Variational Inference for Training Graph Neural Networks in Low-Data Regime through Joint Structure-Label Estimation

Conference Paper · August 2022

DOI: 10.1145/3534678.3539283

CITATIONS
0

READS
11

4 authors, including:

Qitian Wu
Shanghai Jiao Tong University
30 PUBLICATIONS **210** CITATIONS

SEE PROFILE

Junchi Yan
Shanghai Jiao Tong University
270 PUBLICATIONS **5,961** CITATIONS

SEE PROFILE

Variational Inference for Training Graph Neural Networks in Low-Data Regime through Joint Structure-Label Estimation

Danning Lao†
Shanghai Jiao Tong University
Shanghai, China
sjtuldn@sjtu.edu.cn

Xinyu Yang†
Shanghai Jiao Tong University
Shanghai, China
yang_xinyu@sjtu.edu.cn

Qitian Wu
Shanghai Jiao Tong University
Shanghai, China
echo740@sjtu.edu.cn

Junchi Yan*
Shanghai Jiao Tong University
Shanghai, China
yanjunchi@sjtu.edu.cn

ABSTRACT

Graph Neural Networks (GNNs) are one of the prominent methods to solve semi-supervised learning on graphs. However, most of the existing GNN models often need sufficient observed data to allow for effective learning and generalization. In real-world scenarios where complete input graph structure and sufficient node labels might not be achieved easily, GNN models would encounter with severe performance degradation. To address this problem, we propose **WSGNN**[1], short for weakly-supervised graph neural network. WSGNN is a flexible probabilistic generative framework which harnesses variational inference approach to solve graph semi-supervised learning in a label-structure joint estimation manner. It collaboratively learns task-related new graph structure and node representations through a two-branch network, and targets a composite variational objective derived from underlying data generation distribution concerning the inter-dependence between scarce observed data and massive missing data. Especially, under weakly-supervised low-data regime where labeled nodes and observed edges are both very limited, extensive experimental results on node classification and link prediction over common benchmarks demonstrate the state-of-the-art performance of WSGNN over strong competitors. Concretely, when only 1 label per class and 1% edges are observed on Cora, WSGNN maintains a decent 52.00% classification accuracy, exceeding GCN by 75.6%.

CCS CONCEPTS

• **Computing methodologies** → **Semi-supervised learning settings**; *Bayesian network models.*

KEYWORDS

Graph Neural Networks, Semi-Supervised Learning on Graphs, Label-Efficient Learning, Variational Inference

ACM Reference Format:
Danning Lao, Xinyu Yang, Qitian Wu, and Junchi Yan. 2022. Variational Inference for Training Graph Neural Networks in Low-Data Regime through Joint Structure-Label Estimation. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '22), August 14–18, 2022, Washington, DC, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3534678.3539283

1 INTRODUCTION

Applying graph neural networks (GNNs) to process graph-structured data have become a common practice due to their strong capacity and flexibility for handling different graph-related tasks, such as node-level prediction [15, 19, 31] and edge-level prediction [5, 18, 36]. However, there are still two prominent obstacles that limit GNNs for practical applications in real situations.

First, most existing works assume the graph structure is well-established and given, yet this condition could often be violated in real-world scenarios. For example, in citation networks, implicit mutual influence may still exists between papers though they are not declared with citation explicitly [22]. In this case, potential links should be considered instead of formulating the network under the assumption of perfect input structure. Second, current GNNs and other graph representation models often require a large number of node labels which can be costly to obtain from handcrafted efforts and even unrealistic in many cases due to practical constraints [16, 28, 40]. For instance, at the early stage of an infectious disease when confirmed cases are rare, observed data that can be used as supervision can be very limited [2]. To make matters worse, the shortage of link observation and node labeling may coincide in real situations [29], which could exaggerate the deficiency of existing models, especially for those that heavily rely on the explicit and complete graph structure as well as adequate node supervision.

Zooming in on the problems stated above, we frame such a setting as *graph-based semi-supervised learning in low-data regime*, and in Fig. 1, we provide empirical results to illuminate the concerning deficiency of common GNNs. In spite of this observation, such a

*The first two authors contributed equally. Junchi Yan is the correspondence author, who is also with MoE Key Lab of Artificial Intelligence, Shanghai Jiao Tong University, and Shanghai AI Lab. This work was partly supported by National Key Research and Development Program of China (2020AAA0107600), Shanghai Municipal Science and Technology Major Project (2021SHZDZX0102), and NSFC (61972250, 72061127003).

KDD '22, August 14–18, 2022, Washington, DC, USA

ACM ISBN 978-1-4503-9385-0/22/08…$15.00
https://doi.org/10.1145/3534678.3539283

[1]Source code (PyTorch) of WSGNN can be found at https://github.com/Thinklab-SJTU/WSGNN.

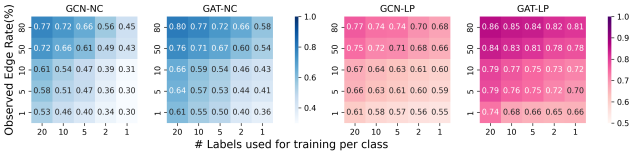


Figure 1: Vanilla GNNs (GCN [19], GAT [31])'s performance degrades notably under low-data regime, namely low rate of observed edges/node labels. Blue: Node classification (NC) accuracy. Red: Link prediction (LP) ROC-AUC score.

problem in fact remains under-explored except for very few recent works [21, 33] (they will be detailedly discussed later in Sec. 2).

On one hand, since the initial unreliable structure may introduce biased modeling, we aim at building a principled approach that unifies the structure inference and representation learning. On the other hand, given very limited supervision at hand, we resort to a strong inference mechanism that exploits scarce observed information and explores massive unsupervised data. To these ends, we devise a probabilistic generative framework based on underlying data-generating process from a bottom-up perspective, and harness variational inference technique to derive a tractable objective for semi-supervised learning on graphs with joint inference on node labels and graph structure. We call our approach **WSGNN** that entails a generative model and recognition model.

More specifically, the generative model assumes graph structure learners and GNN networks as subsequent modules, organized as a two-branch model architecture that can collectively refine node embeddings and latent structure in a feature-structure inter-dependent manner. The graph learners aim at inferring new structure through weighted cosine similarity metrics with node embeddings as input. The GNN networks are derived from basic models e.g. GCN [19] for aggregating and propagating the feature information along the inferred structural path.

Furthermore, the recognition model seeks for inference on missing information in a collaborative manner through variational inference. We model the known information including the available node labels and the initial structure as observed variables, conditioned on which the latent information including the potential links and the unobserved labels will be jointly inferred as missing variables. We further use the derived Evidence Lower Bound as a tractable objective that unifies the learning and inference tasks.

Empirically, we show the practical efficacy of the proposed model on extensive experimental settings over real-world datasets. In particular, when only 1 label per class and 1% edges are observed on Cora [25] dataset, our model WSGNN maintains a decent 52.00% classification accuracy, with a large improvement 75.6% over GCN [19]. Besides, we further show its practical efficacy in a real-world disease propagation dataset where the limited supervision is often the case in practice. The highlights of this paper are:

1) Aspect: We focus on one of the pain points of current GNNs for semi-supervised learning on graphs with very limited supervision. We probe into the intersection of node classification and link prediction with both limited observed edges and labeled nodes.

2) Method: We propose a probabilistic generative framework and leverage variational inference to derive a tractable objective, unifying the learning and inference over graph-structured data. By jointly optimizing the graph structure and node embeddings, the model can exploit the inter-dependence between nodes and edges.

3) Evaluation: We empirically show that WSGNN helps to improve the generalization performance of backbone GNNs on public datasets for node classification and link prediction, under (extreme) low-data regime as well as regular setting, outperforming strong competitors by a large margin.

2 RELATED WORK

Graph neural networks. Recent years have witnessed boosted research on GNNs revolutionizing wide graph-based learning problems [15, 19, 37, 40]. A typical schema of GNN is to iteratively update node representations by aggregating neighboring information according to graph structure [14, 39]. Critical flaws of GNNS exist. For example, GNNs are known error-prone to incomplete structure [8]. Unreliable information e.g. imperfect graph structure will exacerbate the representation quality and have cascading effects. For similar reason, existing solutions suffer from performance degradation given limited training data, not to mention label-structure both missing regime. Few work has explored this scenario, which is the focus of this work.

Graph semi-supervised representation learning. Considerable literature has arisen around the central theme of graph-based semi-supervised learning (SSL) problems [26, 44], which aims to train models with potential to fit labelled data that is partially given and infer the corresponding label information of unlabelled data. Shallow methods utilize label propagation, graph regularization [3, 4], graph laplacian [43] and so on. And ample deep networks [11, 34] include generative models based on poisson learning [6, 33], variational principles [23, 24, 38] and so on. However, the regime of extremely limited supervision on both node and edges is still unexplored and our focus.

Graph structural learning. Graph structural learning [45] targets learning an optimized graph structure to improve the quality of down-stream tasks. Most of the GNN-based graph structural learning methods focus on enhancing the robustness of the model. Typical solutions include bayesian methods [10, 41], bi-level optimization methods [13], graph regulation methods [17, 42]. A recent work VIB-GSL [27] derives the structure learning model from variational information bottleneck principle [1], with the aim of improving model robustness under noisy information and structure perturbations. However, these models require abundant labeled information for learning accurate structure while our work focuses on weak supervision. Besides, our work leverages the learned structure as not only computation graphs for message passing, but also potential links between entities to exploit information from massive missing data.

Comparison with existing low-data-regime-oriented models. Here we further compare our method with two semi-supervised graph models that are tailored for low-data regime [21, 33].

- CGPN [33] vs. WSGNN. CGPN instantiates the generation and inference distribution with a graph poisson network and a GNN model respectively. To train the two models, ELBO loss, supervise loss and an extra contrastive loss are utilized. The differences are two-fold. i) Our WSGNN collaboratively infers the missing links and node labels, instead of merely inference on labels as missing

data like CGPN. Hence, WSGNN can handle more general settings with both low label rate and incomplete graph structure, which is also more difficult. iii) WSGNN harnesses latent structure learning as a critical component for modeling the inter-dependence between observed and unobserved data, while CGPN ignores the information from unobserved edges and only focuses on message passing over input structure.

- Shoestring [21] vs. WSGNN. Shoestring incorporates a graph embedding network and a metric learning network to perform centroid clustering according to similarity of the learned representations. Unlike our variational inference based method, the effectiveness of centroid clustering largely depends on the correctness of class centroids, directly related to quality of existing labels. Therefore, WSGNN is expected to be more effective and robust. Besides, Shoestring does not take graph structure as prediction task either, not to mention the joint learning task.

3 PROBLEM SETUP

We start with formally introducing the problem of graph-based semi-supervised learning. An undirected and unweighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is given with $n = |\mathcal{V}|$ nodes and $m = |\mathcal{E}|$ edges.

Node-Level SSL. The node set $\mathcal{V} = \{v_1, v_2, ...v_n\}$ represents the data samples $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$, where $\boldsymbol{x_i} \in \mathbb{R}^d$ and $\boldsymbol{y_i} \in \mathbb{R}^k$ are the feature and label vectors of node i respectively. We denote $\boldsymbol{X} \in \mathbb{R}^{n\times d}$ as feature matrix with the i-th row formed by the feature vector x_i and $\boldsymbol{Y} \in \mathbb{R}^{n\times k}$ as label matrix with its (i, j)-th element $Y_{i,j} = 1$ if v_i belongs to the j-th class and 0 otherwise.

On top of this, a typical semi-supervised node-level problem is formulated as follows. $\mathcal{V}$ is composed of the labeled set $\mathcal{V}_l = \{v_1, v_2, ...v_{n_l}\}$ where node labels $\boldsymbol{Y}_l = \{y_i\}_{i=1}^{n_l}$ are provided and the unlabelled set $\mathcal{V}_u = \{v_{n_l+1}, ...v_n\}$ where unobserved labels $\boldsymbol{Y}_u = \{y_i\}_{i=n_l+1}^{n}$ are to be predicted. $\boldsymbol{Y} = \boldsymbol{Y}_l \bigcup \boldsymbol{Y}_u$. And in low data regime, $n_l = |\mathcal{V}_l| \ll n_u = |\mathcal{V}_u|$.

Edge-Level SSL. The adjacency matrix of $\mathcal{G}$ is denoted as $\boldsymbol{A}$, where $\boldsymbol{A}_{i,j} = 1$ means that the edge $e_{i,j}$ exists in edge set $\mathcal{E}$. However, in the semi-supervised scenario, one only observes incomplete edges $\mathcal{E}_{obs} \subseteq \mathcal{E}$ and needs to predict the missing links $\mathcal{E}_{miss}$ in $\mathcal{E} \setminus \mathcal{E}_{obs}$. Furthermore, our work aims at the extreme low-data regime, where $|\mathcal{E}_{obs}| \ll |\mathcal{E}_{miss}|$. Similarly, there is $\boldsymbol{A} = \boldsymbol{A}_{obs} \bigcup \boldsymbol{A}_{miss}$.

In a word, the graph-based semi-supervised learning problem assumes the label matrix and adjacency matrix as

Given a dataset with incompletely observed graph $\mathcal{G} = (\boldsymbol{X}, \boldsymbol{Y_l}, \boldsymbol{A}_{obs})$, the goal is to infer the missing labels $\boldsymbol{Y}_u$ or missing links $\boldsymbol{A}_{miss}$.

4 METHODOLOGY

In this section, we first derive the formulation of WSGNN utilizing variational inference approach. Next we present the details of WSGNN's components. Lastly, we illustrate the training procedure.

4.1 Probabilistic Bayesian Model

We propose a flexible generative framework that characterizes the data generation of node features **X**, node labels **Y** and graph adjacency **A**. Y_u, A_{miss} are unknown latent variables, and Y_l, A_{obs} are observed ones.

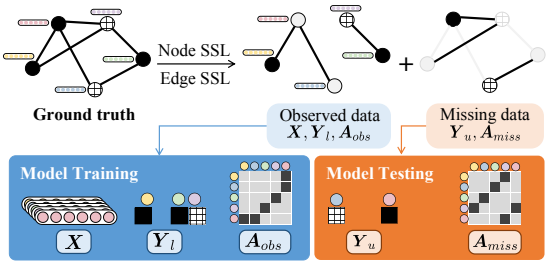


Figure 2: Problem setup illustration. The training information is limited with partially observed graph structure and scarce labeled nodes. In experiment, we mask the ground truth and use the missing data for model evaluation.

4.1.1 Generative Model. We commence by modeling joint distribution $p(X, Y, A)$. Assuming $P(X)$ as empirical data distribution for node features, we can factorize the joint distribution as:

$$\begin{aligned} P(X, Y, A) &= P(A, Y|X)P(X) \\ &= P(A_{miss}, A_{obs}, Y_u, Y_l|X)P(X), \end{aligned}$$

where the conditional distribution $P(A_{miss}, A_{obs}, Y_u, Y_l|X)$ can be modeled by some flexible parametric families of distribution $p_\theta(A_{miss}, A_{obs}, Y_u, Y_l|X)$. For generality, we do not specify the distribution form of $P(X)$ and everything will be conditioned on X later in this paper. We assume that for the joint distribution $P(A_{miss}, A_{obs}, Y_u, Y_l|X)$, its PMF is differentiable almost everywhere w.r.t. both θ and latent variables (Y_u, A_{miss}). Unfortunately, the true generative parameter θ^* and the values of the latent variables are unknown.

In our case, the computation of the marginal distribution

$$p_\theta(A_{obs}, Y_l|X) = \int p_\theta(A_{miss}, A_{obs}, Y_u, Y_l|X)d(A_{miss}, Y_u).$$

is intractable due to the integration over high-dimension variable, and thereby for the conditional distribution of unobserved variables

$$p_\theta(A_{miss}, Y_u|A_{obs}, Y_l, X) = \frac{p_\theta(A_{miss}, Y_u, A_{obs}, Y_l|X)}{p_\theta(A_{obs}, Y_l|X)},$$

which is obtain by Bayes' rule, is also intractable for computation.

4.1.2 Inference/Recognition Model. Since the conditional distribution for latent variables $p_\theta(A_{miss}, Y_u|A_{obs}, Y_l, X)$ is intractable for computation, to infer the missing labels Y_u and missing structure A_{miss}, we harness variational inference approach to approximate the posterior $p_\theta(A_{miss}, Y_u|A_{obs}, Y_l, X)$ by introducing a recognition model q_Φ, representing a variational distribution $q_\Phi(A_{miss}, Y_u|A_{obs}, Y_l, X)$. Its parameter Φ will be jointly optimized with the generative model's parameter θ. Specifically, we can derive the Evidence Lower BOund (ELBO) for observed data log-likelihood as:

$$\begin{aligned} &\log p_\theta(A_{obs}, Y_l|X) \\ \geq &\mathcal{L}_{ELBO} = \mathbb{E}_{q_\Phi(A_{miss}, Y_u|A_{obs}, Y_l, X)}[-\log q_\Phi(A_{miss}, Y_u|A_{obs}, Y_l, X) \\ &+ \log p_\theta(A_{miss}, Y_u, A_{obs}, Y_l|X)]. \end{aligned} \tag{1}$$

Practically, by instantiating $q_\Phi(A_{miss}, Y_u|A_{obs}, Y_l, X)$ as a deep network q_Φ model and $p_\theta(A_{miss}, Y_u, A_{obs}, Y_l|X)$ as another network p_θ, we target the optimal parameters θ, Φ via

$$\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\phi}} = \arg\min_{\boldsymbol{\theta}, \boldsymbol{\phi}} \; -\mathcal{L}_{\text{ELBO}}.$$

By introducing $\mathcal{D}_{KL}(\cdot||\cdot)$, the Kullback-Leibler divergence between two distributions, we can find (see Sec. A.1 of Appendix for derivation):

$$\arg\min_{\boldsymbol{\theta}, \boldsymbol{\Phi}} \; -\mathcal{L}_{\text{ELBO}} = \arg\min_{\boldsymbol{\theta}, \boldsymbol{\Phi}} \mathcal{D}_{KL}(q_\Phi, p_\theta). \tag{2}$$

which illuminates the fact that the optimal model parameters successfully achieve our goal of minimizing the divergence between the variational distribution $q_\Phi(A_{miss}, Y_u|A_{obs}, Y_l, X)$ and the intractable posterior distribution $p_\theta(A_{miss}, Y_u|A_{obs}, Y_l, X)$. The reformulation of loss in code implementation can be found in Appendix.

4.2 WSGNN Instantiations

For the inference model q_Φ, we are making a further approximation from $q_\Phi(A_{miss}, Y_u|A_{obs}, Y_l, X)$ to $q_\Phi(A_{miss}, Y_u|A_{obs}, X)$. This approximation is known as the mean-field method, which is commonly used in variational inference based methods [23]. We specify it as a network that takes (X, A_{obs}) as input and outputs estimations $\hat{Y}$ and $\hat{A}$. For the generative model $p_\theta(A_{miss}, A_{obs}, Y_u, Y_l|X)$, we also adopt the same architecture design except for the input which is only node feature X. We believe such a similar structure can help bridge the divergence between q_Φ and p_θ. The details are presented as follows.

4.2.1 Label-Structure Joint Inference. In this section we present the big picture of how inference is achieved with a label-structure joint learning process. The two-branch network is used as instantiations for both p_θ and q_Φ, except for some slight differences that A_{obs} is not utilized in p model. It meets our assumption that dependencies exist in label and structure during data generation phase. The overall framework is given in Fig. 3.

Two-branch global architecture. Without loss of generality, we do not assume certain directional dependence between A_{miss} and Y_u, and instead resort to inference on them through a two-branch model architecture that combines the two inference paths $(A_{obs}, X) \rightarrow A_{miss} \rightarrow Y_u$ and $(A_{obs}, X) \rightarrow Y_u \rightarrow A_{miss}$, as shown in the left part of Fig. 3 (the BranchAZ and BranchZA).

Each of the branches is composed of a graph neural network module to update node representations based on information aggregation along graph structure, and a graph learner module to update the weight of connectivity based on similarities of node representations. Within each branch, we consider iterative updating for node representations and graph structure according to the important hypothesis of joint learning: better graph structure can lead to better node representations (BranchAZ) and better node representations can also lead to better graph structure (BranchZA). The order of GNN module and the graph learner also follows the hypothesis.

Calibration for structure inference. On BranchZA, the first-layer representation $Z^{(2)}$ learned from A_{obs} may not be sufficient for inferring the graph structure A_{miss} since the observed graph is partially observed and the learned representations based on that could also contain insufficient information. Therefore, we leverage the representation $Z^{(1)}$ given by BranchAZ as supplementary information, which is learned from a more complete latent structure A'. And this is noted in Fig. 3 as the regularization.

Final Estimations Output. The final estimation comes from averaging the learned results of two branches as shown in the middle part of Fig. 3. Therefore we get $\hat{A}$ and $\hat{Y}$, probability vectors for graph adjacency, and for node classification.

4.2.2 Instantiation of representation learning with graph neural network. We introduce our implementation for the GNN module that takes $\boldsymbol{A}$ and $\boldsymbol{Z}$ as inputs and outputs the learned representation $\boldsymbol{Z'}$ to be further processed as probability of $\hat{\boldsymbol{Y}}$. Most of the recent GNNs can be incorporated into our framework, such as GCN [19], GAT [31], APPNP [20]. We choose APPNP as our backbone for most of our later experiments. The variants with other GNN backbones can also be explored. Note that for p model instantiation, the GNN module in BranchZA should be replaced with a MLP module as shown in Fig. 3 since A_{obs} is not available for distribution p.

4.2.3 Instantiation of structure learning with graph learner. As for the graph structure learner which takes adjacency matrix A and representation matrix Z as input and outputs a new structure denoted by A', we involve a two-step process. The first step follows the multi-head weighted cosine similarity learning method previously used by [8], while the second step incorporates the information of the original structure input. Fig. 4 shows the module's details.

We denote $\{\boldsymbol{w}_i\}_{i=1}^m$ as m independent learnable weight vectors. The multi-head weight vectors perform similarly as attention [30], enabling better expressiveness. Without loss of generality, we compute the estimation for edge between node pair (u, v) as

$$a_{uv} = \frac{1}{m} \sum_{i=1}^{m} cos(\boldsymbol{w}_i \odot \boldsymbol{z}_u, \boldsymbol{w}_i \odot \boldsymbol{z}_v),$$

where z_u and z_v are respectively the node representations of a node pair. Note that here for similarity learning, multiple methods can be applied for replacement.

Next, to inherit the information from original graph structure, we combine the learned graph A_{tmp} and initial graph A_{obs} with smoothing parameter β to get output A':

$$A' = \beta \times A_{obs} + (1 - \beta) \times A_{tmp}. \tag{3}$$

We set $\beta = 0$ for p model as A_{obs} is not available for distribution p.

4.3 Training

In Sec. 4.2, we have shown that the generative model p_θ and the recognition model q_Φ are respectively instantiated by a two-branch network. Next we explain their training details.

Supervise loss. To utilize the limited label information, we apply an additional cross-entropy loss $\mathcal{L}_{sup}$:

$$\mathcal{L}_{sup} = -\log q_\Phi(A_{obs}, Y_l|X), \tag{4}$$

which penalizes the difference between the estimation and the ground-truth labels $\boldsymbol{Y}_l$, $\boldsymbol{A}_{obs}$. By this term, the supervised information (both for existing links and labeled nodes) is exploited.

The final loss combines the ELBO and supervised term:

$$\mathcal{L}(\theta, \Phi) = -\mathcal{L}_{ELBO} + \eta \mathcal{L}_{sup}, \tag{5}$$

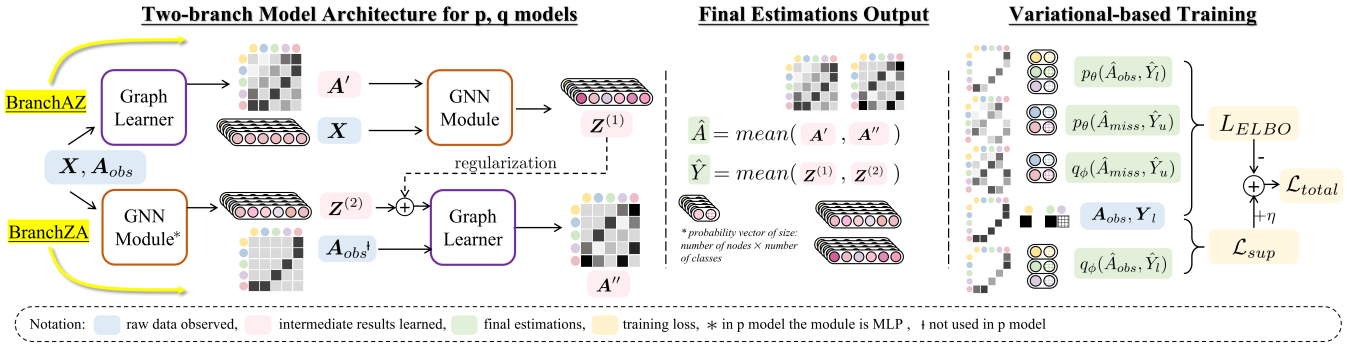


Figure 3: Illustration of the proposed WSGNN's model architecture and training pipeline. Left: The designed two-branch joint learning model. There are two instantiations, for generative model p and for inference model q (see Sec. 4.1). Middle: p, q models' output label/structure predictions as probability vector. Right: p, q models' output composes a composite loss for WSGNN training.

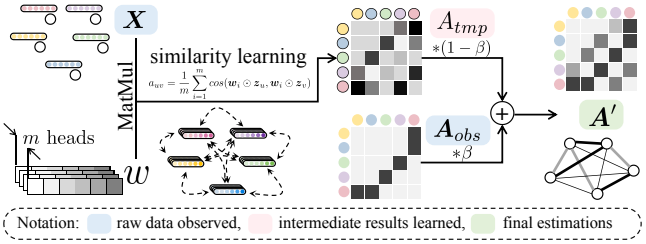


Figure 4: Illustration for the graph learner. The module is applied in p, q models' branches as illustrated in Fig. 3.

where η is a hyper-parameter.

Negative edge sampling. We consider the link prediction task as a positive-unlabeled problem, where the observed edges denoted by A_{obs} are positive samples and any other currently unconnected node pair may have an unlabeled edge between them. Therefore, if we train the network completely using A_{obs}, the model will yield trivial solutions, predicting all edges to be **1**. To avoid this, we adopt negative sampling strategy. We randomly sample node pairs which currently are not connected and assume they have label **0** instead of unlabeled samples. We use these negative samples for computing the supervised loss, which provides contrasting information and can avoid the trivial solution. The negative sampling ratio is set as five. Algorithm 1 shows the overall training process of WSGNN.

Algorithm 1: The training procedure of WSGNN

Input: $\mathcal{G} = (X, A_{obs}, Y_l)$, number of training epochs E, η
Output: Trained parameters Φ in inference model q_Φ, θ in generation model p_θ, predicted structure probability vector $\hat{A}$, predicted label probability vector $\hat{Y}$

```
Parameter initialization;
for e = 1,2,...,E do
    // WSGNN framework setup;
    for p, q models do
        // Feature-structure joint learning;
        A' = Graph Learner(X, A_obs);
        Z^(1) = GNN(X, A');
        Z^(2) = GNN(X, A_obs);
        A'' = Graph Leaner([Z^(1), Z^(2)], A_obs);
        Â = mean(A', A'');
        Ŷ = mean(Z^(1), Z^(2));
    end
    // Optimize and train the model;
    L = −L_ELBO + η L_sup;
    Back-propagate L to update model parameters Φ, θ;
end
```

5 EXPERIMENT

We evaluate WSGNN on semi-supervised node classification and link prediction. We focus on both some common data-scarce settings (e.g., Sec. 5.2, Sec. 5.4) and several more extreme situations (Sec. 5.3). We use the PyTorch-Geometric library [12] and concrete experiment environment can be found in Appendix Sec. B.2.2.

5.1 Experimental Settings

Datasets. Many standard graph learning benchmarks support neither weakly-supervised nor joint learning tasks. We create our weakly-supervised joint learning settings by modifing three commonly used benchmark datasets: Cora, Citeseer and Pubmed[25], each of which refers to a citation network with documents as nodes and citations as edges, and node labels are academic subareas. For constructing the data-scarce setting, we modify the original datasets by randomly choosing labeled nodes and observed edges according to pre-defined ratios. To further evaluate WSGNN and prior methods, we study two disease propagation tree datasets [7]: Disease-NC and Disease-LP. Following previous usage, Disease-NC is only for node classification while Disease-LP is only for link prediction. Each disease dataset represents an infectious disease spreading network built according to SIR model [2], where the node features represent disease susceptibility and node label represents whether a node is infected or not. We summarize further dataset description as in Tab. 5 in Sec. B.2.1 of Appendix.

Baselines. For baselines, we compare against general GNNs including GCN, GAT, APPNP [20], GPR-GNN [9], GRAND [11]. We also include simple method SGC [35], and variational inference method G3NN [23]. In low data regime, we further compare WSGNN with CGPN [32]. We implement the baseline models following original work with necessary modifications to perform joint learning task. Details can be found in Sec. B.2.3 of Appendix.

Training details and evaluation metrics. We denote the task of node classification as NC, the task of link prediction as LP. For all methods, we determine the hyperparameters, including learning rate, weight decay, dropout, number of layers via grid search. The hyperparameter settings are summarized with Tab. 6 in Appendix Sec. B.2.4. Without loss of generality, we measure the performance on the same five random seeds for all methods and report the mean and standard deviation. We use area under the ROC curve to evaluate link prediction task, and use classification accuracy and F1 score to respectively evaluate node classification on citation networks and infectious disease dataset. Further details about experiment environment are in Appendix Sec. B.2.2.

5.2 Results on Normal Data-Scarce Settings

We randomly choose 20 or 10 labeled nodes per class and keep 50%, 10% or 1% edges for training. We use 500 nodes and 10% edges for validation and the remaining for testing. The results on Cora, Citeseer and Pubmed are shown in Tab. 1[3]. The results show that almost for all the settings of all the datasets, WSGNN notably achieves the best node classification accuracy and also the highest link prediction score. Notably, for the case of 10 labels per class and 1% edges on cora, WSGNN beats GCN, GPR-GNN by more than 18%. Furthermore, with the labels becoming more scarce, WSGNN keeps stable performance while the baseline models suffer from great performance degradation. Besides, since our implementation of WSGNN uses APPNP as backbone module, we compare with vanilla APPNP, and the results show that our model can effectively enhance the learning performance of basic GNN modules.

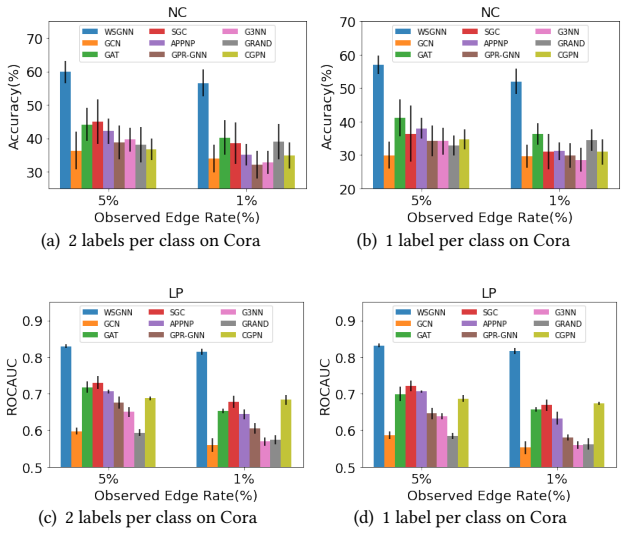
5.3 Results under Extreme Low-Data Regime

We further reduce the number of labels that are known for training from 10 labels per class to 1 or 2 labels per class. In this way, we obtain an extreme low-data regime where the label information for training is extremely limited, and the input graph structure is also largely incomplete with only 5% or 1% edges observed. Other experimental setups are the same as the standard setting. As can be seen from the results on Cora in Fig. 5, WSGNN has an unparalleled advantage over all the baselines, beating G3NN [23] and CGPN [32] which are particularly designed for data-scarce settings by more than 20% in all the four cases of NC task. The increasing gap of the performance between WSGNN and other models in such extremely cold-start situation serves as further evidence of WSGNN's superiority in learning with limited supervision. More results can be found in Tab.3 in Sec. B.1.1 of Appendix.

5.4 Case Study on Infectious Disease Spreading

We further perform evaluation on a synthetic infectious disease network derived from the SIR model. In line with the protocol in [7],

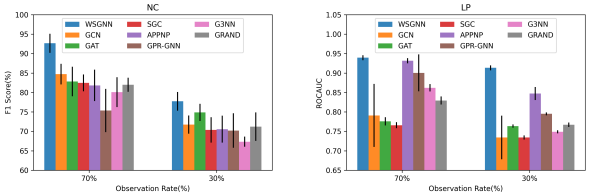
[3]The **bold** marker denotes the best performance on each training setting.



(a) 2 labels per class on Cora (b) 1 label per class on Cora

(c) 2 labels per class on Cora (d) 1 label per class on Cora

Figure 5: Node classification (NC) accuracy and link prediction (LP) ROC-AUC under extreme low data regime (2/1 labels per class and 5%/1% observed edges for training) on Cora.



(a) Disease-NC (b) Disease-LP

Figure 6: Left: Node classification (NC) F1 score on Disease-NC with 30%, 70% training labels. Right: Link prediction (LP) ROC-AUC score on Disease-LP with 30%, 70% training edges. WSGNN outperforms all other baselines.

we separately perform link prediction task on Disease-LP dataset and node classification task on Disease-NC dataset. Therefore, we modify our training loss to suit the requirement of the available information: in NC task we only use Y related loss, while in LP task, we only use A loss. It shows that our model can be degenerated to fit individual learning task and still show excellence.

For both NC and LP task, we perform experiments on the two settings: 70%, 30% edges as observed edges for training, 10% for validation, and the rest for testing. As shown in Fig. 6, WSGNN again outperforms. This suggests its suitability for incomplete disease spreading data, which is common in the early stage of spreading. More results can be found in Tab. 4 in Sec. B.1.2 of Appendix.

5.5 Parameter Sensitivity Study

An important hyperparameter that should be finetuned is the ratio of supervise loss, namely η in Eq. 5. We test the model's sensitivity by varying η from 1 to 100 using the regular experiment setting. The corresponding node classification accuracy and link prediction ROC-AUC result are respectively shown in Fig. 7 with red and blue

Table 1: Node classification (NC) accuracy and link prediction (LP) ROC-AUC score with different label/observed edge rates.

# Labels/class - Edges observed		20-50%	20-10%	20-1%	10-50%	10-10%	10-1%
NC on Cora	GCN [19]	72.50%±1.52%	61.15%±2.46%	53.33%±3.07%	66.21%±1.84%	53.56%±3.37%	45.99%±4.34%
	GAT [31]	75.50%±1.08%	65.65%±1.72%	61.06%±1.47%	71.16%±1.50%	58.86%±1.07%	54.95%±1.25%
	SGC [35]	74.76%±1.29%	64.79%±2.26%	59.09%±2.78%	70.85%±1.43%	59.25%±2.50%	53.26%±1.66%
	APPNP [20]	74.82%±1.54%	63.13%±1.68%	56.83%±2.43%	72.07%±1.75%	57.23%±1.70%	49.55%±1.11%
	GPR-GNN [9]	74.94%±1.20%	62.66%±2.25%	52.37%±1.09%	71.58%±0.80%	55.82%±1.53%	45.41%±1.79%
	G3NN [23]	77.17%±1.57%	68.71%±3.00%	57.41%±3.07%	73.56%±2.24%	64.85%±2.03%	46.90%±1.86%
	GRAND [11]	78.39%±0.69%	68.06%±1.73%	63.87%±0.88%	74.66%±1.12%	60.18%±3.16%	57.15%±0.75%
	WSGNN	**78.41%±0.37%**	**70.92%±0.58%**	**68.10%±0.38%**	**74.72%±0.58%**	**66.14%±3.40%**	**64.15%±2.56%**
LP on Cora	GCN [19]	0.7456±0.0316	0.6674±0.0273	0.6111±0.0298	0.7165±0.0261	0.6428±0.0211	0.5840±0.0262
	GAT [31]	0.8404±0.0042	0.7948±0.0076	0.7370±0.0091	0.8260±0.0063	0.7701±0.0045	0.6760±0.0141
	SGC [35]	0.8606±0.0060	0.7931±0.0123	0.7386±0.0081	0.8459±0.0059	0.7757±0.0138	0.7142±0.0153
	APPNP [20]	0.8733±0.0103	0.7934±0.0133	0.7415±0.0116	0.8608±0.0110	0.7679±0.0046	0.7086±0.0113
	GPR-GNN [9]	0.8809±0.0030	0.7949±0.0092	0.7077±0.0085	0.8743±0.0078	0.7649±0.0026	0.6614±0.0096
	G3NN [23]	0.8744±0.0033	0.7925±0.0132	0.6856±0.0071	0.8673±0.0043	0.7795±0.0046	0.6274±0.0174
	GRAND [11]	0.8276±0.0112	0.7335±0.0092	0.6706±0.0099	0.8016±0.0120	0.6938±0.0049	0.6295±0.0090
	WSGNN	**0.8933±0.0065**	**0.8547±0.0047**	**0.8350±0.0040**	**0.8807±0.0069**	**0.8383±0.0059**	**0.8162±0.0046**
NC on Citeseer	GCN [19]	65.90%±0.43%	58.33%±0.56%	54.07%±2.76%	63.24%±1.89%	52.50%±1.92%	46.27%±2.67%
	GAT [31]	67.86%±1.02%	62.57%±0.80%	60.46%±1.63%	64.52%±1.78%	57.95%±1.53%	55.26%±1.61%
	SGC [35]	66.63%±1.23%	61.35%±1.85%	59.02%±2.28%	63.42%±1.73%	56.13%±3.77%	53.04%±4.35%
	APPNP [20]	68.13%±1.19%	62.17%±0.87%	57.95%±1.61%	65.33%±0.64%	57.72%±0.80%	51.76%±2.27%
	GPR-GNN [9]	67.31%±1.02%	61.49%±0.67%	51.63%±2.79%	65.04%±1.69%	56.42%±2.57%	45.15%±2.12%
	G3NN [23]	**69.96%±0.71%**	65.36%±1.54%	51.64%±2.49%	67.68%±1.17%	60.41%±3.81%	45.82%±3.17%
	GRAND [11]	67.91%±0.90%	63.47%±1.22%	62.19%±1.07%	64.94%±1.66%	59.18%±1.48%	56.28%±2.82%
	WSGNN	69.30%±1.35%	**66.60%±1.29%**	**65.05%±1.14%**	**67.94%±0.55%**	**65.33%±0.92%**	**61.43%±1.15%**
LP on Citeseer	GCN [19]	0.7582±0.0253	0.7060±0.0227	0.6502±0.0273	0.7276±0.0078	0.6666±0.0302	0.6131±0.0338
	GAT [31]	0.8621±0.0046	0.8298±0.0058	0.7752±0.0196	0.8506±0.0068	0.8071±0.0103	0.7349±0.0245
	SGC [35]	0.8326±0.0126	0.7662±0.0175	0.7230±0.0277	0.8006±0.0200	0.7322±0.0258	0.6812±0.0278
	APPNP [20]	0.8892±0.0041	0.8431±0.0030	0.7949±0.0145	0.8837±0.0062	0.8263±0.0065	0.7606±0.0245
	GPR-GNN [9]	0.8883±0.0045	0.8385±0.0062	0.7387±0.0262	0.8941±0.0055	0.8228±0.0126	0.7040±0.0266
	G3NN [23]	0.8763±0.0054	0.8357±0.0060	0.7400±0.0178	0.8757±0.0050	0.8150±0.0115	0.6800±0.0232
	GRAND [11]	0.8138±0.0148	0.7715±0.0078	0.7304±0.0090	0.7966±0.0152	0.7110±0.0213	0.6676±0.0248
	WSGNN	**0.9210±0.0073**	**0.9056±0.0040**	**0.8932±0.0032**	**0.9199±0.0057**	**0.9049±0.0053**	**0.8873±0.0032**
NC on Pubmed	GCN [19]	75.60%±2.50%	70.52%±3.66%	69.81%±3.65%	71.30%±4.57%	65.99%±4.78%	66.20%±5.06%
	GAT [31]	75.58%±2.90%	72.08%±3.51%	71.47%±3.05%	71.45%±4.01%	68.39%±4.44%	67.77%±4.40%
	SGC [35]	75.33%±1.83%	71.66%±2.36%	72.19%±2.34%	71.83%±3.49%	67.13%±5.13%	67.56%±6.00%
	APPNP [20]	76.59%±2.02%	71.98%±2.81%	70.35%±3.00%	73.80%±2.79%	69.78%±2.11%	66.63%±4.85%
	GPR-GNN [9]	77.13%±1.85%	72.11%±3.08%	69.20%±2.47%	**74.29%±2.10%**	68.55%±3.88%	64.39%±5.02%
	G3NN [23]	75.44%±3.11%	72.69%±5.02%	72.45%±3.68%	70.43%±8.23%	66.75%±7.12%	65.69%±7.07%
	GRAND [11]	75.60%±2.50%	70.52%±3.66%	69.81%±3.65%	71.30%±4.57%	65.99%±4.78%	66.20%±5.06%
	WSGNN	**78.74%±1.54%**	**74.94%±2.74%**	**74.70%±2.25%**	73.20%±4.53%	**70.90%±4.67%**	**71.03%±4.43%**
LP on Pubmed	GCN [19]	0.8638±0.0302	0.7380±0.0291	0.7188±0.0300	0.8579±0.0236	0.7216±0.0301	0.7013±0.0258
	GAT [31]	0.8252±0.0080	0.7985±0.0070	0.7940±0.0076	0.8198±0.0103	0.7962±0.0101	0.7895±0.0124
	SGC [35]	0.8758±0.0021	0.8228±0.0094	0.8121±0.0074	0.8669±0.0077	0.8172±0.0055	0.8079±0.0059
	APPNP [20]	0.8843±0.0129	0.8236±0.0123	0.7917±0.0087	0.8818±0.0178	0.8318±0.0116	0.7953±0.0074
	GPR-GNN [9]	0.8839±0.0071	0.8298±0.0044	0.7848±0.0108	0.8939±0.0144	0.8323±0.0046	0.7755±0.0223
	G3NN [23]	0.8870±0.0056	0.8241±0.0079	0.7969±0.0101	0.8910±0.0055	0.8245±0.0134	0.8011±0.0141
	GRAND [11]	0.8638±0.0302	0.7380±0.0291	0.7188±0.0300	0.8579±0.0236	0.7216±0.0301	0.7013±0.0258
	WSGNN	**0.9145±0.0037**	**0.9036±0.0039**	**0.9001±0.0024**	**0.9126±0.0061**	**0.9021±0.0043**	**0.8996±0.0022**

lines. We find that the behavior of WSGNN is stable with the change of η, indicating the effective utilization of missing information.

5.6 Result Visualization

Fig. 8 visualizes the learned node embeddings for classification with a series of t-SNE graphs. Besides, taking Cora as an example, we also visualize the original and learned graph structure as in Fig. 9.

We obtain two sets of results respectively with the setting of [training labels per class-edges] as [20-1%] and [1-1%] on Cora. Fig. 8 shows that under regular setting [20-1%] of the top row, most of the baselines more or less show an effect of feature aggregation of the same class, especially G3NN. However, all the baselines fail to differentiate the nodes under extreme low-data regime [1-1%].

In addition, we visualize the original and predicted graph structure on Cora taking the example training setting of [20%-1%] in Fig. 9. In the first three pictures, we present the complete dataset. We can see that APPNP still mixes the different classes together while our model recovers the structure better. We randomly choose 80

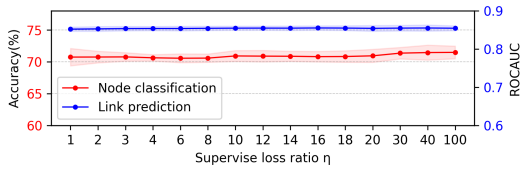


Figure 7: Sensitivity for supervision hyper-parameter η. 20 labels per class and 10% edges are used for training on Cora.

Table 2: Ablation study of the elements in WSGNN.

Models	$A \to Z$	$Z \to A$	$Z^{(2)} \to A''$
WSGNN-AZ	✓		
WSGNN-ZA		✓	
WSGNN-sym	✓	✓	
WSGNN	✓	✓	✓

nodes per class from val/test set and present their connections as shown with (d) to (f). And with (g) to (i), we highlight the distribution of class 1. WSGNN predicts the graph structure better than the vanilla APPNP.

5.7 Ablation Study

As discussed in Sec. 4.2.1, WSGNN contains three modules: branch $A \to Z$, branch $Z \to A$ and the regularization $Z^{(2)} \to A''$. We consider three ablation models as described in Tab. 2.

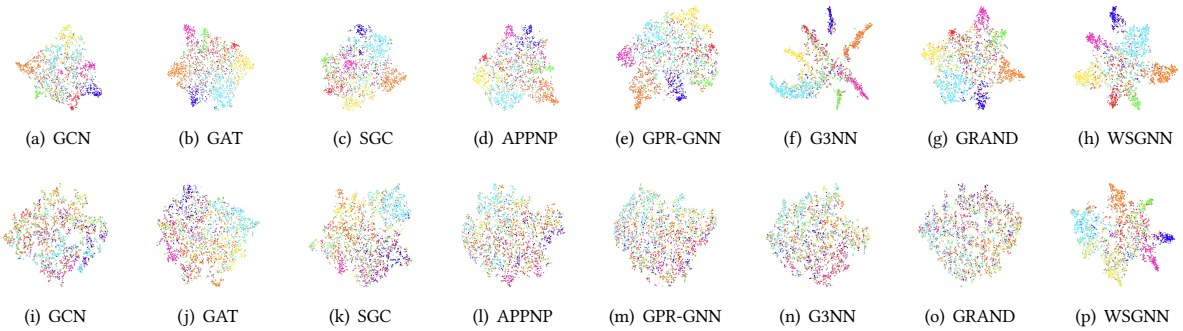
We present the ablation results of the models using the same model and experiment setup as in previous regular setting. As shown in Fig. 10, WSGNN-sym as two-branch model has recognizable performance gain over single-branch models, proving the significance of collaboratively optimizing node representation and graph structure. Furthermore, the regularization component also adds to model effectiveness as shown with the gap between WSGNN-sym and the complete framework.

6 CONCLUSION

In this paper, we have proposed a probabilistic generative framework WSGNN for semi-supervised learning on graphs that is especially designed for working in low data regime, where labeled nodes and observed edges are both very limited. To utilize the underlying information of massive missing data to the best extent, we resort to joint learning of node representations and graph structure and infer them as latent variables in a fully data-driven manner. To exploit the (scarce) labeled information, we also devise a composite principled objective that unifies the learning on observed data and inference on unobserved data. Experimental results demonstrate the practical efficacy and superiority of our model under various datasets and settings with limited supervision from nodes and edges.

REFERENCES

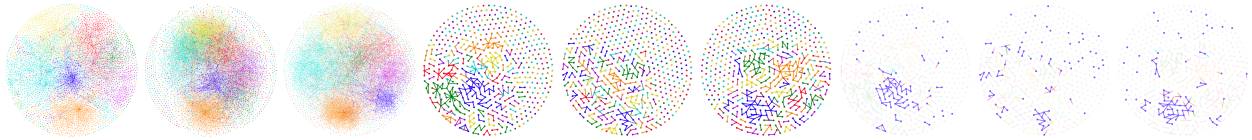
[1] Alexander Amir Alemi, Ian S. Fischer, Joshua V. Dillon, and Kevin P. Murphy. 2017. Deep Variational Information Bottleneck. In ICLR.
[2] Roy M. Anderson and Robert M. May. 1991. Infectious Diseases of Humans: Dynamics and Control.
[3] Mikhail Belkin and Partha Niyogi. 2004. Semi-Supervised Learning on Riemannian Manifolds. Machine Learning 56 (2004), 209–239.
[4] Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. 2006. Manifold Regularization: A Geometric Framework for Learning from Labeled and Unlabeled Examples. J. Mach. Learn. Res. 7 (2006), 2399–2434.
[5] Lei Cai, Jundong Li, Jie Wang, and Shuiwang Ji. 2021. Line Graph Neural Networks for Link Prediction. IEEE TPAMI PP (2021).
[6] Jeff Calder, Brendan Cook, Matthew Thorpe, and Dejan Slepev. 2020. Poisson Learning: Graph Based Semi-Supervised Learning At Very Low Label Rates. In ICML.
[7] Ines Chami, Rex Ying, Christopher Ré, and Jure Leskovec. 2019. Hyperbolic Graph Convolutional Neural Networks. In NeurIPS, Vol. 32. 4869–4880.
[8] Yu Chen, Lingfei Wu, and Mohammed J. Zaki. 2020. Iterative Deep Graph Learning for Graph Neural Networks: Better and Robust Node Embeddings. In NeurIPS.
[9] Eli Chien, Jianhao Peng, Pan Li, and Olgica Milenkovic. 2021. Adaptive Universal Generalized PageRank Graph Neural Network. In ICLR.
[10] Pantelis Elinas, Edwin V. Bonilla, and Louis C. Tiao. 2020. Variational Inference for Graph Convolutional Networks in the Absence of Graph Data and Adversarial Settings. In NeurIPS.
[11] Wenzheng Feng, Junfeng Zhang, Yuxiao Dong, Yu Han, Huanbo Luan, Qian Xu, Qiang Yang, Evgeny Kharlamov, and Jie Tang. 2020. Graph Random Neural Networks for Semi-Supervised Learning on Graphs. In NeurIPS.
[12] Matthias Fey and Jan Eric Lenssen. 2019. Fast Graph Representation Learning with PyTorch Geometric. ICLR 2019 Workshop on Representation Learning on Graphs and Manifolds.
[13] Luca Franceschi, Mathias Niepert, Massimiliano Pontil, and Xiao He. 2019. Learning Discrete Structures for Graph Neural Networks. In ICML.
[14] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. 2017. Neural Message Passing for Quantum Chemistry. In ICML, Vol. 70. 1263–1272.
[15] William L. Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In NeurIPS.
[16] Ziniu Hu, Yuxiao Dong, Kuansan Wang, Kai-Wei Chang, and Yizhou Sun. 2020. GPT-GNN: Generative Pre-Training of Graph Neural Networks. In SIGKDD.
[17] Wei Jin, Yao Ma, Xiaorui Liu, Xianfeng Tang, Suhang Wang, and Jiliang Tang. 2020. Graph Structure Learning for Robust Graph Neural Networks. In SIGKDD.
[18] Thomas Kipf and Max Welling. 2016. Variational Graph Auto-Encoders. ArXiv abs/1611.07308 (2016).
[19] Thomas Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In ICLR.
[20] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. 2019. Predict then Propagate: Graph Neural Networks meet Personalized PageRank. In ICLR.
[21] Wanyu Lin, Zhaolin Gao, and Baochun Li. 2020. Shoestring: Graph-Based Semi-Supervised Classification With Severely Limited Labeled Data. In CVPR. 4173–4181.
[22] Hanwen Liu, Huaizhen Kou, Chao Yan, and Lianyong Qi. 2019. Link prediction in paper citation network to construct paper correlation graph. EURASIP Journal on Wireless Communications and Networking 2019 (2019), 1–12.
[23] Jiaqi Ma, Weijing Tang, Ji Zhu, and Qiaozhu Mei. 2019. A Flexible Generative Framework for Graph-based Semi-supervised Learning. In NeurIPS.
[24] Siddharth Narayanaswamy, Brooks Paige, Jan-Willem van de Meent, Alban Desmaison, Noah D. Goodman, Pushmeet Kohli, Frank D. Wood, and Philip H. S. Torr. 2017. Learning Disentangled Representations with Semi-Supervised Deep Generative Models. In NeurIPS.
[25] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. 2008. Collective Classification in Network Data. AI Mag. 29 (2008), 93–106.
[26] Zixing Song, Xiangli Yang, Zenglin Xu, and Irwin King. 2022. Graph-based Semi-supervised Learning: A Comprehensive Review. IEEE TNNLS PP (2022).
[27] Qingyun Sun, Jianxin Li, Hao Peng, Jia Wu, Xingcheng Fu, Cheng Ji, and Philip S. Yu. 2022. Graph Structure Learning with Variational Information Bottleneck. In AAAI.
[28] Jie Tang, Jing Zhang, Limin Yao, Juan-Zi Li, Li Zhang, and Zhong Su. 2008. ArnetMiner: extraction and mining of academic social networks. In SIGKDD.
[29] Phi Vu Tran. 2018. Learning to Make Predictions on Graphs with Autoencoders. In 5th IEEE International Conference on Data Science and Advanced Analytics.
[30] Ashish Vaswani, Noam M. Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In NeurIPS.
[31] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio', and Yoshua Bengio. 2018. Graph Attention Networks. In ICLR.
[32] Sheng Wan, Shirui Pan, Jian Yang, and Chen Gong. 2021. Contrastive and Generative Graph Convolutional Networks for Graph-based Semi-Supervised Learning. In AAAI.
[33] Sheng Wan, Yibing Zhan, Liu Liu, Baosheng Yu, Shirui Pan, and Chen Gong. 2021. Contrastive Graph Poisson Networks: Semi-Supervised Learning with Extremely Limited Labels. In NeurIPS.
[34] Yiwei Wang, Wenhong Wang, Yuxuan Liang, Yujun Cai, Juncheng Liu, and Bryan Hooi. 2020. NodeAug: Semi-Supervised Node Classification with Data Augmentation. In SIGKDD.



(a) GCN (b) GAT (c) SGC (d) APPNP (e) GPR-GNN (f) G3NN (g) GRAND (h) WSGNN

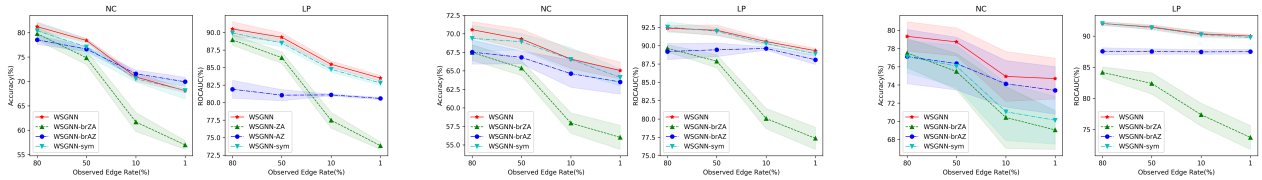
(i) GCN (j) GAT (k) SGC (l) APPNP (m) GPR-GNN (n) G3NN (o) GRAND (p) WSGNN

Figure 8: t-SNE plot on Cora. 20 labels (for top) and 1 label (for bottom) per class and 1% of edges are used for training.

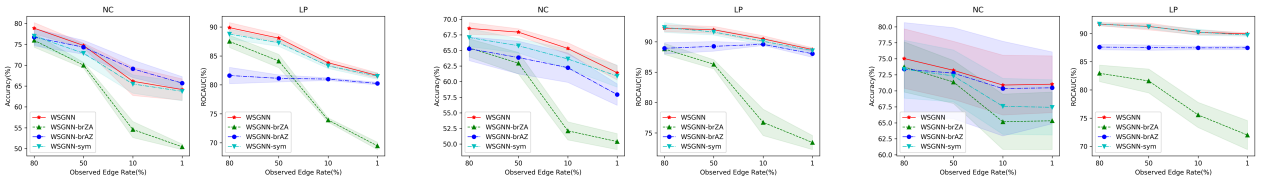


(a) Original Cora (b) APPNP (c) WSGNN (d) Original Cora (e) APPNP (f) WSGNN (g) Original Cora (h) APPNP (i) WSGNN

Figure 9: Link prediction visualization on Cora. 20 labels per class and 1% observed edges are used for training. The permutation of nodes are automatically generated for clearness. Node color represents different classes. (a) to (c): Visualization on complete dataset. (d) to (f): Random 80 nodes per class are chosen for visualization. (g) to (i): Sub-graph of Class 1 are highlighted.



(a) 20 labels per class on Cora (b) 20 labels per class on Citeseer (c) 20 labels per class on Pubmed



(d) 10 labels per class on Cora (e) 10 labels per class on Citeseer (f) 10 labels per class on Pubmed

Figure 10: Ablation study on Cora, Citeseer and Pubmed with 20/10 labels and 80%/50%/10%/1% edges for training. WSGNN-sym shows advantage over the single-branch models, indicating the necessity of two-branch joint learning. In addition, complete WSGNN wins against WSGNN-sym, illustrating the regularization term of $Z^{(2)} \rightarrow A''$ is effective.

[35] Felix Wu, Tianyi Zhang, Amauri H. de Souza, Christopher Fifty, Tao Yu, and Kilian Q. Weinberger. 2019. Simplifying Graph Convolutional Networks. In ICML, Vol. 97. 6861–6871.

[36] Qitian Wu, Hengrui Zhang, Xiaofeng Gao, Junchi Yan, and Hongyuan Zha. 2021. Towards open-world recommendation: An inductive model-based collaborative filtering approach. In ICML. 11329–11339.

[37] Qitian Wu, Hengrui Zhang, Junchi Yan, and David Wipf. 2022. Handling Distribution Shifts on Graphs: An Invariance Perspective. In ICLR.

[38] Tailin Wu, Hongyu Ren, Pan Li, and Jure Leskovec. 2020. Graph Information Bottleneck. In NeurIPS.

[39] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. 2021. Do Transformers Really Perform Bad for Graph Representation?. In NeurIPS.

[40] Hengrui Zhang, Qitian Wu, Junchi Yan, David Wipf, and Philip S Yu. 2021. From canonical correlation analysis to self-supervised graph neural networks. NeurIPS.

[41] Yingxue Zhang, Soumyasundar Pal, Mark J. Coates, and Deniz Üstebay. 2019. Bayesian graph convolutional neural networks for semi-supervised classification. In AAAI.

[42] Cheng Zheng, Bo Zong, Wei Cheng, Dongjin Song, Jingchao Ni, Wenchao Yu, Haifeng Chen, and Wei Wang. 2020. Robust Graph Representation Learning via Neural Sparsification. In ICML.

[43] Xiaojin Zhu, Zoubin Ghahramani, and John D. Lafferty. 2003. Semi-Supervised Learning Using Gaussian Fields and Harmonic Functions. In ICML.

[44] Xiaojin Zhu, John D. Lafferty, and Ronald Rosenfeld. 2005. Semi-supervised learning with graphs.

[45] Yanqiao Zhu, Weizhi Xu, Jinghao Zhang, Qiang Liu, Shu Wu, and Liang Wang. 2021. Deep Graph Structure Learning for Robust Representations: A Survey. ArXiv abs/2103.03036 (2021).

A METHODOLOGY SUPPLEMENTARY

A.1 Derivation of Evidence Lower BOund

Here we prove Eq. 2 start from derivation of the Kullback-Leibler divergence between the intractable distribution $p_\theta(A_{miss}, Y_u|A_{obs}, Y_l, X))$ and the recognition distribution $q_\Phi(A_{miss}, Y_u|A_{obs}, Y_l, X)$.

$$\begin{aligned}
&\mathcal{D}_{KL}(q_\Phi(A_{miss}, Y_u|A_{obs}, Y_l, X)||p_\theta(A_{miss}, Y_u|A_{obs}, Y_l, X)) \\
=&\mathbb{E}_{q_\Phi(A_{miss},Y_u|A_{obs},Y_l,X)}[\log \frac{q_\Phi(A_{miss}, Y_u|A_{obs}, Y_l, X)}{p_\theta(A_{miss}, Y_u|A_{obs}, Y_l, X)}] \\
=&\mathbb{E}_{q_\Phi(A_{miss},Y_u|A_{obs},Y_l,X)}[\log \frac{q_\Phi(A_{miss}, Y_u|A_{obs}, Y_l, X)p_\theta(A_{obs}, Y_l|X)}{p_\theta(A_{miss}, Y_u, A_{obs}, Y_l|X)} \\
=&\mathbb{E}_{q_\Phi(A_{miss},Y_u|A_{obs},Y_l,X)}[-\log \frac{p_\theta(A_{miss}, Y_u, A_{obs}, Y_l|X)}{q_\Phi(A_{miss}, Y_u|A_{obs}, Y_l, X)}] \\
&+\log p_\theta(A_{obs}, Y_l|X)
\end{aligned}$$

By a simple transposition of terms we can get:

$$\begin{aligned}
&\mathcal{L}_{ELBO} = \log p_\theta(A_{obs}, Y_l|X) \\
&- \mathcal{D}_{KL}(q_\Phi(A_{miss}, Y_u|A_{obs}, Y_l, X)||p_\theta(A_{miss}, Y_u|A_{obs}, Y_l, X)).
\end{aligned} \tag{6}$$

B EXPERIMENT SUPPLEMENTARY

B.1 Experiment Results

B.1.1 Extreme low data regime. Here with Tab. 3 we present the numerical results of extreme low data regime setting (Sec. 5.3).

Table 3: Performance on Cora under extreme low data regime[2].

# Labels-Edges	2-5%	2-1%	1-5%	1-1%
Node classification accuracy (%)				
GCN [19]	36.30±5.59*	33.89±4.21*	29.91±4.05*	29.60±3.42*
GAT [31]	44.11±4.88	40.24±5.17	41.06±5.51	36.34±3.20
SGC [35]	44.88±6.65	38.59±6.25	36.36±8.39*	30.98±5.38*
APPNP [20]	42.09±3.81	35.09±3.24*	37.98±3.06	31.11±2.74*
GPR-GNN [9]	38.70±5.04*	32.08±4.03*	34.32±4.60*	29.88±3.71*
G3NN [23]	39.56±3.62*	32.81±3.50*	34.12±3.95*	28.53±3.50*
GRAND [11]	38.08±5.31*	38.96±5.20	32.78±3.04*	34.39±3.19
CGPN [33]	36.64±3.16*	34.84±3.91*	34.68±2.96*	30.96±3.80*
WSGNN	**59.80±3.32**	**56.56±4.11**	**56.96±2.84**	**52.00±3.74**
Link prediction ROC-AUC score				
GCN [19]	0.5975±0.0090*	0.5601±0.0185*	0.5857±0.0105*	0.5525±0.0184*
GAT [31]	0.7178±0.0153	0.6532±0.0064	0.6986±0.0195	0.6571±0.0067
SGC [35]	0.7300±0.0172	0.6778±0.0171	0.7204±0.0144	0.6683±0.0154
APPNP [20]	0.7061±0.0055	0.6436±0.0127	0.7056±0.0030	0.6321±0.0176
GPR-GNN [9]	0.6749±0.0162	0.6058±0.0148*	0.6457±0.0151	0.5804±0.0074*
G3NN [23]	0.6499±0.0128	0.5693±0.0112*	0.6380±0.0076	0.5588±0.0105*
GRAND [11]	0.5931±0.0091*	0.5744±0.0125*	0.5841±0.0076*	0.5625±0.0163*
CGPN [33]	0.6871±0.0056	0.6830±0.0131	0.6867±0.0101	0.6730±0.0044
WSGNN	**0.8290±0.0050**	**0.8143±0.0091**	**0.8314±0.0058**	**0.8157±0.0084**

B.1.2 Case study: infectious disease spreading. Here with Tab. 4 we present the numerical results of infectious disease setting (Sec. 5.4).

[2]We use the asterisk (*) marker to denote the case when WSGNN outperforms the method by over 20% in NC task, or WSGNN achieves an over 0.2 higher score in LP task.

Table 4: Node classification F1 score(%) and link prediction ROC AUC score on Disease-NC, Disease-LP datasets respectively.

Dataset	Disease-NC		Disease-LP	
Observation	70%	30%	70%	30%
GCN [19]	84.73±2.66	71.74±2.35	0.7910±0.0811	0.7344±0.0562
GAT [31]	82.83±3.79	74.88±2.20	0.7759±0.0107	0.7635±0.0042
SGC [35]	82.48±2.17	70.38±3.25	0.7657±0.0077	0.7343±0.0054
APPNP [20]	81.83±4.05	70.57±3.47	0.9318±0.0067	0.8474±0.0169
GPRGNN [9]	75.35±5.56	70.22±4.43	0.9004±0.0474	0.7952±0.0037
G3NN [23]	80.09±3.86	67.34±1.31	0.8621±0.0095	0.7490±0.0039
GRAND [11]	81.99±1.81	71.21±3.65	0.8294±0.0105	0.7670±0.0058
WSGNN	**92.65±2.47**	**77.73±2.39**	**0.9382±0.0054**	**0.9143±0.0092**

B.2 Experimental Settings

B.2.1 Dataset. Table. 5 summarizes the statistics of the five datasets used in our experiments.

Table 5: Dataset statistics.

Datasets	Nodes	Edges	Features	Classes
Cora [25]	2,708	5,429	1,433	7
Citeseer [25]	3,327	4,732	3,703	6
Pubmed [25]	19,717	44,338	500	3
Disease-NC [7]	1,044	1,043	1,000	2
Disease-LP [7]	2,665	2,664	11	2

B.2.2 Experiment Environments. Cora, Citeseer, Disease-LP and Disease-NC experiments are run on GeForce RTX 2080Ti (11GB) and Intel(R) Xeon(R) E5-2678 v3 CPU @ 2.50GHz. Pubmed experiments are run on Quadro RTX 8000 (48GB) and Intel(R) Xeon(R) W-3175X CPU @ 3.10GHz.

Our environment configurations are as follows:

- Ubuntu 16.04
- CUDA 11.0
- Python 3.8
- Pytorch 1.7.1
- Pytorch Geometric 2.0.2

B.2.3 Baseline Implementation. As introduced in Section 5.1, we have 8 GNN-based models as our baselines: GCN [19], GAT [31], SGC [35], APPNP [20], GPR-GNN [9], G3NN [23], GRAND [11], CGPN [32][4].

For joint learning tasks, we extend the output layer and loss of these baselines for link prediction as follows.

[4]Implementation of GCN, GAT, SGC, APPNP and GPR-GNN is adapted from https://github.com/CUAI/Non-Homophily-Benchmarks.
Implementation of G3NN is adapted from https://github.com/jiaqima/G3NN.
Implementation of GRAND is adapted from https://github.com/THUDM/GRAND.
Implementation of CGPN is adapted from https://github.com/LEAP-WS/CGPN.

Table 6: Hyperparameters for WSGNN on all datasets

Hyperparameter	Cora	Citeseer	Pubmed	Cora(low data regime)	Disease -NC	Disease -LP
Learning rate	0.01	0.01	0.01	0.01	0.001	0.01
Weight decay	5e-4	5e-4	5e-4	0	5e-4	0
Training epochs	100	200	200	100	1000	1000
Supervise loss ratio η	10	10	10	10	10	10
Hidden layer size	64	64	64	64	64	64
Dropout rate	0.5	0.5	0.5	0.5	0.5	0.5
Number of MLP layers	2	2	2	2	2	2
Number of APPNP layers	4	4	6	6	3	1
APPNP teleport probability	0.1	0.1	0.1	0.1	0.1	0.1
Graph skip connection β for q model	0.5	0.5	0.5	0.5	0.8	0.8
Number of heads in graph learner	8	8	8	8	8	1

Joint Learning Output Layer. For NC task, we implement the output layer following the original paper, e.g. GCNConv for GCN. For LP task, we estimate the adjacency between node pair (u,v):

$$\hat{a}_{uv} = \text{Sigmoid}(z_u \cdot z_v)^5,$$

where z_u and z_v are respectively the node representations of node pair (u,v) after the last hidden layer.

Joint Learning Loss. For NC task, we define the loss $\mathcal{L}_{NC}$ the same as original paper, e.g. cross-entropy loss for GCN. For LP task, we evaluate the binary cross-entropy error over all observed edge examples:

$$\mathcal{L}_{LP} = -\frac{1}{|\mathcal{E}_{obs}|} \sum_{(u,v)\in\mathcal{E}_{obs}} (a_{uv} \cdot \log \hat{a}_{uv} + (1 - a_{uv}) \cdot \log(1 - \hat{a}_{uv})), \tag{7}$$

where a_{uv} and $\hat{a}_{uv}$ are respectively the true and predicted adjacency of node pair (u,v).

By assigning the weight hyperparameters λ_1 and λ_2 to $\mathcal{L}_{NC}$ and $\mathcal{L}_{LP}$ correspondingly, the total loss is:

$$\mathcal{L}_{baseline} = \lambda_1 \mathcal{L}_{NC} + \lambda_2 \mathcal{L}_{LP}. \tag{8}$$

In our implementation, we set $\lambda_1 = \lambda_2 = 1$ for joint learning tasks, $\lambda_1 = 1, \lambda_2 = 0$ for NC tasks and $\lambda_1 = 0, \lambda_2 = 1$ for LP tasks.

B.2.4 Model Hyperparameters. In table. 6, we show the hyperparameters for WSGNN on all datasets.

[5]The implementation is adapted from https://github.com/rusty1s/pytorch_geometric/blob/master/examples/link_pred.py