Bayesian Graph Convolutional Neural Networks using Node Copying

Soumyasundar Pal [*1] **Florence Regol** [*1] **Mark Coates** [1]

Abstract

Graph convolutional neural networks (GCNN) have numerous applications in different graph based learning tasks. Although the techniques obtain impressive results, they often fall short in accounting for the uncertainty associated with the underlying graph structure. In the recently proposed Bayesian GCNN (BGCN) framework, this issue is tackled by viewing the observed graph as a sample from a parametric random graph model and targeting joint inference of the graph and the GCNN weights. In this paper, we introduce an alternative generative model for graphs based on copying nodes and incorporate it within the BGCN framework. Our approach has the benefit that it uses information provided by the node features and training labels in the graph topology inference. Experiments show that the proposed algorithm compares favourably to the state-of-the-art in benchmark node classification tasks.

1. Introduction

Recently, there has been an increased research focus on graph convolutional neural networks (GCNNs) due to their successful application in various graph based learning problems such as node and graph classification, matrix completion, and learning of node embeddings. Prior work leading to the development of GCNNs includes (Bruna et al., 2013; Henaff et al., 2015; Duvenaud et al., 2015). (Defferrard et al., 2016) propose an approach based on spectral filtering which is also followed in (Levie et al., 2019; Chen et al., 2018a; Kipf & Welling, 2017). Other works (Atwood & Towsley, 2016; Hamilton et al., 2017) consider spatial filtering and aggregation strategies. A general framework for learning on graphs and manifolds with neural networks is derived in (Monti et al., 2017) and this includes various other existing methods as special cases.

Several modifications can improve the performance of the GCNN, including adding attention nodes (Veličković et al., 2018), gates (Li et al., 2016c; Bresson & Laurent, 2017), edge conditioning and skip connections (Sukhbaatar et al., 2016; Simonovsky & Komodakis, 2017). Other approaches involve the use of graph ensembles (Anirudh & Thiagarajan, 2017), multiple adjacency matrices (Such et al., 2017), the dual graph (Monti et al., 2018), or random perturbation (Sun et al., 2019). Employing localized sampling methods (Hamilton et al., 2017), importance sampling (Chen et al., 2018a) or control variate based stochastic approximation (Chen et al., 2018b) has been shown to improve the scalability of these methods for processing large graphs.

The majority of the existing approaches process the graph as the ground truth. However, in many practical settings, the graph is often derived from noisy data or inaccurate modelling assumptions. As a result, spurious edges may be present or edges between very similar nodes might be omitted. This can lead to deterioration in the performance of the learning algorithms. Various existing approaches like the graph attention network (Veličković et al., 2018) and graph ensemble based approach (Anirudh & Thiagarajan, 2017) address this issue partially. Nevertheless, neither of these methods has the flexibility to add edges that could be missing from the observed graph. A principled way to address the uncertainty in the graph structure is to consider the graph as a random sample drawn from a probability distribution over graphs. The Bayesian framework of (Zhang et al., 2019) proposes to use a parametric random graph model as the generative model of the graph and formulates the learning task as the inference of the joint posterior distribution of the graph and the weights of the GCNN. Despite the effectiveness of the approach, the choice of a suitable random graph model is crucial and heavily dependent on the learning task and datasets. Furthermore, the method in (Zhang et al., 2019) conducts the posterior inference of the graph solely conditioned on the observed graph topology. This results in a complete disregard of any information provided by the node features and the training labels, which is undesirable if these data are highly correlated with the true graph structure.

In this paper, we introduce a novel generative model for graphs based on copying nodes from one location to another. While this idea is similar to the full duplication process presented in (Chung et al., 2003), we do not grow the graph since we only copy existing nodes rather than adding new ones. This results in a formulation in which the posterior inference of the graph is carried out conditioned on the features and training labels as well as the observed graph topology. Experimental results demonstrate the efficacy of

[*]Equal contribution [1]Dept. of Electrical and Computer Engineering, McGill University, Montréal, Canada. Correspondence to: Soumyasundar Pal <soumyasundar.pal@mail.mcgill.ca>.

Presented at the ICML 2019 Workshop on Learning and Reasoning with Graph-Structured Data.

our approach for the semi-supervised node classification task, particularly if a limited number of training labels is available. The rest of the paper is organized as follows. We provide a brief review of the GCNN in Section 2 and present the proposed approach in Section 3. We report the results of the numerical experiments in Section 4 and make concluding remarks in Section 5.

2. Graph convolutional neural networks

Although graph convolutional neural networks are suitable for a variety of learning tasks, here we restrict ourselves to the discussion of the node classification problem on a graph for brevity. In this setting, an observed graph $\mathcal{G}_{obs} = (\mathcal{V}, \mathcal{E})$ is available, where $\mathcal{V}$ is the set of N nodes and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ denotes the set of edges. There is a feature vector $\boldsymbol{x}_i \in \mathbf{R}^{d\times 1}$ associated with each node i and its class label is denoted by $\boldsymbol{y}_i$. The labels are known only for the nodes in the training set $\mathcal{L} \subset \mathcal{V}$. The goal is to predict the labels of the remaining nodes using the information provided by the observed graph $\mathcal{G}_{obs}$, the feature matrix $\mathbf{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N]^T$ and the training labels $\mathbf{Y}_{\mathcal{L}} = \{\boldsymbol{y}_i : i \in \mathcal{L}\}$.

In a GCNN, learning is performed using graph convolution operations within a neural network architecture. A layerwise propagation rule for the simpler architectures (Defferrard et al., 2016; Kipf & Welling, 2017) is written as:

$$\mathbf{H}^{(1)} = \sigma(\hat{\mathbf{A}}_{\mathcal{G}}\mathbf{X}\mathbf{W}^{(0)})\,, \tag{1}$$

$$\mathbf{H}^{(l+1)} = \sigma(\hat{\mathbf{A}}_{\mathcal{G}}\mathbf{H}^{(l)}\mathbf{W}^{(l)})\,. \tag{2}$$

The normalized adjacency operator $\hat{\mathbf{A}}_{\mathcal{G}}$ is derived from the observed graph and it controls the aggregation of the output features across the neighbouring nodes at each layer. σ denotes a pointwise non-linear activation function and $\mathbf{H}^{(l)}$ are the output features from layer $l-1$. $\mathbf{W}^{(l)}$ represents the weights of the neural network at layer l. We use $\mathbf{W} = \{\mathbf{W}^l\}_{l=1}^L$ to denote the collection of GCNN weights across all layers. In an L-layer network, the final output is collected from the last layer $\mathbf{Z} = \mathbf{H}^{(L)}$. The weights of the neural network $\mathbf{W}$ are learned via backpropagation with the objective of minimizing an error metric between the training labels $\mathbf{Y}_{\mathcal{L}}$ and the network predictions $\mathbf{Z}_{\mathcal{L}} = \{\boldsymbol{z}_i : i \in \mathcal{L}\}$ at the nodes in the training set.

3. Methodology

In the Bayesian paradigm, the observed graph is viewed as a random quantity and the posterior inference for the underlying graph is required. We postulate a model which allows sampling of a random graph by copying the observed graph and then replacing each node's edges with a high probability by the edges of a similar node randomly selected from the observed graph, while the node features remain unchanged.

3.1. Node-Copying Graph Model

In order to sample graph $\mathcal{G}$ from the proposed model, we introduce an auxiliary random vector $\boldsymbol{\zeta} \in \{1, 2, \ldots N\}^N$, where the j'th entry ζ^j denotes the node whose edges are to replace the edges of the j'th node in the observed graph. The entries in $\boldsymbol{\zeta}$ are assumed to be mutually independent. For sampling the ζ^js, we use a base classification algorithm using the observed graph $\mathcal{G}_{obs}$, the features $\mathbf{X}$ and the training labels $\mathbf{Y}_{\mathcal{L}}$ to obtain labels $\hat{c}_\ell \in \{1, 2, \ldots K\}$ for each node ℓ in the graph. Then for each class $1 \le k \le K$, we collect the nodes with predicted label k into the set $\mathcal{C}_k$:

$$\mathcal{C}_k = \{\ell \mid 1 \le \ell \le N, \hat{c}_\ell = k\}\,. \tag{3}$$

We define the posterior distribution of $\boldsymbol{\zeta}$ as follows:

$$p(\boldsymbol{\zeta}|\mathcal{G}_{obs}, \mathbf{X}, \mathbf{Y}_{\mathcal{L}}) = \prod_{j=1}^{N} p(\zeta^j|\mathcal{G}_{obs}, \mathbf{X}, \mathbf{Y}_{\mathcal{L}})\,,$$

$$p(\zeta^j = m|\mathcal{G}_{obs}, \mathbf{X}, \mathbf{Y}_{\mathcal{L}}) = \begin{cases} \dfrac{1}{|\mathcal{C}_k|}, & \text{if } \hat{c}_j = \hat{c}_m = k \\ 0, & \text{otherwise}\,, \end{cases} \tag{4}$$

for $1 \le j, m \le N$ and $1 \le k \le K$. Sampling ζ^j from this model boils down to selecting a node at random from the collection of nodes that have the same predictive label as the j'th node. Conditioned on $\boldsymbol{\zeta}$ and the observed graph $\mathcal{G}_{obs}$, the sampling of graph $\mathcal{G}$ is carried out by copying the ζ^j'th node of $\mathcal{G}_{obs}$ in the place of the j'th node of $\mathcal{G}$, independently for all $1 \le j \le N$ with a high probability. More formally, the generative model is given as:

$$p(\mathcal{G}|\mathcal{G}_{obs}, \boldsymbol{\zeta}) = \prod_{j=1}^{N} \epsilon^{\mathbb{1}_{\{\mathcal{G}_j = \mathcal{G}_{obs,j}\}}} (1-\epsilon)^{\mathbb{1}_{\{\mathcal{G}_j = \mathcal{G}_{obs,\zeta^j}\}}}\,, \tag{5}$$

where, $0 < \epsilon \ll 1$ is a hyperparameter and $\mathbb{1}_{\{\mathcal{G}_j = \mathcal{G}_{obs,q}\}}$ denotes the indicator function of copying q'th node of $\mathcal{G}_{obs}$ in place of the j'th node of $\mathcal{G}$. The copying operation involves changing the set of neighbours of the j'th node of $\mathcal{G}$ to be the same as the set of neighbours of the q'th node of $\mathcal{G}_{obs}$.

3.2. Bayesian Graph Convolutional Neural Networks

As in (Zhang et al., 2017), we compute the marginal posterior probability of the node labels via marginalization with respect to the graph and the GCNN weights.

$$p(\mathbf{Z}|\mathbf{Y}_{\mathcal{L}}, \mathbf{X}, \mathcal{G}_{obs}) = \int p(\mathbf{Z}|\mathbf{W}, \mathcal{G}_{obs}, \mathbf{X}) p(\mathbf{W}|\mathbf{Y}_{\mathcal{L}}, \mathbf{X}, \mathcal{G})$$
$$p(\mathcal{G}|\mathcal{G}_{obs}, \boldsymbol{\zeta}) p(\boldsymbol{\zeta}|\mathcal{G}_{obs}, \mathbf{Y}_{\mathcal{L}}, \mathbf{X})\, d\mathbf{W}\, d\mathcal{G}\, d\boldsymbol{\zeta}\,. \tag{6}$$

Here $\mathbf{W}$ denotes the random weights of a Bayesian GCNN over the graph $\mathcal{G}$ and $\boldsymbol{\zeta}$ is an N-dimensional random vector associated with the proposed node copying model. In a node classification problem with K classes, the term

Algorithm 1 Bayesian GCN with node copying

Input: $\mathcal{G}_{obs}$, $\mathbf{X}$, $\mathbf{Y}_{\mathcal{L}}$
Output: $p(\mathbf{Z}|\mathbf{Y}_{\mathcal{L}}, \mathbf{X}, \mathcal{G}_{obs})$
Initialization: train a base classifier to obtain $\hat{c}_\ell$ for $1 \leq \ell \leq N$, form $\mathcal{C}_k$ using eq. (3) for $1 \leq k \leq K$.
for $v = 1$ **to** V **do**
  Sample $\boldsymbol{\zeta}_v \sim p(\boldsymbol{\zeta}|\mathcal{G}_{obs}, \mathbf{X}, \mathbf{Y}_{\mathcal{L}})$ using eq. (4).
  for $i = 1$ **to** N_G **do**
    Sample graph $\mathcal{G}_{i,v} \sim p(\mathcal{G}|\mathcal{G}_{obs}, \boldsymbol{\zeta}_v)$ using eq. (5).
    for $s = 1$ **to** S **do**
      Sample weights $W_{s,i,v}$ using MC dropout by training a GCNN over the graph $\mathcal{G}_{i,v}$.
    end for
  end for
end for
Approximate $p(\mathbf{Z}|\mathbf{Y}_{\mathcal{L}}, \mathbf{X}, \mathcal{G}_{obs})$ using eq. (7).

$p(\mathbf{Z}|\mathbf{Y}_{\mathcal{L}}, \mathbf{X}, \mathcal{G}_{obs})$ is modelled using a K-dimensional categorical distribution by applying a softmax function to the output of the GCNN. In (Zhang et al., 2019), $\mathcal{G}_{obs}$ is viewed as a sample realization from a collection of graphs associated with a parametric random graph model and posterior inference of $p(\mathcal{G}|\mathcal{G}_{obs})$ is targeted via marginalization of the random graph parameters. Their approach thus ignores any possible dependence of the graph $\mathcal{G}$ on the features $\mathbf{X}$ and the labels $\mathbf{Y}_{\mathcal{L}}$. By contrast, our approach models the marginal posterior distribution of the graph $\mathcal{G}$ as $p(\mathcal{G}|\mathcal{G}_{obs}, \mathbf{X}, \mathbf{Y}_{\mathcal{L}})$. This allows us to incorporate the information provided by the features $\mathbf{X}$ and the training labels $\mathbf{Y}_{\mathcal{L}}$ in the graph inference process. The integral in equation (6) is not analytically tractable. Hence, a Monte Carlo approximation is formed as follows:

$$p(\mathbf{Z}|\mathbf{Y}_{\mathcal{L}}, \mathbf{X}, \mathcal{G}_{obs}) \approx \frac{1}{V}\sum_{v=1}^{V}\frac{1}{N_G S}\sum_{i=1}^{N_G}\sum_{s=1}^{S} p(\mathbf{Z}|\mathbf{W}_{s,i,v}, \mathcal{G}_{obs}, \mathbf{X}). \quad (7)$$

In this approximation, V samples $\boldsymbol{\zeta}_v$ are drawn from $p(\boldsymbol{\zeta}|\mathcal{G}_{obs}, \mathbf{Y}_{\mathcal{L}}, \mathbf{X})$. The N_G graphs $\mathcal{G}_{i,v}$ are sampled from $p(\mathcal{G}|\mathcal{G}_{obs}, \boldsymbol{\zeta}_v)$ and subsequently S weight matrices $\mathbf{W}_{s,i,v}$ are sampled from $p(\mathbf{W}|\mathbf{Y}_{\mathcal{L}}, \mathbf{X}, \mathcal{G}_{i,v})$ from the Bayesian GCN corresponding to the graph $\mathcal{G}_{i,v}$.

Sampling graphs from the node-copying model in Section 3.1 has several advantages compared to the graph inference technique based on mixed membership stochastic block models (MMSBMs) (Airoldi et al., 2009), which was adopted in (Zhang et al., 2019). First, the sampling of $\boldsymbol{\zeta}$ is computationally much cheaper than the inference of parameters of the parametric model, which becomes more severe as the size of the graph increases. Second, it is in general extremely difficult to carry out accurate inference for high dimensional MMSBM parameters (Li et al., 2016b) and inaccuracies in parameter estimates results in sampling of graphs which are very different from the observed graph. This can impact classification performance adversely, particularly if the observed graph does not fit the MMSBM well. However, for the proposed copying model, the similarity between the sampled graph and the observed graph depends mostly on the performance of the base classifier. If a state-of-the-art graph based classification method (e.g., GCNN) is used, we can obtain more representative graph samples from this model, particularly for large graphs. The expected graph edit distance between the random graphs and the observed graph can be controlled by the choice of the parameter ϵ. A low value of ϵ is chosen since it causes high variability among the random graph samples which was found to be effective empirically. Third, sampling a graph from the MMSBM scales as $\mathcal{O}(N^2)$ whereas the proposed method offers $\mathcal{O}(N)$ complexity.

For the Bayesian inference of GCNN weights, we can use various techniques including expectation propagation (Hernández-Lobato & Adams, 2015), variational inference (Gal & Ghahramani, 2016; Sun et al., 2017; Louizos & Welling, 2017), and Markov Chain Monte Carlo methods (Neal, 1993; Korattikara et al., 2015; Li et al., 2016a). Similar to (Zhang et al., 2019), we train a GCNN on $\mathcal{G}_{i,v}$ and use Monte Carlo dropout (Gal & Ghahramani, 2016) to sample $\mathbf{W}_{s,i,v}$. This is equivalent to sampling the weights from a variational approximation of $p(\mathbf{W}|\mathbf{Y}_{\mathcal{L}}, \mathbf{X}, \mathcal{G}_{i,v})$, with a particular structure. The resulting algorithm is summarized in Algorithm 1.

4. Numerical Experiments and Results

We address a semi-supervised node classification task for three citation networks (Sen et al., 2008): Cora, CiteSeer, and Pubmed. In these datasets each node represents a scientific publication and an undirected edge is formed between two nodes if any one of them cites the other. Each node has a sparse bag-of-words feature vector and the label describes the topic of the document. During training, we have access to the labels of only a few nodes per class and the goal is to infer labels for the other nodes.

We consider two different strategies for splitting the data into training and test sets, as specified in (Zhang et al., 2019). In the first setting, we use the fixed split from (Yang et al., 2016), which contains 20 labels per class in the training set. For the cases with 5 and 10 training labels per class in the fixed split scenario, the first 5 and 10 labels in the original partition of (Yang et al., 2016) are used. The second type of split is constructed by sampling the training and test sets randomly for each trial. Since a specific split of data can impact the classification performance significantly, random splitting provides a more robust comparison of performance of the algorithms.

We compare the proposed BGCN in this paper with ChebyNet (Defferrard et al., 2016), GCNN (Kipf & Welling, 2017), GAT (Veličković et al., 2018) and the BGCN in (Zhang et al., 2019). The hyperparameters of GCNN are set according to (Kipf & Welling, 2017) and the same values are used for the BGCN algorithms as well. For the proposed BGCN, we use GCNN (Kipf & Welling, 2017) as the base classification method. For both splitting strategies, each algorithm is run for 50 trials with random weight initializations. The average accuracies for Cora, Citeseer and Pubmed datasets along with their standard errors are reported in Table 1, 2 and 3 respectively.

Random split	**5 labels**	**10 labels**	**20 labels**
ChebyNet	61.7±6.8	72.5±3.4	78.8±1.6
GCNN	70.0±3.7	76.0±2.2	79.8±1.8
GAT	70.4±3.7	76.6±2.8	79.9±1.8
BGCN	**74.6±2.8**	77.5±2.6	80.2±1.5
BGCN (ours)	73.8±2.7	**77.6±2.6**	**80.3±1.6**
Fixed split			
ChebyNet	67.9±3.1	72.7±2.4	80.4±0.7
GCNN	74.4±0.8	74.9±0.7	**81.6±0.5**
GAT	73.5±2.2	74.5±1.3	81.6±0.9
BGCN	**75.3±0.8**	76.6±0.8	81.2±0.8
BGCN (ours)	75.1±1.3	**76.7±0.7**	81.4±0.6

Table 1. Classification accuracy (in %) for Cora dataset.

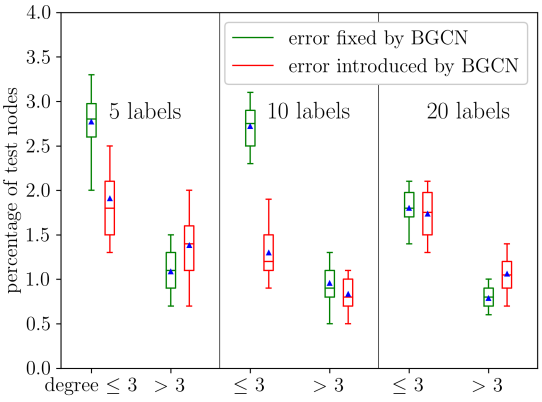
Random split	**5 labels**	**10 labels**	**20 labels**
ChebyNet	58.5±4.8	65.8±2.8	67.5±1.9
GCNN	58.5±4.7	65.4±2.6	67.8±2.3
GAT	56.7±5.1	64.1±3.3	67.6±2.3
BGCN	63.0±4.8	**69.9±2.3**	**71.1±1.8**
BGCN (ours)	**63.9±4.2**	68.5±2.3	70.2±2.0
Fixed split			
ChebyNet	53.0±1.9	67.7±1.2	70.2±0.9
GCNN	55.4±1.1	65.8±1.1	70.8±0.7
GAT	55.4±2.6	66.1±1.7	70.8±1.0
BGCN	57.3±0.8	**70.8±0.6**	**72.2±0.6**
BGCN (ours)	**61.4±2.3**	69.6±0.6	71.9±0.6

Table 2. Classification accuracy (in %) for Citeseer dataset.

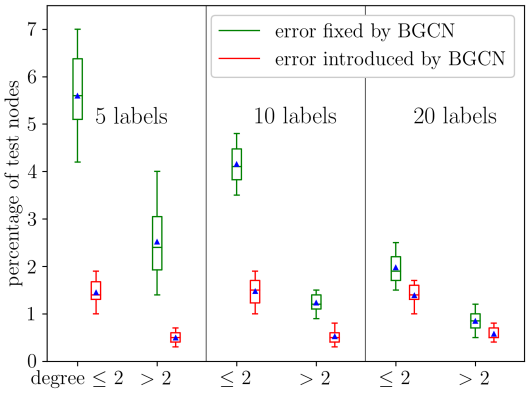
Random split	**5 labels**	**10 labels**	**20 labels**
ChebyNet	62.7±6.9	68.6±5.0	74.3±3.0
GCNN	69.7±4.5	73.9±3.4	77.5±2.5
GAT	68.0±4.8	72.6±3.6	76.4±3.0
BGCN	70.2±4.5	73.3±3.1	76.0±2.6
BGCN (ours)	**71.0±4.2**	**74.6±3.3**	**77.5±2.4**
Fixed split			
ChebyNet	68.1±2.5	69.4±1.6	76.0±1.2
GCNN	69.7±0.5	72.8±0.5	78.9±0.3
GAT	70.0±0.6	71.6±0.9	76.9±0.5
BGCN	70.9±0.8	72.3±0.8	76.6±0.7
BGCN (ours)	**71.2±0.5**	**73.6±0.5**	**79.1±0.4**

Table 3. Classification accuracy (in %) for Pubmed dataset.

We observe that the proposed BGCN algorithm obtains higher classification accuracy compared to its competitors in most cases. The improvement in accuracy compared to GCNN is more significant when the number of available labels is limited to 5 or 10. From Figure 1, we observe that in most cases, for the Cora and the Citeseer datasets, the proposed BGCN algorithm corrects more errors of the GCNN base classifier for nodes with lower degree.



(a)



(b)

Figure 1. Boxplot of different categories of nodes in the (a) Cora and (b) Citeseer datasets based on the classification results of the GCNN and the proposed BGCN algorithms. The two groups are formed by thresholding the degree of the nodes in the test set at the median value. The box shows 25-75 percentiles; the triangle represents the mean value; and the median is indicated by a horizontal line. Whiskers are drawn at the 5 and 95 percentiles of data points.

5. Conclusion

In this paper, we present a Bayesian GCNN using a node copying based generative model for graph. The proposed algorithm exhibits superior performance in the semi-supervised node classification task when the amount of available labels for training is limited. Future work will involve conducting a more thorough experimental evaluation and exploring ways to extend the methodology to other graph based learning tasks.

References

Airoldi, E. M., Blei, D. M., Fienberg, S. E., and Xing, E. P. Mixed membership stochastic blockmodels. In *Proc. Adv. Neural Inf. Proc. Systems*, pp. 33–40, 2009.

Anirudh, R. and Thiagarajan, J. J. Bootstrapping graph convolutional neural networks for autism spectrum disorder classification. *arXiv:1704.07487*, 2017.

Atwood, J. and Towsley, D. Diffusion-convolutional neural networks. In *Proc. Adv. Neural Inf. Proc. Systems*, 2016.

Bresson, X. and Laurent, T. Residual gated graph convnets. *arXiv:1711.07553*, 2017.

Bruna, J., Zaremba, W., Szlam, A., and LeCun, Y. Spectral networks and locally connected networks on graphs. In *Proc. Int. Conf. Learning Representations*, Scottsdale, AZ, USA, 2013.

Chen, J., Ma, T., and Xiao, C. FastGCN: fast learning with graph convolutional networks via importance sampling. In *Proc. Int. Conf. Learning Representations*, 2018a.

Chen, J., Zhu, J., and Song, L. Stochastic training of graph convolutional networks with variance reduction. In *Proc. Int. Conf. Machine Learning*, 2018b.

Chung, F., Lu, L., Dewey, T. G., and Galas, D. J. Duplication models for biological networks. *J. of Computat. Biology*, 10(5):677–687, 2003.

Defferrard, M., Bresson, X., and Vandergheynst, P. Convolutional neural networks on graphs with fast localized spectral filtering. In *Proc. Adv. Neural Inf. Proc. Systems*, 2016.

Duvenaud, D., Maclaurin, D., et al. Convolutional networks on graphs for learning molecular fingerprints. In *Proc. Adv. Neural Inf. Proc. Systems*, 2015.

Gal, Y. and Ghahramani, Z. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *Proc. Int. Conf. Machine Learning*, 2016.

Hamilton, W., Ying, R., and Leskovec, J. Inductive representation learning on large graphs. In *Proc. Adv. Neural Inf. Proc. Systems*, 2017.

Henaff, M., Bruna, J., and LeCun, Y. Deep convolutional networks on graph-structured data. *arXiv:1506.05163*, 2015.

Hernández-Lobato, J. M. and Adams, R. Probabilistic backpropagation for scalable learning of Bayesian neural networks. In *Proc. Int. Conf. Machine Learning*, 2015.

Kipf, T. and Welling, M. Semi-supervised classification with graph convolutional networks. In *Proc. Int. Conf. Learning Representations*, 2017.

Korattikara, A., Rathod, V., Murphy, K., and Welling, M. Bayesian dark knowledge. In *Proc. Adv. Neural Inf. Proc. Systems*, 2015.

Levie, R., Monti, F., Bresson, X., and Bronstein, M. M. Cayleynets: Graph convolutional neural networks with complex rational spectral filters. *IEEE Trans. Signal Processing*, 67(1), Jan. 2019.

Li, C., Guo, X., and Mei, Q. Deepgraph: Graph structure predicts network growth. *arXiv:1610.06251*, 2016a.

Li, W., Ahn, S., and Welling, M. Scalable MCMC for mixed membership stochastic blockmodels. In *Proc. Artificial Intelligence and Statistics*, pp. 723–731, 2016b.

Li, Y., Tarlow, D., Brockschmidt, M., and Zemel, R. Gated graph sequence neural networks. In *Proc. Int. Conf. Learning Representations*, 2016c.

Louizos, C. and Welling, M. Multiplicative normalizing flows for variational Bayesian neural networks. *arXiv:1703.01961*, 2017.

Monti, F., Boscaini, D., et al. Geometric deep learning on graphs and manifolds using mixture model CNNs. In *Proc. IEEE Conf. Comp. Vision and Pattern Recognition*, Jul. 2017.

Monti, F., Shchur, O., et al. Dual-primal graph convolutional networks. *arXiv:1806.00770*, 2018.

Neal, R. M. Bayesian learning via stochastic dynamics. In *Proc. Adv. Neural Inf. Proc. Systems*, pp. 475–482, 1993.

Sen, P., Namata, G., et al. Collective classification in network data. *AI Magazine*, 29(3):93, 2008.

Simonovsky, M. and Komodakis, N. Dynamic edge-conditioned filters in convolutional neural networks on graphs. *arXiv:1704.02901*, 2017.

Such, F., Sah, S., et al. Robust spatial filtering with graph convolutional neural networks. *IEEE J. Sel. Topics Signal Proc.*, 11(6):884–896, Sept 2017.

Sukhbaatar, S., Szlam, A., and Fergus, R. Learning multiagent communication with backpropagation. In *Proc. Adv. Neural Inf. Proc. Systems*, 2016.

Sun, K., Koniusz, P., and Wang, J. Fisher-Bures adversary graph convolutional networks. *arXiv e-prints, arXiv : 1903.04154*, 2019.

Sun, S., Chen, C., and Carin, L. Learning structured weight uncertainty in Bayesian neural networks. In *Proc. Artificial Intelligence and Statistics*, 2017.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph attention networks. In *Proc. Int. Conf. Learning Representations*, Vancouver, Canada, Apr. 2018.

Yang, Z., Cohen, W. W., and Salakhutdinov, R. Revisiting semi-supervised learning with graph embeddings. *arXiv preprint arXiv:1603.08861*, 2016.

Zhang, X., Moore, C., and Newman, M. Random graph models for dynamic networks. *Eur. Phys. J. B*, pp. 90–200, 2017.

Zhang, Y., Pal, S., Coates, M., and Üstebay, D. Bayesian graph convolutional neural networks for semi-supervised classification. In *Proc. AAAI Conf. Artificial Intelligence*, 2019.