Drop "Noise" Edge: An Approximation of the Bayesian GNNs

Xiaoling Zhou and Ou Wu[✉]

National Center for Applied Mathematics, Tianjin University, Tianjin 300072, China
{xiaolingzhou,wuou}@tju.edu.cn

Abstract. Graph neural networks (GNNs) have proven to be powerful tools for graph analysis. The key idea is to recursively propagate and gather information along the edges of a given graph. Although they have been successful, they are still limited by over-smoothing and noise in the graph. Over-smoothing means that the representation of each node will converge to the similar value as the number of layers increases. "Noise" edges refer to edges with no positive effect on graph representation in this study. To solve the above problems, we propose DropNEdge (Drop "Noise" Edge), which filters useless edges based on two indicators, namely, feature gain and signal-to-noise ratio. DropNEdge can alleviate over-smoothing and remove "noise" edges in the graph effectively. It does not require any changes to the network's structure, and it is widely adapted to various GNNs. We also show that the use of DropNEdge in GNNs can be interpreted as an approximation of the Bayesian GNNs. Thus, the models' uncertainty can be obtained.

Keywords: Drop "Noise" Edge · Over-smoothing · Bayesian GNNs

1 Introduction

Graph neural networks (GNNs) and their many variants have achieved success in graph representation learning by extracting high-level features of nodes from their topological neighborhoods. However, several studies have shown that the performances of GNNs decrease significantly with the increase in the number of neural network layers [15,16]. The reason is that the nodes' characteristics will converge to similar values with the continuous aggregation of information. Some existing methods (DropEdge [2], Dropout [3]) solve over-smoothing by dropping some information in the graph randomly. Although these methods are efficient, they cannot guarantee that the dropped information is harmful or beneficial. Hence, they can only bring sub-optimal effect.

In addition, the effect of GNNs is affected by noise edges [1,17]. Many graphs in real world have noise edges which requires GNNs to have the ability to identify

Supported by ZJFund 2019KB0AB03, NSFC 62076178, TJ-NSF (19JCZDJC31300, 19ZXAZNGX00050).

and remove noise edges. The recursive aggregation mode of GNNs makes it susceptible to the influence of surrounding nodes. Therefore, finding a principled way to decide what information not to aggregate will have a positive effect on GNNs' performance. Topological denoising [1] is an effective solution to solve this problem by removing noise edges. We can trim off the edges with no positive impact on the task to avoid GNNs from aggregating unnecessary information.

In this paper, we propose DropNEdge (Drop "Noise" Edge), which takes the structure and content information of a graph as input and deletes edges with no or little positive effect on the final task based on the node's signal-to-noise ratio and feature gain. The differences between our method and DropEdge are detailed as follows. First, DropNEdge treats the edges unequally and deletes edges based on the graph's information, which is a more reasonable and effective method to solve the limitations of GNNs. Second, deleting edges from the above two aspects can ensure that the dropped edges have no or little positive effect on the final task. Therefore, it can not only alleviate over-smoothing, but also remove "noise" edges. DropNEdge is widely adapted to most GNNs and does not need to change the networks' structure. Because DropNEdge changes the topology of the graph, it can be used as a graphical data enhancement method.

Considering that Dropout can be used as a Bayesian approximation for general neural networks, we prove that DropNEdge can be used as a Bayesian approximation for GNNs. If we use DropNEdge during the training and test phase, then the models' uncertainty can be obtained.

The main contributions of our work are presented as follows:

- We propose DropNEdge, which is a plug-and-play layer that is widely adapted to various GNNs. It can effectively alleviate the over-smoothing phenomenon and remove "noise" edges in the graph.
- We show that the use of DropNEdge in GNNs is an approximation of the Bayesian GNNs. In this way, the uncertainty of GNNs can be obtained.

2 Related Work

Deep stacking of layers usually results in a significant decrease in the performance of GNNs, such as GCN [13] and GAT [14]. Chen et al. [5] measured and alleviated the over-smoothing problem of GNNs from a topological perspective. Hou et al. [6] proposed two over-smoothing indicators to measure the quantity and quality of information obtained from graphic data and designed a new GNN model called CS-GNN. To prevent node embeddings from being too similar, PairNorm [7] was proposed which is a normalization layer based on the analysis of graph convolution operations. DropEdge [2] also effectively relieves the over-smoothing phenomenon by randomly removing a given percentage of edges in the graph.

Another limitation is the noise in the graph. A large number of papers show that GNNs are not robust to noise. Recently, graph sampling has been investigated in GNNs for the rapid calculation and to improve the generalization ability of GNNs, including neighbor-level [8], node-level [9] and edge-level sampling methods [10]. Unlike these methods that randomly sample edges during the training phase, PTDNet [4] uses a parametric network to actively remove "noise" edges for specific tasks. Moreover, it has been proved that the graph data enhancement strategy can effectively improve the robustness of GNNs [17].

Bayesian network is a hot topic which is critical for many machine learning systems. Since exact Bayesian inference is intractable, many approximation methods have been proposed such as Laplace approximation [20], Markov chain Monte Carlo (MCMC) [21], stochastic gradient MCMC [22], and variational inference methods [23]. Bernoulli Dropout and its extensions are commonly used in practice because they are fast in calculation and easy to be implemented. Bayesian neural networks also have some applications in GNNs. Zhang et al. [19] proposed a Bayesian graph convolutional neural networks for semi-supervised classification. Hasanzadeh et al. [25] proposed a unified framework for adaptive connection sampling in GNNs. And GNNs training with adaptive connection sampling is shown to be equivalent to an efficient approximation of training Bayesian GNNs.

3 Notations

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ represent the input graph of size N with nodes $v_i \in \mathcal{V}$ and edges $(v_i, v_j) \in \mathcal{E}$. The node features are denoted as $\boldsymbol{X} = \{x_1, x_2, \cdots, x_N\} \in R^{N \times C}$ and the adjacent matrix is defined as $\mathcal{A} \in \mathcal{R}^{N \times N}$ which associates each edge (v_i, v_j) with its element $\mathcal{A}_{ij}$. The node degrees are given by $d = \{d_1, d_2, \cdots, d_N\}$ where d_i computes the sum of edge weights connected to node i. $\mathcal{N}_{v_i} = \{v_j : (v_i, v_j) \in \mathcal{E}\}$ denotes the set of neighbors of node v_i.

4 Methodology

4.1 Drop "Noise" Edge

The GNNs are superior to the existing Euclidean-based methods because they obtain a wealth of information from the nodes' neighbors. Therefore, the performance improvement brought by graphic data is highly related to the quantity and quality of domain information [11]. DropEdge [2] randomly drops edges in the graph. Although it is efficient, it does not consider the influence of adjacent nodes' information on the current node. Therefore, it can not determine whether the deleted information is beneficial or harmful to the task. Compared with DropEdge, DropNEdge treats the edges as unequal based on the influence of adjacent nodes' information on the current node and deletes edges with no or little positive impact on the final task. We use signal-to-noise ratio and feature gain indexes to measure the influence of adjacent nodes on the current node.

Feature Gain. Feature gain is used to measure the information gain of adjacent nodes' information relative to the current node. Considering that Kullback-Leibler (KL) divergence can measure the amount of information lost when an approximate distribution is adopted, it is used to calculate the information gain of the current node from its adjacent nodes [6]. The definition of KL divergence is stated as follows.

Definition 1. *$C(K)$ refers to the probability density function (PDF) of $\tilde{c}^k_{v_i}$, which is the ground truth and can be estimated by non-parametric methods with a set of samples. Each sample point is sampled with probability $|\mathcal{N}_{v_i}|/2|\mathcal{E}|$. $S(k)$ is the PDF of $\sum_{v_j \in N_{v_i}} a^{(k)}_{i,j} \cdot \tilde{c}^k_{v_j}$, which can be estimated with a set of samples $\{\sum_{v_j \in N_{v_i}} a^{(k)}_{i,j} \cdot \tilde{c}^k_{v_j}\}$. Each point is also sampled with probability $|\mathcal{N}_{v_i}|/2|\mathcal{E}|$ [6]. The information gain can be computed by KL divergence [12] as:*

$$D_{KL}\left(S^{(k)}||C^{(k)}\right) = \int_{x_k} S^{(k)}(x) \cdot log \frac{S^{(k)}(x)}{C^{(k)}(x)} dx. \tag{1}$$

In the actual calculation, the true and simulated distributions of the data are unknown. Thus, we use the feature gain to approximate KL divergence which measures the feature difference between the current node and its adjacent nodes. The definition of feature gain of node v is

$$FG_v = \frac{1}{|\mathcal{N}_v|} \sum_{v' \in \mathcal{N}_v} ||x_v - x_{v'}||^2, \tag{2}$$

where $|\mathcal{N}_v|$ is the number of adjacent nodes of node v, and x_v is the representation of node v. Moreover, the feature gain has the following relationship with KL divergence.

Theorem 1. *For a node v with feature x_v in space $[0,1]^d$, the information gain of the node from the surrounding $D_{KL}(S||C)$ is positively related to its feature gain FG_v; (i.e., $D_{KL}(S||C) \sim FG_v$). In particular, $D_{KL}(S||C) = 0$, when $FG_v = 0$.*

Thus, we know that the information gain is positively correlated to the feature gain. That is, the greater the feature gain means that the node can obtain more information from adjacent nodes. Therefore, we should first deal with nodes with less information gain. If the feature similarity of the nodes on both sides of a edge exceeds a given threshold, the edge should be dropped. In this way, edges with a significant impact on the task can be retained. The proof process of Theorem 1 is shown as follows.

Proof. For $D_{KL}(S||C)$, since the PDFs of C and S are unknown, a non-parametric way is used to estimate the PDFs of C and S. Specifically, the feature space $X = [0,1]^d$ is divided uniformly into r^d bins $\{H_1, H_2, \cdots, H_{r^d}\}$, whose length is $1/r$ and dimension is d. To simplify the use of notations, $|H_i|_C$ and $|H_i|_S$ are used to denote the number of samples that are in bin H_i. Thus, we yield

$$\begin{aligned}D_{KL}(S\|C) &\approx D_{KL}(\hat{S}||\hat{C})\\ &= \sum_{i=1}^{r^d} \frac{|H_i|_S}{2|\mathcal{E}|} \cdot \log \frac{\frac{|H_i|S}{2|\mathcal{E}|}}{\frac{|H_i|_C}{2|\mathcal{E}|}}\\ &= \frac{1}{2|\mathcal{E}|} \cdot \sum_{i=1}^{r^d} |H_i|_S \cdot \log \frac{|H_i|_S}{|H_i|_C}\\ &= \frac{1}{2|\mathcal{E}|} \cdot \left(\sum_{i=1}^{r^d} |H_i|_S \cdot \log |H_i|_S - \sum_{i=1}^{r^d} |H_i|_S \cdot \log |H_i|_C \right)\\ &= \frac{1}{2|\mathcal{E}|} \cdot \left(\sum_{i=1}^{r^d} |H_i|_S \cdot \log |H_i|_S - \sum_{i=1}^{r^d} |H_i|_S \cdot \log \left(|H_i|_S + \Delta_i\right) \right), \quad (3)\end{aligned}$$

where $\Delta_i = |H_i|_C - |H_i|_S$. Regard Δ_i as an independent variable, we consider the term $\sum_{i=1}^{r^d} |H_i|_S \cdot \log \left(|H_i|_S + \Delta_i\right)$ with second-order Taylor approximation at point 0 as

$$\sum_{i=1}^{r^d} |H_i|_S \cdot \log \left(|H_i|_S + \Delta_i\right) \approx \sum_{i=1}^{r^d} |H_i|_S \cdot \left(\log |H_i|_S + \frac{\ln 2}{|H_i|_S} \cdot \Delta_i - \frac{\ln 2}{2\left(|H_i|_S\right)^2} \cdot \Delta_i^2 \right). \quad (4)$$

Note that the number of samples for the context and the surrounding are the same, where we have

$$\sum_{i=1}^{r^d} |H_i|_C = \sum_{i=1}^{r^d} |H_i|_S = 2 \cdot |\mathcal{E}|. \quad (5)$$

Thus, we obtain $\sum_{i=1}^{r^d} \Delta_i = 0$. Therefore, $D_{KL}(\hat{S}||\hat{C})$ can be written as

$$\begin{aligned}D_{KL}\left(S||C\right) &\approx D_{KL}\left(\hat{S}||\hat{C}\right)\\ &= \frac{1}{2|\mathcal{E}|} \cdot \left(\sum_{i=1}^{r^d} |H_i|_S \cdot \log |H_i|_S - \sum_{i=1}^{r^d} |H_i|_S \cdot \log \left(|H_i|_S + \Delta_i\right) \right)\\ &\approx \frac{1}{2|\mathcal{E}|} \cdot \left(\sum_{i=1}^{r^d} |H_i|_S \cdot \left(-\frac{\ln 2}{|H_i|_S} \cdot \Delta_i + \frac{\ln 2}{2\left(|H_i|_S\right)^2} \cdot \Delta_i^2 \right) \right)\\ &= \frac{1}{2|\mathcal{E}|} \cdot \sum_{i=1}^{r^d} \left(\frac{\ln 2}{2|H_i|_S} \cdot \Delta_i^2 - \ln 2 \cdot \Delta_i \right)\\ &= \frac{\ln 2}{4|\mathcal{E}|} \cdot \sum_{i=1}^{r^d} \frac{\Delta_i^2}{|H_i|_S}. \quad (6)\end{aligned}$$

If we regard $|H_i|_S$ as constant, we have: if Δ_i^2 is large, then the information gain $D_{KL}(S||C)$ tends to be large. The above proof process is borrowed from Reference [6].

Considering the case of a node and its adjacent nodes, the samples of C are equal to x_v and the samples of S are sampled from $\{x_{v'} : v' \in \mathcal{N}_v\}$. For the distribution of the difference between the surrounding and the context, we consider $x_{v'}$ as noises on the "expected" signal and x_v is the "observed" signal. Then the difference between C and S is $\frac{1}{|\mathcal{N}_v|}\sum_{v' \in \mathcal{N}_v} ||x_v - x_{v'}||^2$, which is also the definition of FG_v. Thus, we obtain

$$\sum_{i=1}^{r^d} \Delta_i^2 \sim FG_v. \tag{7}$$

Therefore,

$$D_{KL}(S\|C) = \frac{\ln 2}{4|\mathcal{E}|}\sum_{i=1}^{r^d}\frac{\Delta_i^2}{|H_i|_S} \sim FG_v. \tag{8}$$

And if $FG_v = 0$, the feature vectors of the current node and its adjacent nodes are the same. Thus, $D_{KL}(S||C) = 0$. □

Signal-to-Noise Ratio. The reason for over-smoothing of GNNs is the low signal-to-noise ratio of received information. When aggregations among samples in different categories are excessive, the node representations in different classes will be similar. Thus, we assume that the aggregation of nodes among different categories is harmful, thereby bringing noise of information, and the aggregation of nodes in the same category brings useful signal. Here the signal-to-noise ratio is defined as

$$In_v = \frac{ds_v}{dh_v}, \tag{9}$$

where ds_v and dh_v represent the sum of edge weights connected to homogeneous and heterogeneous nodes of node v, respectively. Therefore, for a node with a small signal-to-noise ratio, we will drop the edges connected to heterogeneous nodes until the signal-to-noise ratio of the node is bigger than the given threshold.

Algorithm of DropNEdge. The specific approach of DropNEdge is shown in Algorithm 1. In this algorithm, if the ratio of deleted "noise" edges r_1 is set to 0, DropNEdge can be reduced to DropEdge.

4.2 Connection with Bayesian GNNs

Considering that Dropout can be an approximation of the Bayesian neural networks [18]; hence, we show that DropNEdge can be an approximation of Bayesian GNNs. We target the inference of the joint posterior of the random graph parameters, the weights in the GNN and the nodes' labels. Given that we are usually not directly interested in inferring the graph parameters, posterior estimates of the labels are obtained by marginalization [13]. The goal is to compute the posterior probability of labels, which is

$$p\left(\boldsymbol{Z} \mid \boldsymbol{Y}, \boldsymbol{X}, \mathcal{G}_{obs}\right)=\int p(\boldsymbol{Z} \mid W, \mathcal{G}, \boldsymbol{X}) p(W \mid \boldsymbol{Y}, \boldsymbol{X}, \mathcal{G}) p(\mathcal{G} \mid \lambda) p\left(\lambda \mid \mathcal{G}_{obs}\right) d W d \mathcal{G} d \lambda \quad (10)$$

where W is a random variable that represents the weights of the Bayesian GNN over graph $\mathcal{G}$, and λ characterizes a family of random graphs. This integral is intractable, we can adopt a number of strategies, including variational

Algorithm 1: Drop "Noise" Edge.

```
Input: The adjacency matrix $\mathcal{A}$, the number of edges $|\mathcal{E}|$, the ratio of deleted
       "noise" edges $r_1$, the ratio of randomly deleted edges $r_2$, the
       signal-to-noise ratio threshold $\delta_1$, the feature similarity threshold $\delta_2$, the
       ratio $q$ which controls the proportion of edges deleted according to the
       two indexes.
Output: The adjacency matrix $\mathcal{A}'$.
1  Initialization: $N_1 = 0$ , $N_2 = 0$, $\mathcal{A}' = \mathcal{A}$ ;
2  Randomly set $|\mathcal{E}| \times r_2$ elements with value 1 in $\mathcal{A}'$ to 0;
3  Calculate $In_v$ and $FG_v$ of each node;
4  Re-sort the nodes according to the two indexes' values from small to large to
   form nodes_1 and nodes_2 lists;
5  while $N_1 < |\mathcal{E}| \times r_1 q$ do
6      for node in nodes_1 do
7          for _node in $\mathcal{N}_{node}$ do
8              while $In_{node} < \delta_1$ do
9                  if node and _node in different class then
10                     $\mathcal{A}'[node, \_node] = 0$;
11                     $N_1 += 1$ ;
12                     Update $In_{node}$;
13                 end
14             end
15         end
16     end
17 end
18 while $N_2 < |\mathcal{E}| \times r_1 (1-q)$ do
19     for node in nodes_2 do
20         for _node in $\mathcal{N}_{node}$ do
21             if feature_similarity [node, _node] $> \delta_2$ then
22                 $\mathcal{A}'[node, \_node] = 0$;
23                 $N_2 += 1$ ;
24             end
25         end
26     end
27 end
28 Return: The adjacency matrix $\mathcal{A}'$
```

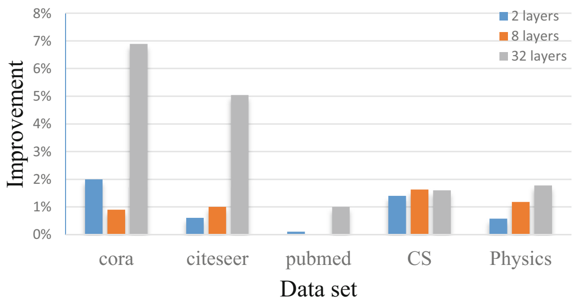

Fig. 1. The average absolute improvement by DropNEdge.

methods [24] and Markov Chain Monte Carlo (MCMC) [21], to approximate it. A Monte Carlo approximation of it is [13]

$$p\left(\boldsymbol{Z}|\boldsymbol{Y},\boldsymbol{X},\mathcal{G}_{obs}\right) \approx \frac{1}{V}\sum_{v}^{V}\frac{1}{N_G S}\sum_{i=1}^{N_G}\sum_{s=1}^{S} p\left(\boldsymbol{Z}|W_{s,i,v},\mathcal{G}_{i,v},\boldsymbol{X}\right). \tag{11}$$

In the approximation, V samples λ_v are drawn from $p\left(\lambda|\mathcal{G}_{obs}\right)$, N_G graphs $\mathcal{G}_{i,v}$ are sampled from $p\left(\mathcal{G}|\lambda_v\right)$, S weight matrices $W_{s,i,v}$ are sampled from $p\left(W|\boldsymbol{Y},\boldsymbol{X},\mathcal{G}_{i,v}\right)$ in the Bayesian GNNs that correspond to the graph $\mathcal{G}_{i,v}$ [19]. The sampled $w_{s,i,v}$ and $G_{i,v}$ can be obtained from GNNs with DropNEdge. Thus, if we turn on DropNEdge during the training and test phase, the model's uncertainty can be obtained.

5 Experiments

5.1 Performance Comparison

We compare the performances of the four GNN models with DropNEdge (DNE) and DropEdge (DE). The results are shown in Table 1. The performances of the

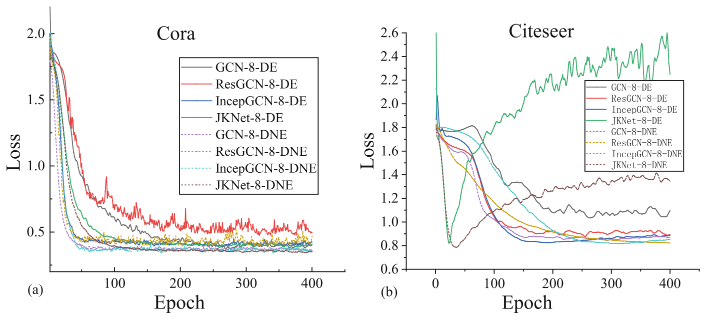

Fig. 2. (a) and (b) show the validation loss of different backbones with DropNEdge or DropEdge on Cora and Citeseer data sets.

Table 1. Accuracies of models with DropNEdge or DropEdge.

Dataset	Backbone	2 layers		8 layers		32 layers	
		DropEdge	DNE	DropEdge	DNE	DropEdge	DNE
Cora	GCN	0.865	0.867	0.858	**0.873**	0.746	0.796
	JKNet	–	–	0.878	**0.886**	0.876	0.878
	IncepGCN	–	–	0.882	**0.887**	0.877	0.886
	ResGCN	–	–	0.869	**0.877**	0.868	0.876
Citeseer	GCN	0.787	0.793	0.772	**0.802**	0.614	0.799
	JKNet	–	–	0.802	**0.810**	0.800	0.803
	IncepGCN	–	–	0.805	0.801	0.803	**0.815**
	ResGCN	–	–	0.788	**0.790**	0.779	0.781
Pubmed	GCN	0.912	**0.913**	0.909	0.909	0.862	0.901
	JKNet	–	–	0.912	0.912	0.913	**0.914**
	IncepGCN	–	–	**0.915**	**0.915**	0.905	0.905
	ResGCN	–	–	0.905	0.905	**0.911**	**0.911**
Coauthor CS	GCN	0.926	**0.934**	0.907	0.930	0.904	0.926
	JKNet	–	–	0.918	**0.926**	0.904	0.904
	IncepGCN	–	–	0.904	**0.936**	0.920	0.932
	ResGCN	–	–	0.898	0.900	0.896	**0.926**
Coauthor physics	GCN	0.954	**0.965**	0.940	0.950	0.932	0.946
	JKNet	–	–	0.941	**0.955**	0.936	0.950
	IncepGCN	–	–	0.936	**0.956**	0.936	0.953
	ResGCN	–	–	0.930	**0.933**	0.894	0.920

four models have been improved in most cases by DropNEdge. The improvement is more clearly depicted in Fig. 1, which counts the average improvement of different number of layers brought by DropNEdge. For example, on Cora data set, DropNEdge brings 6.9% average improvement to the models with 32 layers.

The results of models with and without DropNEdge are shown in Table 2. The effects of all models with DropNEdge have been consistently improved compared with models without DropNEdge. Thus the effect of DropNEdge is demonstrated. Figure 2 (a) and (b) show the comparison of the verification loss of models with DropNEdge or DropEdge which indicate that models with DropNEdge converge faster, and their losses are smaller.

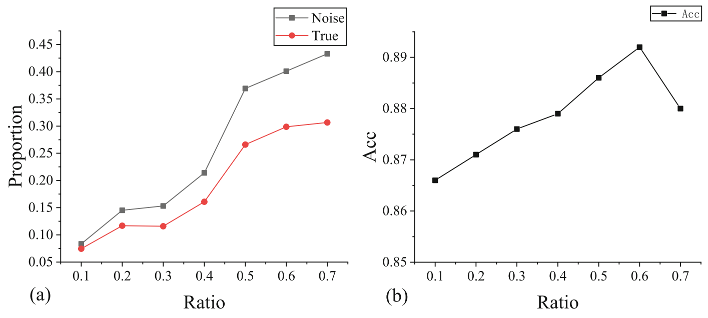

Fig. 3. The effect of removing noise edges.

Table 2. Accuracies of models with and without DropNEdge. "OOM" represents out of memory.

Dataset	Backbone	2 layers		8 layers		32 layers	
		Original	DNE	Original	DNE	Original	DNE
Cora	GCN	0.861	0.867	0.787	**0.873**	0.716	0.796
	JKNet	–	–	0.867	**0.886**	0.871	0.878
	IncepGCN	–	–	0.867	**0.887**	0.874	0.886
	ResGCN	–	–	0.854	**0.877**	0.851	0.876
Citeseer	GCN	0.759	0.793	0.746	**0.802**	0.592	0.799
	JKNet	–	–	0.792	**0.810**	0.717	0.803
	IncepGCN	–	–	0.796	0.801	0.726	**0.815**
	ResGCN	–	–	0.778	**0.790**	0.744	0.781
Pubmed	GCN	0.912	**0.913**	0.901	0.909	0.846	0.901
	JKNet	–	–	0.906	0.912	0.892	**0.914**
	IncepGCN	–	–	0.902	**0.915**	OOM	0.905
	ResGCN	–	–	0.896	0.905	0.902	**0.911**
Coauthor CS	GCN	0.885	**0.934**	0.885	0.930	0.846	0.926
	JKNet	–	–	0.901	**0.926**	0.863	0.904
	IncepGCN	–	–	0.930	**0.936**	0.897	0.932
	ResGCN	–	–	0.857	0.900	0.863	**0.926**
Coauthor physics	GCN	0.918	**0.965**	0.875	0.950	0.791	0.946
	JKNet	–	–	0.902	**0.955**	0.921	0.950
	IncepGCN	–	–	0.844	**0.956**	OOM	0.953
	ResGCN	–	–	0.847	**0.933**	0.910	0.920

The superiority of DropNEdge lies in the following reasons: (1) DropNEdge avoids excessive aggregation of node information by dropping "noise" edges, which alleviates the over-smoothing phenomenon effectively. (2) It can remove "noise" edges and retain meaningful edges which prevents the transmission of harmful information. (3) DropNEdge can be used as a graphical data enhancement method.

5.2 Remove "Noise" Edges

We randomly add a given proportion of edges to the graph of Cora data set which is set to 0.3 in this experiment. The added edges are considered to be "noise" edges. We change the ratio of deleted "noise" edges r_1. Subsequently, the proportions of deleted added edges to total added edges (r_N) and deleted non-added edges to the real edges (r_T) in the graph are counted. The model used is GCN-8 and the results are shown in Fig. 3 (a) and (b). From Fig. 3 (a), we can see that DropNEdge can remove "noise" edges because r_N is always greater than r_T no matter what the ratio r_1 is. Figure 3 (b) shows that the model's accuracy also increases as r_1 increases. However, when too many edges are deleted, the meaningful aggregation of information will also decrease. Thus, the accuracy of the model decreases.

5.3 Suppress Over-Smoothing

When the top-level output of GNNs converges to a subspace and becomes irrelevant to the input as the depth increases, over-smoothing phenomenon occurs. Considering that the convergent subspace cannot be derived explicitly, we measure the degree of smoothing by calculating the difference between the output of the current layer and the previous layer. Euclidean distance is used to calculate the difference. The smaller the distance, the more severe the over-smoothing. This experiment is carried out on GCN-8 with Cora data set whose results are shown in Fig. 4.

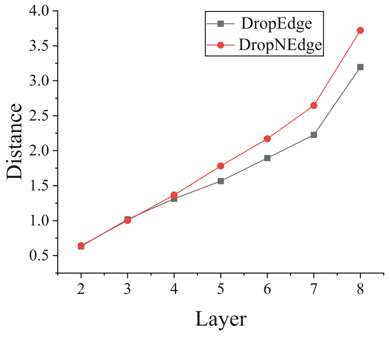

Fig. 4. The effect comparison between DropEdge and DropNEdge of suppressing over-smoothing.

DropNEdge is better than DropEdge in suppressing over-smoothing. As the number of layers increases, the distances between layers in models with DropEdge and DropNEdge both increase. Furthermore, the distance's increasing speed of the model with DropNEdge is faster than that of the model with DropEdge.

5.4 Layer Independent DropNEdge

The DropNEdge mentioned above is that all layers share the same perturbation adjacency matrix. In fact, we can perform it for each individual layer. Different layers can have different adjacent matrices. This layer-independent (LI) version brings more randomness and distortion of the original data. We experimentally compare its performance with the shared DropNEdge's performance on Cora data set. The model used is GCN-8 and the comparisons of the verification loss and training loss between shared and independent DropNEdge are shown in Fig. 5 (a). Although hierarchical DropNEdge may achieve better results, we still prefer to use shared DropNEdge which can not only reduce the risk of overfitting, but also reduce the computational complexity.

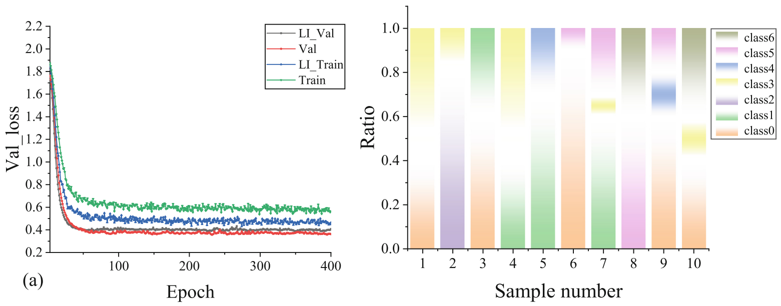

Fig. 5. (a) shows the comparison of the training and verification loss between DropNEdge and LI DropNEdge. (b) shows the uncertainty of model with DropNEdge.

5.5 Model Uncertainty

To obtain the model's uncertainty, we turn on DropNEdge during the training and test phase and set the ratio of deleted "noise" edges to 0.3. The experiment is carried out on GCN-8 with Cora data set. After the model predicts multiple times, different predictions may be produced for a sample. Figure 5 (b) shows the ratio of different labels obtained in 10 predictions for 10 samples. For example, for sample one, 40% of the ten predictions are class 0 and 60% of the predictions are class 3. Thus, the confidence of the predictions can be obtained by using DropNEdge. For high-confidence samples, that is, samples with consistent results after multiple predictions, the model's predictions can be used directly. If the model's predictions of some samples change greatly, other models should be further used or they should be artificially determined to get more reasonable predictions.

6 Conclusion

This paper proposes DropNEdge, a novel and effective method to alleviate the over-smoothing phenomenon and remove "noise" edges in graphs. It mainly considers two indicators based on the graph's information, namely, feature gain and signal-to-noise ratio. By using DropNEdge, the over-smoothing of GNNs is alleviated and the "noise" edges with no positive impact on the final task can be removed, thereby improving the performance of GNNs. DropNEdge does not need to change the network's structure and is widely adapted to various GNNs.

References

1. Dongsheng, L., et al.: Learning to drop: robust graph neural network via topological denoising. In: Proceedings of the 14th ACM International Conference on Web Search and Data Mining (WSDM 2021), pp. 779–787. Association for Computing Machinery, Online (2021) . https://doi.org/10.1145/3437963.3441734

2. Yu, R., Wenbing, H., Tingyang, X., Junzhou, H.: DropEdge: towards deep graph convolutional networks on node classification. In: Proceedings of the 8th International Conference on Learning Representations (ICLR 2020), International Conference on Learning Representations, Addis Ababa (2020) . https://openreview.net/forum?id=Hkx1qkrKPr
3. Nitish, S., Geoffrey, H., Alex, K., Ilya, S.: Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. **15**(1), 1929–1958 (2014)
4. Qimai, Li., Zhichao, H., Xiao-Ming, W.: Deeper insights into graph convolutional networks for semi-supervised learning. In: The Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18), pp. 3538–3545. AAAI press, Louisiana (2018). https://arxiv.org/abs/1801.07606
5. Deli, C., Yankai, L., Wei, L., Peng, L., Jie, Z., Xu, S.: Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In: The Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI-20), pp. 3438–3445. AAAI press, New York (2020) . https://doi.org/10.1609/aaai.v34i04.5747
6. Yifan, H., et al.: Measuring and improving the use of graph information in graph neural network. In: The Eighth International Conference on Learning Representations (ICLR 2020), Addis Ababa (2020) . https://openreview.net/forum?id=rkeIIkHKvS
7. Lingxiao, Z., Leman, A.: PairNorm: tackling oversmoothing in GNNs. In: Proceedings of the 8th International Conference on Learning Representations (ICLR 2020), Addis Ababa (2020) . https://arxiv.org/abs/1909.12223
8. William, L. H., Rex, Y., Jure, L.: Inductive representation learning on large graphs. In: Proceedings of the 31st Annual Conference on Neural Information Processing Systems (NIPS 2017), pp. 1025–1035. Neural Information Processing Systems Foundation, California (2017). https://arxiv.org/abs/1706.02216
9. Jie, C., Tengfei, M., Cao, X.: FastGCN: fast learning with graph convolutional networks via importance sampling. In: Proceedings of the 6th International Conference on Learning Representations (ICLR 2018), Vancouver (2018). https://arxiv.org/abs/1801.10247
10. Santo, F.: Community detection in graphs. Phys. Rep. **486**, 75–174 (2010)
11. Jie, Z., et al.: Graph neural networks: a review of methods and applications. arXiv:1812.08434 (2021)
12. Kullback, S., Leibler, R.A.: On information and sufficiency. Ann. Math. Stat. **22**(1), 79–86 (1951)
13. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: Proceedings of the 5th International Conference on Learning Representations (ICLR 2017), Toulon (2017). https://openreview.net/pdf?id=SJU4ayYgl
14. Petar, V., Guillem, C., Arantxa, C., Adriana, R., Pietro, L., Yoshua, B.: Graph attention networks. In: Proceedings of the 6th International Conference on Learning Representations (ICLR 2018), Vancouver (2018) . https://arxiv.org/abs/1710.10903
15. Chaoqi, Y., Ruijie, W., Shuochao, Y., Shengzhong, L., Tarek, A.: Revisiting oversmoothing in deep GCNs. arXiv:2003.13663 (2020)
16. Chen, C., Yusu, W.: A note on over-smoothing for graph neural networks. In: The Thirty-seventh International Conference on Machine Learning (ICML 2020), International Machine Learning Society, Online (2020) . https://arxiv.org/abs/2006.13318

17. James, F., Sivasankaran, R.: How robust are graph neural networks to structural noise? arXiv:1912.10206 (2019)
18. Yarin, G., Zoubin, G.: Dropout as a Bayesian approximation: representing model uncertainty in deep learning. In: The Thirty-Third International Conference on Machine Learning (ICML 2016), International Machine Learning Society, New York (2016) . https://arxiv.org/abs/1506.02142v1
19. Yingxue, Z., Soumyasundar, P., Mark, C., Deniz, Ü.: Bayesian graph convolutional neural networks for semi-supervised classification. In: The Thirty-Third AAAI Conference on Artificial Intelligence (AAAI 2019), pp. 5829–5836. AAAI Press, Hawaii (2019). https://arxiv.org/abs/1811.11103
20. David, J.C.M.: Bayesian Methods for Adaptive Models. California Institute of Technology, California (1992)
21. Radford, M.N.: Bayesian Learning for Neural Networks. Springer, New York (1996)
22. Welling, M., Yee, W.T.: Bayesian learning via stochastic gradient Langevin dynamics. In: Proceedings of the 28th International Conference on Machine Learning (ICML 2011), pp. 681–688. Association for Computing Machinery, Washington (2011). https://dl.acm.org/doi/10.5555/3104482.3104568
23. David, M.B., Alp, K., Jon, D.M.: Variational inference: a review for statisticians. J. Am. Stat. Assoc. **112**(518), 859–877 (2017)
24. Matthew, D.H., David, M.B., Chong, W., John, P.: Stochastic variational inference. J. Mach. Learn. Res. **14**(4), 1303–1347 (2013)
25. Arman, H., et al.: Bayesian graph neural networks with adaptive connection sampling. arXiv:2006.04064 (2020). https://arxiv.org/abs/2006.04064