Learning on Graphs with Out-of-Distribution Nodes

Yu Song
songyu132@outlook.com
Westlake University
Hangzhou, China

Donglin Wang*
wangdonglin@westlake.edu.cn
Westlake University
Hangzhou, China

ABSTRACT

Graph Neural Networks (GNNs) are state-of-the-art models for performing prediction tasks on graphs. While existing GNNs have shown great performance on various tasks related to graphs, little attention has been paid to the scenario where out-of-distribution (OOD) nodes exist in the graph during training and inference. Borrowing the concept from CV and NLP, we define OOD nodes as nodes with labels unseen from the training set. Since a lot of networks are automatically constructed by programs, real-world graphs are often noisy and may contain nodes from unknown distributions. In this work, we define the problem of *graph learning with out-of-distribution nodes*. Specifically, we aim to accomplish two tasks: 1) detect nodes which do not belong to the knwon distribution and 2) classify the remaining nodes to be one of the known classes. We demonstrate that the connection patterns in graphs are informative for outlier detection, and propose Out-of-Distrubution Graph Attention Network (OODGAT), a novel GNN model which explicitly models the interaction between different kinds of nodes and separate inliers from outliers during feature propagation. Extensive experiments show that OODGAT outperforms existing outlier detection methods by a large margin, while being better or comparable in terms of in-distribution classification.

CCS CONCEPTS

• **Mathematics of computing** → **Graph algorithms**; • **Computing methodologies** → Neural networks; **Anomaly detection**.

KEYWORDS

Graph Neural Networks; Outlier Detection

ACM Reference Format:
Yu Song and Donglin Wang. 2022. Learning on Graphs with Out-of-Distribution Nodes. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '22), August 14–18, 2022, Washington, DC, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3534678.3539457

1 INTRODUCTION

Graphs neural networks (GNNs) have become the *de facto* tool for performing prediction tasks on graphs. Among various applications, one of the most important tasks of GNNs is semi-supervised node classification (SSNC) [11]. In SSNC, GNNs aggregate information from adjacent nodes and generate representations that are smooth within neighborhoods, alleviating the difficulty of classification.

In recent years, many studies have begun to consider graph learning tasks in realistic settings, such as graphs with label noise [4], low labeling rates [27] and distribution shifts [2, 33]. However, very few work has considered the scenario where out-of-distribution (OOD) nodes exist in the graph on which one performs SSNC. By using the term 'OOD', we borrow the notion from CV and NLP, which means samples with labels not seen in the training set. In the graph domain, this can be quite common as graphs are usually constructed in an incremental way where new nodes are added due to the connectivity with existing ones, and for most cases there is no guarantee that nodes must connect to others from the same distribution. For example, we want to classify papers in a citation network into AI-related topics, e.g., deep learning, reinforcement learning and optimization methods. The network is obtained using a web crawler which adopts a breadth first search (BFS) strategy and keeps exploring papers referring to existing ones for a number of iterations. When searching stops, the resulting network is not guaranteed to contain nodes only from the known categories, as it is common for a scientific paper to refer to articles in less relevant research areas, for example, an AI paper may cite papers in neuroscience and mathematics. To make the task even more distinct from the traditional outlier detection, the OOD classes may contain nodes in comparable size with ID ones, if not larger. *Given a noisy graph as such, our task is to predict the label for nodes which correspond to one of the known classes, and identify nodes that do not belong to any of them.*

In CV and NLP, OOD detection has been a hot research area with a long history. [9] demonstrates that neural networks tend to assign higher softmax probabilities to in-distribution (ID) samples than to out-of-distribution (OOD) ones, and propose to use the maximum softmax probability (MSP) produced by the neural network as the score for OOD detection. Other approaches attempt to improve detection performance by modifying the model structure [30, 34], employing customized uncertainty measures [14] or exploiting labeled outliers [10].

Different from the above methods which only focus on identifying OOD samples at inference time, the presence of OOD nodes in graphs makes the task more challenging. First, in traditional settings of CV and NLP, outliers only occur in the test set, while in the graph domain one is usually given the entire graph for training which consists of both inliers and outliers, transferring the problem from detecting *unknown-unknown* to *known-unknown*. How to leverage the availability of outliers is the key to success. Second, the classifier in CV and NLP is usually trained in a fully supervised

*Corresponding author.

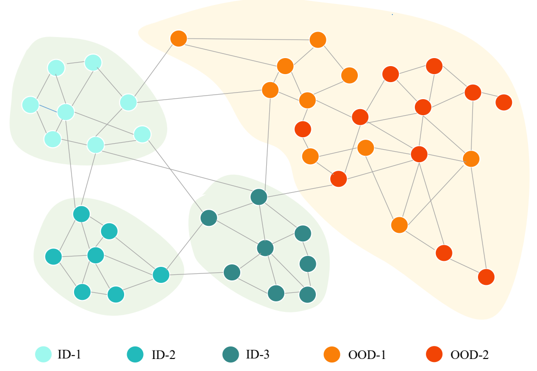


Figure 1: An illustration of graph learning with out-of-distribution nodes. In this task, we aim to accomplish two things: 1) separate ID nodes from OOD nodes and 2) classify the ID nodes correctly. Colors of nodes indicate their labels, while the shaded areas represent a set of possible decision boundaries to achieve the goal. Unlike traditional anomaly detection, for graphs, the OOD part can contain nodes that are comparable in size to the ID part, or even larger.

manner with abundant labeled data, while for graphs the most common approach for node classification is to train a GNN with limited labeled data in a semi-supervised way. Due to the message-passing framework adopted by GNNs, the latent features of ID and OOD nodes can be affected by each other. Therefore, it is important to study how the information flow between inliers and outliers can affect the performance of in-distribution classification and outlier detection. A similar question arises in [21], where the authors investigate the performance of semi-supervised learning (SSL) methods when labeled and unlabeled data are drawn from different distributions. However, the problem remains unexplored in the field of graph-based SSNC. Third, since our purpose is to address node classification and outlier detection in a joint framework, a natural question is how to combine the two tasks into a unified model, and how to balance the impact of one task on the other.

In this work, we first analyze the impact of OOD nodes on graph learning tasks with GNNs. We demonstrate that for graphs with high homophily, message-passing GNNs are inherently good at detecting outliers due to the smoothness effect caused by feature propagation. Furthermore, we find that removing inter-edges between ID and OOD nodes while preserving intra-edges within each cluster can lead to the overall best performance. Motivated by these findings, we propose a novel GNN model called OODGAT which utilizes attention mechanism and explicitly models the interaction between inliers and outliers. Experiments show that OODGAT outperforms all baselines in terms of both detection and classification, and even surpasses post-hoc detection methods which are tuned directly on the test set.

To the best of our knowledge, we are the first to formally define the problem of *graph learning with OOD nodes.* [36] considiers a similar setting where the graph also contains OOD nodes. They developed a Bayesian framework to detect outliers by calculating multiple uncertainty measures. Our work differs in that we analyze the fundamental advantages of GNNs from the perspective of network geometry, and exploit the information contained in the graph structure to solve the problem in an efficient and elegant way.

To summarize, our work makes the following contributions:

- We formalize the problem of *graph learning with OOD nodes* and identify its challenges.
- We analyze the problem from the perspective of graph structure and present the basic design choice to achieve good performance.
- We propose a novel GNN model called OODGAT which explicitly distinguishes inliers from outliers during feature propagation and solves the problem of node classification and outlier detection in the same framework.
- We conduct extensive experiments on various graph datasets to demonstrate the effectiveness of the proposed method.

2 RELATED WORKS

2.1 Graph Neural Networks

Graph neural networks (GNNs) have shown great performance in various applications related to graphs. In this work, we focus on the problem of semi-supervised node classification (SSNC) [11]. In SSNC, GNNs aggregate features from neighboring nodes and produce a latent space where the similarity between node embeddings corresponds to the connection patterns between nodes in the geometry space. The most commonly used GNNs include graph convolutional network (GCN) [11], graph attention network (GAT) [29] and GraphSAGE [7].

2.2 Outlier Detection

Outlier detection, also known as OOD detection, has been a hot research area in various domains. Based on the availability of OOD data during training, OOD detectors can be classified into three types, namely unsupervised, supervised and semi-supervised methods.

Unsupervised Methods. Unsupervised methods only utilize in-distribution data to train the outlier detector. Among various techniques proposed by researchers, the most common approaches include ODIN [16] and Mahalanobis-distance [14]. These methods are called *post-hoc detectors* as they assume the classification network is already trained on in-distribution data, and the detector is built on top of the pretrained classifier by calibrating its output probabilities or exploiting its latent space. Other approaches like [25, 28, 30] require training an additional model which is designed specifically for OOD detection, apart from the original classification network. Unsupervised methods do not utilize the abundant unlabeled data during training and can only find sub-optimal solutions since they treat the classification and outlier detection as two independent tasks.

Supervised Methods. Supervised methods assume access to a set of OOD samples during training [8, 10, 13]. Such methods train the classifier in an end-to-end fashion using cross-entropy loss on the ID training data to minimize classification error, together with a confidence penalty loss on the labeled OOD data to maintain low prediction confidence. For example, [13] applies a KL-divergence term to OOD samples to ensure their predictions are close to uniform distribution. Supervised detectors generally outperform unsupervised ones for they manage to exploit the distributional information provided by training OOD data. However, the OOD samples are either from a different but related dataset [10] or generated by GANs [18], limiting its usage in the graph domain where one cannot find such proxy OOD dataset or generate pseudo OOD data easily.

Semi-supervised Methods. Inspired by semi-supervised learning, recent studies in OOD detection also consider the setting where an unlabeled set is available during training [1, 34, 37]. [37] defines a novel task called 'semi-supervised OOD detection', where one is given a limited set of labeled inliers and a large mixed set of both inliers and outliers, whose identities cannot be known during training. They employ contrastive learning to obtain latent representations of unlabeled samples and compute their distance to the centers of in-distribution data to get the OOD score. [1] adopts a similar setting but solves the problem with model ensemble. The drawbacks of these methods include 1) they are not designed for graph data and thus cannot utilize the structural information; 2) they usually require training an additional detection model and cannot handle classification and detection in the same framework.

2.3 Semi-supervised Learning With Distribution Mismatch

Another way to understand the proposed task is to consider it as a semi-supervised learning problem on graphs. SSL assumes access to only a small set of labeled data and a relatively large set of samples without label information. Oliver et al. [21] points out a surprising fact that existing SSL methods tend to degrade the original classification performance when there exists a class distribution mismatch between labeled and unlabeled data. Following their discovery, researchers have developed SSL methods that are robust against OOD samples, with performance being at least as good as fully-supervised learning [6, 12, 35]. The key idea of such methods is straightforward: they attempt to detect and remove the OOD part of the unlabeled data and apply SSL techniques only on the remaining purified set. This setting resembles ours in that they also treat the problem as two tasks, i.e., semi-supervised learning on in-distribution data and outlier detection on the unlabeled set, where each task has its influence on the other. However, these approaches perform SSL by adding regularization terms to the original classification loss (e.g., cross-entropy), like VAT [20] and minimum entropy regularization [5], while in the graph domain, SSNC is usually done with GNNs which achieve semi-supervised learning in an implicit way.

3 LEARNING ON GRAPHS WITH OUT-OF-DISTRIBUTION NODES

In this section, we define the problem of *graph learning with OOD nodes* and motivate the design choice of OODGAT.

3.1 Problem Formulation

Consider a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ denotes the set of nodes and $\mathcal{E}$ denotes the set of edges. The graph structure is represented by an binary adjacency matrix $\mathbf{A} \in \{0,1\}^{|\mathcal{V}|\times|\mathcal{V}|}$. Each node v in the graph is associated with a feature vector $\mathbf{x}_v$ and a label y_v, and the overall feature matrix and class vector can be represented by $\mathbf{X}$ and $\mathbf{y}$, respectively. In SSNC, the node set can be further divided into $\mathcal{V} = \mathcal{V}_l \cup \mathcal{V}_u$ where $\mathcal{V}_l$ refers to the set of nodes whose labels are accessible during training. Similarly, the feature matrix and class vector can be divided into $\mathbf{X} = \left[\mathbf{X}_l{}^\top, \mathbf{X}_u{}^\top\right]^\top$ and $\mathbf{y} = [\mathbf{y}_l \parallel \mathbf{y}_u]$. The aim of SSNC is to predict the labels for nodes in $\mathcal{V}_u$ using the training set $(\mathbf{X}_l, \mathbf{y}_l)$, the unlabeled features $\mathbf{X}_u$ and the graph structure $\mathbf{A}$. Different from traditional close-world SSNC which assumes that nodes in $\mathcal{V}_l$ and $\mathcal{V}_u$ are sampled from the same distribution, we generalize the problem into a more realistic setting where nodes in $\mathcal{V}_u$ may come from different distributions than nodes in $\mathcal{V}_l$. Due to the distribution shift between labeled and unlabeled data, the class vector $\mathbf{y}_u$ may contain labels not seen in $\mathbf{y}_l$, and the label space $\mathcal{Y}$ is enlarged by $\mathcal{Y} = \mathcal{Y}_l \cup \mathcal{Y}_u$. For simplicity, we denote nodes with labels from $\mathcal{Y}_l$ by ID nodes or inliers and nodes with labels from $\mathcal{Y}_u \setminus \mathcal{Y}_l$ by OOD nodes or outliers. We call this setting *graph learning with OOD nodes* and the purpose is to 1) assign labels from $\mathcal{B} = \{0, 1\}$ to nodes in $\mathcal{V}_u$ where 0 stands for ID and 1 for OOD and 2) for nodes tagged as ID, we further classify them to be one of the classes in $\mathcal{Y}_l$. Note that for both tasks, we are presented with the whole graph $\mathcal{G}$ during training, leading to a *semi-supervised* and *transductive* setting. In the remaining of the article, we call the two tasks Semi-Supervised Outlier Detection (SSOD) and Semi-Supervised Node Classification (SSNC) for the sake of simplicity.

3.2 Semi-supervised Outlier Detection

Unlike previous outlier detection methods which are designed primarily for CV and NLP tasks and derive the detector using only in-distribution data, the uniqueness of graphs makes us wonder: *can we leverage the unlabeled data* $\mathbf{X}_u$ *and the graph structure* $\mathbf{A}$ *for better OOD detection?* To answer the question, we first take a brief review at the most common task on graphs, namely, SSNC. In SSNC, a GNN is used to propagate information between adjacent nodes and produce a latent space where features are distributed smoothly w.r.t. the graph structure [15]. The smoothness is desirable due to the widely adopted homophily assumption, i.e., connected nodes tend to share the same label [22]. We argue that, like SSNC, the connection pattern between nodes can also provide information for distinguishing ID from OOD. We start by giving the following proposition:

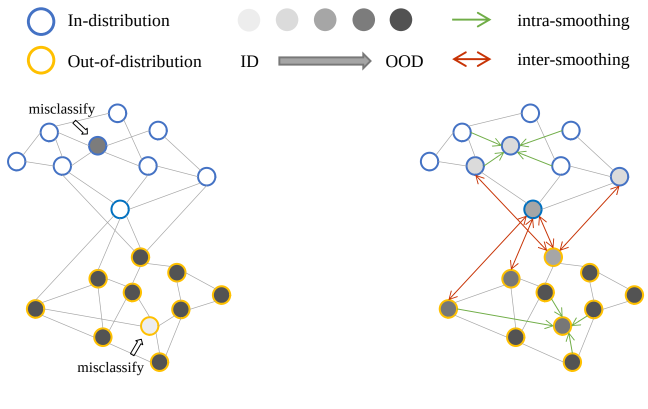
Proposition. *Given a graph $\mathcal{G}$, the set of original labels $\mathcal{Y} = \mathcal{Y}_l \cup \mathcal{Y}_u$, and the set of identity labels $\mathcal{B} = \{0, 1\}$. Assume that:*

(1) There exists a mapping $f : \mathcal{Y} \mapsto \mathcal{B}$ which maps each label in $\mathcal{Y}$ to be ID or OOD;

(2) $\mathcal{G}$ is homophilic w.r.t. to $\mathcal{Y}$, i.e., edges in $\mathcal{G}$ tend to connect nodes with the same label in $\mathcal{Y}$.

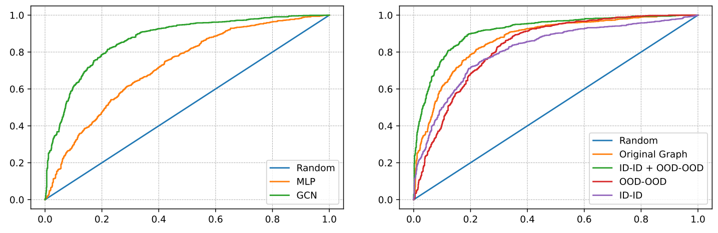
Then, $\mathcal{G}$ is also homophilic w.r.t. $\mathcal{B}$.

The proof of the proposition is presented in Appendix A. From the proposition, we make the following hypothesis: GNNs are naturally well suited for SSOD because they are inherently equipped with a regularizer that pushes the predicted OOD scores to be close within densely connected communities, which is helpful for graphs with high homophily. We illustrate this in Figure 2. The left figure shows the OOD scores obtained without considering the graph structure. Overall, the scores of OOD nodes are higher than ID nodes, with an exception in each community due to the weakness of modern neural networks [8]. By smoothing features according



(a) Before smoothing (b) After smoothing

Figure 2: Smoothness helps OOD detection. The border of circles represents the true identity of nodes, while the darkness of the inner color represents the predicted OOD score. The arrows indicate the smoothing effect of GNNs.



(a) ROC curves of OOD detection using MLP and GCN as detectors. GCN uses the original graph as input.

(b) ROC curves of GCN detectors. Different lines are trained on graphs with different subsets of edges.

Figure 3: Effect of feature smoothing on Cora.

to the graph structure (Figure 2b), GNN manages to recover the true scores of nodes from their neighbors (green arrows). We also notice the edges that connect different kinds of nodes, which lead to undesirable feature aggregation and compromise the separation between inliers and outliers (red arrows). However, since the number of intra-edges significantly exceeds that of inter-edges (for graphs with high homophily), the overall performance should be better than not utilizing structural information at all.

To verify the hypothesis, we conduct an experiment on Cora [32] using Multilayer Perceptron (MLP) and GCN [11] as predictors, and calculate the entropy of the predicted class distribution as the OOD score as in [17, 23, 36]. The higher the entropy, the more likely the model considers the node to be OOD. The ROC curves for both methods are plotted in Figure 3a, from which we can see the GCN detector outperforms the MLP counterpart by a large margin, validating that the graph structure is useful for detecting outliers. To better understand the impact of different kinds of connections, we test the detection performance on graphs with different subsets of edges, and the results are shown in Figure 3b. As expected, removing inter-edges from the graph leads to improved detection performance (green line vs orange line). However, the performance drops sharply when we further remove edges within ID (red line) or OOD (purple line) communities, indicating that smoothness within the same type of nodes is critical for successful detection.

3.3 Semi-supvervised Node Classification

It is known that the distribution mismatch between labeled and unlabeled data can hurt the performance of semi-supervised learning [21]. For graph-based SSNC, unlabeled nodes convey their influence to the model through connections to labeled nodes, so it is natural to expect the same performance drop observed in [21] when the graph contains edges connecting inliers and outliers. However, the problem here is more sophisticated. On the one hand, the information exchange between ID and OOD data may introduce noise to the interested distribution, making the model prone to overfitting and leading to poor generalization; On the other hand, the addition of inter-connections can enhance the connectivity of the graph and facilitate the propagation of supervision signals among nodes. Moreover, the connection patterns between inliers and outliers may provide knowledge about classifying ID nodes. Therefore, it is difficult to tell whether the presence of inter-connections is beneficial or detrimental to the task. To find out the impact of inter-connections, we conducted experiments on some commonly used graph datasets using GCN as the classifier and report the mean accuracy across 9 runs in Table 1. For each graph, we test the classification accuracy in three cases: preserving all inter-edges, randomly dropping half of them, and removing all inter-edges. Empirically, we conclude that removing inter-edges does not hurt generalization on in-distribution data, instead it can be beneficial for some datasets.

Table 1: Effect of Inter-Edge Removal

Dataset	Remove=0	Remove=0.5	Remove=1.0
Cora	92.0	92.5	**92.7**
CoauthorCS	92.8	92.6	**93.0**
Amazon-Photo	97.0	97.0	**97.2**
Amazon-Computers	81.2	81.5	**83.2**

4 OODGAT: END-TO-END MODEL FOR SSOD AND SSNC

In this section, we first introduce the attention architecture adopted by OODGAT, and then propose three regularization terms to guide the learning process of OODGAT.

4.1 Attention Mechanism: From Node to Edge

Since it is important to separate in-distribution nodes from OOD nodes, it is natural to resort to attention mechanism which adaptively computes the weights for aggregating information from neighbors. The general form of graph convolution with attention is:

$$\mathbf{h}_i' = \sigma\left(\sum_{j\in\mathcal{N}(i)\cup\{v_i\}} \alpha_{ij}\mathbf{W}\mathbf{h}_j\right) \tag{1}$$

where α_{ij} is the attention weight for aggregating information from v_j to v_i. The difference between various graph attention networks lies in the way the attention values are calculated. For example, GAT [29] propose to compute the (unnormalized) attention weights between v_i and v_j by $e_{ij} = LeakyReLU\left(\mathbf{a}^\top \left[\mathbf{W}\mathbf{h}_i \,\|\, \mathbf{W}\mathbf{h}_j\right]\right)$, where they use a single layer neural network parameterized by $\mathbf{a}$ to output attention weights. However, none of the previous methods takes OOD nodes into account, and the attention coefficients obtained from their methods are not guaranteed to contain knowledge about how to distinguish inliers from outliers.

In OODGAT, we propose to explicitly model the interaction between inliers and outliers. Based on the discussion in Section 3, we summarize three properties the attention mechanism should have : 1) allow messages to pass within in-distribution nodes, 2) allow message passing within out-of-distribution nodes and 3) block information flow between inliers and outliers. Therefore, we propose the following attention form:

$$e_{ij} = 1 - |w(i) - w(j)| \tag{2}$$

where $w(i)$ and $w(j)$ are the attention scores for v_i and v_j, respectively. If we imagine $w(v)$ as a binary classifier that assigns different weights to inliers and outliers, we can find that Equation (2) satisfies all the properties discussed above. We illustrate this in Figure 4. Without loss of generosity, when w_v and $w_{c'}$ are large, say $w_v = w_{c'} = 1$, and w_u and w_c are small, say $w_u = w_c = 0$, the attention weights for intra-edges become $e_{cu} = e_{c'v} = 1$, while the weights for inter-edges become $e_{cv} = e_{c'u} = 0$. We also note that, for any node v_i, the weight with which it attends to itself is fixed to be $e_{ii} = 1$, i.e., the maximum value possible for all node pairs. This is desirable since maintaining more self-information can be helpful when the neighborhood may contain contaminated features. After obtaining e_{ij}, we normalize them in each neighborhood using softmax to keep the embedding scale unchanged before and after aggregation:

$$\alpha_{ij} = softmax_j(e_{ij}) = \frac{exp(e_{ij})}{\sum_{k\in\mathcal{N}(i)\cup\{v_i\}} exp(e_{ik})} \tag{3}$$

The binary classifier $w(v)$ can be defined in various forms. To avoid too many parameters and a complex model, we simply implement it as a logistic regression classifier parameterized by $\mathbf{a} \in \mathbb{R}^{d'}$ over the latent space of a GNN layer, i.e., $w(v) = \sigma\left(\mathbf{a}^\top \mathbf{W}\mathbf{h}_v\right)$, where $\mathbf{W} \in \mathbb{R}^{d'\times d}$ is the weight matrix of the GNN layer, $\mathbf{h}_v \in \mathbb{R}^d$ is the input of the layer, and σ is the sigmoid function. The classifier aims to find a partition of the latent space such that inliers and outliers are well separated from each other. To enhance the expressiveness of the model, we extend the attention computation to a multi-head variant, similar to [29]:

$$\mathbf{h}'_i = \Big\|_{k=1}^{K} \sigma\left(\sum_{j\in\mathcal{N}(i)\cup\{v_i\}} \alpha_{ij}^k \mathbf{W}^k \mathbf{h}_j\right) \tag{4}$$

where K is the number of attention heads and $\|$ means concatenation. In the prediction layer, the concatenation is replaced with averaging to keep the dimension reasonable for classification:

$$\mathbf{z}_i = softmax\left(\frac{1}{K}\sum_{k=1}^{K} \sum_{j\in\mathcal{N}(i)\cup\{v_i\}} \alpha_{ij}^k \mathbf{W}^k \mathbf{h}_j\right) \tag{5}$$

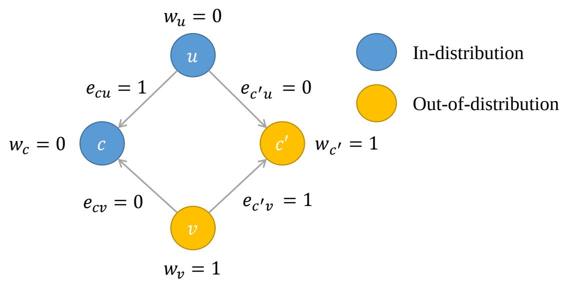


Figure 4: Attention computation of OODGAT. c and c' indicate central nodes, while u and v are their neighbors. Arrows indicates the direction of feature propagation. OODGAT first computes the node-level scores of nodes being ID or OOD, then transfers them to edge-level attention weights for feature propagation.

where $\mathbf{z}_i$ is the predicted class distribution of v_i, $\sum_{k=1}^{|\mathcal{Y}_l|} z_{ik} = 1$.

4.2 Regularizer

Of course, the cross-entropy loss alone is not sufficient to make OODGAT work in the expected way. In particular, we want the embedded binary classifier to learn knowledge about how to distinguish inliers from outliers. Therefore, we propose three regularizers to guide the learning process of OODGAT, i.e., consistency loss, entropy loss and discrepancy loss. The architecture of OODGAT is shown in Figure 5, which we will explain in detail in the following sections.

Consistency Regularizer. OODGAT integrates a bianry classifer to measure the OOD score for nodes in the graph, and transfers node-level scores to edge-level attention weights for aggregating features. Besides the scores predicted by the classifier, we can also obtain the output distribution of nodes at the final layer of the model, from which the entropy can be calculated as another kind of OOD measurement. We denote the scores predicted by the classifier as w, and the scores given by entropy as e. To coordinate the relationship between w and e, we design the following regularization term called consistency loss:

$$\mathcal{L}_{con} = -\cos\left(\mathbf{w}, \mathbf{e}\right) \tag{6}$$

where $\mathbf{w}$ represents the vector of OOD scores predicted by the classifier:

$$\begin{gathered} \mathbf{w} = [w_1, w_2, \cdots, w_{|\mathcal{V}|}]^\top \\ w_i = \sigma(\mathbf{a}^\top \mathbf{W}\mathbf{h}_i) \end{gathered} \tag{7}$$

and $\mathbf{e}$ denotes the vector of OOD scores given by entropy:

$$\begin{gathered} \mathbf{e} = [\sigma(\tilde{e_1}), \sigma(\tilde{e_2}), \cdots, \sigma(\tilde{e_{|\mathcal{V}|}})]^\top \\ \tilde{e_i} = \frac{e_i - \mu_e}{\sigma_e} \\ e_i = H(\mathbf{z}_i) = -\sum_{j=1}^{|\mathcal{Y}_l|} z_{ij} log(z_{ij}) \end{gathered} \tag{8}$$

where μ_e and σ_e denotes the mean and standard deviation of all e_is, and $H(\mathbf{z})$ is the entropy of the predicted class distribution given by the last layer of OODGAT.

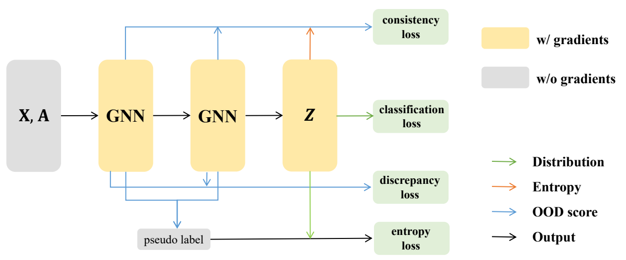


Figure 5: Architecture of OODGAT. Arrows of different colors indicate different information to extract from the layer for loss computation. Yellow and grey rectangles represent layers w/ and w/o gradients propagation, respectively.

In Equation (6), we use cosine similarity to constrain the directions of $\mathbf{w}$ and $\mathbf{e}$ to be close, i.e., the two methods should give similar predictions across all nodes. The intuition behind the consistency regularizer is the causal relationship between the attention mechanism and the model's final output. That is, when the scores given by the classifier change, the attention weights used for aggregating features will also change, which in turn affects the final output of the model. If we regard the change of the classifier as the cause, then the change of the model's output can be viewed as the effect. By aligning cause and effect, the hypothesis space of the model is reduced and gradient descent is more likely to find solutions that are closer to ground truth. Imagine an extreme case where the classifier works perfectly and produces OOD scores close to ground-truth. In this case, the attention weights of edges also become near perfect and the model becomes extremely powerful in detecting outliers as it smooths representations for all ID and OOD clusters and prevents information flow between ID and OOD communities. As a result, the OOD scores computed from entropy are also close to reality, making the angle between $\mathbf{w}$ and $\mathbf{e}$ small. From another perspective, we can interpret the consistency loss as a kind of supervised learning: the entropy provides supervision to the classifier and vice versa. As training progresses, the classifier not only learns from the final output but also teaches the model to produce more reliable predictions by differentiating ID and OOD better in the latent space. Thus, the two modules play a chasing game and benefit each other.

For OODGAT with two layers, the consistency loss is computed for both layers, and in each layer, the score vector $\mathbf{w}$ is averaged across all heads:

$$\mathcal{L}_{con} = -\frac{1}{2} \times \left(\cos(\mathbf{w}^1, \mathbf{e}) + \cos(\mathbf{w}^2, \mathbf{e})\right)$$
$$\mathbf{w}^l = \left[\frac{1}{K}\sum_{k=1}^{K} w_1^{lk}, \frac{1}{K}\sum_{k=1}^{K} w_2^{lk}, \cdots, \frac{1}{K}\sum_{k=1}^{K} w_{|\mathcal{V}|}^{lk}\right]^\top \tag{9}$$

Entropy Regularizer. In OODGAT, we use entropy as the measure of predictive uncertainty. As training progresses, the cross-entropy loss continuously pulls the outputs of labeled nodes toward one-hot distribution, pushing their entropy to the lowest level. Due to the generalization ability of neural networks, nodes outside the training set may also produce low-entropy predictions, especially those with attributes similar to or closely connected to the labeled ones, resulting in some low-entropy regions in the graph. In contrast to the classification loss, we want to keep the uncertainty of outliers as high as possible to counteract the entropy-reducing effect caused by cross-entropy. However, the true identities of nodes cannot be obtained during training, so we take the predictions given by the binary classifier as pseudo-labels, and make the outputs of pseudo-OOD nodes close to uniform distribution to enhance the distinguishability between inliers and outliers. Thus, we get the following entropy loss:

$$\mathcal{L}_{ent} = \frac{\sum_{i=1}^{|\mathcal{V}|} CE\left(\mathbf{u}, \mathbf{z}_i\right)\delta\big(w(i) > \epsilon\big)}{\sum_{i=1}^{|\mathcal{V}|}\delta\big(w(i) > \epsilon\big)} \tag{10}$$

where $\mathbf{u}$ is uniform distribution, $\mathbf{z}_i$ is the predicted class distribution of v_i, ϵ is the threshold for selecting pseudo-OOD nodes, δ means the Kronecker delta.

Discrepancy Regularizer. For OODGAT with two graph convolutional layers, we further constrain the difference between the OOD scores computed by the two layers by minimizing the following discrepancy loss:

$$\mathcal{L}_{dis} = -\cos\left(\mathbf{w}^1, \mathbf{w}^2\right) \tag{11}$$

Final Objective. Therefore, the optimal parameters of OODGAT are obtained by minimizing the following loss:

$$\mathcal{L}_{OODGAT} = -\frac{1}{|\mathcal{V}_l|}\sum_{i=1}^{|\mathcal{V}_l|} log(z_{iy_i}) + a^{b\times t}(\beta\mathcal{L}_{con} + \gamma\mathcal{L}_{ent} + \zeta\mathcal{L}_{dis}) \tag{12}$$

where β, γ and ζ are the balance parameters of regularizers. In addition, $a^{b\times t}$ is used to decay the weights of regularizers gradually as training progresses so as to control the balance point between ID classification and OOD detection. a is a number between 0 and 1, b is a small number and t is the iteration step. In the experiments, we set a and b to be 0.9 and 0.01, respectively. By combining the three regularizers with cross-entropy, OODGAT not only learns to classify in-distribution nodes, but also learns to separate inliers from outliers in the latent space, as shown in Figure 6.

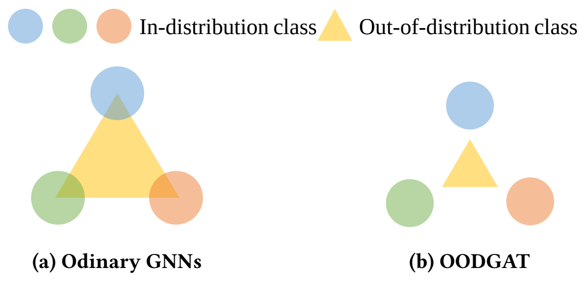


Figure 6: Latent space of ordinay GNNs and OODGAT. By combining the attention module and the tailored regularizers, OODGAT manages to produce a clearer gap between inliers and outliers, while being more effective in distinguishing ID classes.

5 EXPERIMENT

In this section, we test OODGAT on various real world graph datasets to demonstrate its effectiveness. Some visualization results are presented in Appendix F due to space limitation.

5.1 Experimental setup

Evaluation Metrics. In the setting of *graph learning with OOD nodes*, we aim to accomplish two tasks simultaneously, namely 1) node classification and 2) outlier detection. For the first task, we adopt classification accuracy as the evaluation metric. For the second task, we calculate two metrics commonly found in the OOD detection literature, namely the area under ROC curve (AUROC) and the false positive rate when the true positive rate achieves 95% (FPR@95). Note that in all experiments we view the outliers as positive. To comprehensively evaluate the performance of the two tasks, we consider them together as a multi-class classification problem with N+1 classes, i.e., N in-distribution classes and one OOD class. We call this task *joint classification*, and the performance can be evaluated by weighted-F1[1].

Datasets. We test OODGAT on six commonly used graph datasets, which are Cora, AmazonComputers, AmazonPhoto, CoauthorCS, LastFMAsia and Wiki-CS [19, 24, 26, 32]. For each dataset, we divide all classes into in-distribution and out-of-distribution, such that the ID part contains classes with a relatively balanced number of nodes, and the number of ID classes is at least three to avoid too easy classification. Similar to traditional SSNC, we randomly select 20 nodes per ID class as the training set. Besides, we construct a small validation set which contains 10 nodes from each ID class, and the same number of outliers randomly sampled from OOD classes. Statistics for the datasets are listed in Appendix C.

Methods. Here we compare the following methods:

- **End-to-end Methods**, which accomplish SSOD and SSNC in the same framework. Specifically, we choose MLP, GCN [11], GraphSAGE [7], GAT [29], and GATv2 [3] as the end-to-end baselines. MLP is used to test the performance without considering graph topology, while the other four are representative GNN models w/ or w/o graph attention. For all methods, we use the entropy of the predicted distribution as the OOD score.
- **Post-hoc OOD Detectors**, which require training an additional outlier detector on top of the pretrained classifier. We employ ODIN [16], Mahalanobis-distance [14], and CaGCN [31] as the post-hoc detectors. ODIN uses temparature sacling and input preprocesssing to calibrate the output distribution, while Mahalanobis-distance leverages the latent space of the pretrained classifier to compute the distance between testing samples and known inliers. For each method, we use the metric described in the original paper for OOD detection, i.e., MSP for [16] and Mahalanobis-distance for [14]. CaGCN is a recently published method for calibrating the output confidence of GNNs. Intuitively, we can use the calibrated confidence as the score for outlier detection.
- **GKDE** [36], the abbreviation for Graph-based Kernel Dirichlet distribution Estimation, a method specifically designed to detect outliers on graphs. It proposes a multi-source uncertainty framework using various types of predictive uncertainties from both deep learning and belief theory, and shows that *vacuity* is the best metric for OOD detection.
- **OODGAT**, the method proposed in this paper. It has two versions: OODGAT-ENT which uses the entropy of the predicted distribution as the measure of outliers and OODGAT-ATT which uses the score given by the binary classifier for decision making.

[1]The details of joint classification are explained in Appendix B.

Implementation Details. For all graphs, we perform 3 random splits to obtain training, validation, and test sets. For each split, we initialize the model with 3 random seeds.[2] As a result, each experiment was performed 9 times. Unless specially mentioned, we tune the hyperparameters using grid search and select the best performing results according to the validation set. Specifically, we choose the learning rate from [0.01, 0.1], the dropout probability from [0, 0.5]. For models with multi-head attentions, the number of attention heads is chosen from [1, 4, 8], and the drop edge probability is set to 0.6. It is known that weight decay is helpful in preventing models from giving arbitrary high confidence, so we also choose the weight decay factor from [0, 5e-5, 5e-4, 5e-3]. We set the maximum iterations of training to be 1000 and perform early-stopping when $(AUROC + Accuracy)$ stops to increase for 200 epochs. All experiments are done using Pytorch Geometric, and the source code is made publicly available on Github.[3]

5.2 Main Results

Comparison with End-to-end Methods. We first compare our method with end-to-end approaches. The results are listed in Table 2. From the table, we make the following observations:

1) On all datasets, GNNs outperform MLP in both SSOD and SSNC by a large margin, suggesting that the graph structure is helpful for both tasks, as indicated in Section 3.

2) GraphSAGE surpasses GCN in terms of AUROC on 5 out of the 6 datasets, which may be attributed to the strategy of separating self and neighboring representations during feature propagation.

3) Across all baseline models and datasets, GAT and/or GATv2 achieve the best performance in outlier detection. The results show that even the naive attention mechanism helps to distinguish nodes from different distributions.

4) For SSOD, OODGAT outperforms all baselines on the six datasets by a considerable margin. On easy datasets such as AmazonPhoto and CoauthorCS, OODGAT achieves an AUROC of over 0.98, while for difficult tasks like LastFMAsia and Wiki-CS, OODGAT greatly improves the detection ability and achieves decent performance, demonstrating the effectiveness of the proposed propagation strategy.

5) For SSNC, OODGAT achieves better or comparable results than other approaches. For example, OODGAT outperforms GAT and/or GATv2 by 3% and 1% in terms of classification accuracy on AmazonComputers and LastFMAsia. The results show that by removing the interference brought by OOD data, the classifier is more likely to converge to points with better generalization ability.

[2]In the comparison with GKDE, we use the split given by the original author in https://github.com/zxj32/uncertainty-GNN

[3]https://github.com/SongYYYY/KDD22-OODGAT

Table 2: Comparison with End-to-end Methods

	Cora	AmazonCS	AmazonPhoto	CoauthorCS	LastFMAsia	Wiki-CS
	Acc ↑ / AUROC ↑ / FPR@95 ↓ / F1 ↑					
MLP	74.1/72.4/75.5/63.1	68.4/65.7/84.6/54.6	91.8/80.2/71.9/72.8	88.6/95.0/28.9/84.8	54.5/57.4/87.0/51.2	78.6/71.7/76.4/64.0
GCN [11]	92.1/88.9/46.0/80.5	81.2/83.3/61.9/70.3	97.1/88.3/44.6/80.7	**92.7**/94.5/32.2/86.4	79.8/72.1/74.7/66.5	80.9/71.7/76.6/63.0
SAGE [7]	90.8/87.7/46.6/79.2	83.2/84.6/54.9/71.7	97.1/93.5/32.0/87.2	92.6/97.0/16.8/89.1	79.3/73.7/68.9/67.0	78.6/73.0/65.3/66.2
GAT [29]	91.6/90.1/40.8/81.5	82.3/88.5/42.9/76.5	96.9/92.5/31.7/86.1	92.0/96.6/16.7/89.0	82.3/81.1/49.6/75.0	79.9/79.8/63.6/70.0
GATv2 [3]	91.5/90.4/40.0/81.9	83.5/88.6/45.7/76.3	95.7/94.4/21.1/88.4	91.7/96.6/19.1/88.7	81.9/79.7/52.3/73.5	**81.4**/80.9/58.9/70.6
OODGAT-ENT	**92.3**/92.9/31.4/84.4	**86.6**/92.2/**40.4**/81.4	**97.6**/98.2/8.1/93.4	92.4/98.9/3.7/92.6	**83.3**/**93.4** /**22.4**/**83.5**	**81.4**/**88.8**/**48.5** /**76.6**
OODGAT-ATT	**92.3**/**93.6**/**26.1** /**85.1**	**86.6**/**93.1**/45.2/**82.2**	**97.6**/**98.3**/**5.8** /**93.9**	92.4/**99.6**/**1.6** /**93.5**	**83.3**/91.9/27.7/81.0	**81.4**/88.3/51.2/73.7

Table 3: Comparison with Post-hoc OOD Detectors

	GAT (base)[29]	ODIN [16]	Mahalanobis -Distance[14]	CaGCN [31]	OODGAT -ENT	OODGAT -ATT
	AUROC ↑ / FPR@95 ↓					
Cora	90.7/36.8	90.7/37.2	87.3/50.3	89.9/45.7	93.4/29.6	**94.1**/**25.0**
AmazonCS	84.1/51.9	84.4/51.2	81.8/78.8	83.6/56.2	91.3/**47.2**	**92.3**/52.0
AmazonPhoto	94.3/21.7	94.3/26.5	77.1/59.6	94.4/24.1	98.3/7.3	**98.4**/**4.2**
CoauthorCS	96.2/19.6	96.1/19.8	94.0/25.3	95.8/22.1	99.1/2.4	**99.6**/**1.4**
LastFMAsia	78.5/60.7	81.1/52.9	83.4/51.0	89.6/30.4	**91.4**/**25.4**	90.5/26.8
Wiki-CS	80.4/62.5	80.4/62.5	74.0/74.4	82.7/54.7	**88.7**/50.0	88.6/**49.0**

Table 4: Ablation Study

loss	Acc ↑	AUROC ↑	F1↑
(1) CE	82.1	56.9	50.7
(2) CE+con	86.3	90.0	78.6
(3) CE+ent	82.9	53.5	48.9
(4) CE+dis	79.3	61.2	46.1
(5) CE+con+ent	85.9	92.8	81.3
(6) CE+con+dis	**87.0**	89.4	78.7
(7) OODGAT	86.6	**93.1**	**82.2**

6) From the perspective of *joint classification*, OODGAT consistently outperforms all competitors, making it the most powerful method for *graph learning with OOD nodes*.

Comparison with Post-hoc OOD Detectors. We also compare OODGAT with ODIN [16], Mahalanobis-distance [14] and CaGCN [31]. The comparison is unfair as these methods either require additional data preprocessing or involve multiple training stages, while OODGAT accomplishes the mission without introducing additional complexity. For all experiments except OODGAT, we pretrain a GAT as the base classifier, and employ different post-hoc detectors for OOD detection. Note that unlike the original paper, we tune the detectors directly on the test set to eliminate the possibility of bad hyperparameter configurations. For OODGAT, we do not utilize the test set for training or tuning. Table 3 reports the detection performance of all methods. As we can see, only in a few cases can the post-hoc detectors improve the detection ability (shaded cells). Apart from that, all methods lose their power due to the characteristics of graph data such as lack of supervision and non-continuous input. By inspecting the last two columns, we find that despite being extremely unfair, OODGAT outperforms all post-hoc detectors by a large margin. The superiority of OODGAT comes from the end-to-end optimization strategy which simultaneously handles feature extraction and OOD detection, whereas other methods use a two-stage update framework which trains the classifier and the detector separately and can only find sub-optimal solutions.

Comparison with GKDE. We now compare OODGAT with GKDE [36]. To ensure a fair comparison, we test our method on the same datasets used in the original paper and adopt the same preprocessing procedures. (See Appendix C for details.) We report the AUROC and AUPR for outlier detection in Table 5, where the results for GKDE are obtained from the original paper. As we can see, although OODGAT is much more efficient than GKDE which requires multiple forward passes due to the Bayesian framework, it still outperforms GKDE in both AUROC and AUPR on all three datasets. The results show that it is not enough to simply embed existing GNNs into frameworks of uncertainty computation. Instead, making better use of the information implicit in the graph structure is the key to success.

5.3 Ablation Study

The success of OODGAT comes from the combination of the unique propagation strategy and the tailored regularizers for guiding the training process. In this section, we perform ablation analysis in Table 4 to demonstrate the importance of each module proposed in Section 4.2. Experiments are done on AmazonComputers using OODGAT-ATT, and the weight for each loss is the same as the best-performing result in Table 2. In (1), we train the model with only cross-entropy loss. The AUROC for outlier detection is around 50% which is similar to random guessing, indicating the use of cross-entropy alone is not sufficient to learn the classification of ID and OOD. We then add one of the proposed regularizers in (2),(3) and (4), respectively. The results show that consistency loss can effectively improve the discriminative ability of the binary classifier, while entropy loss and discrepancy loss contribute little or negatively when used without the help of consistency regularizer. This is expected since the other two losses rely on the accurate prediction of the binary classifier, which is learned through consistency loss. Comparing (2) and (5), we find that the entropy loss can further improve the detection ability when used together with consistency loss. Similarly, the comparison between (2) and (6) indicates that the addition of discrepancy regularizer can help the classification of in-distribution samples. The best result is obtained in (7) where we combine all three regularizers with cross-entropy loss. In summary, all regularizers contribute to the final performance, among which

Table 5: Comparison with GKDE

Dataset	AUROC		AUPR	
	GKDE	OODGAT	GKDE	OODGAT
Cora	87.6	**91.4**	78.4	**82.9**
Citeseer	84.8	**87.7**	86.8	**89.0**
Pubmed	74.6	**81.1**	69.6	**76.0**

consistency loss plays the most important role. For the information about the sensitivity of hyperparameters, please see Appendix E for details.

6 CONCLUSION

In this paper, we propose and study the problem of *graph learning with OOD nodes*. We demonstrate that GNNs are inherently suitable for outlier detection on graphs with high homophily, and propose an end-to-end model OODGAT to tackle the problem of SSOD and SSNC. Extensive experiments show that while existing methods such as input preprocessing and temparature scaling cannot handle the problem well, OODGAT consistently yields decent performance in both in-distribution classification and outlier detection. In the future, we aim to extend OODGAT to more realistic settings such as few-shot learning and incremental learning.

ACKNOWLEDGMENTS

This work has been supported by Science and Technology Innovation 2030-Major Project (2022ZD0208800): Brain Science and Brain-like Research and NSFC General Program (No.62176215).

REFERENCES

[1] Țifrea Alexandru, Stavarache Eric, and Yang Fanny. 2021. Novelty detection using ensembles with regularized disagreement. *arXiv preprint arXiv:1811.05868* (2021).

[2] Beatrice Bevilacqua, Yangze Zhou, and Bruno Ribeiro. 2021. Size-Invariant Graph Representations for Graph Classification Extrapolations. *arXiv preprint arXiv:2103.05045* (2021).

[3] Shaked Brody, Uri Alon, and Eran Yahav. 2021. How Attentive are Graph Attention Networks? *arXiv preprint arXiv:2105.14491* (2021).

[4] Enyan Dai, Charu Aggarwal, and Suhang Wang. 2021. NRGNN: Learning a Label Noise-Resistant Graph Neural Network on Sparsely and Noisily Labeled Graphs. *arXiv preprint arXiv:2106.04714* (2021).

[5] Yves Grandvalet, Yoshua Bengio, et al. 2005. Semi-supervised learning by entropy minimization. *CAP* 367 (2005), 281–296.

[6] Lan-Zhe Guo, Zhen-Yu Zhang, Yuan Jiang, Yu-Feng Li, and Zhi-Hua Zhou. 2020. Safe deep semi-supervised learning for unseen-class unlabeled data. In *International Conference on Machine Learning*. PMLR, 3897–3906.

[7] William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 1025–1035.

[8] Matthias Hein, Maksym Andriushchenko, and Julian Bitterwolf. 2019. Why relu networks yield high-confidence predictions far away from the training data and how to mitigate the problem. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 41–50.

[9] Dan Hendrycks and Kevin Gimpel. 2016. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136* (2016).

[10] Dan Hendrycks, Mantas Mazeika, and Thomas Dietterich. 2018. Deep anomaly detection with outlier exposure. *arXiv preprint arXiv:1812.04606* (2018).

[11] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[12] Saito Kuniaki, Kim Donghyun, and Saenko Kate. 2021. OpenMatch: Open-set Consistency Regularization for Semi-supervised Learning with Outliers. *Advances in Neural Information Processing Systems* 34 (2021).

[13] Kimin Lee, Honglak Lee, Kibok Lee, and Jinwoo Shin. 2017. Training confidence-calibrated classifiers for detecting out-of-distribution samples. *arXiv preprint arXiv:1711.09325* (2017).

[14] Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. 2018. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. *Advances in neural information processing systems* 31 (2018).

[15] Qimai Li, Zhichao Han, and Xiao-Ming Wu. 2018. Deeper insights into graph convolutional networks for semi-supervised learning. In *Thirty-Second AAAI conference on artificial intelligence*.

[16] Shiyu Liang, Yixuan Li, and Rayadurgam Srikant. 2017. Enhancing the reliability of out-of-distribution image detection in neural networks. *arXiv preprint arXiv:1706.02690* (2017).

[17] David Macêdo, Tsang Ing Ren, Cleber Zanchettin, Adriano LI Oliveira, and Teresa Ludermir. 2021. Entropic out-of-distribution detection. In *2021 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 1–8.

[18] Petr Marek, Vishal Ishwar Naik, Vincent Auvray, and Anuj Goyal. 2021. OodGAN: Generative Adversarial Network for Out-of-Domain Data Generation. *arXiv preprint arXiv:2104.02484* (2021).

[19] Péter Mernyei and Cătălina Cangea. 2020. Wiki-cs: A wikipedia-based benchmark for graph neural networks. *arXiv preprint arXiv:2007.02901* (2020).

[20] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. 2018. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence* 41, 8 (2018), 1979–1993.

[21] Avital Oliver, Augustus Odena, Colin Raffel, Ekin D Cubuk, and Ian J Goodfellow. 2018. Realistic evaluation of deep semi-supervised learning algorithms. *arXiv preprint arXiv:1804.09170* (2018).

[22] Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. 2020. Geom-gcn: Geometric graph convolutional networks. *arXiv preprint arXiv:2002.05287* (2020).

[23] Jie Ren, Peter J Liu, Emily Fertig, Jasper Snoek, Ryan Poplin, Mark Depristo, Joshua Dillon, and Balaji Lakshminarayanan. 2019. Likelihood Ratios for Out-of-Distribution Detection. *Advances in Neural Information Processing Systems* 32 (2019), 14707–14718.

[24] Benedek Rozemberczki and Rik Sarkar. 2020. Characteristic functions on graphs: Birds of a feather, from statistical descriptors to parametric models. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 1325–1334.

[25] Vikash Sehwag, Mung Chiang, and Prateek Mittal. 2021. Ssd: A unified framework for self-supervised outlier detection. *arXiv preprint arXiv:2103.12051* (2021).

[26] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. 2018. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868* (2018).

[27] Ke Sun, Zhouchen Lin, and Zhanxing Zhu. 2020. Multi-stage self-supervised learning for graph convolutional networks on graphs with few labeled nodes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 5892–5899.

[28] Jihoon Tack, Sangwoo Mo, Jongheon Jeong, and Jinwoo Shin. 2020. Csi: Novelty detection via contrastive learning on distributionally shifted instances. *arXiv preprint arXiv:2007.08176* (2020).

[29] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).

[30] Apoorv Vyas, Nataraj Jammalamadaka, Xia Zhu, Dipankar Das, Bharat Kaul, and Theodore L Willke. 2018. Out-of-distribution detection using an ensemble of self supervised leave-out classifiers. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 550–564.

[31] Xiao Wang, Hongrui Liu, Chuan Shi, and Cheng Yang. 2021. Be Confident! Towards Trustworthy Graph Neural Networks via Confidence Calibration. *Advances in Neural Information Processing Systems* 34 (2021).

[32] Zhilin Yang, William Cohen, and Ruslan Salakhudinov. 2016. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*. PMLR, 40–48.

[33] Gilad Yehudai, Ethan Fetaya, Eli Meirom, Gal Chechik, and Haggai Maron. 2021. From local structures to size generalization in graph neural networks. In *International Conference on Machine Learning*. PMLR, 11975–11986.

[34] Qing Yu and Kiyoharu Aizawa. 2019. Unsupervised out-of-distribution detection by maximum classifier discrepancy. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 9518–9526.

[35] Qing Yu, Daiki Ikami, Go Irie, and Kiyoharu Aizawa. 2020. Multi-task curriculum framework for open-set semi-supervised learning. In *European Conference on Computer Vision*. Springer, 438–454.

[36] Xujiang Zhao, Feng Chen, Shu Hu, and Jin-Hee Cho. 2020. Uncertainty Aware Semi-Supervised Learning on Graph Data. *Advances in Neural Information Processing Systems* 33 (2020).

[37] Zhi Zhou, Lan-Zhe Guo, Zhanzhan Cheng, Yu-Feng Li, and Shiliang Pu. 2021. STEP: Out-of-Distribution Detection in the Presence of Limited In-distribution Labeled Data. *Advances in Neural Information Processing Systems* 34 (2021).

A PROOF OF PROPOSITION

We now prove the proposition in Section 3.

Proof. The homophily is the fraction of edges in a graph which connect nodes that have the same label. In [22], the node homophily ratio is defined as:

$$h = \frac{1}{|\mathcal{V}|} \sum_{v \in \mathcal{V}} \frac{|\{(u,v) : u \in \mathcal{N}(v) \land y_u = y_v\}|}{|\mathcal{N}(v)|}$$

Assuming a graph $\mathcal{G}$ whose node homophily ratio w.r.t. $\mathcal{Y}$ is h. By definition, we can derive the node homophily ratio w.r.t. $\mathcal{B}$ as:

$$h' = \frac{1}{|\mathcal{V}|} \sum_{v \in \mathcal{V}} \frac{|\mathcal{A}(v)| + \sum_{u \in \mathcal{B}(v)} \delta(f(y_u) = f(y_v))}{|\mathcal{N}(v)|}$$

where $\mathcal{A}(v) = \{u : u \in \mathcal{N}(v) \land y_u = y_v\}$, $\mathcal{B}(v) = \{u : u \in \mathcal{N}(v) \land y_u \neq y_v\}$, and $\mathcal{N}(v) = \mathcal{A}(v) \cup \mathcal{B}(v)$. f is a mapping from $\mathcal{Y}$ to $\mathcal{B}$.

Since for any v, we have

$$|\mathcal{A}(v)| = |\{(u,v) : u \in \mathcal{N}(v) \land y_u = y_v\}|$$

and

$$\sum_{u \in \mathcal{B}(v)} \delta(f(y_u) = f(y_v)) \geq 0$$

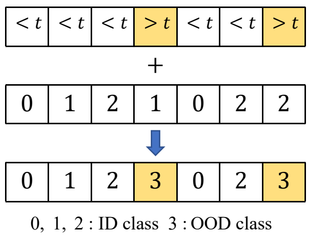
we can derive that

$$h' \geq h$$

Therefore, if a graph $\mathcal{G}$ is homophilic w.r.t. $\mathcal{Y}$, it is also homophilic w.r.t. to $\mathcal{B}$. □

B JOINT CLASSIFICATION

The joint classification includes two stages: first, it classifies nodes to be inliers or outliers according to the OOD scores predicted by the model; then, it assigns in-distrubtion labels for nodes tagged as ID using their output distributions. An illustration is shown in Figure 7. Since the first stage is a binary classification task, the value of weighted-F1 is dependent on the threshold chosen. In the experiments, we report the best F1 value under all possible thresholds.



$< t$ $< t$ $< t$ $> t$ $< t$ $< t$ $> t$
+
0 1 2 1 0 2 2
0 1 2 3 0 2 3
0, 1, 2 : ID class 3 : OOD class

Figure 7: Illustration of joint classification. First row represents the predicted OOD scores, where t is the threshold. Second row represents the predicted labels (3 classes in this case). Combining the two rows, we derive the final result of joint classification (3+1 classes in this case).

C DATASET STATISTICS

Statistics for datasets used in the main experiments are listed in Table 6.

Table 6: Statistics for Main Datasets

Dataset	#Nodes	#Edges	#Features	#Classes
Cora	2708	10556	1433	7
Amazon-Computer	13752	491722	767	10
Amazon-Photo	7650	238162	745	8
Coauthor-CS	18333	163788	6805	15
LastFMAsia	7624	55612	128	18
Wiki-CS	11701	297110	300	10

Experimental setup for main datasets is shown in Table 7.

Table 7: Experimental Setup for Main Datasets

Dataset	OOD class	OOD ratio
Cora	[0, 1, 3]	0.51
Amazon-Computer	[0, 3, 4, 5, 9]	0.49
Amazon-Photo	[1, 6, 7]	0.52
Coauthor-CS	[0, 1, 2, 3, 4, 9, 13]	0.51
LastFMAsia	[1, 2, 3, 4, 5, 9, 10, 12, 17]	0.53
Wiki-CS	[0, 2, 4, 5]	0.50

Statistics for citation datasets are listed in Table 8.

Table 8: Statistics for Citation Datasets

Dataset	#Nodes	#Edges	#Features	#Classes
Cora	2708	10556	1433	7
Citeseer	3327	9104	3703	6
Pubmed	19717	88648	500	3

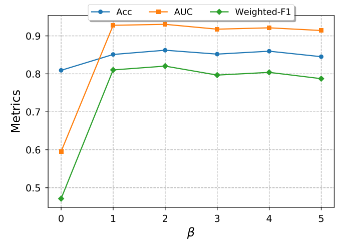
Experimental setup for citation datasets is shown in Table 9.

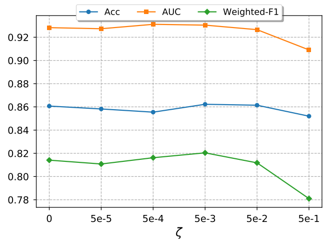
Table 9: Experimental Setup for Citation Datasets

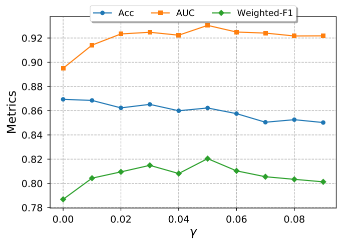
Dataset	OOD classes	OOD ratio
Cora	[0, 1, 2, 3]	0.38
Citeseer	[0, 1, 2]	0.55
Pubmed	[0, 1]	0.40

D EXPERIMENT DETAILS

The results from Table 2 are obtained with the following hyperparameter configurations:


(a) β: consistency loss


(b) ζ: discrepancy loss


(c) γ: entropy loss

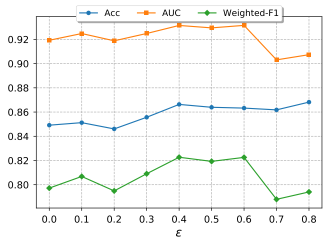

(d) ϵ: threshold

Figure 8: Hyperparameter analysis on AmazonComputers.

Table 10: Hyperparameter Configurations of Main Results

Data	lr	dropout	β	γ	ζ	ϵ	heads
Cora	0.01	0.5	2	0.05	0.005	0.6	4
Amazon-Computer	0.01	0.5	2	0.05	0.005	0.4	4
Amazon-Photo	0.01	0.5	3	0.1	0.005	0.4	4
Coauthor-CS	0.01	0.5	4	0.05	0.005	0.6	4
LastFMAsia	0.01	0.5	3	0.3	0.005	0.5	1
Wiki-CS	0.01	0.5	3	0.2	0.005	0.5	4

The results from Table 5 are obtained with hyperparameters in Table 11.

Table 11: Hyperparameter Configurations of Citation Datasets

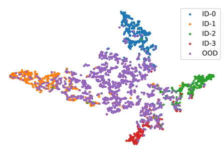
Data	lr	dropout	β	γ	ζ	ϵ	heads
Cora	0.01	0.5	1	0.01	0.005	0.5	4
Citeseer	0.01	0.5	2	0.01	0.005	0.5	4
Pubmed	0.01	0.5	1	0.1	0.005	0.4	4

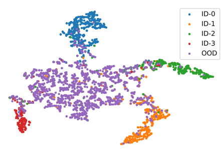
E INFLUENCE OF HYPERPARAMETERS

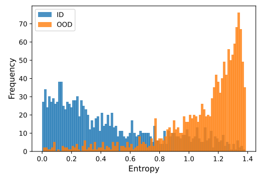
The training of OODGAT involves four hyperparameters: β, γ, ζ and ϵ. The former three are the balance parameters of regularizers, while the last is the threshold to determine the set of nodes on which the model encourages uniform distribution. We only present the impact of the hyperparameters on AmazonComputers due to space limitation, while similar trends are observed on other datasets. From Figure 8a, we observe that consistency loss is the key to the success of OOD detection. In Figure 8b, the performance is slightly improved when the weight of discrepancy loss is around 5e-3. The results in Figure 8c show that while the addition of the entropy regularizer can improve the detection ability, it also leads to a decrease in the performance of in-distribution classification. However, by choosing an appropriate trade-off parameter, we can achieve better detection capability without having too much impact on the classification, and thus improve the overall performance. The effect of the threshold ϵ is shown in Figure 8d. When the threshold is 0, the entropy loss simply forces all nodes to behave like outliers by increasing the uncertainty of predictions, regardless of their real identities. When moving the threshold to an appropriate range, OODGAT manages to reduce the confidence only for outliers while not affecting in-distribution data, resulting in the highest overall performance.

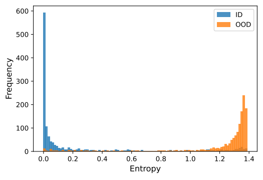
F VISUALIZATION RESULTS

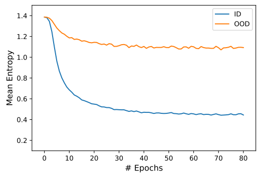
We present some visualization results about GCN and OODGAT in Figure 9.


(a) GCN


(b) OODGAT


(c) GCN


(d) OODGAT


(e) GCN

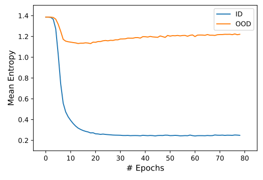

(f) OODGAT

Figure 9: Visualization Results of GCN and OODGAT. Experiments done on Cora. (a) and (b): the t-SNE plot of latent space, (c) and (d): distribution of nodes' predictive uncertainties, (e) and (f): training dynamics of the mean entropy of inliers and outliers. In (a) and (b), OODGAT shows a clearer boundary between ID and OOD classes. In (c) and (d), OODGAT produces scores with less overlap between ID and OOD. In (e) and (f), OODGAT maintains a larger gap between the entropy of inliers and outliers during the whole training stage.