End-to-End Open-Set Semi-Supervised Node Classification with Out-of-Distribution Detection

Tiancheng Huang[1,2,3], **Donglin Wang**[2,3*], **Yuan Fang**[4], **and Zhengyu Chen**[2,3]
[1] Zhejiang University, Hangzhou, China
[2] Westlake University, Hangzhou, China
[3] Westlake Institute for Advanced Study, Hangzhou, China
[4] Singapore Management University, Singapore
{huangtiancheng, wangdonglin, chenzhengyu}@westlake.edu.cn, yfang@smu.edu.sg

Abstract

Out-Of-Distribution (OOD) samples are prevalent in real-world applications. The OOD issue becomes even more severe on graph data, as the effect of OOD nodes can be potentially amplified by propagation through the graph topology. Recent works have considered the OOD detection problem, which is critical for reducing the uncertainty in learning and improving the robustness. However, no prior work considers simultaneously OOD detection and node classification on graphs in an end-to-end manner. In this paper, we study a novel problem of *end-to-end open-set semi-supervised node classification (OSSNC)* on graphs, which deals with node classification in the presence of OOD nodes. Given the lack of supervision on OOD nodes, we introduce a latent variable to indicate in-distribution or OOD nodes in a variational inference framework, and further propose a novel algorithm named as *Learning to Mix Neighbors* (LMN) which learns to dampen the influence of OOD nodes through the messaging-passing in typical graph neural networks. Extensive experiments on various datasets show that the proposed method outperforms state-of-the-art baselines in terms of both node classification and OOD detection.

1 Introdution

Graph Neural Networks (GNNs) present the state-of-the-art learning algorithm for graph-structured data via message passing between graph nodes [Wu *et al.*, 2020]. Recently, various GNNs have been proposed with promising performance, such as graph convolutional network (GCN) [Kipf and Welling, 2017], graph attention network (GAT) [Veličković *et al.*, 2018], GraphSAGE [Hamilton *et al.*, 2017], and many others [Defferrard *et al.*, 2016; Wu *et al.*, 2019; Klicpera *et al.*, 2018; Kim and Oh, 2020; Chen *et al.*, 2020]. Generally, these GNNs have been proposed for node classification under the closed-set assumption that the training and test data are assumed to be sampled from the same distribution. However, in the real world, it is difficult to know the complete set of labels for all nodes, and it is not guaranteed that all unlabeled nodes are sampled from the known classes especially on a large graph. Therefore, the learning algorithm dedicated for in-distribution scenarios cannot cover the situation above [Geisa *et al.*, 2021]. Recently, the Out-Of-Distribution (OOD) detection has attracted increasing attention for computer vision applications [Hendrycks and Gimpel, 2016; Liang *et al.*, 2018; Vernekar *et al.*, 2019; Hsu *et al.*, 2020] and graph-structured data [Zhao *et al.*, 2020; Stadler *et al.*, 2021]. However, to the best of our knowledge, no prior research on GNNs has considered both OOD detection and graph node classification in an end-to-end manner.

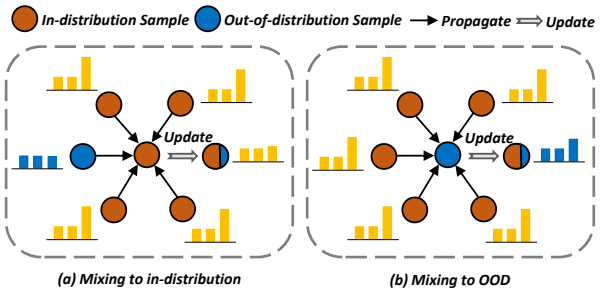


Figure 1: The illustration of the across-distribution mixture on graphs: (a) Mixing to in-distribution; (b) Mixing to OOD.

In this paper, we study a novel ***end-to-end Open-set Semi-Supervised Node Classification (OSSNC)*** problem, where a graph consists of both in-distribution and OOD nodes. In real-world scenarios, a graph likely contains both labeled nodes belonging to known classes that can be regarded to be in-distribution, and unlabeled nodes belonging to known or unknown classes [Stadler *et al.*, 2021], where the unknown classes are regarded to be OOD. The GNN may suffer from severe performance degradation due to **across-distribution mixture** [Bitterwolf *et al.*, 2022], as shown in Fig. 1. We consider two situations on graphs: mixing to in-distribution and mixing to OOD. 1) First, when the central node is in-distribution, aggregating the OOD node information in the neighbors will cause an across-distribution mixture. 2) When the central node is OOD, aggregating information from in-distribution nodes will smooth the node embedding, which increases the risk of misclassification. To avoid across-distribution mixture between nodes, the prop-

*Corresponding author.

agation and aggregation of GNNs should be well-designed. For *OSSNC*, while the attention-based GAT [Veličković *et al.*, 2018] can cope with the first scenario where the central node is in-distribution to some extent by assigning a small attention value to OOD neighbors, it cannot deal with the second scenario since the attention coefficients on the neighbors are normalized which inevitably introduces the across-distribution mixture between them.

Thus, a crucial question remains to be addressed: *How to train the GNN to perform OSSNC with OOD detection in end-to-end manner*? This problem faces the following challenges: 1) The lack of supervision on whether a node is in-distribution or OOD. 2) There exist graph edges between in-distribution and OOD nodes, promoting the information propagation between them and causing across-distribution mixture. 3) GNNs are also prone to over-fitting [Rong *et al.*, 2019; Xiao *et al.*, 2021], which could become more severe when we utilize an additional parameterized model to predict if each node is in-distribution or OOD.

To address these challenges, we adopt the principle of the probabilistic generative model and propose a novel algorithm called *Learning to Mix Neighbors* (LMN) in a variational inference framework. To handle challenge 1), we adopt variational inference, by introducing a latent variable to indicate in-distribution or OOD node to approximate the intractable true posterior distribution. To handle challenge 2), we design a parameterized predictor for OOD node detection, and instantiate it as a weight predictor which generates real-valued weights to softly mix neighboring nodes. Then, we adopt GCNII [Chen *et al.*, 2020] as the backbone and take the learned weights to adjust the information from neighbors, which effectively mitigates across-distribution mixture. To handle challenge 3), a bi-level optimization algorithm is adopted to iteratively update the GNN parameters and the predictor parameters. To summarize, we list our contributions:

1) We study a novel end-to-end *OSSNC* problem for simultaneous node classification and OOD detection. To the best of our knowledge, we are the first to investigate this problem.

2) We propose an algorithm LMN in a variational inference framework, which learns to mix neighbors to mitigate the propagation to and from OOD nodes without needing any OOD labels. Furthermore, we employ a bi-level optimization to reduce over-fitting introduced by additional parameters.

3) We conduct extensive experiments on four graphs and the results demonstrate the effectiveness of our method.

2 Related Work

2.1 GNNs and Robust GNNs

Recently, GNNs have been proposed and they generally can be categorized into three categories, i.e., *spectral-based methods* [Defferrard *et al.*, 2016; Kipf and Welling, 2017; Wu *et al.*, 2019], *spatial-based methods* [Hamilton *et al.*, 2017; Veličković *et al.*, 2018] and *deeper GNNs* [Klicpera *et al.*, 2018; Chen *et al.*, 2020]. Next, we review methods about robust GNNs for alleviating the noise issue. For example, GAT [Veličković *et al.*, 2018] aggregates neighboring nodes with weights from attention mechanism to prevent the robustness. And some methods (e.g., DropEdge [Rong *et al.*, 2019]) based on graph sparsification and sampling to enable efficient computation, and improve robustness. Besides, the other related works are *anomaly detection on graphs*. The works on anomaly detection can be found in surveys [Pimentel *et al.*, 2014; Ma *et al.*, 2021]. The core of such work is how to find anomaly samples. However, OOD detection is usually addressed by modifying the model, which requires not only to effectively detect OOD samples, but also to ensure that the performance of the model is not affected. In this paper, we study the end-to-end *OSSNC* problem, and aim to train a GNN model with OOD detection in end-to-end manner, which is empowered to detect OOD nodes during inference, while we still take advantage of semi-supervised training for the original classification task.

2.2 Out-of-distribution Detection

Out-of-distribution detection task plays an important role in AI safety [Bitterwolf *et al.*, 2022]. Recently, OOD detection has been investigated in computer vision for better robustness and prediction, including softmax-based methods [Hendrycks and Gimpel, 2016; Liang *et al.*, 2018; Hsu *et al.*, 2020], generative-model-based methods [Denouden *et al.*, 2018; Lee *et al.*, 2018], and uncertainty-based methods [DeVries and Taylor, 2018; Mukhoti *et al.*, 2021]. For *softmax-based* models, they aim to distinguish between in-distribution and OOD samples by the softmax prediction probability. For example, ODIN [Liang *et al.*, 2018] improved the baseline [Hendrycks and Gimpel, 2016] with two strategies, i.e., temperature scaling and input preprocessing, to further distinguish in-distribution and OOD data. For *generative-model-based* models, [Lee *et al.*, 2018] generates near-OOD samples to surround the manifold of the entire in-distribution data. And other work [Vernekar *et al.*, 2019] generates these samples using a manifold learning network (e.g., variational autoencoder) and then trains an n+1 classifier for OOD detection. The *uncertainty-based* methods are to use the confidence of the model's prediction to measure whether a sample is OOD or not. For example, the work [DeVries and Taylor, 2018] augments with a confidence estimation branch to produce a confidence estimation. For graph-structural data, the previous works [Zhao *et al.*, 2020; Stadler *et al.*, 2021] expect to detect OOD nodes by uncertainty, and focus on OOD detection task, rather than consider both OOD detection and graph node classification in an end-to-end manner.

3 Preliminaries and Analysis

3.1 OSSNC

The *OSSNC* consists of *semi-supervised node classification* as a primary task with *OOD detection* as an auxiliary task. 1) *Semi-Supervised Node Classification.* Given a graph $\mathcal{G} = \{V, E, \mathbf{X}, Y_L\}$, where V indicates the node set, E indicates the edge set, $\mathbf{X}$ is the node feature and Y_L is the node label. And $V = V_L \cup V_U$, where V_L is labeled set and V_U is unlabeled set. We also can use an adjacency matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$ to represent the graph $\mathcal{G}$, where $\mathbf{A}_{v,u} = 1$ indicates that there exists an edge between nodes v and u; otherwise, $\mathbf{A}_{v,u} = 0$. For the *semi-supervised node classification*, the model takes a small amount of labeled nodes with a large amount of unlabeled

nodes during training. Overall, the goal of the node classification is to predict the labels of unlabeled nodes. 2) *OOD Detection.* Let an input node $\mathbf{x} \in \mathbf{X}$ and label $y \in Y$ with the joint data distribution $P(\mathbf{x}, y|O{=}in) = P(y|\mathbf{x}, O{=}in)P(\mathbf{x}|O = in)$, where O indicates in-distribution or OOD node. We assume that a classifier $P_\theta(y|\mathbf{x})$ is trained using in-distribution nodes from $P(\mathbf{x}, y|O = in)$, where θ denotes the model parameter. However, the partial nodes during inference phase may come from out-of-distribution $P(\mathbf{x}|O = out)$. The OOD detection task is to determine whether the input node $\mathbf{x}$ is from $P(O = in)$ or $P(O = out)$. For each input node during inference, we measure some score based on the uncertainty of prediction (e.g., Entropy), and compare the score to a threshold γ. If the score is above γ, the node is considered as an OOD node; otherwise, in-distribution node.

3.2 Analysis on Across-distribution Mixture

In this subsection, we first analyze the across-distribution mixture on graphs. According to [Bitterwolf *et al.*, 2022]:

Theorem 3.1 *Across-distribution Mixture. It is assumed that the sample feature conforms to the Normal distribution with the mean μ and variance σ. The mixing feature of a sample comes from in-distribution and out-of-distribution:*

$$P_{mix}(\boldsymbol{x}) = P(\boldsymbol{x}|O = in)P(O = in) + P(\boldsymbol{x}|O = out)P(O = out) \sim \mathcal{N}(\mu_1, \sigma_1) + \mathcal{N}(\mu_2, \sigma_2), \tag{1}$$

where $P(\boldsymbol{x}|O{=}in)$, $P(\boldsymbol{x}|O{=}out) \sim \mathcal{N}(\mu_i, \sigma_i)$, $i \in \{1, 2\}$.

On this basis, we can further analyze the across-distribution mixture on graphs, more complex due to the propagation and aggregation between nodes.

Lemma 3.1 *Across-distribution Mixture on Graphs. Take one-layer aggregation of GNN as an example, the distribution mixture comes from the central node and its neighbors:*

$$P_{mix}(\boldsymbol{x}_i) \sim \mathcal{N}(\mu_i, \sigma_i) + \sum_{v=1}^{|\mathbb{N}(u)|} w_{v,u}\mathcal{N}(\mu_{j_v}, \sigma_{j_v}) \tag{2}$$

where $i, j_v \in \{1, 2\}$, $w_{v,u}$ denotes weights between node u and v, and $\mathbb{N}(u)$ denotes neighbors of node u.

Visualization. To visualize the across-distribution mixture, we assign different weight $w_{v,u}$ for mixing to in-distribution (Fig. 2 (a) $\sim$ (c)) and mixing to OOD (Fig. 2 (d) $\sim$ (f)). The weight $w_{v,u}$ is set as 1 for all cases in (b) and (e) while close to 0 for all cases in (c) and (f). Consistent with intuition, (b) and (e) depicts a severe distribution mixture while (c) and (f) have a slight across-distribution mixture. From Fig. 2 (b) and (e), the OOD has a significant effect on distribution variation. It is necessary to accurately detect OOD samples.

3.3 Problem Definition

In this paper, we consider the input pair $(\mathbf{x}, y)$ with partial class y not belonging to any in-distribution class C_{in}, which is regarded as OOD nodes. Thus, the extended label set includes in-distribution classes C_{in} and OOD classes C_{out}, i.e., $C{=}C_{in} \cup C_{out}$. In traditional setting using the closed-set assumption, all nodes in the unlabeled set V_U come from the same distribution encountered in V_L. In our open-set scenarios, it is not necessary for unlabeled nodes V_U coming from the same distribution as labeled nodes V_L. In other words, unlabeled nodes V_U may contain nodes of unseen classes, called OOD nodes. In summary, our goal is to learn a GNN with OOD detection in an end-to-end manner for avoiding the performance degradation on semi-supervised node classification.

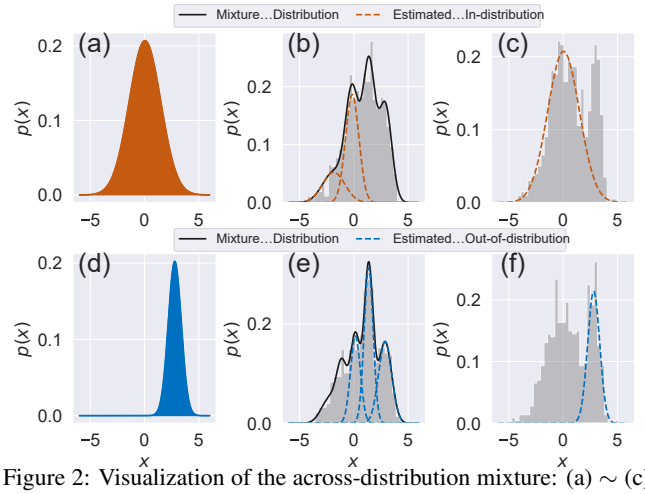


Figure 2: Visualization of the across-distribution mixture: (a) $\sim$ (c) Mixing to in-distribution; (d) $\sim$ (f) Mixing to OOD.

4 Methodology

Given no supervision, a generative model is proposed for modeling the joint distribution of Y and O conditioned on features and graphs, $P(Y, O|\mathbf{X}, \mathbf{A})$, and formulate the Learning to Mix Neighbors (LMN) as a variational objective.

4.1 Unified Learning Framework

To avoid the across-distribution mixture as Fig. 2, we need to effectively distinguish data in terms of distribution. We expect that the primary classification task can be guided with the auxiliary OOD detection task in an end-to-end manner. Hence, we propose an unified learning framework for the *OSSNC* problem. The joint probability distribution of labels Y and O in the *OSSNC* task can be described as

$$P_\theta(Y, O|\mathbf{X}, \mathbf{A}) = P_\theta(Y|\mathbf{X}, \mathbf{A}, O)P(O|\mathbf{X}, \mathbf{A}), \tag{3}$$

where θ represents GNN parameter and the probability distribution of OOD detection can be described as $P(O|\mathbf{X}, \mathbf{A})$. From the Bayesian perspective, the learning process includes: i) Learning the GNN parameter by maximizing the likelihood

$$\log P_\theta(Y, O|\mathbf{X}, \mathbf{A}) = \log \sum_k P_\theta(Y|\mathbf{X}, \mathbf{A}, O_k) \cdot P(O_k|\mathbf{X}, \mathbf{A}), \tag{4}$$

where $O_k \in \{in, out\}$ indicates in-distribution or OOD case. ii) Inferring the following posterior of the latent variable O as

$$P_\theta(O_k|\mathbf{X}, \mathbf{A}, Y) = \frac{P_\theta(Y|\mathbf{X}, \mathbf{A}, O_k)}{\sum_j P_\theta(Y|\mathbf{X}, \mathbf{A}, O_j)}, \tag{5}$$

where $j \in \{in, out\}$ also indicates in-distribution or OOD.

However, it is challenging to address such two learning problems. On one hand, we cannot directly learn the GNN parameter θ because it involves marginalizing the latent variable O, which is generally time-consuming and intractable [Kingma and Welling, 2013]. On the other hand, we lack of supervision for test nodes for inference.

Variational Inference. To solve the challenges, the variational inference principle is considered and the lower bound of the marginal log-likelihood in Eq. (4) produces the variational objective:

$$\mathcal{L}(\theta) = \mathbb{E}_{Q(O_k)}\left[\log P_\theta(Y|\mathbf{X}, \mathbf{A}, O_k)\right] - \mathrm{KL}(Q(O_k)||P(O_k)), \tag{6}$$

where $Q(O_k)$ is the introduced variational distribution. Maximizing the evidence lower bound (ELBO) above is equivalent to simultaneously maximizing Eq. (4) and making the variational distributions $Q(O_k)$ close to its intractable true posteriors. We consider an end-to-end iterative algorithm to minimize negative ELBO by introducing the parameterized posterior $Q_\phi(O|\mathbf{X}, \mathbf{A})$ with parameter ϕ, into Eq. (6) and directly optimize ELBO using reparameterization trick [Kingma and Welling, 2013].

Specifically, we introduce the variational distribution to minimize Kullback-Leibler (KL) divergence:

$$\mathrm{KL}(Q_\phi \| P) = \mathbb{E}_Q\left[\log \frac{Q_\phi(O_k|\mathbf{X}, \mathbf{A})}{P(O_k)}\right]. \tag{7}$$

where $P(O_k)$ follows Bernoulli distribution. Specifically, we minimize the following negative ELBO in Eq. (6) with the reparameterization trick

$$\mathcal{L}(\theta, \phi) = -\mathbb{E}_{Q_\phi}[\log P_\theta(Y|\mathbf{X}, \mathbf{A}, O)] + \mathrm{KL}(Q_\phi \| P), \tag{8}$$

where the $\mathbb{E}_{Q_\phi}[\log P_\theta(Y|\mathbf{X}, \mathbf{A}, O)]$ is the semi-supervised node classification loss. According to Eq. (7), minimizing Eq. (8) makes the variational distribution $Q_\phi(O|\mathbf{X}, \mathbf{A})$ close to its intractable true posterior distribution.

4.2 Learning to Mix Neighbors

To learn the variational distribution $Q_\phi(O|\mathbf{X}, \mathbf{A})$, we introduce a novel method *learning to mix neighbors*. Recall our analysis for across-distribution mixture on graphs (Eq. (2)), our underlying intuition is that the central node should aggregate neighbor nodes discriminately. This gives us motivation to selectively mix our neighbors via learned weights.

Learning Weights. Instead of binary dropping of edges [Rong *et al.*, 2019; Luo *et al.*, 2021], we adopt a predictor f_ϕ, which generates the weights w to mix neighbors and gives a single scalar between 0 and 1 (parametrized as a sigmoid):

$$Q_\phi(w_{v,u}|\mathbf{X}, \mathbf{A}) = \frac{1}{1 + exp(-\mathbf{W}^T[\mathbf{H}_v \| \mathbf{H}_u])}, \tag{9}$$

where $\mathbf{W}$ is a trainable parameter of predictor f_ϕ, $\mathbf{H_v}$ and $\mathbf{H_u}$ are the representation of node v and u, and $w_{v,u}$ represents the weight to aggregate the neighbor nodes. The learned weight is generated by taking the concatenate or element-wise multiplication features as input of the predictor f_ϕ.

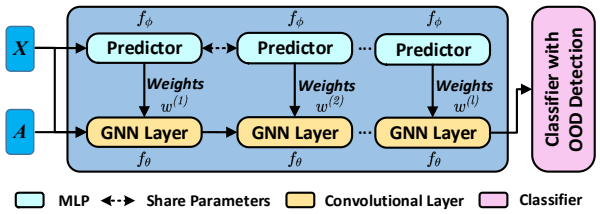


Figure 3: Illustrations of LMN framework. The architecture consists of a MLP-based predictor with shared parameters, a multi-layer GNN and a classifier with OOD detection [Mukhoti *et al.*, 2021].

Then, the latent variable $w_{v,u} \in [0, 1]$ is treated as an aggregating weight from node v to u. For our model, the graph convolution based on message passing can be described as

$$\mathbf{H}^{(l+1)} = f_{\theta^{(l)}}(\mathbf{H}^{(l)}, \mathbf{A}, w^{(l)}_{v,u}), \tag{10}$$

where $f_{\theta^{(l)}}$ is GNN with parameter θ at layer l, $\mathbf{H}^{(0)} = \mathbf{X}$.

Objective Function for Learning. By substituting Eq. (9) into (8), and (8) into (7), the objective becomes

$$\begin{aligned}\mathcal{L}(\theta, \phi) &= -\mathbb{E}_{Q_\phi}[\log P_\theta(Y|\mathbf{X}, \mathbf{A}, O)] \\ &\quad + \mathrm{KL}(Q_\phi(w_{v,u}|\mathbf{X}, \mathbf{A})||P(O_k)) \\ &= -\mathbb{E}_{Q_\phi}[\log P_\theta(Y|\mathbf{X}, \mathbf{A}, O)] \\ &\quad + \mathrm{KL}\left(\frac{1}{1 + exp(-\mathbf{W}^T[\mathbf{H}_v \| \mathbf{H}_u])}||P(O_k)\right),\end{aligned} \tag{11}$$

where $Q_\phi(w_{v,u}|\mathbf{X}, \mathbf{A})$ is the output of predictor f_ϕ. Consequently, minimizing Eq. (11) allows each node to learn effective weights for the *OSSNC* task.

Framework. In Section 4.1, we present a ***unified*** learning framework, where Q_ϕ in Eq. (7) denotes a general formula and can be used for various considerations (such as nodes, edges). However, in Eq. (9), Q_ϕ is a dedicated probability distribution for edges in our specific LMN method (in Section 4.2). The architecture of LMN in our implementation is shown in Fig. 3, which likely consists of a multi-layer GNN and achieves good performance on both node classification and OOD detection tasks in an end-to-end manner.

4.3 Bi-level Optimization

For the optimization, we calculate the optimal MLP-based predictor ϕ as well as the GNN with OOD detection classifier $\theta = \{\theta_g, \theta_c\}$, which is formulated by using two nested loops of optimization, i.e., an outer loop for the predictor ϕ and an inner loop for the GNN θ. Thus, the objective is given as

$$\begin{aligned}&\min_\phi \mathcal{L}_{val}(\theta^*(\phi), \phi), \\ &s.t.\ \theta^*(\phi) = \arg\min_\theta \mathcal{L}_{train}(\theta, \phi),\end{aligned} \tag{12}$$

where $\mathcal{L}_{train}$ and $\mathcal{L}_{val}$ are consistent with the objective of semi-supervised node classification task with the variational inference, i.e., Eq. (11), where $\mathcal{L}_{train}$ samples from training sets and $\mathcal{L}_{val}$ samples from validation sets. We employ

an alternating optimization strategy to iteratively update the GNN θ and the predictor ϕ. However, the nested optimization results in a huge computational complexity and a heavy GPU memory occupation. To alleviate these issues, we adopt an approximate alternating optimization via the first-order approximation trick same as [Xiao *et al.*, 2021].

Updating inner level θ_g and θ_c. This updating is different from calculating the optimal parameters θ_g^* and θ_c^* at each iteration, we only perform a hyper-parameter T steps stochastic gradient descent [Dai *et al.*, 2021].

$$\theta_g^{t+1} = \theta_g^t - \alpha_g \nabla \mathcal{L}_{train}(\theta_g^t, \theta_c^t, \phi^t),$$
$$\theta_c^{t+1} = \theta_c^t - \alpha_c \nabla \mathcal{L}_{train}(\theta_g^t, \theta_c^t, \phi^t), \quad (13)$$

where α_g and α_c are the learning rate of GNN and classifier.

Updating outer level ϕ. We use the parameters after updating inner level θ_g^T and θ_c^T to approximate the optimal parameters θ_g^* and θ_c^*. Then, the convergence of the updating via first-order approximation is proved in [Xiao *et al.*, 2021].

$$\phi^{n+1} = \phi^n - \alpha_\phi \nabla_\phi \mathcal{L}_{val}\left(\theta_g^T, \theta_c^T, \phi^n\right), \quad (14)$$

where α_ϕ is the learning rate of the predictor.

5 Experiments

In the section, we aim to answer the following questions:

(RQ1) How does our proposed LMN perform compared with the baselines in term of *node classification*?

(RQ2) How is our LMN performed compared with the baselines in term of *OOD detection*?

(RQ3) Could the *bi-level optimization* alleviate the overfitting issue?

(RQ4) How does our method perform with different mixing neighbor strategies?

5.1 Experiment Settings

Datasets. For semi-supervised node classification, we employ four widely used benchmark datasets: 1) Cora; 2) Citeseer; 3) Pubmed; and 4) ogbn-arXiv [Hu *et al.*, 2020]. The statistics of datasets are presented in the Table 1. For the split of OOD classes, we strictly follow the standard OOD detection benchmark on graphs [Stadler *et al.*, 2021].

Baselines. In this paper, we implement 10 representative GNN models covering *Spectral-based*, *Spatial-based*, *Deeper* and *Robust* GNNs, which are GCN [Kipf and Welling, 2017], ChebNet [Defferrard *et al.*, 2016], GraphSAGE [Hamilton *et al.*, 2017], GAT [Veličković *et al.*, 2018], SGC [Wu *et al.*, 2019], JKNet [Xu *et al.*, 2018], APPNP [Klicpera *et al.*, 2018], SuperGAT [Kim and Oh, 2020], GCNII [Chen *et al.*, 2020], and DropEdge [Rong *et al.*, 2019]. We implement all baselines and our proposed models in *PyTorch* and *PyTorch-Geometric*.

Evaluation Metrics. For semi-supervised node classification task, we evaluate the performance of the node classification in terms of the in-distribution by using the accuracy metric. For OOD detection task, we implement a unified OOD detection method based on the uncertainty of prediction (e.g., Entropy) [Mukhoti *et al.*, 2021]. And we use Area Under the Receiver Operating Characteristic curve (AUROC) as the evaluation metric, which is a threshold-independent performance evaluation. The average results and standard deviations of five runs are reported on all datasets.

	Cora	Citeseer	Pubmed	arXiv
# Nodes	2,708	3,327	19,717	169,343
# Edges	10,556	9,104	88,648	2,315,598
# Features	1,433	3,703	500	128
# Labels	7	6	3	40
# $\lvert\mathcal{C}_{out}\rvert$	3	2	1	15
# Fraction	33.38%	33.18%	39.94%	39.11%

Table 1: Statistics of the experimental datasets on semi-supervised node classification. Further, we provide the number of OOD classes and the fraction of OOD nodes for OOD detection.

Methods	Cora	Citeseer	Pubmed	arXiv
GCN	87.4±0.3	66.0±0.6	89.0±0.2	47.4±0.6
ChebNet	85.6±0.4	65.0±0.6	88.4±0.3	46.5±0.4
GraphSAGE	85.3±1.2	65.8±0.7	89.6±0.6	46.8±0.9
GAT	88.7±0.6	69.6±0.6	90.6±0.9	49.8±1.5
SGC	87.2±0.3	69.2±0.2	91.5±0.6	40.5±2.6
JKNet	86.7±1.1	67.3±0.7	93.1±0.1	50.6±0.6
APPNP	88.2±0.4	68.3±0.5	93.2±0.1	51.3±0.9
SuperGAT	88.3±0.5	69.3±0.8	91.3±1.0	49.2±0.7
GCNII	88.7±0.3	69.4±1.4	93.0±0.7	51.6±1.7
DropEdge	88.9±0.7	69.6±1.2	93.0±0.9	51.7±2.7
LMN(Ours)	**89.7**±0.6	**71.1**±0.6	**93.4**±0.1	**54.1**±1.4

Table 2: Comparison of semi-supervised node classification accuracy (%) ↑ results. The best performance is highlighted in bold.

5.2 Node Classification Task (RQ1)

Following semi-supervised node classification setting [Rong *et al.*, 2019], we apply the standard fixed training/validation/testing split, where 20 labeled nodes per class are for training, 500 nodes are for validation, and the remaining nodes are for testing. We report the average accuracy of baselines and LMN in Table 2. From this table, we have the following observations: 1) Attention-based models like GAT and SuperGAT generally outperform other shallow GNNs including GCN, ChebNet and SGC. It implies the attention mechanism is helpful to focus on the self- or neighbor-information and beneficial to node classification. 2) The deeper GNNs achieve higher performance than shallow GNN, which indicates that deeper GNNs can improve the performance of model. 3) The DropEdge outperforms its backbone GCNII and other baselines, which indicates that DropEdge can improve the robustness of model even if it adopts a random way to drop edges. 4) Our proposed LMN outperforms all baselines because LMN can effectively identify data to be in-distribution or OOD and thus alleviate across-distribution mixture.

Methods	Cora	Citeseer	Pubmed	arXiv
GCN	77.8±0.4	73.1±2.2	63.3±1.4	56.1±0.5
ChebNet	73.5±1.3	69.7±4.0	62.2±1.2	57.1±0.8
GraphSAGE	75.6±1.8	72.8±3.1	59.5±2.0	56.9±1.0
GAT	80.2±1.4	77.9±3.1	61.6±4.2	58.0±1.0
SGC	70.0±0.8	75.5±2.3	61.4±1.8	51.8±1.5
JKNet	76.3±1.8	70.8±3.4	64.4±1.8	52.9±0.6
APPNP	77.8±0.5	72.3±2.7	64.3±0.8	53.7±0.3
SuperGAT	78.5±1.6	78.1±1.6	63.2±3.9	54.1±0.8
GCNII	78.0±1.3	72.4±2.1	65.2±3.9	56.3±2.2
DropEdge	79.3±0.9	75.2±3.5	63.0±2.1	57.9±0.9
LMN(Ours)	**80.5**±1.2	**78.5**±3.2	**68.7**±1.3	**60.4**±0.3

Table 3: Comparison of OOD detection AUROC (%) ↑ results. The best performance is highlighted in bold.

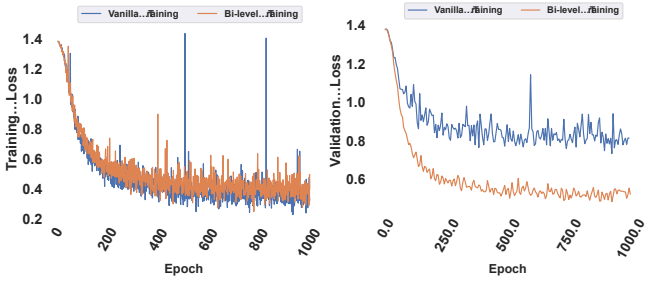


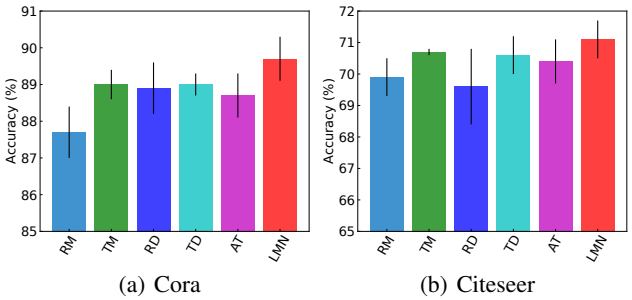
Figure 4: The training and validation losses on Cora.

5.3 OOD Detection Task (RQ2)

To comprehensively evaluate the OOD detection performance, we adopt AUROC as the evaluation metric. The AUROC is a threshold-independent performance evaluation metric. The results of AUROC are reported in Table 3. From this table, we have following observations: 1) Attention-based models like GAT and SuperGAT get better results than weighted average models since the attention weighted aggregation is helpful to OOD detection. 2) The DropEdge outperforms its backbone GCNII except Pubmed; however, it adopts a random dropping leading to a lack of interpretability. 3) Our proposed LMN achieves the best performance, which indicates once again that LMN alleviates the across-distribution mixture between in-distribution and OOD samples.

5.4 The Effect of Bi-level Optimization (RQ3)

To demonstrate the effectiveness of bi-level optimization for over-fitting issue, we analyze the model loss of vanilla training (we simultaneously optimize ϕ and θ on training data without validation) as well as the loss of the bi-level optimization with validation. Fig. 4 shows the learning curves of training loss and validation loss on the Cora dataset. From such figure, we can find that the over-fitting issue leads to a low training loss in (a) but a high validation loss in (b). And the bi-level optimization has a lower validation loss than vanilla training, which indicates that the bi-level optimization with validation can effectively alleviate the over-fitting issue.



(a) Cora (b) Citeseer

Figure 5: Performance of OOD detection scheme with GCNII.

Methods	OOD Modules	Cora	Citeseer
GCNII	**None**	88.7±0.3	69.4±1.4
LMN	**Mixing Neighbors**	**89.7**±0.6	**71.1**±0.6

Table 4: Ablation study on two datasets.

5.5 Mixing Strategies (RQ4)

We further consider five OOD models for comparison, RandomMask (RM), TruthMask (TM), RandomDrop (RD), TruthDrop (TD), and ATtention (AT). Specifically, 1) RM masks neighbor nodes randomly with different masking proportions. 2) TM masks neighbor nodes by using OOD ground truth. 3) RD removes edges randomly. 4) RD drops edges by OOD ground truth. 5) AT adopts attention mechanism. For RM and RD, we report the best result on dropping proportions $\{0.1, 0.2, 0.3, 0.4, 0.5\}$. To make a fair comparison, a state-of-the-art graph neural network architecture GCNII [Chen *et al.*, 2020] is adopted as backbone.

Results. We report the results on Cora and Citeseer datasets for page limitation but the observation is consistent for all datasets. From Fig. 5, we observe that: 1) TM and TD outperform AT, indicating that AT cannot avoid across-distribution mixture well. 2) Both TM and TD surpass RM and RD. It shows that the randomly discarding on nodes and edges cannot obtain satisfactory results. 3) LMN outperforms RM, RD, TD, and AT consistently, indicating that LMN alleviates across-distribution mixture effectively.

5.6 Ablation Study

Table 4 shows the results for brief ablation study on two datsets. Our LMN uses GCNII with OOD detection while GCNII has no OOD detection module. LMN outperforming GCNII demonstrates the effectiveness of our detection scheme.

6 Conclusions

Current efforts on promoting GNNs mostly focus on classification accuracy on closed-set assumption. In this paper, we study a novel problem of end-to-end open-set semi-supervised node classification. The novel method LMN in a variational inference framework has been proposed for simultaneous node classification and OOD detection in an end-to-end manner. Extensive experiments on four various datasets demonstrate the effectiveness of our proposed method.

Acknowledgements

Authors would like to thank funding support from NSFC General Program (No.62176215). Dr. Yuan Fang's work in this research is supported by the Lee Kong Chian Fellowship awarded by Singapore Management University.

References

[Bitterwolf *et al.*, 2022] Julian Bitterwolf, Alexander Meinke, Maximilian Augustin, and Matthias Hein. Revisiting ood detection: A simple baseline is surprisingly effective. In *ICLR 2022 Submitted*, 2022.

[Chen *et al.*, 2020] Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. Simple and deep graph convolutional networks. In *ICML*, 2020.

[Dai *et al.*, 2021] Enyan Dai, Zhimeng Guo, and Suhang Wang. Label-wise message passing graph neural network on heterophilic graphs. In *CoRR*, 2021.

[Defferrard *et al.*, 2016] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *NeurIPS*, 2016.

[Denouden *et al.*, 2018] Taylor Denouden, Rick Salay, Krzysztof Czarnecki, Vahdat Abdelzad, Buu Phan, and Sachin Vernekar. Improving reconstruction autoencoder ood detection with mahalanobis distance. *CoRR*, 2018.

[DeVries and Taylor, 2018] Terrance DeVries and Graham W Taylor. Learning confidence for out-of-distribution detection in neural networks. *CoRR*, 2018.

[Geisa *et al.*, 2021] Ali Geisa, Ronak Mehta, Hayden S Helm, Jayanta Dey, Eric Eaton, Jeffery Dick, Carey E Priebe, and Joshua T Vogelstein. Towards a theory of out-of-distribution learning. *CoRR*, 2021.

[Hamilton *et al.*, 2017] William L Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *NeurIPS*, 2017.

[Hendrycks and Gimpel, 2016] Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *CoRR*, 2016.

[Hsu *et al.*, 2020] Yen-Chang Hsu, Yilin Shen, Hongxia Jin, and Zsolt Kira. Generalized odin: Detecting out-of-distribution image without learning from out-of-distribution data. In *CVPR*, 2020.

[Hu *et al.*, 2020] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. In *NeurIPS*, 2020.

[Kim and Oh, 2020] Dongkwan Kim and Alice Oh. How to find your friendly neighborhood: Graph attention design with self-supervision. In *ICLR*, 2020.

[Kingma and Welling, 2013] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *CoRR*, 2013.

[Kipf and Welling, 2017] Thomas N Kipf and Max Welling. Semi-supervised classification with graph neural networks. In *ICLR*, 2017.

[Klicpera *et al.*, 2018] Johannes Klicpera, Aleksandar Bojchevski, and Stephan Günnemann. Predict then propagate: Gnns meet personalized pagerank. *CoRR*, 2018.

[Lee *et al.*, 2018] Kimin Lee, Honglak Lee, Kibok Lee, and Jinwoo Shin. Training confidence-calibrated classifiers for detecting out-of-distribution samples. In *ICLR*, 2018.

[Liang *et al.*, 2018] Shiyu Liang, Yixuan Li, and R Srikant. Enhancing the reliability of out-of-distribution image detection in neural networks. In *ICLR*, 2018.

[Luo *et al.*, 2021] Dongsheng Luo, Wei Cheng, Wenchao Yu, Bo Zong, Jingchao Ni, Haifeng Chen, and Xiang Zhang. Learning to drop: Robust graph neural network via topological denoising. In *WSDM*, 2021.

[Ma *et al.*, 2021] Xiaoxiao Ma, Jia Wu, Shan Xue, Jian Yang, Quan Z Sheng, and Hui Xiong. A comprehensive survey on graph anomaly detection with deep learning. *CoRR*, 2021.

[Mukhoti *et al.*, 2021] Jishnu Mukhoti, Andreas Kirsch, Joost van Amersfoort, Philip HS Torr, and Yarin Gal. Deep deterministic uncertainty: A simple baseline. In *CoRR*, 2021.

[Pimentel *et al.*, 2014] Marco AF Pimentel, David A Clifton, Lei Clifton, and Lionel Tarassenko. A review of novelty detection. *Signal Processing*, 2014.

[Rong *et al.*, 2019] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. Dropedge: Towards deep graph convolutional networks on node classification. In *ICLR*, 2019.

[Stadler *et al.*, 2021] Maximilian Stadler, Bertrand Charpentier, Simon Geisler, Daniel Zügner, and Stephan Günnemann. Graph posterior network: Bayesian predictive uncertainty for node classification. In *NeurIPS*, 2021.

[Veličković *et al.*, 2018] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. In *ICLR*, 2018.

[Vernekar *et al.*, 2019] Sachin Vernekar, Ashish Gaurav, Vahdat Abdelzad, Taylor Denouden, Rick Salay, and Krzysztof Czarnecki. Out-of-distribution detection in classifiers via generation. *CoRR*, 2019.

[Wu *et al.*, 2019] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *ICML*, 2019.

[Wu *et al.*, 2020] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on gnns. *TNNLS*, 2020.

[Xiao *et al.*, 2021] Teng Xiao, Zhengyu Chen, Donglin Wang, and Suhang Wang. Learning how to propagate messages in graph neural networks. In *SIGKDD*, 2021.

[Xu *et al.*, 2018] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka. Representation learning on graphs with jumping knowledge networks. In *ICML*, 2018.

[Zhao *et al.*, 2020] Xujiang Zhao, Feng Chen, Shu Hu, and Jin-Hee Cho. Uncertainty aware semi-supervised learning on graph data. In *CoRR*, 2020.