

---

# A Flexible Generative Framework for Graph-based Semi-supervised Learning

---

**Jiaqi Ma\***

School of Information  
University of Michigan  
jiaqima@umich.edu

**Weijing Tang\***

Department of Statistics  
University of Michigan  
weijtang@umich.edu

**Ji Zhu**

Department of Statistics  
University of Michigan  
jizhu@umich.edu

**Qiaozhu Mei**

School of Information and Department of EECS  
University of Michigan  
qmei@umich.edu

## Abstract

We consider a family of problems that are concerned about making predictions for the majority of unlabeled, graph-structured data samples based on a small proportion of labeled examples. Relational information among the data samples, often encoded in the graph or network structure, is shown to be helpful for these semi-supervised learning tasks. Conventional graph-based regularization methods and recent graph neural networks do not fully leverage the interrelations between the features, the graph, and the labels. We propose a flexible generative framework for graph-based semi-supervised learning, which approaches the joint distribution of the node features, labels, and the graph structure. Borrowing insights from random graph models in network science literature, this joint distribution can be instantiated using various distribution families. For the inference of missing labels, we exploit recent advances of scalable variational inference techniques to approximate the Bayesian posterior. We conduct thorough experiments on benchmark datasets for graph-based semi-supervised learning. Results show that the proposed methods outperform state-of-the-art models under most settings.

## 1 Introduction

Traditional machine learning methods typically treat data samples as independent and approximate a mapping function from the features to the outcome of each individual sample. However, many real-world data collections, such as social media or scientific articles, often come with richer relational information among the individual samples. We consider a family of such scenarios where the relational information is stored in a graph structure with the data samples as nodes, and the learning task is to predict the outcome of unlabeled nodes based on the graph structure as well as a set of labeled nodes. In these scenarios, breaking the independence assumption and utilizing such relational information in the prediction models have been shown to be helpful [24, 23, 1, 9, 4, 13, 11]. However, there still lacks a principled way to best synergize and utilize the relational information stored in the graph together with the information stored in individual nodes. In this paper, we consider the problem of graph-based semi-supervised learning and try to approach this question by presenting a flexible generative framework that explicitly models the joint relationship among the three key types of information in this context: features, outcomes (or labels), and the graph.

---

\*The authors contribute equally to this paper.

There are two major classes of existing methods for graph-based semi-supervised learning. The first class is graph-based regularization methods [24, 23, 1, 13, 11], where explicit regularizations are posed to smooth the outcome predictions or feature representations over a local neighborhood. This class of methods shares an assumption that some kind of smoothness (e.g., the outcomes of adjacent nodes are likely to be the same) should present in the local and global graph structure. The second class consists of graph neural networks [9, 4, 19], where the node features within a local neighborhood are aggregated into a hidden representation for the central node and predictions are made on top of the hidden representations. These existing methods either do not treat the graph as a random variable (but rather as a fixed observation) or do not jointly model the data features, graph, and outcomes.

While having not been well-explored in graph-based semi-supervised learning, we believe that modeling the joint distribution of the data, graph, and labels with generative models has several unique advantages over the above methods.

First, generative models can learn succinct underlying structures of the graph data. Rich literature in network science [15] has shown that underlying structures often exist in real-world graph data. And there have been many probabilistic generative models [6, 5] that can learn the underlying structures well from observed graph data. Most of the existing graph-based semi-supervised learning methods described above view the graph as a fixed observation and treat it as ground truth. In reality, however, an observed graph is often noisy. We expect that through treating features, outcomes, and graph as random variables, a generative model can capture more general patterns among these entities and learn low-dimensional representations of the data that can take account for the noise in the graph.

Second, modeling the joint distribution can extract more general relationship among features, outcomes, and the graph. We argue that both classes of the existing graph-based semi-supervised learning methods only utilize restricted relationships among them. The graph-based regularization methods usually make strong assumptions about smoothness over adjacent nodes. Such assumptions often restrict the model capacity, making the models fail to fully utilize the relational information. The graph neural networks, although more flexible in aggregating node features through the graph structure, usually implicitly assume conditional independence over the outcomes given node features and the graph. This might still be sub-optimal in utilizing the relational information. Directly modeling the joint distribution with flexible models allows us to better utilize the relational information.

Lastly, generative models can better handle the missing-data situation. In real-world applications, we are often faced with imperfect data, where either node features or edges in the graph are missing. Generative models excel in such situations.

A few previous work [22, 12] trying to apply generative models in graph-based semi-supervised learning have been restricted to relatively simple model families due to the difficulty in efficient training of generative models. Thanks to recent advances of scalable variational inference techniques [8, 7], we are able to propose a flexible generative framework for graph-based semi-supervised learning. In this work, we use neural networks, latent space models [5], and stochastic block models [6] to form the generative models. And we use graph neural networks as the approximate posterior models in the scalable variational inference. We evaluate the proposed framework with four variants of instantiations on three semi-supervised classification benchmark datasets. Experiments show that our models achieve better performances than the state-of-the-art models under most settings.

## 2 Related Work

This paper mainly focuses on the problem of graph-based semi-supervised learning, where the data samples are connected by a graph and the outcome labels are only available to part of the samples. The goal is to infer the unobserved labels based on both the labeled and unlabeled data as well as the graph structure.

### 2.1 Graph-based Regularization for Semi-supervised Learning

One of the most popular types of graph-based semi-supervised learning methods is the graph-based regularization methods. The general assumption of such methods is that the data samples are located

in a low-dimensional manifold where each local neighborhood is a high-dimensional Euclidean space, and the graph stores the similarity or proximity of these data samples. Various graph regularizations are posed to smooth the outcome predictions of the model or the feature representations of the data samples over the local neighborhood in the graph. Suppose there are  $n$  data samples in total and  $m$  of them are labeled, the graph-based regularization methods generally conduct semi-supervised learning by optimizing the following objective function:

$$\sum_{i=1}^m \mathcal{L}_i + \eta \sum_{i,j=1}^n w_{i,j} \mathcal{R}(\mathbf{f}_i, \mathbf{f}_j),$$

where  $\mathcal{L}_i$  is the supervised loss function of sample  $i$ ;  $\mathcal{R}(\cdot, \cdot)$  is a regularization function and  $w_{i,j}$  is a graph-based coefficient;  $\mathbf{f}_i, \mathbf{f}_j$  could be the outcome predictions [24, 23, 1] or the feature representations [13, 20, 11] of nodes  $i$  and  $j$ ;  $\eta$  is a hyper-parameter trading-off the supervised loss and the graph-based regularization. Different methods can have different variants of the regularization term. Most commonly, it is set as a graph Laplacian regularizer [24, 23, 1, 13, 20]. Such type of models heavily rely on the smoothness assumption over the graph, which restricts the modeling capacity [9].

## 2.2 Graph Neural Networks for Semi-supervised Learning

Another class of methods that have gained great attention recently are the graph neural networks [9, 4, 19]. A graph neural network aggregates the node features within a local neighborhood into a hidden representation for the central node. Such aggregation operations can also be stacked on top of the hidden representations to form deeper neural networks. Generally, a single aggregation operation for node  $i$  at depth  $l$  can be represented as follows,

$$h_i^l = \sigma\left(\sum_{j \in \mathcal{N}_i} \alpha_{i,j} \mathbf{W} h_j^{l-1}\right),$$

where  $h_i^l$  is the hidden representation of  $i$  at  $l^{\text{th}}$  layer;  $\mathcal{N}_i$  is the neighbor set of  $i$ ;  $\mathbf{W}$  is a learnable linear transformation matrix;  $\sigma$  is an element-wise nonlinear activation function; and different models have different definitions for  $\alpha_{i,j}$ . For Graph Convolutional Networks [9],  $\alpha_{i,j} = 1/d_i$  or  $\alpha_{i,j} = 1/\sqrt{d_i d_j}$ , where  $d_i$  is the number of neighbors of  $i$ . For Graph Attention Networks [19],  $\alpha_{i,j}$  is defined as an attention function between  $i$  and  $j$ . Finally, the predictions of each node are made on top of hidden representations in the last layer. Such methods usually model the mapping from the features and the graph to the outcome of an individual node, which assume the outcomes are conditionally independent given the features and the graph. This assumption prevents the model from utilizing the joint relationship among the outcomes over the graph. Our framework models the joint distribution of the features, outcomes, and the graph, and it is not restricted to this assumption.

## 2.3 Generative Methods for Graph-based Semi-supervised Learning

Most methods from the above two classes treat the graph as fixed observation and only a few methods [16, 22, 12] treat the graph as a random variable and model it with generative models. Among them, Ng et al. [16] focus more on an active learning setting on graphs; Zhang et al. [22] models the graph along with a stochastic block model and does not consider the interaction between the graph and the features or the labels in the generative model of the graph; the work of Liu [12] is the most similar one as our framework, but is restricted in the linear models. To our best knowledge, our work is the first to propose a generative framework for graph-based semi-supervised learning that models the joint distribution of features, outcomes, and the graph with flexible nonlinear models.

Finally, as a side note, there has been a recently active area of deep generative models for graphs [10, 2]. But these models focus more on generating realistic graph topology structures and are less related to our problem of interest.

# 3 Approach

## 3.1 Problem Setup

We start by formally introducing the problem of graph-based semi-supervised learning. Given a set of data samples  $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^n$ ,  $\mathbf{x}_i \in \mathbb{R}^d$  and  $\mathbf{y}_i \in \mathbb{R}^l$  are the feature and outcome vectors

of sample  $i$  respectively. We further denote  $\mathbf{X} \in \mathbb{R}^{n \times d}$  and  $\mathbf{Y} \in \mathbb{R}^{n \times l}$  as the matrices formed by feature and outcome vectors. The dataset also comes with a graph  $G = (\mathcal{V}, \mathcal{E})$  with the data samples as nodes, where  $\mathcal{V} = \{1, 2, \dots, n\}$  is the set of nodes and  $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$  is the set of edges. In the semi-supervised learning setting, only  $0 < m < n$  samples have observed their outcome labels and the outcome labels of other samples are missing. Without loss of generality, we assume the outcomes of the samples  $1, 2, \dots, m$  are observed and that of  $m+1, \dots, n$  are missing. Therefore we can partition the outcome matrix as

$$\mathbf{Y} = \begin{bmatrix} \mathbf{Y}_{\text{obs}} \\ \mathbf{Y}_{\text{miss}} \end{bmatrix}.$$

In the most general form, the goal of graph-based semi-supervised learning is to infer  $\mathbf{Y}_{\text{miss}}$  based on  $(\mathbf{X}, \mathbf{Y}_{\text{obs}}, G)$ . For discriminative learning, this is usually done by first learning a prediction model  $\mathbf{y} = f(\mathbf{x}; \mathbf{X}, G)$  using empirical risk minimization, optionally with regularizations:

$$\hat{f} = \arg \min_f \frac{1}{m} \sum_{i=1}^m \mathcal{L}(\mathbf{y}_i, f(\mathbf{x}_i; \mathbf{X}, G)) + \lambda \mathcal{R}(f; G),$$

where  $\mathcal{L}(\cdot, \cdot)$  is a discriminative loss function,  $\mathcal{R}(\cdot; G)$  is a graph-based regularization, and  $\lambda$  is a hyper-parameter controlling the strength of the regularization. Then we use the learned model  $\hat{f}$  to conduct the inference of  $\mathbf{Y}_{\text{miss}}$ .

There are two specific learning settings, namely transductive learning and inductive learning, which are common in graph-based semi-supervised learning. Transductive learning assumes that  $\mathbf{X}$  and  $G$  are fully observed during both the learning stage and the inference stage while inductive learning assumes  $\mathbf{X}_{m+1:n}$  and the nodes of  $m+1, \dots, n$  in  $G$  are missing during the learning stage but available during the inference stage. In the following of this paper, we will mainly focus on the transductive learning setting but our method can be easily extended to the inductive learning setting.

### 3.2 A Flexible Generative Framework for Graph-based Semi-supervised Learning

In discriminative learning, the graph  $G$  is usually viewed as a fixed observation treated as ground truth. In reality, however, there is usually considerable noise in the graph. Moreover, we want to take advantage of the underlying structure among  $\mathbf{X}$ ,  $\mathbf{Y}$ , and  $G$  to improve the prediction performance under this semi-supervised learning setting. In this work, we propose a flexible generative framework that can model a wide range of the forms of the joint distribution  $p(\mathbf{X}, \mathbf{Y}, \mathbf{G})$ , where  $\mathbf{X}$ ,  $\mathbf{Y}$ , and  $\mathbf{G}$  are the random variables corresponding to  $\mathbf{X}$ ,  $\mathbf{Y}$ , and  $G$ . We also denote  $\mathbf{Y}_{\text{obs}}$  and  $\mathbf{Y}_{\text{miss}}$  as the random variable counterparts of  $\mathbf{Y}_{\text{obs}}$  and  $\mathbf{Y}_{\text{miss}}$ .

**Generation process.** Inspired by the random graph models from the area of network science [15], we assume the graph is generated based on the node features and outcomes. The generation process can be illustrated by the following factorization of the joint distribution:

$$p(\mathbf{X}, \mathbf{Y}, \mathbf{G}) = p(\mathbf{G}|\mathbf{X}, \mathbf{Y})p(\mathbf{Y}|\mathbf{X})p(\mathbf{X}),$$

where the conditional probabilities  $p(\mathbf{G}|\mathbf{X}, \mathbf{Y})$  and  $p(\mathbf{Y}|\mathbf{X})$  will be modeled by some flexible parametric families distributions  $p_{\theta}(\mathbf{G}|\mathbf{X}, \mathbf{Y})$  and  $p_{\theta}(\mathbf{Y}|\mathbf{X})$  with parameters  $\theta$ . By "flexible" we mean that the only restriction on the PMFs of these conditional probabilities is that they need to be differentiable almost everywhere w.r.t.  $\theta$ ; and we do not assume the integral of the marginal distribution  $p_{\theta}(\mathbf{G}|\mathbf{X}) = \int p_{\theta}(\mathbf{Y}|\mathbf{X})p_{\theta}(\mathbf{G}|\mathbf{Y}, \mathbf{X})d\mathbf{Y}$  is tractable. For simplicity, we do not specify the distribution  $p(\mathbf{X})$  and everything will be conditioned on  $\mathbf{X}$  later in this paper.

**Model inference.** To infer the missing outcomes  $\mathbf{Y}_{\text{miss}}$ , we would need the posterior distribution  $p_{\theta}(\mathbf{Y}_{\text{miss}}|\mathbf{X}, \mathbf{Y}_{\text{obs}}, \mathbf{G})$ , which is usually intractable under many flexible generative models. Following the recent advances in scalable variational inference [8, 7], we introduce a recognition model  $q_{\phi}(\mathbf{Y}_{\text{miss}}|\mathbf{X}, \mathbf{Y}_{\text{obs}}, \mathbf{G})$  parameterized by  $\phi$  to approximate the true posterior  $p_{\theta}(\mathbf{Y}_{\text{miss}}|\mathbf{X}, \mathbf{Y}_{\text{obs}}, \mathbf{G})$ .

**Model learning.** We train the model parameters  $\theta$  and  $\phi$  by optimizing the Evidence Lower Bound (ELBO) of the observed data  $(\mathbf{Y}_{\text{obs}}, G)$  conditioned on  $\mathbf{X}$ . The negative ELBO loss  $\mathcal{L}_{\text{ELBO}}$

is defined as follows,

$$\log p(\mathbf{Y}_{\text{obs}}, G|\mathbf{X}) \geq \mathbb{E}_{q_\phi(\mathbf{Y}_{\text{miss}}|\mathbf{X}, \mathbf{Y}_{\text{obs}}, G)}(\log p_\theta(\mathbf{Y}_{\text{miss}}, \mathbf{Y}_{\text{obs}}, G|\mathbf{X}) - \log q_\phi(\mathbf{Y}_{\text{miss}}|\mathbf{X}, \mathbf{Y}_{\text{obs}}, G)) \\ \triangleq -\mathcal{L}_{\text{ELBO}}(\theta, \phi; \mathbf{X}, \mathbf{Y}_{\text{obs}}, G).$$

And the optimal model parameters are obtained by minimizing this loss:

$$\hat{\theta}, \hat{\phi} = \arg \min_{\theta, \phi} \mathcal{L}_{\text{ELBO}}(\theta, \phi; \mathbf{X}, \mathbf{Y}_{\text{obs}}, G).$$

### 3.3 Instantiations

For practical use, it remains to specify the parametric forms of the generative model  $p_\theta(\mathbf{G}|\mathbf{X}, \mathbf{Y})$  and  $p_\theta(\mathbf{Y}|\mathbf{X})$ , and the approximate posterior model  $q_\phi(\mathbf{Y}_{\text{miss}}|\mathbf{X}, \mathbf{Y}_{\text{obs}}, \mathbf{G})$ . As proof of the effectiveness of our general framework, we intend to instantiate its components with simple models and leave room for optimizing its performance with more complex instantiations. The generative framework proposed above does not restrict the type of outcomes. As proof of concept, we focus on multi-class classification outcomes in the following of the paper and denote the number of classes as  $K$ . Interested readers can work out the initialization for other types of outcome labels.

#### 3.3.1 Instantiations of the Generative Model

For  $p_\theta(\mathbf{Y}|\mathbf{X})$  in the generative model, we simply instantiate it with a multi-layer perceptron. For  $p_\theta(\mathbf{G}|\mathbf{X}, \mathbf{Y})$  in the generative model, we have come up with two instantiations inspired by the generative network models from network science literature. There are two major classes of generative models for complex networks: the latent space models (LSM) [5] and the stochastic block models (SBM) [6]. We instantiate a simple model from each class as our generative models.

A general assumption used by both classes of generative models for networks is that the edges are conditionally independent. Let  $\mathbf{e}_{i,j} \in \mathcal{V} \times \mathcal{V}$  be the binary edge random variable between node  $i$  and  $j$ .  $\mathbf{e}_{i,j} = 1$  indicates the edge between  $i$  and  $j$  exists and vice versa. Based on the conditional independence assumption of edges, the conditional probability of the graph  $\mathbf{G}$  can be factorized as

$$p_\theta(\mathbf{G}|\mathbf{X}, \mathbf{Y}) = \prod_{i,j} p_\theta(\mathbf{e}_{i,j}|\mathbf{X}, \mathbf{Y}).$$

Next we will specify the instantiations of  $p_\theta(\mathbf{e}_{i,j}|\mathbf{X}, \mathbf{Y})$ .

**Instantiation with a LSM.** The latent space model assumes that the nodes lie in a latent space and the probability of  $\mathbf{e}_{i,j}$  only depends the representation of node  $i$  and  $j$ . i.e.,  $p_\theta(\mathbf{e}_{i,j}|\mathbf{X}, \mathbf{Y}) = p_\theta(\mathbf{e}_{i,j}|\mathbf{x}_i, \mathbf{y}_i, \mathbf{x}_j, \mathbf{y}_j)$ . We specify the PMF as a logit model:

$$p_\theta(\mathbf{e}_{i,j} = 1|\mathbf{x}_i, \mathbf{y}_i, \mathbf{x}_j, \mathbf{y}_j) = \sigma([(U\mathbf{x}_i)^T, \mathbf{y}_i^T, (U\mathbf{x}_j)^T, \mathbf{y}_j^T]\mathbf{w}),$$

where  $\sigma(\cdot)$  is the logistic function and  $\mathbf{w}$  is the parameters of the logistic function and is included in  $\theta$ ;  $U$  is a linear transformation matrix with learnable parameters; the class labels  $\mathbf{y}_i, \mathbf{y}_j$  are represented as one-hot vectors, and we concatenate the transformed features and the class labels of a pair of nodes as input of the logistic function.

**Instantiation with a SBM.** The stochastic block model assumes there are  $C$  types of nodes and each node  $i$  has a (latent) type variable  $z_i \in \{1, 2, \dots, C\}$  and the probability of edge  $\mathbf{e}_{i,j}$  only depends on node types  $z_i, z_j$ . In a general SBM, we have  $p_\theta(\mathbf{e}_{i,j} = 1|z_i, z_j) = p_{z_i, z_j}$ , and the  $p_{u,v}$  for all  $u, v \in \{1, 2, \dots, C\}$  form a probability matrix  $\mathbf{P}$  and are free parameters to be fitted.

In our model, we assume the node types are the class labels, i.e.,  $C = K$ ,  $z_i = \arg \max \mathbf{y}_i$ , and  $p_\theta(\mathbf{e}_{i,j}|\mathbf{X}, \mathbf{Y}) = p_\theta(\mathbf{e}_{i,j}|\mathbf{y}_i, \mathbf{y}_j)$ . Note that in our notation  $\mathbf{y}_i$  is a one-hot vector. We specify  $p_\theta(\mathbf{e}_{i,j}|\mathbf{y}_i, \mathbf{y}_j)$  with the simplest SBM, which is also called planted partition model. That is,

$$\mathbf{e}_{i,j}|\mathbf{y}_i, \mathbf{y}_j \sim \begin{cases} \text{Bernoulli}(p_0) & \text{if } \mathbf{y}_i = \mathbf{y}_j \\ \text{Bernoulli}(p_1) & \text{if } \mathbf{y}_i \neq \mathbf{y}_j \end{cases}.$$

This means the probability matrix  $\mathbf{P}$  are a constant  $p_0$  on the diagonal and another constant  $p_1$  off the diagonal.

### 3.3.2 Instantiations of the Approximate Posterior Model

For the approximate posterior model  $q_\phi(\mathbf{Y}_{\text{miss}}|\mathbf{X}, \mathbf{Y}_{\text{obs}}, \mathbf{G})$ , in principle we need a strong function approximator that takes  $(\mathbf{X}, \mathbf{Y}_{\text{obs}}, \mathbf{G})$  as the input and outputs the probability of  $\mathbf{Y}_{\text{miss}}$ . Here we consider two recently invented graph neural networks: Graph Convolutional Network (GCN) [9] and Graph Attention Network (GAT) [19]. Note that by doing this we are making a further approximation from  $q_\phi(\mathbf{Y}_{\text{miss}}|\mathbf{X}, \mathbf{Y}_{\text{obs}}, \mathbf{G})$  to  $q_\phi(\mathbf{Y}_{\text{miss}}|\mathbf{X}, \mathbf{G})$ , as the graph neural networks by design only take  $(\mathbf{X}, \mathbf{G})$  as the input.

### 3.4 Training

Finally, we end this section by introducing two practical details of model training.

**Supervised loss.** As our main task is to conduct classification with the approximate posterior model, similar with Kingma et al. [7], we add an additional supervised loss to better train the approximate posterior model:

$$\mathcal{L}_s(\phi; \mathbf{X}, \mathbf{Y}_{\text{obs}}, G) = -\log q_\phi(\mathbf{Y}_{\text{obs}}|\mathbf{X}, G).$$

The total loss is controlled by a weight hyper-parameter  $\eta$ ,

$$\mathcal{L}(\theta, \phi) = \mathcal{L}_{\text{ELBO}}(\theta, \phi; \mathbf{X}, \mathbf{Y}_{\text{obs}}, G) + \eta \cdot \mathcal{L}_s(\phi; \mathbf{X}, \mathbf{Y}_{\text{obs}}, G).$$

We could rewrite the total loss in an alternative way as follows,

$$\begin{aligned} \mathcal{L}(\theta, \phi) = & \mathbb{E}_{q_\phi(\mathbf{Y}_{\text{miss}}|\mathbf{X}, G)} -\log p_\theta(G|\mathbf{X}, \mathbf{Y}_{\text{obs}}, \mathbf{Y}_{\text{miss}}) + \mathcal{D}_{\text{KL}}(q_\phi(\mathbf{Y}_{\text{miss}}|\mathbf{X}, G) || p_\theta(\mathbf{Y}_{\text{miss}}|\mathbf{X})) \\ & -\log p_\theta(\mathbf{Y}_{\text{obs}}|\mathbf{X}) - \eta \cdot \log q_\phi(\mathbf{Y}_{\text{obs}}|\mathbf{X}, G), \end{aligned}$$

which provides a connection between the proposed generative framework and existing graph neural networks. The fourth term  $-\eta \cdot \log q_\phi(\mathbf{Y}_{\text{obs}}|\mathbf{X}, G)$  provides supervised information from labeled data for the approximate posterior GCN or GAT, while the other three terms can be viewed as additional regularizations: the learned GCN or GAT is encouraged to support the generative model and also not go far away from the marginal  $p_\theta(\mathbf{Y}|\mathbf{X})$  without information from the graph.

**Negative edge sampling.** For both LSM and SBM based models, the probability  $p_\theta(\mathbf{G}|\mathbf{X}, \mathbf{Y})$  factorizes to the production of probabilities of all possible edges. Calculating the log-likelihood of it requires enumeration of all possible  $(i, j)$  pairs where  $i, j \in \{1, 2, \dots, n\}$ , which results in a  $\mathcal{O}(n^2)$  computational cost at each epoch. In practice, instead of going through all  $(i, j)$  pairs, we only calculate the probabilities of the edges observed in the graph and a set of "negative edges" randomly sampled from the  $(i, j)$  pairs where edges do not exist. This practical trick, named negative sampling, is commonly used in the training of word embeddings [14] and graph embeddings [18].

## 4 Experiments

In this section, we evaluate the proposed framework on several benchmark datasets for graph-based semi-supervised learning. We test the models under both the standard benchmark setting [21] as well as two data-scarce settings.

### 4.1 Standard Benchmark Setting

We first consider a standard benchmark setting in recent graph-based semi-supervised learning works [21, 9, 19].

**Datasets.** We use three standard semi-supervised learning benchmark datasets for graph neural networks, Citeseer, Cora, and Pubmed [17, 21]. The graph  $G$  of each dataset is a citation network with documents as nodes and citations as edges. The feature vector of each node is a bag-of-words representation of the document and the class label represents the research area this document belongs to. We adopt these datasets from the PyTorch-Geometric library [3] in our experiments. We closely follow the dataset setup in Yang et al. [21] and Kipf and Welling [9].

**Models for comparison.** For the proposed framework, we implement four variants ( $2 \times 2$ ) of models by combining the two generative model instantiations with the two approximate posterior model instantiations: LSM-GCN, SBM-GCN, LSM-GAT, SBM-GAT.

Table 1: Classification accuracy under the standard benchmark setting. The **bold** marker denotes the best performance on each dataset. The underline marker denotes that the generative model outperforms its discriminative counterpart, e.g., LSM-GCN outperforms GCN; and the asterisk (\*) marker denotes the difference is statistically significant by a t-test at significance level 0.05. The ( $\pm$ ) error bar denotes the standard deviation of the test performance of 10 independent trials.

	Cora	Pubmed	Citeseer
MLP	0.583 $\pm$ 0.009	0.734 $\pm$ 0.002	0.569 $\pm$ 0.008
GCN	0.815 $\pm$ 0.002	<b>0.794</b> $\pm$ 0.004	0.718 $\pm$ 0.003
GAT	0.825 $\pm$ 0.005	0.785 $\pm$ 0.004	0.715 $\pm$ 0.007
LSM_GCN	<u>0.825</u> $\pm$ 0.002*	0.779 $\pm$ 0.004	<u>0.744</u> $\pm$ 0.003*
LSM_GAT	<b>0.829</b> $\pm$ 0.003	0.776 $\pm$ 0.007	<u>0.731</u> $\pm$ 0.005*
SBM_GCN	<u>0.822</u> $\pm$ 0.002*	0.784 $\pm$ 0.006	<b>0.745</b> $\pm$ 0.004*
SBM_GAT	<u>0.829</u> $\pm$ 0.003	0.774 $\pm$ 0.004	<u>0.740</u> $\pm$ 0.003*

For baselines, we compare against two state-of-the-art models for the graph-based semi-supervised learning methods, GCN [9] and GAT [19]. We also include a multi-layer perceptron (MLP), which is a fully connected neural network without using any graph information, as a reference.

We use the original architectures of GCN and GAT models in both the baselines and the proposed methods. We grid search the number of hidden units from (16, 32, 64) and the learning rate from (0.001, 0.005, 0.01). GAT uses a multi-head attention mechanism. In our experiments, we fix the number of heads as 8 and try to set the total number of hidden units as (16, 32, 64) and to set the number of hidden units of a single head as (16, 32, 64). In the proposed methods, we set the generative model for  $p_\theta(\mathbf{Y}|\mathbf{X})$  as a two-layer MLP having the same number of hidden units as the corresponding GCN or GAT in the posterior model. For the MLP baseline, we also set the number of layers as 2 and grid search the number of hidden units and learning rate like other models. For the proposed generative models, we grid search the coefficient of the supervised loss  $\eta$  from (0.5, 1, 10). The number of negative edges is set to be the same number as the observed edges in the graph. For LSM models, the dimensions of the feature transformation matrix  $U$  is fixed to  $8 \times d$ , where  $d$  is the feature size. For SBM models, we use two settings of  $(p_0, p_1)$ : (0.9, 0.1) and (0.5, 0.6). We use Adam optimizer to train all the models and apply early stopping with the cross-entropy loss on the validation set. We adopt the implementations of GCN and GAT from the PyTorch-Geometric library in all our experiments.

**Results.** The performance of the baselines and proposed models under the standard benchmark setting is summarized in Table 1. We report the mean and the standard deviation of the test accuracy of 10 independent trials for each model. The results show that on all datasets except for Pubmed, the proposed methods achieve the best test accuracy on the standard benchmark setting. Notably, every instantiation model of the proposed generative framework outperforms their corresponding discriminative baseline (GCN or GAT) in most cases. The proposed models do not outperform the baselines on Pubmed, perhaps because the edges are denser in the graph of Pubmed than those of Cora and Citeseer so the generative model can only offer limited help.

## 4.2 Data-Scarce Settings

Generative models usually have better sample efficiency than discriminative models. Therefore, we expect the proposed generative framework to show bigger advantage when data are scarce. Next, we evaluate the models on the citation datasets under two such settings: missing-edge setting and reduced-label setting.

### 4.2.1 Missing-Edge Setting

In the standard benchmark setting, we assume that all samples are connected to the graph identically and the training, validation, and test set are split randomly in the datasets. In practice, however, the samples we are interested in the test period may not be well connected to the graph. For example, we may not have connection information for new users in social networks yet other than their profile information. In this cold-start situation, one may expect to make predictions purely based on the

Table 2: Classification accuracy under the missing-edge setting. The **bold** marker, the underline marker, the asterisk (\*) marker, and the ( $\pm$ ) error bar share the same definitions in Table 1.

	Cora	Pubmed	Citeseer
MLP	$0.583 \pm 0.009$	$0.734 \pm 0.002$	$0.569 \pm 0.008$
GCN	$0.665 \pm 0.007$	$0.746 \pm 0.004$	$0.652 \pm 0.005$
GAT	$0.682 \pm 0.004$	$0.744 \pm 0.006$	$0.642 \pm 0.004$
LSM_GCN	<u><math>0.711 \pm 0.005^*</math></u>	<b><u><math>0.766 \pm 0.006^*</math></u></b>	<u><math>0.704 \pm 0.002^*</math></u>
LSM_GAT	<u><math>0.710 \pm 0.007^*</math></u>	<u><math>0.766 \pm 0.004^*</math></u>	<u><math>0.691 \pm 0.005^*</math></u>
SBM_GCN	<b><u><math>0.718 \pm 0.004^*</math></u></b>	<u><math>0.762 \pm 0.005^*</math></u>	<b><u><math>0.716 \pm 0.004^*</math></u></b>
SBM_GAT	<u><math>0.716 \pm 0.007^*</math></u>	<u><math>0.761 \pm 0.005^*</math></u>	<u><math>0.709 \pm 0.008^*</math></u>

Table 3: Classification accuracy under the reduced-label setting. The **bold** marker, the underline marker, the asterisk (\*) marker, and the ( $\pm$ ) error bar share the same definitions in Table 1.

	Cora	Pubmed	Citeseer
MLP	$0.498 \pm 0.004$	$0.674 \pm 0.005$	$0.493 \pm 0.010$
GCN	$0.750 \pm 0.003$	<b><math>0.724 \pm 0.005</math></b>	$0.666 \pm 0.003$
GAT	$0.771 \pm 0.004$	$0.711 \pm 0.006$	$0.675 \pm 0.005$
LSM_GCN	<u><math>0.777 \pm 0.002^*</math></u>	$0.709 \pm 0.003$	<u><math>0.691 \pm 0.005^*</math></u>
LSM_GAT	<u><math>0.792 \pm 0.004^*</math></u>	$0.699 \pm 0.003$	<u><math>0.691 \pm 0.004^*</math></u>
SBM_GCN	<u><math>0.780 \pm 0.002^*</math></u>	$0.710 \pm 0.004$	<b><u><math>0.703 \pm 0.006^*</math></u></b>
SBM_GAT	<b><u><math>0.796 \pm 0.008^*</math></u></b>	$0.699 \pm 0.003$	<u><math>0.698 \pm 0.003^*</math></u>

profile information. However, we believe that the relational information stored in the graph of the training data can still help us learn a better and more generalizable model even if some of the predictions are made only based on the node features. And we expect the proposed generative models to work better than the discriminative baselines in this case because they can better distill the relationship among the data.

To mimic such situations, we create an extreme missing-edge setting, where we remove all the edges of the test nodes from the graph. Note that this setting is different from the inductive learning setting in previous works [4, 19] where the edges for the test data are absent during the training stage but present during the test stage. In the missing-edge setting, the edges for the test data are absent during both stages. We follow the same experimental setup as in the standard benchmark setting except for the modification of the graph.

**Results.** The performance under the missing-edge setting is shown in Table 2. Not surprisingly, as we lose part of the graph information, the performances of all models except for MLP (which does not use the graph at all) drop compared to the standard benchmark setting in Table 1. However, the proposed generative models perform better than their corresponding discriminative baselines by a large margin. Remarkably, even without knowing any edges of test nodes, the accuracy of SBM-GCN on Citeseer can achieve the state-of-the-art level of GCN under the standard benchmark setting.

#### 4.2.2 Reduced-Label Setting

Another common data-scarce situation is the lack of labeled data. Therefore, we create a reduced-label setting, where we drop half of the training data for each class compared to the standard benchmark setting. All other experiment and model setups are the same as the standard benchmark setting.

**Results.** The performance under the reduced-label setting is shown in Table 3. As can be seen from the results, the proposed generative models achieve the best test accuracy on Cora and Citeseer again. And the gap of the performances between the proposed generative models and the corresponding discriminative models are larger on Cora and Citeseer.



## 5 Conclusion

In this paper, we have presented a flexible generative framework for graph-based semi-supervised learning. By applying scalable variational inference, this framework is able to take the advantages of both recently developed graph neural networks and the wisdom of random graph models from classical network science literature. We further implemented 4 simple instantiations of the proposed framework, where we use a latent space model and a stochastic block model as the generative models with GCN and GAT as the approximate posterior models. Through thorough experiments, We demonstrated that these instantiation models outperform the state-of-the-art graph-based semi-supervised learning methods on most benchmark datasets under the standard benchmark setting. We also showed that the proposed generative framework has great potential in data-scarce situations.

For future work, we expect more complex instantiations of generative models to be developed using this framework and to optimize the graph-based semi-supervised learning.

## References

- [1] Belkin, M., Niyogi, P., and Sindhwani, V. (2006). Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of machine learning research*, 7(Nov):2399–2434.
- [2] De Cao, N. and Kipf, T. (2018). Molgan: An implicit generative model for small molecular graphs. *arXiv preprint arXiv:1805.11973*.
- [3] Fey, M. and Lenssen, J. E. (2019). Fast graph representation learning with pytorch geometric. *arXiv preprint arXiv:1903.02428*.
- [4] Hamilton, W., Ying, Z., and Leskovec, J. (2017). Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pages 1024–1034.
- [5] Hoff, P. D., Raftery, A. E., and Handcock, M. S. (2002). Latent space approaches to social network analysis. *Journal of the american Statistical association*, 97(460):1090–1098.
- [6] Holland, P. W., Laskey, K. B., and Leinhardt, S. (1983). Stochastic blockmodels: First steps. *Social networks*, 5(2):109–137.
- [7] Kingma, D. P., Mohamed, S., Rezende, D. J., and Welling, M. (2014). Semi-supervised learning with deep generative models. In *Advances in neural information processing systems*, pages 3581–3589.
- [8] Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.
- [9] Kipf, T. N. and Welling, M. (2016a). Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.
- [10] Kipf, T. N. and Welling, M. (2016b). Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*.
- [11] Li, T., Levina, E., Zhu, J., et al. (2019). Prediction models for network-linked data. *The Annals of Applied Statistics*, 13(1):132–164.
- [12] Liu, B. (2019). *Statistical learning for networks with node features*. PhD thesis, University of Michigan.
- [13] Mei, Q., Zhang, D., and Zhai, C. (2008). A general optimization framework for smoothing language models on graph structures. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 611–618. ACM.
- [14] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.
- [15] Newman, M. (2010). *Networks: an introduction*. Oxford university press.

- [16] Ng, Y. C., Colombo, N., and Silva, R. (2018). Bayesian semi-supervised learning with graph gaussian processes. In *Advances in Neural Information Processing Systems*, pages 1683–1694.
- [17] Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., and Eliassi-Rad, T. (2008). Collective classification in network data. *AI magazine*, 29(3):93–93.
- [18] Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., and Mei, Q. (2015). Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web*, pages 1067–1077. International World Wide Web Conferences Steering Committee.
- [19] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. (2017). Graph attention networks. *arXiv preprint arXiv:1710.10903*.
- [20] Weston, J., Ratle, F., Mobahi, H., and Collobert, R. (2012). Deep learning via semi-supervised embedding. In *Neural Networks: Tricks of the Trade*, pages 639–655. Springer.
- [21] Yang, Z., Cohen, W. W., and Salakhutdinov, R. (2016). Revisiting semi-supervised learning with graph embeddings. *arXiv preprint arXiv:1603.08861*.
- [22] Zhang, Y., Pal, S., Coates, M., and Üstebay, D. (2018). Bayesian graph convolutional neural networks for semi-supervised classification. *arXiv preprint arXiv:1811.11103*.
- [23] Zhou, D., Bousquet, O., Lal, T. N., Weston, J., and Schölkopf, B. (2004). Learning with local and global consistency. In *Advances in neural information processing systems*, pages 321–328.
- [24] Zhu, X., Ghahramani, Z., and Lafferty, J. D. (2003). Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the 20th International conference on Machine learning (ICML-03)*, pages 912–919.