	–	–
	JKNet	85.25	75.85	88.94	92.40	94.80
	JKNet (Drop)	87.46	75.96	89.45	92.65	95.13
	Incep (Drop)	86.86	76.83	89.18	92.14	95.07
	GCNII	88.49	77.08	89.57	91.98	95.12
Label enhanced GCNs	APPNP	87.87	76.53	89.40	92.31	94.97
	M3S	88.19	78.15	86.65	91.48	95.37
	GCN-LPA	88.50	78.70	87.80	**94.80**	**96.90**
	C&S (Base Prediction)	74.75	70.51	87.19	89.92	94.17
	C&S (Autoscale)	88.73	76.75	89.93	91.72	95.27
	C&S (FDiff-scale)	89.05	76.22	89.74	91.51	94.39
GCNs and LaenNet enhanced GCNs	GCN	87.29	77.05	85.92	92.42	95.29
	GCN + LaenNet	**90.49**	78.69	87.32	92.87	95.88
	SGC	87.45	75.63	83.28	92.62	95.00
	SGC + LaenNet	88.38	78.12	85.49	92.79	96.07
	GAT	88.26	78.38	88.18	–	–
	GAT + LaenNet	89.27	**80.29**	88.76	–	–

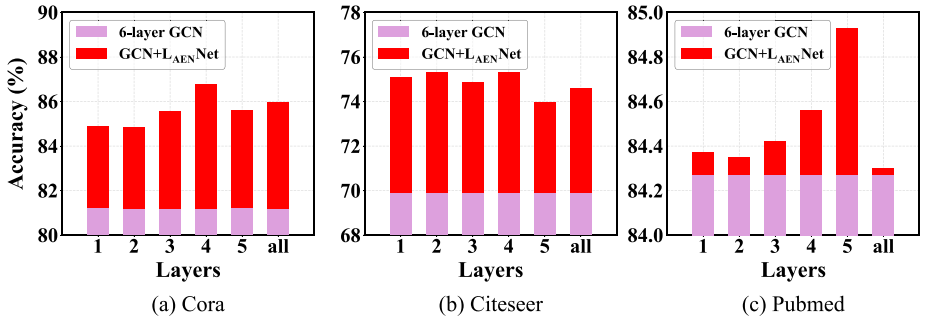


Fig. 8. Results of integrating LaenNet into different layers of a 6-layer GCN on Cora, Citeseer and Pubmed.

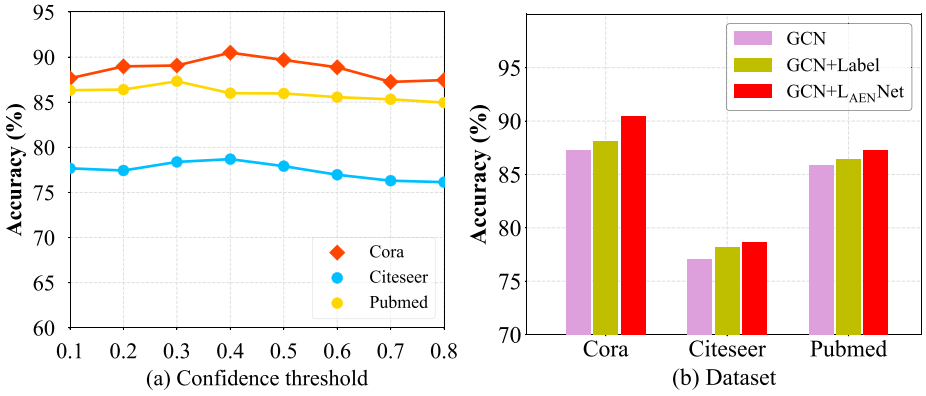


Fig. 9. (a) Impact of different confidence threshold. The horizontal axis is the value of confidence threshold and the vertical axis is the node classification accuracy. (b) Results of ablation study on Cora, Citeseer and Pubmed. To verify the effectiveness of supervision propagation, we simplify the model by integrating $\widetilde{\mathbf{Y}}$ with GCNs' hidden representation and obtain GCN+Label. Then, we show the performance of GCN, GCN+Label and GCN+LaenNet, respectively.

LaenNet_h. Besides, we integrate LaenNet into all layers of GCN, named LaenNet_all. Experimental results are shown in Fig. 8. From the results, we can see that all the layers enhanced by LaenNet can improve GCN significantly. Particularly, the performance is better when integrating LaenNet closer to the output layer.

Impact of confidence threshold Ψ.

We set the confidence threshold Ψ from 0.1 to 0.8 with an interval of 0.1 to validate its impact on performance. As shown in Fig. 9(a), the model's performance first increases and then slightly decreases. When the confidence threshold is small, almost all the label predictions would be used for neighborhood aggregation, which may introduce interference due to inaccurate predictions. However, if the confidence threshold

is set too large, most of the predictions will not be adopted and using zero vectors may hurt model's performance. Hence, a moderate confidence threshold, *e.g.,* 0.4 is desirable to achieve satisfactory performance.

4.4. Ablation studies (Q4)

To further explore LaenNet's capacity, we conduct an ablation study to investigate the contribution of integrating labels into GCNs. As claimed in Eq. (6), we first propagate the augmented prediction matrix $\widetilde{\mathbf{Y}}$ via graph and then concatenate the results to a latent feature layer of GCNs. To verify the effectiveness of supervision propagation, we construct a simplified model which integrates $\widetilde{\mathbf{Y}}$ into GCNs' hidden layer directly, that is, we modify the concatenation operation as,

$$\mathbf{Z}_h \leftarrow [\mathbf{Z}_h, \widetilde{\mathbf{Y}}], \quad h \in 1, \ldots, l-1 \tag{8}$$

Specifically, we take GCN as the backbone and obtain the simplified model GCN+Label. Experimental results are shown in Fig. 9(b).

As we can see, GCN+Label also improves GCN consistently on three datasets, which demonstrates the effectiveness of introducing labels into GCNs once more. However, the performance of GCN+Label is inferior to LaenNet. In the label matrix, each label only depicts the label information of a single node. As the labels of neighbor nodes contain relevant information about the target node, it is meaningful to spread label information on graph before integrating into GCNs, which is also supported by a common assumption in semi-supervised graph learning, *i.e.,* nodes that are close to each other are more likely to share a label.

5. Conclusion

In this paper, we explore how to improve the robustness of GCNs under imperfect graph data scenarios. Our basic idea is to take label as auxiliary features to participate in message passing, leading to an enhanced neighborhood aggregation schema. Particularly, we present a simple-yet-effective LaenNet architecture, which can be added to any hidden layer of GCNs. To verify the effectiveness of LaenNet, we instantiate it with three representative GCNs and validate it by semi-supervised node classification under four challenging noisy and sparse graph data scenarios, including noisy features, sparse features, noisy labels and sparse labels, and the generic scenario. Thorough experiments and analysis show the effectiveness and flexibility of the proposed method. As a lightweight framework, LaenNet can be easily generalized to the variants of GCNs, which takes little extra time cost according to time complexity analysis.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request

Acknowledgments

This work was supported by the National Key R&D Program of China under Grant Nos. 2021ZD0112501 and 2021ZD0112502; the National Natural Science Foundation of China under Grant Nos. U22A 2098, 62172185, 62206105 and 62202200.

References

Atwood, J., & Towsley, D. (2016). Diffusion-convolutional neural networks. In *Advances in neural information processing systems* (pp. 1993–2001).

Bianchi, F. M., Grattarola, D., Livi, L., & Alippi, C. (2022). Graph neural networks with convolutional arma filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 44*(7), 3496–3507.

Bruna, J., Zaremba, W., Szlam, A., & LeCun, Y. (2014). Spectral networks and deep locally connected networks on graphs. In *International conference on learning representations*.

Carlson, A., Betteridge, J., Kisiel, B., Settles, B., Hruschka, E. R., & Mitchell, T. M. (2010). Toward an architecture for never-ending language learning. In *AAAI conference on artificial intelligence*.

Chen, X., Chen, S., Yao, J., Zheng, H., Zhang, Y., & Tsang, I. W. (2020). Learning on attribute-missing graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 44*(2), 740–757.

Chen, M., Wei, Z., Huang, Z., Ding, B., & Li, Y. (2020). Simple and Deep Graph Convolutional Networks. In *International conference on machine learning* (pp. 1725–1735).

Dai, E., Aggarwal, C., & Wang, S. (2021). NRGNN: Learning a label noise resistant graph neural network on sparsely and noisily labeled graphs. In *KDD* (pp. 227–236).

Dai, E., Zhao, T., Zhu, H., Xu, J., Guo, Z., Liu, H., et al. (2022). *A comprehensive survey on trustworthy graph neural networks: Privacy, robustness, fairness, and explainability*. arXiv preprint arXiv:2204.08570.

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

Entezari, N., Al-Sayouri, S. A., Darvishzadeh, A., & Papalexakis, E. E. (2020). All you need is low (rank) defending against adversarial attacks on graphs. In *Proceedings of the 13th international conference on web search and data mining* (pp. 169–177).

Errica, F., Podda, M., Bacciu, D., & Micheli, A. (2020). A Fair Comparison of Graph Neural Networks for Graph Classification. In *International conference on learning representations*.

Fan, W., Liu, X., Jin, W., Zhao, X., Tang, J., & Li, Q. (2022). Graph Trend Networks for Recommendations. In *International ACM SIGIR conference on research and development in information retrieval*.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., & Dahl, G. E. (2017). Neural message passing for quantum chemistry. In *International conference on machine learning* (pp. 1263–1272). PMLR.

Guo, S., Lin, Y., Feng, N., Song, C., & Wan, H. (2019). Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In *AAAI conference on artificial intelligence,Vol. 33,no. 01* (pp. 922–929).

Guo, S., Lin, Y., Wan, H., Li, X., & Cong, G. (2021). Learning dynamics and heterogeneity of spatial-temporal graph data for traffic forecasting. *IEEE Transactions on Knowledge and Data Engineering*.

Hamilton, W. L., Ying, R., & Leskovec, J. (2017). Inductive representation learning on large graphs. In *Advances in neural information processing systems* (pp. 1025–1035).

Hamilton, W. L., Ying, Z., & Leskovec, J. (2017). Inductive representation learning on large graphs. In *Advances in neural information processing systems* (pp. 1024–1034).

Hao, Z., Lu, C., Huang, Z., Wang, H., Hu, Z., Liu, Q., et al. (2020). ASGN: An active semi-supervised graph neural network for molecular property prediction. In *ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 731–752).

Hu, X., Tang, L., Tang, J., & Liu, H. (2013). Exploiting social relations for sentiment analysis in microblogging. In *ACM international conference on web search and data mining* (pp. 537–546).

Huang, Q., He, H., Singh, A., Lim, S., & Benson, A. R. (2021). Combining Label Propagation and Simple Models out-performs Graph Neural Networks. In *International conference on learning representations*.

Jin, W., Ma, Y., Liu, X., Tang, X., Wang, S., & Tang, J. (2020). Graph structure learning for robust graph neural networks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 66–74).

Kipf, T. N., & Welling, M. (2016). Variational graph auto-encoders. In *Advances in neural information processing systems workshop on bayesian deep learning*.

Kipf, T. N., & Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In *International conference on learning representations*.

Klicpera, J., Bojchevski, A., & Günnemann, S. (2019). Predict then Propagate: Graph Neural Networks meet Personalized PageRank. In *International conference on learning representations*.

Kollias, G., Kalantzis, V., Idé, T., Lozano, A., & Abe, N. (2022). Directed Graph Auto-Encoders. In *AAAI conference on artificial intelligence*.

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

Li, R., Wang, S., & Chen-Chuan, K. (2012). Multiple location profiling for users and relationships from social network and content. *Proceedings of the VLDB Endowment, 5*(11).

Li, Y., Yu, R., Shahabi, C., & Liu, Y. (2018). Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. In *International conference on learning representations*.

Liu, Q., Allamanis, M., Brockschmidt, M., & Gaunt, A. (2018). Constrained graph variational autoencoders for molecule design. *Advances in Neural Information Processing Systems, 31*.

Liu, X., Jin, W., Ma, Y., Li, Y., Liu, H., Wang, Y., et al. (2021). Elastic graph neural networks. In *International conference on machine learning* (pp. 6837–6849). PMLR.

Luo, D., Cheng, W., Yu, W., Zong, B., Ni, J., Chen, H., et al. (2021). Learning to drop: Robust graph neural network via topological denoising. In *Proceedings of the 14th ACM international conference on web search and data mining* (pp. 779–787).

Monti, F., Bronstein, M., & Bresson, X. (2017). Geometric matrix completion with recurrent multi-graph neural networks. In *Advances in neural information processing systems* (pp. 3697–3707).

Namata, G., London, B., Getoor, L., Huang, B., & EDU, U. (2012). Query-driven active surveying for collective classification. In *International workshop on mining and learning with graphs,Vol. 8*.

Pan, S., Hu, R., Long, G., Jiang, J., Yao, L., & Zhang, C. (2018). Adversarially regularized graph autoencoder for graph embedding. In *International joint conference on artificial intelligence* (pp. 2609–2615).

Pei, H., Wei, B., Chang, K. C.-C., Lei, Y., & Yang, B. (2020). Geom-gcn: Geometric graph convolutional networks. In *International conference on learning representations*.

Petar, V., Guillem, C., Arantxa, C., Adriana, R., Pietro, L., & Yoshua, B. (2018). Graph attention networks. In *International conference on learning representations*.

Raghavan, U. N., Albert, R., & Kumara, S. (2007). Near linear time algorithm to detect community structures in large-scale networks. *Physical Review E, 76*(3), Article 036106.

Rong, Y., Huang, W., Xu, T., & Huang, J. (2019). Dropedge: Towards deep graph convolutional networks on node classification. In *International conference on learning representations*.

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

Seo, Y., Defferrard, M., Vandergheynst, P., & Bresson, X. (2018). Structured sequence modeling with graph convolutional recurrent networks. In *International conference on neural information processing* (pp. 362–373).

Shchur, O., Mumme, M., Bojchevski, A., & Günnemann, S. (2018). Pitfalls of graph neural network evaluation. In *Advances in neural information processing systems workshop*.

Shi, Y., Huang, Z., Feng, S., Zhong, H., Wang, W., & Sun, Y. (2021). Masked label prediction: Unified message passing model for semi-supervised classification. In *International joint conference on artificial intelligence* (pp. 1548–1554).

Sun, K., Lin, Z., & Zhu, Z. (2020). Multi-Stage Self-Supervised Learning for Graph Convolutional Networks on Graphs with Few Labeled Nodes. In *AAAI conference on artificial intelligence* (pp. 5892–5899).

Taguchi, H., Liu, X., & Murata, T. (2021). Graph convolutional networks for graphs containing missing features. *Future Generation Computer Systems, 117*, 155–168.

Tibshirani, R. (1996). Regression shrinkage and selection via the Lasso. *Journal of the Royal Statistical Society. Series B. Statistical Methodology, 58*(1), 267–288.

van der Maaten, L., & Hinton, G. (2008). Visualizing data using t-SNE. *Journal of Machine Learning Research, 9*(86), 2579–2605.

Wang, X., He, X., Cao, Y., Liu, M., & Chua, T.-S. (2019). Kgat: Knowledge graph attention network for recommendation. In *ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 950–958).

Wang, H., & Leskovec, J. (2021). Combining graph convolutional neural networks and label propagation. *ACM Transactions on Information Systems (TOIS), 40*(4), 1–27.

Wang, W., Wei, X., Suo, X., Wang, B., Wang, H., Dai, H.-N., et al. (2021). Hgate: Heterogeneous graph attention auto-encoders. *IEEE Transactions on Knowledge and Data Engineering*, (01), 1.

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

Wu, H., Wang, C., Tyshetskiy, Y., Docherty, A., Lu, K., & Zhu, L. (2019). Adversarial examples for graph data: deep insights into attack and defense. In *Proceedings of the 28th international joint conference on artificial intelligence* (pp. 4816–4823).

Wu, F., Zhang, T., Souza Jr, A. H. d., Fifty, C., Yu, T., & Weinberger, K. Q. (2019). Simplifying graph convolutional networks. In *International conference on machine learning*.

Xu, K., Hu, W., Leskovec, J., & Jegelka, S. (2019). How powerful are graph neural networks?. In *International conference on learning representations*.

Xu, K., Li, C., Tian, Y., Sonobe, T., Kawarabayashi, K., & Jegelka, S. (2018). Representation learning on graphs with jumping knowledge networks. In *International conference on machine learning* (pp. 5449–5458).

Yang, Z., Cohen, W. W., & Salakhutdinov, R. (2016). Revisiting Semi-Supervised Learning with Graph Embeddings. In *International conference on machine learning* (pp. 40–48).

Ying, R., He, R., Chen, K., Eksombatchai, P., Hamilton, W. L., & Leskovec, J. (2018). Graph convolutional neural networks for web-scale recommender systems. In *ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 974–983).

Ying, Z., You, J., Morris, C., Ren, X., Hamilton, W., & Leskovec, J. (2018). Hierarchical graph representation learning with differentiable pooling. In *Advances in neural information processing systems* (pp. 4805–4815).

Yu, B., Yin, H., & Zhu, Z. (2018). Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. In *International joint conference on artificial intelligence* (pp. 3634–3640).

Zhang, M., & Chen, Y. (2018). Link prediction based on graph neural networks. In *Advances in neural information processing systems* (pp. 5171–5181).

Zhang, Z., Cui, P., & Zhu, W. (2020). Deep learning on graphs: A survey. *IEEE Transactions on Knowledge and Data Engineering, 34*(1), 249–270.

Zhang, X., & Zitnik, M. (2020). Gnnguard: Defending graph neural networks against adversarial attacks. *Advances in Neural Information Processing Systems, 33*, 9263–9275.

Zhou, D., Bousquet, O., Lal, T. N., Weston, J., & Schölkopf, B. (2004). Learning with local and global consistency. In *Advances in neural information processing systems* (pp. 321–328).

Zhu, H., & Koniusz, P. (2020). Simple spectral graph convolution. In *International conference on learning representations*.