You Never Walk Alone: A Generalizable and Nonparametric Structure Learning Framework

Jiaqiang Zhang, Xinrui Wang, and Songcan Chen, *Senior Member, IEEE*

***Abstract*—The graph-structured learning (GSL) aims to assist graph neural networks (GNNs) to yield effective node embeddings for downstream tasks, especially in scenarios with the absence of structures or the existence of unreliable edges. Most GSL models are built on i.i.d. assumption across training and testing data. However, this assumption can be violated, where testing data contain out-of-distribution (OOD) samples. Consequently, those models are limited in generalization, which will lead to a poor structure. On the other hand, while they have made great progress, additional optimized parameters are required due to their implementation with parametric models. To tackle the above problems, we propose a novel generalizable and nonparametric structure learning framework named GNS, which can be easily and effectively applied to various tasks. GNS neither relies on i.i.d. assumption nor even involves any parameters being optimized, instead to find an appropriate similarity between nodes and an associated threshold to establish desirable structures. Specifically, we first incorporate the candidate neighbor distributions for nodes to refine the similarity. Then, we introduce an adaptive threshold discovery method inspired by Fisher's criterion to determine final structures. Extensive experiments demonstrate that GNS excels not only in OOD scenarios but also in the general classification and regression prediction tasks.**

***Index Terms*—Graph neural networks (GNNs), graph-structured learning (GSL), out-of-distribution (OOD) detection, representation learning.**

I. Introduction

GRAPHS are pervasive data structures in numerous domains [1], such as social networks and citation networks. In recent years, graph neural networks (GNNs) have been considered a powerful tool for learning the representation of graph-structured data [2]. GNNs are fundamentally based on a message-passing mechanism, where the representation of each node is obtained by aggregating and transforming its neighbor information according to the graph structure [3]. Due to their impressive performance, GNNs have been applied to a variety of analytical tasks, such as node classification [4], traffic prediction [5], and text classification [6].

Received 13 June 2024; revised 14 November 2024 and 1 April 2025; accepted 10 June 2025. This work was supported by the National Natural Science Foundation of China under Grant 62376126 and Grant 62076124. *(Corresponding author: Songcan Chen.)*

The authors are with the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China, and also with the MIIT Key Laboratory of Pattern Analysis and Machine Intelligence, Nanjing 211106, China (e-mail: zhangjq@nuaa.edu.cn; wangxinrui@nuaa.edu.cn; s.chen@nuaa.edu.cn).

Digital Object Identifier 10.1109/TNNLS.2025.3580101

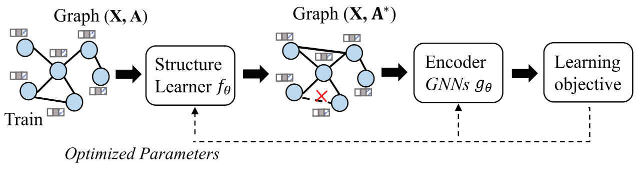
Graph $(\mathbf{X}, \mathbf{A})$
Train
Structure Learner f_θ
Graph $(\mathbf{X}, \mathbf{A}^*)$
Encoder GNNs g_θ
Learning objective
Optimized Parameters

Fig. 1. Parametric GSL paradigm: the structure learner is optimized jointly with GNNs according to the learning objective.

Although promising results have been achieved, existing studies [7], [8], [9] have shown the performance of GNNs is closely related to the quality of the given graph structures. Unfortunately, as graph structures are usually generated from complex interacting systems, they inevitably contain noise edges [10]. For example, users mistakenly follow someone they do notknow in a social network or a wrong citation in a citation network. Moreover, structural information are often missing in image or text datasets. Capturing implicit relationships between samples in these datasets, such as two images containing the same object [11], would enhance the broad applicability of GNNs [12].

Graph-structured learning (GSL) emerges as the latest driving force to not only improve graph structures but also enhance the effectiveness of node embeddings for downstream tasks [13], [14], [15]. Most GSL methods design a structured learner by resorting to parametric models, as shown in Fig. 1, where the structure learner and the encoder (such as GNNs) are optimized jointly under the label supervision [16], [17]. For example, VGCN [18] performs parametric learning on the adjacency matrix with the task of node classification. GSSC [19] applies sparsification techniques to remove potentially uninformative or noisy edges in the neighborhood and then performs contrastive learning on the sparse neighborhood to obtain robust node representations. However, these learned models are limited by the assumption of i.i.d.: the training set and testing set are drawn from the same distribution, which is always violated in the real world. Taking two graphs, ES and DE, from Twitch that come from different distributions [20] as an example, if the model applies structure learning on DE and is then used to model ES, it may result in generating poor graph structures. GraphGLOW [21] is a generalizable method for this problem from a probabilistic data-generative aspect. However, its structure learning module still relies on heavy multilayer neural network optimization. Moreover, for each

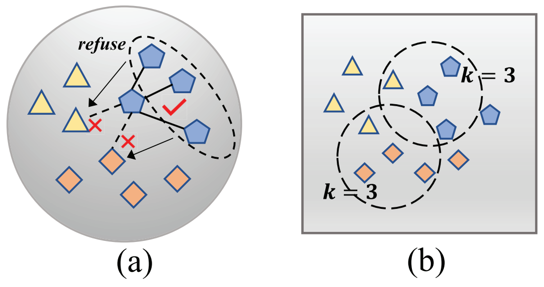

Fig. 2. Two challenges of nonparametric GSL paradigm. (a) Only feature similarity: the nodes with closer distances in the feature space are not necessarily of the same class. (b) One-k-fits-all: the same number of neighbors is set for each node. The above problems lead to undesirable structures.

dataset with different distributions, it requires additional GNN encoders with the label information for supervised learning. In this case, largely overlooked nonparametric methods can be considered since they neither rely on i.i.d. assumption nor introduce additional optimized parameters [22], instead to calculate the similarity between nodes based on features and then set an associated threshold to construct the structures. At this time, we turn to address the following issues.

1) *How to choose a suitable similarity?* Simply using *single-point* information, such as cosine feature similarity, to determine whether two nodes are connected is untrustworthy. As shown in Fig. 2, the points with closer distances in the feature space are not necessarily connected.
2) *How to choose an appropriate number of neighbors for each node?* Current methods set the same threshold to choose neighbors for each node, which will inevitably introduce noisy edges and impair the quality of node embeddings in the information aggregation [23]. Meanwhile, it is unrealistic to manually select an appropriate threshold for each node and each dataset.

To solve the above dilemmas, we propose a novel generalizable and nonparametric structure learning framework named GNS, which does not impose i.i.d. assumption and can be effectively applied to various tasks without any optimized parameters. Meanwhile, we theoretically explain the benefits of this framework based on the message-passing mechanism. Specifically, we suggest that in addition to the *single-point* information, the *contextual* information of nodes is also beneficial, i.e., when the neighbors of a node and its candidate node exhibit resemblance, they are more likely to constitute a true pair. As shown in Fig. 2, it can be intuitively seen that the blue node can reject unrelated connections based on most of its surrounding neighbors with the same class. Thus, we first introduce the candidate neighbor distribution to refine the similarity. Then, we propose an adaptive threshold discovery method according to similarity statistics. In detail, as the proposed similarity is naturally comparable 1-D data, we regard the selection of neighbors as a binary classification problem. The classification criterion is inspired by Fisher's discriminant analysis [24], which is to minimize the variance within classes and maximize the variance between classes. As a result, we can identify a threshold, which could be used to optimally classify samples into two classes: one class is the neighbor node with high similarity and the other with low similarity. At last, we apply the derived structure to conduct extensive experiments on various benchmark datasets. Specifically, we first evaluate the model in the presence of out-of-distribution (OOD) data. GNS can adaptively generate structure given the features of nodes under different distributions without training, thereby avoiding the problem that existing GSL models will learn inductive biases under different distributions. Meanwhile, GNS also demonstrates its impressive performance in the general classification (node, image, and text classification), regression prediction tasks (spatiotemporal sequence prediction), and node clustering tasks. It is worth noting that GNS is model-agnostic and training-free, resulting in greater generality and flexibility. In this article, the main contributions are summarized as follows.

1) *Problem:* We propose an improved nonparametric structure learning for GNNs, which does not impose i.i.d. assumption and can be effectively applied to various tasks without any optimized parameters.
2) *Algorithm:* We propose and theoretically elaborate a novel GSL method GNS, which first introduces the similarity enhanced by candidate neighbor distribution, and then designs an adaptive threshold discovery mechanism based on Fisher's criterion.
3) *Evaluations:* We conduct extensive experiments compared with SOTA methods. The results verify the effectiveness of our model in OOD detection, image/text/node classification, and spatiotemporal sequence prediction tasks, showcasing the broad applicability.

II. Related Works

A. Graph Neural Networks

GNN is a powerful deep learning tool for learning low-dimensional embeddings of graph-structured data, which takes neighbor relations and instance features as input [1], [2]. Existing GNN can be divided into two categories: spatial-based and spectral-based [25], [26]. Among them, the operation process of the former is relatively simple and direct. For each node, the information of neighbor nodes are aggregated according to the topology. The aggregation mechanism has various designs according to the needs, including mean, max, LSTM, and attention [4], [27]. As for the spectral-based GNN, they follow the operation in the graph signal processing, where the convolutional operation is defined in the spectral domain. For example, typical work [28] uses Fourier bias to decompose the graph signal. CNN graph [29] takes the Chebyshev expansion of graph Laplacian to improve efficiency. Recently, techniques have emerged to decouple high- and low-frequency signals in the frequency domain, which enables us to gain a deeper understanding of graph learning in the spectral domain [25].

B. Graph-Structured Learning

GSL aims to explicitly refine graph topology to assist in the expression of GNN [8], [21], [30]. Most existing GSL models are based on deep learning, which estimates the structure using learnable parameters and then optimizes jointly with the encoder (GNN) under the supervision of downstream tasks, such as node classification. For example, LDS [7] models each edge as a Bernoulli random variable and optimizes the graph structure with an encoder. SDEP [31] introduces an innovative prior label constraint matrix in the context of semi-supervised structure learning, refining the graph structure by identifying label consistency. Furthermore, LatGRL [32] is unique in constructing fine-grained homophilic and heterophilic latent graphs to guide the heterogeneous graph representation learning. However, these methods may not work properly in applications, where label information are lacking. SUBLIME [17] and DeepRicci [33] eliminate the need for task-specific supervision that is typically required for GSL with the self-supervised learning paradigm. However, it comes at the cost of a heavy contrastive learning module, which imposes certain limitations on scalability and flexibility. Moreover, Nodeformer [11] and Difformer [12] introduce an improved attention mechanism to efficiently capture the structural relationships between instances. Recently, new learning paradigms based on LLM have emerged, such as GraphEdit [34]. Unlike methods that rely solely on unimodal information, it leverages large language models to incorporate the rich textual data associated with nodes in graph structures. In addition, there are some nonparametric methods, such as kNN-based method [35] and Dropedge [36]. Neuralsparse [37] uses an edge-discarding strategy to denoise graph structure. For applications that GSL can be combined with, G-Splice [38] is a recent representative work. It introduces a novel structure-based extrapolation data augmentation technique to enhance the ability of graph encoders (GCNs) to learn node representations in distribution shift scenarios. An interesting direction is to utilize GSL combined with extrapolation techniques to explore an approach capable of generating graph OOD samples.

C. *OOD Detection on Graph*

In recent years, as a wide range of OOD detection methods have been developed in visual [39] and natural language [40], improving the capabilities of GNNs on OOD data have received increasing attention [41], [42], [43]. GraphDE [44] proposes a generative framework involving node features, labels, and graph structure while deriving the posterior distribution for OOD detection. Bazenov et al. [45] utilized several uncertainty estimation methods and discuss the dependence of OOD detection performance on the structure of graph representations. GOOD-D [41] introduces a carefully designed hierarchical graph contrastive learning model to detect OOD graphs based on their semantic inconsistency. AAGOD [46] designs a learnable amplifier generator a regularized learning strategy for the reusing-based graph OOD detection problem. These mentioned works focus on graph-level detection. For the node level, graph-based kernel Dirichlet distribution estimation (GKDE) method [47] designs a multisource uncertainty framework for detecting OOD nodes. Graph posterior network (GPN) [48] estimates the uncertainty of each node with the help of Bayesian posterior and density estimation. GNNSafe [20] is an advanced learning framework for OOD detection based on energy scores and propagation. NodeSafe [49] further enhances this approach by introducing two optimization terms to constrain negative energy scores and mitigate logit shifts, thereby reducing extreme node scores. However, these methods do not consider the potential impact of noise in structure-based score propagation or the utilization of global structural information in the feature space.

III. Problem Definition

In this section, we first provide the necessary terminology and main problem definitions. A graph can be represented as $\mathcal{G} = (V, E, \mathbf{X})$, where V is the set of N nodes and E is the set of edges. $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N]^{\mathrm{T}} \in \mathbb{R}^{N\times d}$ is the feature matrix, where $\mathbf{x}_i$ is the feature vector of the node v_i. Note that the edges here can be predefined (explicit) or learned by GSL (implicit). The structure can also be described as an adjacency matrix $\mathbf{A} \in \mathbb{R}^{N\times N}$. For graphs, GCN is a typical encoder that iteratively aggregates neighbor information. Formally, given a graph $\mathcal{G} = (V, E, \mathbf{X})$ as input, the lth GCN layer can be written as follows:

$$\mathrm{GCN}(\mathbf{A}, \mathbf{H}^{(l)}) = \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}\mathbf{H}^{(l-1)}\mathbf{W}^{l} \tag{1}$$

where $\mathbf{H}^{(l)}$ represents the embedding of nodes in the lth layer, and $\mathbf{H}^{(0)} = \mathbf{X}$, $\mathbf{D}$ is the degree matrix of $\mathbf{A}$. $\mathbf{W}^{l}$ is the trainable parameter matrix in the lth layer.

Definition 1 (GSL): Given the feature matrix $\mathbf{X} \in \mathbb{R}^{N\times d}$, the goal of GSL is to learn a graph topology $\mathbf{A}_f \in [0, 1]^{N\times N}$. The topology is required to reflect correlations between instances. Specifically, $\mathbf{A}_f^{ij} \in [0, 1]$ indicates whether an edge exists between two instance x_i and x_j. Generally speaking, GSL can be achieved by a parametric model that is jointly optimized with tasks, which is illustrated in Fig. 1. Meanwhile, it can be nonparametric, such as kNN.

Definition 2 (OOD Detection on Graph): The general node classification task is usually under the assumption of i.i.d. assumption: the testing set and the training set come from the same distribution, which is violated in the real world. OOD detection on the graph is to hope that the classifier has good performance to distinguish between in-distribution data and OOD data. Formally, the goal is to find a good decider G associated with the classifier h for each node v on the graph $\mathcal{G} = (V, E, \mathbf{X})$

$$G(v, \mathcal{G}; h) = \begin{cases} 1, & v \text{ is an in-distribution node} \\ 0, & v \text{ is an OOD node.} \end{cases} \tag{2}$$

IV. Methodology

In this section, we will introduce the designed model in detail. Specifically, the model is divided into three parts, which are shown in Fig. 3. The first part is dedicated to obtaining a good similarity measure and will be introduced in Section IV-A. Based on this, the purpose of the second part is to adaptively find different numbers of neighbors for each node to construct an adjacency matrix, which will be introduced in Section IV-B. Finally, the resulting matrix is applied to downstream tasks as described in Section IV-C.

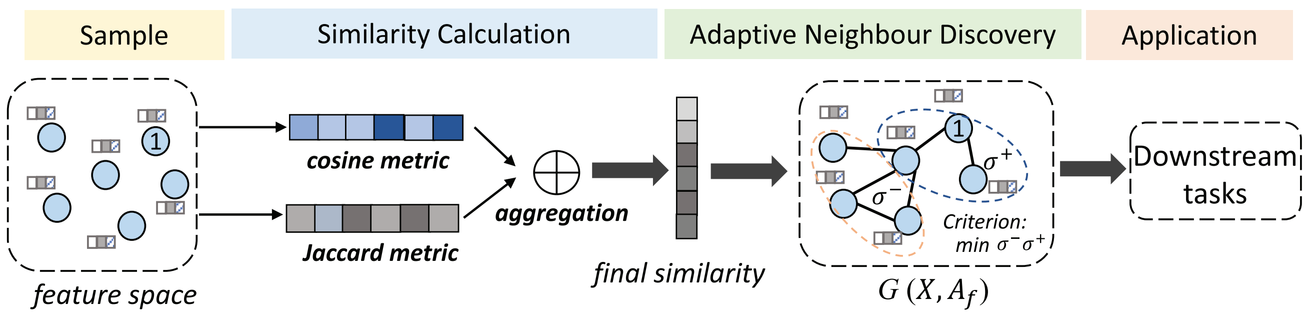

Fig. 3. Overall pipeline of GNS.

A. *Neighbor Distribution Enhanced Similarity*

Most existing GSL methods [11], [23], [35] only use the cosine similarity of original or pretrained features to construct graphs. However, it is unreliable to judge whether they have established the same class of relationship, as shown in Fig. 2. There is a risk of introducing noise edges only by this kind of *single-point* information, which in turn propagates erroneous information when using GNNs for information aggregation. To this end, we intuitively introduce candidate neighbor distribution, i.e., *contextual* information, to encode more structure information, thereby making the calculated similarity more reliable. Specifically, we argue that if the neighbors of a node and its candidate node exhibit resemblance, they are more likely to constitute a true pair. Here, neighbors can be obtained from the original adjacency matrix. However, we consider scenarios where the original structure is missing or noisy. Therefore, for each node, we select the top-k nodes as its neighbors based on the cosine similarity of their features. In practice, we first construct a preliminary feature similarity between each node by cosine metric

$$s_f\left(v_i, v_j\right) = \frac{\mathbf{x}_i \cdot \mathbf{x}_j}{|\mathbf{x}_i||\mathbf{x}_j|}. \tag{3}$$

For the similarity measurement, common methods include cosine, Mahalanobis, Manhattan, and Hamming distances [50]. Depending on the data characteristics in different scenarios, various methods can be chosen to improve model performance. In our work, we chose cosine similarity based on prior studies [17], [23] and data type. Then, for each node v, the initial similarity ranking list can be denoted by $R(v) = \{v_1, v_2, \ldots, v_n\}$, where $\mathbf{s}_f(v, v_i) > \mathbf{s}_f(v, v_{i+1})$. Naturally, the k-nearest neighbors of each node v can be defined as follows:

$$\mathrm{Ne}(v, k) = \{v_1, \ldots, v_k\}, \quad |\mathrm{Ne}(v, k)| = k. \tag{4}$$

Therefore, the similarity of candidate neighbor distributions with the other node can be calculated by the Jaccard metric

$$\begin{aligned}\mathbf{s}_{ne}(v_i, v_j) &= \mathrm{Jaccard}(\mathrm{Ne}(v_i, k), \mathrm{Ne}(v_j, k))\\ &= \frac{|\mathrm{Ne}(v_i, k) \cap \mathrm{Ne}(v_j, k)|}{|\mathrm{Ne}(v_i, k) \cup \mathrm{Ne}(v_j, k)|}\end{aligned} \tag{5}$$

where $\mathrm{Jaccard}(\cdot, \cdot)$ measures the similarity of two sets, which is calculated by dividing the intersection size of the two sets by their union size, where the sets are the connected neighbors of each node and $|\cdot|$ denotes the number of nodes in the set. Finally, the neighbor distribution enhanced similarity is defined as follows:

$$\mathbf{s}(v_i, v_j) = (1 - \eta)\mathbf{s}_f(v_i, v_j) + \eta \mathbf{s}_{ne}(v_i, v_j) \tag{6}$$

where $\eta \in [0, 1]$ is a hyperparameter. It should be noted that the choice of k is insensitive, as be verified in Section V-G.

In summary, Jaccard similarity and cosine similarity can be seen as the *contextual* information and the *single-point* information, respectively. Meanwhile, the designed Jaccard similarity can be regarded as a first-order Weisfeiler-Lehman subtree kernel [51] from the perspective of WL-test in the feature space. WL-test is a classic discriminant graph isomorphism discrimination method, which is proven to be the performance upper bound of GNN. The core step of this method includes calculating the kernel similarity between nodes. The specific explanation is presented as follows.

For two nodes v and u, the calculation of WL-subtree kernel [51] can be defined as follows:

$$k^1_{\mathrm{WLsubtree}}(v, u) = g\left(\varphi^1_{\mathrm{WLsubtree}}(v), \varphi^1_{\mathrm{WLsubtree}}(u)\right) \tag{7}$$

where $g(\cdot, \cdot)$ is the similarity calculation function. $\varphi^1_{\mathrm{WLsubtree}}(v)$ is the information expression of node v based on one-step neighbor aggregation. Specifically, for the node v on the graph, it is defined as follows:

$$\varphi^1_{\mathrm{WLsubtree}}(v) = \mathrm{Hash}\{h_v, f(h_k | k \in N(v))\} \tag{8}$$

where $\mathrm{Hash}(\cdot)$ corresponds to the labeling process of neighbor nodes, which can be defined as taking the index of each node in the dataset as its label, h_v and h_u are the index of node v and node u, $f(\cdot)$ is the information aggregation function, and $N(v)$ is the set of neighbors of node v. In this article

$$\begin{aligned}k^1_{\mathrm{WLsubtree}}(v, u) &= g\left(\varphi^1_{\mathrm{WLsubtree}}(v), \varphi^1_{\mathrm{WLsubtree}}(u)\right)\\ &= \mathrm{Jaccard}(\mathrm{Hash}\{h_v, f(h_k | k \in N(v))\}\\ &\quad \times \mathrm{Hash}\{h_u, f(h_k | k \in N(u))\}.\end{aligned} \tag{9}$$

The index set of neighbor nodes by the $\mathrm{Hash}(\cdot)$ operation can be defined as follows:

$$\mathrm{Hash}\{h_v, f(h_k | k \in N(v))\} = \mathrm{Ne}(v, k). \tag{10}$$

Thus,

$$\begin{aligned}k^1_{\mathrm{WLsubtree}}(v, u) &= \mathrm{Jaccard}(\mathrm{Ne}(v, k), \mathrm{Ne}(u, k))\\ &= \frac{|\mathrm{Ne}(v, k) \cap \mathrm{Ne}(u, k)|}{|\mathrm{Ne}(v, k) \cup \mathrm{Ne}(u, k)|}.\end{aligned} \tag{11}$$

Moreover, the benefits of this contextual information can be further explained from the perspectives of social networks and kNN classifier.

1) For Jaccard similarity, our assumption is that if the neighbor overlap between an anchor node and a candidate node is high, the probability that they are true pairs is also high. This assumption is supported by a fundamental principle in social networks: *two persons who have many common friends would be much likely to be good friends.* This concept has also been applied and validated in the area of reidentification [53], [54].
2) The k-nearest neighbors algorithm is a widely used supervised learning classifier. Its core idea is to first find the k-nearest candidate samples of an anchor sample in the feature space and then perform a voting process, i.e., the label of the anchor sample is determined by the most frequent class among these candidates. In our approach, cosine similarity corresponds to the process of finding k-nearest neighbors in the feature space, while Jaccard similarity aligns with the voting process. Since we lack label information, the voting process is instead replaced by calculating the overlap between candidate samples. A higher overlap indicates a greater probability that they belong to the same class.

B. Adaptive Threshold Discovery Inspired by Fisher's Criterion

After obtaining the similarity between nodes, existing methods [10], [35] generally adopt a *one-size-fits-all* solution, which sets a unified threshold to choose neighbors for all nodes. However, it is common sense that each node exhibits a distinct neighbor pattern. Meanwhile, such solutions usually require a lot of hyperparameter tuning effort to choose the appropriate threshold for neighbor selection. Even if other methods [11], [12] further parameterize the structure for learning, additional model parameters will be introduced. To this end, in this section, we introduce an adaptive nonparametric threshold discovery method to select neighbors. In detail, we regard the selection of neighbors as a binary classification problem. The classification criterion is inspired by Fisher's criterion, which is to minimize the similarity variance within a class and maximize the similarity variance between classes. Based on this, we can identify a threshold in the distribution of similarity scores, thereby dividing the data into two classes: one class is the neighbor nodes with high similarity, and the other is unrelated ones with low similarity. Specifically, for each node v, the objective function can be defined as follows:

$$\min_{C_1,C_2} \frac{\sum_{v_i\in C_1}(x_i-\mu_1)^2}{|C_1|}+\frac{\sum_{v_i\in C_2}(x_i-\mu_2)^2}{|C_2|} \tag{12}$$

$$\text{s.t. } C_1\cap C_2=\emptyset,\quad C_1\cup C_2=v_1,v_2,\dots,v_N \tag{13}$$

where x_i is our derived neighbor distribution enhanced similarity for node v_i, C_1 and C_2 are the two classes, μ_i is the mean of class C_i, and $|C_i|$ is the cardinality of C_i.

The achievement of the objective function (12) is as follows. According to Algorithm 1, we first need to sort the similarity values. Then, we use a recursive approach to compute the mean (ascending order: $\bar{X}^+$ and descending order: $\bar{X}^-$) and

Algorithm 1 Adaptive Threshold Discovery Method

```
Input: Sequence of **neighbor distribution enhanced similarity** values $x_i$ for $i \in 1,\dots,N$
Output: Class-break index $b$
 1: Sort $\mathcal{X}=\{x_i, 1\le i\le N\}$ to a strictly increasing sequence.
 2: $\sigma_1^{2+}\leftarrow 0$; $\bar{X}_1^+\leftarrow x_1$; $\sigma_N^{2-}-\leftarrow 0$; $\bar{X}_1^-\leftarrow x_n$; $b\leftarrow 0$; $s\leftarrow\infty$
 3: for $n=2$ to $N$ do
 4:     $\bar{X}_n^+=\frac{1}{n}x_n+\frac{n-1}{n}\bar{X}_{n-1}^+$
 5:     $\sigma_n^{2+}=\frac{n-2}{n-1}\sigma_{n-1}^{2+}+\frac{1}{n}(\bar{X}_n^+-\bar{X}_{n-1}^+)^2$
 6: end for
 7: for $n=N-1$ to 1 do
 8:     $\bar{X}_n^-=\frac{1}{n}x_n+\frac{n-1}{n}\bar{X}_{n-1}^-$
 9:     $\sigma_n^{2-}=\frac{n-2}{n-1}\sigma_{n-1}^{2-}+\frac{1}{n}(\bar{X}_n^--\bar{X}_{n-1}^-)^2$
10: end for
11: for $n=1$ to $N-1$ do
12:     if $\sigma_{n+1}^{2-}+\sigma_n^{2+}<s$ then
13:         $s=\sigma_{n+1}^{2-}+\sigma_n^{2+}$; $b=n$
14:     end if
15: end for
```

variance (ascending order: σ^{2+} and descending order: σ^{2-}) of the sequence [lines (3–10)]. Here, we provide the derivation of the mean and variance below for lines (4 and 5) and (8 and 9). Note that for the sake of brevity, we do not introduce class symbols $(+,-)$

$$\bar{X}_n=\frac{1}{n}x_n+\frac{n-1}{n}\bar{X}_{n-1} \tag{14}$$

where $\bar{X}_n$ is the averaged value of the first n values in the sequence. Then, we can have

$$\begin{aligned}
&(n-1)\sigma_n^2\\
&\quad=\sum_{i=1}^{n}\Big[(x_i-\bar{X}_{n-1})^2+(\bar{X}_{n-1}-\bar{X}_n)^2\\
&\qquad\qquad+2(x_i-\bar{X}_{n-1})(\bar{X}_{n-1}-\bar{X}_n)\Big]\\
&\quad=\sum_{i=1}^{n-1}(x_i-\bar{X}_{n-1})^2+(x_n-\bar{X}_{n-1})^2+n(\bar{X}_{n-1}-\bar{X}_n)^2\\
&\qquad+2(\bar{X}_{n-1}-\bar{X}_n)\sum_{i=1}^{n}(x_i-\bar{X}_{n-1})\\
&\quad=(n-2)\sigma_{n-1}^2+(x_n-\bar{X}_{n-1})^2+n(\bar{X}_{n-1}-\bar{X}_n)^2\\
&\qquad+2(\bar{X}_{n-1}-\bar{X}_n)(x_n-\bar{X}_{n-1})\\
&\quad=(n-2)\sigma_{n-1}^2+(n^2-n)(\bar{X}_{n-1}-\bar{X}_n)^2\\
&\quad=(n-2)\sigma_{n-1}^2+\frac{n-1}{n}(x_n-\bar{X}_{n-1})^2.
\end{aligned} \tag{15}$$

Finally, it is natural for us to have a record for the sum of variances for every possible split. Following the lines (11–15) in Algorithm 1, we can make sure (12) and (13) are held, from which the threshold b is found. The candidate nodes whose index are greater than b are considered as neighbor nodes, and their set is defined as $C_f(v)$. Therefore, we can derive the adjacency matrix $\mathbf{A}_f$

$$\mathbf{A}_f^{ij}=\begin{cases}1, & v_j\in C_f(v_i)\\ 0, & \text{Otherwise.}\end{cases} \tag{16}$$

In addition, for the conditions of use of the algorithm, we do not make additional assumptions about similarity distribution. If the sample size is N, it must be greater than 1. Meanwhile, the sum of elements in the final neighbor and nonneighbor sets is N. Since we focus on similarity ranking, the input dimension of the sample feature is 1-D, and the feature values cannot be identical.

C. Downstream Applications of Structures

In this section, we integrate the adjacency matrix $\mathbf{A}_f$ into graph learning tasks. It can be found that GNS is model-agnostic and has no preset distribution assumptions. Therefore, we apply it to downstream tasks including OOD detection, and general tasks (classification and regression) due to its flexibility.

1) Applications on the OOD Detection: Here, we first choose the GNNSafe [20] as the basic model with GNS to perform node-level OOD detection. Meanwhile, we also combined GNS with the recent SOTA method NodeSafe [49], whose basic framework is consistent with GNNSafe, but has a unique design in the loss function. Specifically, for a given graph $(\mathbf{A}, \mathbf{X})$, we can first define its embedding matrix at the lth layer graph convolutional network

$$\mathbf{Z}^{(l)} = \sigma\left(\mathbf{D}^{-1/2}\tilde{\mathbf{A}}\mathbf{D}^{-1/2}\mathbf{Z}^{(l-1)}\mathbf{W}^{(l)}\right)$$
$$\mathbf{Z}^{(l-1)} = \left[z_i^{(l-1)}\right]_{i\in\mathcal{I}} \quad (17)$$

where $\mathbf{Z}^0 = \mathbf{X}$, $\tilde{\mathbf{A}}$ is an adjacency matrix with self-loop on $\mathbf{A}$, $\mathbf{D}$ is its degree matrix, $\mathbf{W}^{(l)}$ is the weight matrix at the lth layer, σ is a nonlinear activation, and $\mathcal{I}$ is the node set. After the Lth layer convolutional operation, for each node, the GCN outputs a C-dimensional vector $\mathbf{z}_i^{(L)}$ as logits, denoted by: $h_\theta(\mathbf{x}, \mathbf{A}) = \mathbf{z}_i^{(L)}$, where θ denotes the trainable parameters of the GCN h. The logits are used to derive a categorical distribution with the softmax function for classification

$$p(y \mid \mathbf{x}, \mathbf{A}) = \frac{e^{h_\theta(\mathbf{x},\mathbf{A})_{[y]}}}{\sum_{c=1}^{C} e^{h_\theta(\mathbf{x},\mathbf{A})_{[c]}}} \quad (18)$$

where C denotes the class number and $\mathbf{x}$ is the feature vector of node v. At this point, without changing the parameterization of the graph convolutional network, we can express the energy score [54] in terms of the denominator in (18)

$$E(\mathbf{x}, \mathbf{A}; h_\theta) = -\log\sum_{c=1}^{C} e^{h_\theta(\mathbf{x},\mathbf{A})_{[c]}}. \quad (19)$$

Next, the propagation of energy score on the structure can be defined as follows:

$$\mathbf{E}^{(K)} = \alpha\mathbf{E}^{(K-1)} + (1-\alpha)\mathbf{D}^{-1}\mathbf{A}\mathbf{E}^{(K-1)}$$
$$\mathbf{E}^{(K)} = \left[E_i^{(K)}\right]_{i\in\mathcal{I}} \quad (20)$$

where α is a parameter controlling the concentration of the energy of the node itself and other neighbors, K is the number of propagation times, and E_i^K is the energy score of node v_i at Kth propagation. Here, we introduce $\mathbf{A}_f$ derived by GNS into the propagation process

$$\mathbf{E}^{(K)} = (\mathbf{E}(s))^{(K)} + \lambda(\mathbf{E}(f))^{(K)} \quad (21)$$

where $(\mathbf{E}(s))^{(K)}$ and $(\mathbf{E}(f))^{(K)}$ take the original observation adjacency matrix $\mathbf{A}_s$ and the $\mathbf{A}_f$ as input, respectively, and λ is a hyperparameter. Finally, the resulting energy score serves as the predictive value for OOD.

In fact, the aggregation of energy scores between nodes on the structure can push the energy toward the majority of neighbored nodes. For example, if the average energy score of node v_i's one-hop neighbors is lower than its own energy score, its energy score will decrease after propagation. Meanwhile, this is also equivalent to making the score of each node closer to their class center. Therefore, the energy gap between the nodes in the distribution and the OOD will be further amplified. This process can be proved as follows.

If $\left(\sum_j a_{ij}(E_j(f))^{(k-1)}/\sum_j a_{ij}\right) < (E_i(f))^{(k-1)}$ (the former is the averaged energy scores of one-hop neighbored before the kth propagation), we can define the update in (20) as the scalar form

$$\begin{aligned}(E_i(f))^{(k)} &= \alpha(E_i(f))^{(k-1)} + (1-\alpha)\frac{\sum_j a_{ij}(E_j(f))^{(k-1)}}{\sum_j a_{ij}}\\ &< \alpha(E_i(f))^{(k-1)} + (1-\alpha)(E_i(f))^{(k-1)}\\ &= (E_i(f))^{(k-1)}. \end{aligned} \quad (22)$$

2) Applications on the General Condition: Under the i.i.d. assumption, we deploy $\mathbf{A}_f$ into semi-supervised classification and regression prediction to enhance their performance. Specifically, we choose a two-layer GCN as the basic model to implement, the operation can be defined as follows:

$$\tilde{\mathbf{Y}} = f(\mathbf{X}, \mathbf{A}_f) = \hat{\mathbf{A}}_f\left(\text{ReLu}\left(\hat{\mathbf{A}}_f\mathbf{X}\mathbf{W}^{(0)}\right)\right)\mathbf{W}^{(1)} \quad (23)$$

where $\hat{\mathbf{A}}_f = \tilde{\mathbf{D}}_f^{-1/2}\tilde{\mathbf{A}}_f\tilde{\mathbf{D}}_f^{-1/2}, \tilde{\mathbf{A}}_f = \mathbf{A}_f + \mathbf{I}$, $\mathbf{D}_f$ is the degree matrix of $\tilde{\mathbf{A}}_f$, $\mathbf{W}^{(0)}$ is an input-to-hidden weight matrix, and $\mathbf{W}^{(1)}$ is a hidden-to-output weight matrix. The predicted values $\tilde{\mathbf{Y}}$ are used to calculate the loss function $\mathcal{L}$, where $\mathcal{L}$ can be cross entropy for classification or mean square error for regression. Similar to the energy score, for the embedding of any node pair in graph $\mathbf{A}_f$, their distance will decrease with the process of message passing, thus making the intraclass distance more compact. The specific proof based on the spectral graph theory is shown as follows. For the process of one-step message passing

$$\boldsymbol{Z}^1 = \tilde{\boldsymbol{D}}_f^{-\frac{1}{2}}\tilde{\boldsymbol{A}}_f\tilde{\boldsymbol{D}}_f^{-\frac{1}{2}}\boldsymbol{Z}$$

where $\boldsymbol{Z}^1$ is the embedding of all nodes after one step and $\boldsymbol{Z}$ is the initial feature matrix. This operation can be rewritten as follows:

$$\boldsymbol{Z}^1 = (\boldsymbol{I} - \boldsymbol{L})\boldsymbol{Z}$$

where $\boldsymbol{L} = \tilde{\boldsymbol{D}}_f^{-(1/2)}\tilde{\boldsymbol{L}}\tilde{\boldsymbol{D}}_f^{-(1/2)}$, $\tilde{\boldsymbol{L}} = \tilde{\boldsymbol{D}}_f - \tilde{\boldsymbol{A}}_f$. Let $(\lambda_1, \ldots, \lambda_N)$ and $(\boldsymbol{e}_1, \ldots, \boldsymbol{e}_N)$ be the eigenvalue and eigenvector of matrix $(\boldsymbol{I} - \boldsymbol{L})$, respectively. Due to $\boldsymbol{L}$ is the symmetric Laplacian matrix, we can obtain

$$-1 < \lambda_1 < \lambda_2 < \cdots < \lambda_N = 1, \quad \boldsymbol{e}_N = \tilde{\boldsymbol{D}}_f^{-\frac{1}{2}}[1, 1, \ldots, 1]^{\mathrm{T}}.$$

After t times of propagation, we can rewrite $\boldsymbol{Z}_i^{(t)} - \boldsymbol{Z}_j^{(t)}$ as follows:

$$\begin{aligned}\boldsymbol{Z}_i^{(t)} - \boldsymbol{Z}_j^{(t)} &= [(\boldsymbol{I}-\boldsymbol{L})^t\boldsymbol{Z}]_i - [(\boldsymbol{I}-\boldsymbol{L})^t\boldsymbol{Z}]_j \\ &= \left[\lambda_1^t\left(\boldsymbol{e}_1^{(i)} - \boldsymbol{e}_1^{(j)}\right), \dots, \lambda_{n-1}^t\left(\boldsymbol{e}_{N-1}^{(i)} - \boldsymbol{e}_{N-1}^{(j)}\right), 0\right]\hat{\boldsymbol{Z}}\end{aligned}$$

where $\boldsymbol{e}_p^{(i)}$ is the ith element of eigenvector $\boldsymbol{e}_p$, $p \in [1, N-1]$. $\hat{\boldsymbol{Z}}$ is the coordinate matrix of $\boldsymbol{Z}$. Now

$$\left\|\boldsymbol{Z}_i^{(t)} - \boldsymbol{Z}_j^{(t)}\right\|_2 = \sqrt{\sum_{q=1}^{N}\left[\sum_{p=1}^{N-1}\lambda_k^t\left(\boldsymbol{e}_k^{(i)} - \boldsymbol{e}_k^{(j)}\right)\hat{\boldsymbol{Z}}_{pq}\right]^2}.$$

Because $-1 < \lambda_1 < \lambda_2 < \cdots < \lambda_{N-1} < 1$, with the increase of t, $\|\boldsymbol{Z}_i^{(t)} - \boldsymbol{Z}_j^{(t)}\|_2 \leq \|\boldsymbol{Z}_i^{(0)} - \boldsymbol{Z}_j^{(0)}\|_2$ is always been established.

3) Complexity Analysis: Regarding the complexity of the model, assuming the dataset size is N, the complexity for similarity measurement and neighbor selection processes is $O(N^2)$ and $O(N^2 \log N)$. Therefore, the complexity of the overall algorithm is $O(N^2 \log N)$. Nevertheless, it is important to note that this computation is performed *only once* and is separate from the model training process. In this article, *our goal* is to abandon the heavy network of previous GSL and instead design a nonparametric, easy-to-use, and generalizable model, which is a different technological route. Empirically, GNS ensures performance across six tasks on 13 datasets without introducing any optimized parameters to the basic model (GCNs), showcasing its broad applicability and flexibility. One direction worth exploring is to focus on exploring nonparametric models with better computational efficiency in the future and combine them with our GNS.

V. Experiments

A. Experimental Settings

1) Datasets: For OOD detection, we choose two real-world datasets used in recent work [20]: Cora and Twitch. The former selects OOD data based on features, structures, and labels, and the latter is based on different subgraphs. We divide the ID data into training/validation/testing sets according to a 1:1:8 ratio or provided splits. The main results are shown in Tables I and II. For node classification, we test the performance on three widely used networks: Cora, Citeseer, and Pubmed [3]. For image classification, we use all 13000 images from STL-10, each of which belongs to one of ten classes. Meanwhile, we select 1500 images from each of the ten classes of CIFAR-10. For the text classification, we choose the 20News for evaluation, which is a dataset consisting of 9607 instances [11], [12]. For spatiotemporal prediction, we consider three datasets: WikiMath, Chickenpox, Covid [55], the description of these datasets is presented in Table III. Meanwhile, for the datasets under the i.i.d. setting, their data splitting is based on the previous work [12].

2) Baselines and Evaluation: We compare the following models regarding OOD detection. First, we choose the methods that come from vision: MSP [56], ODIN [39], Mahalanobis [57], OE [58], and Energy [54]. We replace their backbone with GCN for processing graphs. The second type is designed for processing graph data, and we compare it with four SOTA methods: GKDE [47], GPN [48], the energy propagation-based method: GNNSafe [20] and NodeSafe [49] (the experimental results on the Twitch dataset combined with GNS are shown in Table IV), and the advanced graph structure learning: SLAPS method [23]. For other experiments, our main comparison methods are as follows: MLP, ManiReg [59], GCN-kNN [3], GAT-kNN, DenseGAT [4], ARGA [60], SGC [61], GLCN [62], Difformer [12], Nodeformer [11], and GSSC [19], which including the SOTA method. As for the evaluation of models, we first follow three widely used metrics in OOD detection, namely, AUROC↑, AUPR↑, and FPR95 ↓. For other experiments, ACC is used for classification, and mse is used for regression tasks. The results of spatiotemporal prediction, image and text classification are presented in Tables V and VI.

3) Hyperparameter Specifications: Tables VII and VIII summarize the important hyperparameter configurations for all datasets, including the candidate neighbor nodes k in Section IV-A, the layers and hidden size of the GCN, the tradeoff parameter λ and α in Section IV-C for OOD scenario, η in Section IV-A for neighbor distribution enhanced similarity. The search range for these hyperparameters is determined based on the previous works [12], [20]. Meanwhile, since the goal is to learn low-dimensional embeddings, the hidden layer size is generally not larger than the initial feature dimension, so it is set to 4 for spatiotemporal datasets. In addition, for experiments with different hyperparameters, we will choose the best model with the validation set for testing.

B. Performance on OOD Detection

In Tables I and II, we report the comparison results of GNS with the baseline models. Overall, from the table, we can see that GNS is better than its competitors in most cases. Specifically, compared with GNNSafe, the highest improvements in the indicators of our model under the three OOD settings of Cora and Twitch are: AUROC increased by 3.66% and 11.79%, AUPR increased by 4.68% and 10.92%, and FPR95 decreased 24.3% and 29.9%. Meanwhile, it outperforms OE on most datasets, even though this method uses additional OOD data in training. Compared with GNS, the unique aspect of SLAPS lies in its design of an additional self-supervised learning task: noisy reconstruction learning. This strategy enhances the method's generalizability and shows good compatibility with the Twitch dataset, resulting in comparable performance. Furthermore, we visualize the energy scores on two datasets, as shown in Figs. 4 and 5, with GNS (right column) and GNNSafe (middle column). It can be intuitively found that our model can indeed further widen the difference in scores between the in-distribution and the OOD data.

C. Performance on Spatiotemporal Prediction

Here, we consider evaluating the performance of GNS on spatiotemporal sequence prediction. In detail, we insert the structures exported by GNS into the GCN for implementation, where the comparison methods take the original structure or kNN graph as input. From the experimental results reported in Table V, we make observations which are described as follows. First of all, compared with the SOTA parametric method Difformer, the performance of GNS is competitive, with the best improvement of 4.7% on wikiMath. It should be

TABLE I
OOD DETECTION PERFORMANCE MEASURED BY AUROC(↑)/AUPR(↑)/FPR95(↓) ON CORA WITH THREE OOD TYPES

Model	Cora-Structure			Cora-Feature			Cora-Label		
	AUROC	**AUPR**	**FPR95**	**AUROC**	**AUPR**	**FPR95**	**AUROC**	**AUPR**	**FPR95**
MSP	70.90	45.73	87.30	85.39	73.70	64.88	91.36	78.03	34.99
ODIN	49.92	27.01	100.0	49.88	26.96	100.0	49.80	24.27	100.0
Mahalanobis	46.68	29.03	98.19	49.93	31.95	99.93	67.62	42.31	90.77
Energy	71.73	46.08	88.74	86.15	74.42	65.81	91.40	78.14	41.08
GKDE	68.61	44.26	84.34	82.79	66.52	68.24	57.23	27.50	88.95
GPN	77.47	53.26	76.22	85.88	73.79	56.17	90.34	77.40	37.42
OE	67.98	46.93	95.31	81.83	70.84	83.79	89.47	77.01	46.55
SLAPS	85.18	72.64	73.08	93.69	88.61	41.84	92.77	84.08	35.29
GNNSafe	87.17	77.15	74.19	93.44	88.19	38.92	92.70	82.07	31.44
+GNS	88.42	79.67	67.61	97.08	92.87	14.62	94.02	84.98	23.12
NodeSafe	91.50	80.30	37.63	95.71	91.02	21.71	92.29	79.72	31.03
+GNS	**93.31**	**83.85**	**33.94**	**96.03**	**91.48**	**17.54**	**92.79**	**81.28**	**29.21**

TABLE II
OOD DETECTION PERFORMANCE MEASURED BY AUROC(↑)/AUPR(↑)/FPR95(↓) ON OOD SUBGRAPHS OF TWITCH

Model	OOD Expo	Twitch-ES			Twitch-FR			Twitch-RU		
		AUROC	**AUPR**	**FPR95**	**AUROC**	**AUPR**	**FPR95**	**AUROC**	**AUPR**	**FPR95**
MSP	No	37.72	53.08	98.09	21.82	38.27	99.25	41.23	56.06	95.01
Mahalanobis	No	45.66	58.82	95.48	40.40	46.69	95.54	55.68	66.42	90.13
Energy	No	38.80	54.26	95.70	57.21	61.48	91.57	57.72	66.68	87.57
GKDE	No	48.70	61.05	95.37	49.19	52.94	95.04	46.48	62.11	95.62
GPN	No	53.00	64.24	95.05	51.25	55.37	93.92	50.89	65.14	99.93
OE	Yes	**55.97**	**69.49**	94.94	45.66	54.03	95.48	55.72	70.18	95.07
SLAPS	No	49.21	58.62	95.31	65.45	67.52	**86.60**	87.94	89.95	50.90
GNNSafe	No	49.07	57.62	93.98	63.49	66.25	90.80	87.90	89.05	43.95
+GNS	No	49.13	56.99	**93.78**	**75.28**	**77.17**	87.39	**94.89**	**94.41**	**14.03**

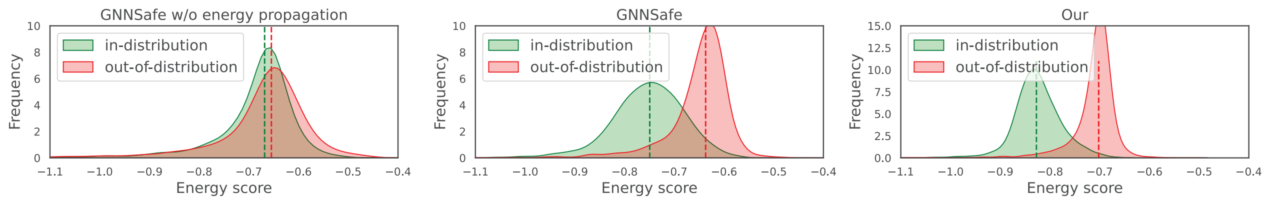

Fig. 4. Score distributions on Twitch.

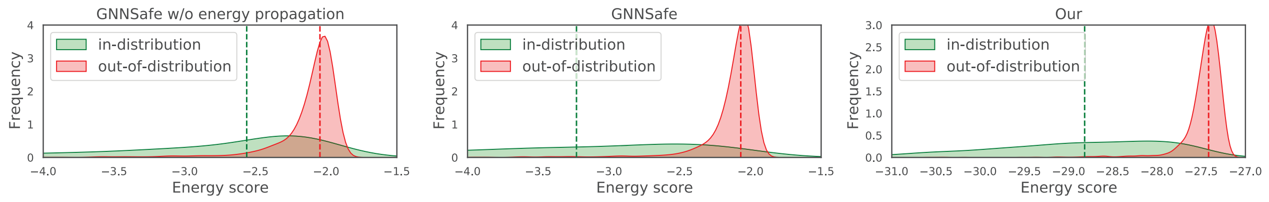

Fig. 5. Score distributions on Cora-feature.

noted that our method is nonparametric, thereby avoiding the introduction of additional optimization parameters for training. Second, we can see that our method is better than the base model GCN and GAT, which reflects the necessity of the introduction of structures we learned. Noise edges will inevitably be introduced into the original structure due to their generation from complex networks. Third, the performance of GCN-kNN and GAT-kNN is poor, which shows the vulnerability of simply using the nearest neighbor for all nodes.

D. *Performance on Node Classification and Clustering*

To verify the effectiveness of the GNS on node classification, we select three benchmark datasets: Cora, Citeseer, and Pubmed for experiments. Similarly, we introduced GNS into the basic models: GCN and GAT. The specific experimental results are shown in Fig. 6. First of all, from the left side of the figure, we can find that the introduced structure by the model improves the performance of GCN and GAT.

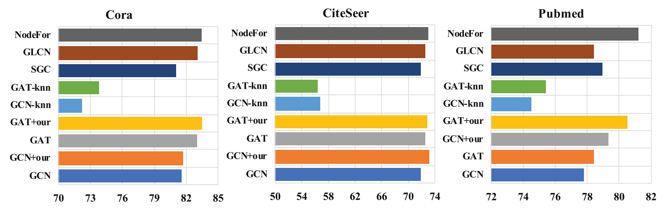

Fig. 6. Testing accuracy on node classification.

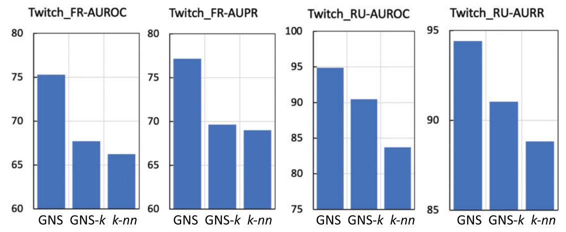

Fig. 7. Ablation study on Twitch FR and Twitch RU under the OOD setting, with AUPR and AUROC as measurement indicators.

TABLE III

SUMMARY STATISTICS OF THE SPATIOTEMPORAL DATASETS WITH INFORMATION ABOUT WHETHER THE GRAPH STRUCTURE IS DYNAMIC OR STATIC, THE CORRESPONDING DIMENSION (D), THE NUMBER OF SNAPSHOTS (T), AND THE NUMBER OF NODES ($|V|$)

Dataset	Graph structure	D	Frequency	T	$\|V\|$
Chickenpox	Static	4	Weekly	522	20
Covid	Dynamic	8	Daily	61	129
WikiMath	Static	14	Daily	731	1,068

Second, compared with the method of using kNN for structure learning, our method shows better performance in terms of performance, which reflects the importance of the selection of similarity and neighbor nodes. In addition, compared with two powerful parametric GSL methods: GLCN and Nodeformer, our method also has good performance without introducing additional optimized parameters. Moreover, we integrate GNS with SUBLIME [17] in the downstream task of node clustering, where we observe performance improvements across various clustering metrics in Table IX.

E. Performance on Image and Text Classification

Here, we conduct experiments on CIFAR-10, STL-10, and 20News datasets for standard classification tasks with a limited label rate. Following the work [12], since these datasets contain no graph structures, we chose kNN for composition as the input of graph models: GCN and GAT. Table VI reports GNS's accuracy versus competitors. It can be found that our method achieves consistently good results. Specifically, compared with the GCN-kNN, and GAT-kNN, the structure derived by GNS through carefully designed metrics and neighbor discovery can better capture the correlation between instances. Meanwhile, our method also achieves desirable performance compared with the parameterized methods GLCN and Differmer.

F. Ablation Study

To verify the effectiveness of each component in the model, we design an ablation experiment, where GNS-k removes

TABLE IV

OOD DETECTION PERFORMANCE WITH NODESAFE ON TWITCH MEASURED BY AUROC (↑)/AUPR (↑)/FPR95 (↓)

Model	Twicth-ES			Twitch-RU		
	AUROC	AUPR	FPR95	AUROC	AUPR	FPR95
NodeSafe	96.40	98.02	21.28	77.81	86.42	88.03
+GNS	**98.81**	**99.35**	**3.23**	**85.30**	**90.97**	**75.42**

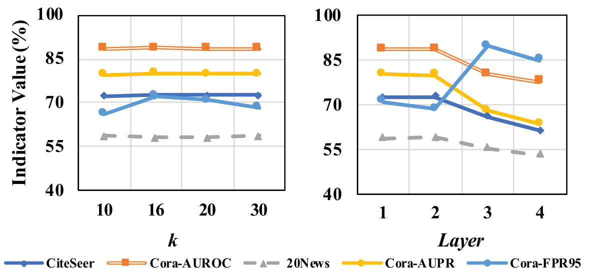

Fig. 8. Sensitivity of hyperparameters. For Citeseer and 20news, the evaluation metric is ACC (↑). For Cora, the metrics include ACROC (↑), AUPR (↑), and FPR95 (↓).

the neighbor distribution similarity in GNS, and kNN further removes the adaptive neighbor discovery mechanism. Meanwhile, we also conducted experiments on replacing the kNN graph in GNS with a random graph. As shown in Table X, it can be found that the kNN graph is relatively reliable, and too much noise on the random graph will affect the process of structural learning. Then, as shown in Fig. 7 and Table XI, both neighbor distribution-enhanced similarity and adaptive threshold selection play crucial roles in structure learning. In both OOD and general settings, the variant GNS-k demonstrates suboptimal results, underscoring the effectiveness of enhanced similarity design through neighbor distribution. In GNS-k, only single-point similarity (i.e., cosine) is used, disregarding structural information in the feature space, which leads to unreliable similarity assessments. For the kNN variant without adaptive threshold selection, performance is the lowest. This is because applying a uniform threshold to each sample introduces more noisy edges, impairing GNNs' ability to aggregate node representations effectively. These findings further highlight the importance of our adaptive selection mechanism in accurately modeling diverse neighbor patterns for each sample. In addition, the visualizations in Figs. 4 and 5 also verify the effectiveness of the method in capturing the correlation between nodes.

G. Hyperparameter and Robustness Study

In this section, we investigate the sensitivity of hyperparameters, which include the number of candidate neighbor nodes k in Section IV-A and the number of layers in GCN. We conduct experiments on three tasks, including OOD detection, node classification, and text classification. Specifically, we have set four options, for k they are 10, 16, 20, and 30, for the number of layers, they are 1–4. The results are shown in Fig. 8. It can be found that the performance of the method is not sensitive to the k. Specifically, good results can be achieved

TABLE V
MEAN AND STANDARD DEVIATION OF MSE ON SPATIOTEMPORAL PREDICTION DATASETS

Dataset	MLP	Difformer	GCN	GAT	Dense GAT	GAT-kNN	GCN-kNN	GCN+GNS
Chickenpox	0.924 (±0.001)	0.914 (±0.008)	0.923 (±0.001)	0.924 (±0.002)	0.935 (±0.005)	0.926 (±0.004)	0.936 (±0.004)	**0.912 (±0.003)**
Covid	0.956 (±0.198)	0.779 (±0.037)	1.080 (±0.162)	1.052 (±0.336)	1.524 (±0.319)	0.861 (±0.123)	1.475 (±0.560)	**0.755 (±0.031)**
WikiMath	1.073 (±0.042)	0.731 (±0.007)	1.292 (±0.125)	1.339 (±0.073)	0.826 (±0.070)	0.882 (±0.015)	1.023 (±0.058)	**0.697 (±0.014)**

TABLE VI
TESTING ACCURACY FOR IMAGE (CIFAR AND STL) AND TEXT (20NEWS) CLASSIFICATION

Dataset		MLP	LP	ManiReg	Difformer	GCN-kNN	GAT-kNN	DenseGAT	GLCN	GCN+GNS
CIFAR	500 labels	73.2 ± 0.4	70.6	72.6 ± 1.2	74.0 ± 0.6	72.9 ± 0.4	72.4 ± 0.5	OOM	72.8 ± 0.5	**74.9 ± 0.6**
	1000 labels	75.4 ± 0.6	71.9	74.3 ± 0.4	75.9 ± 0.3	74.7 ± 0.5	74.1 ± 0.5	OOM	74.7 ± 0.3	**76.0 ± 0.3**
STL	500 labels	73.0 ± 0.8	71.8	72.5 ± 0.5	72.9 ± 0.7	72.1 ± 0.8	72.0 ± 0.8	OOM	72.4 ± 1.3	**73.2 ± 0.5**
	1000 labels	75.0 ± 0.8	72.7	74.2 ± 0.5	75.3 ± 0.6	73.7 ± 0.4	73.9 ± 0.6	OOM	74.3 ± 0.7	**75.5 ± 0.4**
20News	1000 labels	54.1 ± 0.9	55.9	56.3 ± 1.2	57.9 ± 0.7	56.1 ± 0.6	55.2 ± 0.8	54.6 ± 0.2	56.2 ± 0.8	**58.8 ± 0.3**
	2000 labels	57.8 ± 0.9	57.6	60.0 ± 0.8	61.3 ± 1.0	60.6 ± 1.3	59.1 ± 2.2	59.3 ± 1.4	60.2 ± 0.7	**62.0 ± 0.4**

TABLE VII
HYPERPARAMETER SPECIFICATIONS FOR GENERAL TASKS

Dataset	k	η	GCN_layer	$Hidden\_Size$
Cora	12	1	2	32
Citeseer	12	1	2	32
Pubmed	12	1	2	32
STL	5	0.7	2	64
CIFAR	10	0.6	2	32
20News	10	0.6	2	32
Covid	10	0.7	2	4
WikiMath	10	0.7	2	4
Chickenpox	3	0.7	2	4

TABLE VIII
HYPERPARAMETER SPECIFICATIONS FOR OOD DETECTION

OOD-Dataset	$Layer$	$Hidden$	k	λ	α	η
Twitch	2	64	30	0.1	0.1	0.7
Cora-Label	2	64	5	3	0.1	0.7
Cora-Feature	2	64	5	2	0.1	0.7
Cora-Structure	2	64	5	2	0.1	0.7

TABLE IX
NODE CLUSTERING PERFORMANCE (FOUR METRICS IN PERCENTAGE) WITH SUBLIME

Model	Cora				Citeseer			
	C-ACC	NMI	F1	ARI	C-ACC	NMI	F1	ARI
K-means	50.0	31.7	37.6	23.9	54.4	31.2	41.3	28.5
ARGA	64.0	44.9	61.9	35.2	57.3	35.0	54.6	34.1
SUBLIME	70.1	53.9	62.7	47.8	68.4	44.6	62.9	44.3
SUBLIME+GNS	**71.6**	**54.2**	**63.8**	**51.4**	**69.2**	**44.8**	**63.7**	**45.4**

TABLE X
COMPARISON BETWEEN RANDOM GRAPH AND KNN GRAPH

Datasets	Random	kNN
Cora	80.9 ± 0.9	81.7 ± 0.5
Citeseer	71.1 ± 0.7	72.9 ± 0.5
CIFAR	74.5 ± 0.6	76.0 ± 0.3
20News	57.5 ± 0.6	58.8 ± 0.3
STL10	74.3 ± 0.7	75.5 ± 0.4
Covid	0.885 ± 0.090	0.755 ± 0.031

TABLE XI
ABLATION STUDY UNDER THE I.I.D. SETTING, WITH ACC AND MSE AS MEASUREMENT INDICATORS

Model	Metric	GNS-k	kNN	GNS
Chickenpox	MSE (↓)	0.916	0.926	0.912
Covid		0.859	0.861	0.755
CIFAR	ACC (↑)	75.2	74.7	76.0
20News		58.2	56.1	58.8
STL10		74.7	73.7	75.5

on different selected values, which verifies the robustness of our method. Regarding the number of GCN layers, we find that better results are achieved when the number of layers is set to 1 or 2 layers. The reason may be that stacking too many layers will cause the node representation to be over-smoothing. To further demonstrate the model's robustness, we evaluate the model under the label noise with different ratios: 20%, 40%, and 60%, where noise formation here follows the method GSSC [19]. From the results in Table XII, we can see that the proposed method GNS is effective in this scenario.

TABLE XII
CLASSIFICATION ACCURACY ± STD (%) WITH DIFFERENT LABEL NOISE RATIOS

Dataset	Rate	GCN	GSSC	**GSSC+GNS**
Cora	20%	77.77 ± 0.62	81.35 ± 0.43	**81.72 ± 0.52**
	40%	69.39 ± 0.70	75.26 ± 0.10	**76.21 ± 0.40**
	60%	52.17 ± 0.93	65.72 ± 0.80	**67.69 ± 0.81**
Citeseer	20%	66.91 ± 0.58	72.41 ± 0.27	**72.81 ± 0.35**
	40%	61.65 ± 0.59	71.26 ± 0.81	**72.96 ± 0.53**
	60%	54.83 ± 0.63	70.19 ± 0.75	**70.87 ± 0.56**

Benefiting from its flexibility, it can be compatible with GSSC to further improve performance.

VI. CONCLUSION

In this article, we propose and theoretically illustrate a novel structure learning framework named GNS that can be easily and effectively applied to various tasks. Specifically, we first design a similarity measure enhanced by candidate neighbor distribution. Then, inspired by Fisher's criterion, an adaptive threshold selection scheme is applied to model different neighbor patterns. Finally, extensive experiments on multiple tasks verify the effectiveness of the GNS, showcasing its broad applicability. GNS ensures performance without introducing any model parameters to the basic model (e.g., GCN).

REFERENCES

[1] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 1, pp. 4–24, Mar. 2020.

[2] J. Zhou et al., "Graph neural networks: A review of methods and applications," *AI Open*, vol. 1, pp. 57–81, Jan. 2020.

[3] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. 5th Int. Conf. Learn. Represent.*, Jan. 2016.

[4] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lió, and Y. Bengio, "Graph attention networks," in *Proc. 6th Int. Conf. Learn. Represent.*, Jan. 2017, pp. 1–12.

[5] S. Wang, J. Cao, and P. S. Yu, "Deep learning for spatio-temporal data mining: A survey," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 8, pp. 3681–3700, Aug. 2022.

[6] H. Linmei, T. Yang, C. Shi, H. Ji, and X. Li, "Heterogeneous graph attention networks for semi-supervised short text classification," in *Proc. Conf. Empirical Methods Natural Lang. Process. 9th Int. Joint Conf. Natural Lang. Process. (EMNLP-IJCNLP)*, 2019, pp. 4821–4830.

[7] L. Franceschi, M. Niepert, M. Pontil, and X. He, "Learning discrete structures for graph neural networks," in *Proc. 36th Conf. Mach. Learn.*, 2019, pp. 1972–1982.

[8] W. Jin, Y. Ma, X. Liu, X. Tang, S. Wang, and J. Tang, "Graph structure learning for robust graph neural networks," in *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2020, pp. 66–74.

[9] M. Liu et al., "Self-supervised temporal graph learning with temporal and structural intensity alignment," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 36, no. 4, pp. 6355–6367, Apr. 2025.

[10] N. Liu, X. Wang, L. Wu, Y. Chen, X. Guo, and C. Shi, "Compact graph structure learning via mutual information compression," in *Proc. ACM Web Conf.*, Apr. 2022, pp. 1601–1610.

[11] Q. Wu, W. Zhao, Z. Li, D. Wipf, and J. Yan, "NodeFormer: A scalable graph structure learning transformer for node classification," in *Proc. Adv. Neural Inf. Process. Syst.*, Jan. 2023, pp. 27387–27401.

[12] Q. Wu, C. Yang, W. Zhao, Y. He, D. Wipf, and J. Yan, "DIFFormer: Scalable (graph) transformers induced by energy constrained diffusion," in *Proc. 11th Int. Conf. Learn. Represent.*, Jan. 2023.

[13] Y. Zhu et al., "A survey on graph structure learning: Progress and opportunities," 2021, *arXiv:2103.03036*.

[14] S. Fu et al., "Multilevel contrastive graph masked autoencoders for unsupervised graph-structure learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 36, no. 2, pp. 3464–3478, Feb. 2025.

[15] R. Wang, P. Wang, D. Wu, Z. Sun, F. Nie, and X. Li, "Multi-view and multi-order structured graph learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 10, pp. 14437–14448, Oct. 2024.

[16] D. Zou et al., "SE-GSL: A general and effective graph structure learning framework through structural entropy optimization," in *Proc. ACM Web Conf.*, Apr. 2023, pp. 499–510.

[17] Y. Liu, Y. Zheng, D. Zhang, H. Chen, H. Peng, and S. Pan, "Towards unsupervised deep graph structure learning," in *Proc. ACM Web Conf.*, 2022, pp. 1392–1403.

[18] P. Elinas, E. V. Bonilla, and L. C. Tiao, "Variational inference for graph convolutional networks in the absence of graph data and adversarial settings," in *Proc. Adv. Neural Inf. Process. Syst.*, Jan. 2019, pp. 6530–6539.

[19] L. Wu, H. Lin, G. Zhao, C. Tan, and S. Z. Li, "Learning to model graph structural information on MLPs via graph structure self-contrasting," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. PP, no. 99, pp. 1–12, Sep. 2024.

[20] Q. Wu, Y.-T. Chen, C. Yang, and J. Yan, "Energy-based out-of-distribution detection for graph neural networks," in *Proc. 11th Int. Conf. Learn. Represent. (ICLR)*, Jan. 2023.

[21] W. Zhao, Q. Wu, C. Yang, and J. Yan, "GraphGLOW: Universal and generalizable structure learning for graph neural networks," in *Proc. 29th ACM SIGKDD Conf. Knowl. Discovery Data Mining*, Aug. 2023, pp. 3525–3536.

[22] Y. Sun, Y. Ming, X. Zhu, and Y. Li, "Out-of-distribution detection with deep nearest neighbors," in *Proc. Int. Conf. Mach. Learn.*, 2022, pp. 20827–20840.

[23] B. Fatemi, L. E. Asri, and S. M. Kazemi, "SLAPS: Self-supervision improves structure learning for graph neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, Jan. 2021, pp. 22667–22681.

[24] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Ann. Eugenics*, vol. 7, no. 2, pp. 179–188, Sep. 1936.

[25] D. Bo, X. Wang, C. Shi, and H. Shen, "Beyond low-frequency information in graph convolutional networks," in *Proc. AAAI Conf. Artif. Intell.*, vol. 35, no. 5, 2021, pp. 3950–3957.

[26] S. Geisler, A. Kosmala, D. Herbst, and S. Günnemann, "Spatio-spectral graph neural networks," in *Proc. NeurIPS*, May 2024, pp. 580–591.

[27] W. L. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. NIPS*, 2017, pp. 1024–1034.

[28] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," in *Proc. 2nd Int. Conf. Learn. Represent. (ICLR)*, Dec. 2013.

[29] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Proc. Annu. Conf. Neural Inf. Process. Syst.*, Jan. 2016, pp. 3837–3845.

[30] Q. Sun et al., "Graph structure learning with variational information bottleneck," in *Proc. AAAI Conf. Artif. Intell.*, 2022, vol. 36, no. 4, pp. 4165–4174.

[31] R. Yuan, Y. Tang, Y. Wu, J. Niu, and W. Zhang, "Semi-supervised graph structure learning via dual reinforcement of label and prior structure," *IEEE Trans. Cybern.*, vol. 54, no. 11, pp. 6943–6956, Nov. 2024.

[32] Z. Shen and Z. Kang, "When heterophily meets heterogeneous graphs: Latent graphs guided unsupervised representation learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 36, no. 6, pp. 10283–10296, Jun. 2025.

[33] L. Sun et al., "DeepRicci: Self-supervised graph structure-feature co-refinement for alleviating over-squashing," in *Proc. IEEE Int. Conf. Data Mining (ICDM)*, Dec. 2023, pp. 558–567.

[34] Z. Guo et al., "GraphEdit: Large language models for graph structure learning," 2024, *arXiv:2402.15183*.

[35] X. Wang, M. Zhu, D. Bo, P. Cui, C. Shi, and J. Pei, "AM-GCN: Adaptive multi-channel graph convolutional networks," in *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2020, pp. 1243–1253.

[36] Y. Rong, W. Huang, T. Xu, and J. Huang, "DropEdge: Towards deep graph convolutional networks on node classification," in *Proc. 8th Int. Conf. Learn. Represent. (ICLR)*, Addis Ababa, Ethiopia, Jan. 2019.

[37] C. Zheng et al., "Robust graph representation learning via neural sparsification," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 11458–11468.

[38] X. Li, S. Gui, Y. Luo, and S. Ji, "Graph structure extrapolation for out-of-distribution generalization," in *Proc. 41st Int. Conf. Mach. Learn.*, vol. 235, Jul. 2024, pp. 27846–27874.

[39] S. Liang, Y. Li, and R. Srikant, "Enhancing the reliability of out-of-distribution image detection in neural networks," in *Proc. 6th Int. Conf. Learn. Represent. (ICLR)*, Jan. 2017.
[40] W. Zhou, F. Liu, and M. Chen, "Contrastive out-of-distribution detection for pretrained transformers," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2021, pp. 1100–1111.
[41] Y. Liu, K. Ding, H. Liu, and S. Pan, "GOOD-D: On unsupervised graph out-of-distribution detection," in *Proc. 16th ACM Int. Conf. Web Search Data Mining*, Feb. 2023, pp. 339–347.
[42] L. Wang et al., "GOODAT: Towards test-time graph out-of-distribution detection," in *Proc. AAAI Conf. Artif. Intell.*, Jan. 2024, pp. 15537–15545.
[43] X. Shen, Y. Wang, K. Zhou, S. Pan, and X. Wang, "Optimizing OOD detection in molecular graphs: A novel approach with diffusion models," in *Proc. 30th ACM SIGKDD Conf. Knowl. Discovery Data Mining*, Aug. 2024, pp. 2640–2650.
[44] Z. Li, Q. Wu, F. Nie, and J. Yan, "GraphDE: A generative framework for debiased learning and out-of-distribution detection on graphs," in *Proc. Adv. Neural Inf. Process. Syst.*, Nov. 2022, pp. 30277–30290.
[45] G. Bazhenov, S. Ivanov, M. Panov, A. Zaytsev, and E. Burnaev, "Towards OOD detection in graph classification from uncertainty estimation perspective," 2022, *arXiv:2206.10691*.
[46] Y. Guo, C. Yang, Y. Chen, J. Liu, C. Shi, and J. Du, "A data-centric framework to endow graph neural networks with out-of-distribution detection ability," in *Proc. 29th ACM SIGKDD Conf. Knowl. Discovery Data Mining*, Aug. 2023, pp. 638–648.
[47] X. Zhao, F. Chen, S. Hu, and J.-H. Cho, "Uncertainty aware semi-supervised learning on graph data," in *Proc. Adv. Neural Inf. Process. Syst.*, Jan. 2020, pp. 12827–12836.
[48] M. Stadler, B. Charpentier, S. Geisler, D. Zügner, and S. Günnemann, "Graph posterior network: Bayesian predictive uncertainty for node classification," in *Proc. Adv. Neural Inf. Process. Syst.*, Jan. 2021, pp. 18033–18048.
[49] S. Yang, B. Liang, A. Liu, L. Gui, X. Yao, and X. Zhang, "Bounded and uniform energy-based out-of-distribution detection for graphs," in *Proc. 41st Int. Conf. Mach. Learn.*, vol. 235, Jul. 2024, pp. 56216–56234.
[50] W. I. D. Mining, *Introduction to Data Mining*. Cham, Switzerland: Springer, 2006.
[51] N. Shervashidze, P. Schweitzer, E. J. van Leeuwen, K. Mehlhorn, and K. M. Borgwardt, "Weisfeiler–Lehman graph kernels," *J. Mach. Learn. Res.*, vol. 12, no. 77, pp. 2539–2561, Feb. 2011.
[52] M. Ye et al., "Person reidentification via ranking aggregation of similarity pulling and dissimilarity pushing," *IEEE Trans. Multimedia*, vol. 18, no. 12, pp. 2553–2566, Dec. 2016.
[53] Q. Leng, R. Hu, C. Liang, Y. Wang, and J. Chen, "Bidirectional ranking for person re-identification," in *Proc. IEEE Int. Conf. Multimedia Expo. (ICME)*, Jul. 2013, pp. 1–6.
[54] W. Liu, X. Wang, J. D. Owens, and Y. Li, "Energy-based out-of-distribution detection," in *Proc. 34th Int. Conf. Neural Inf. Process. Syst.*, Dec. 2020, pp. 21464–21475.
[55] B. Rozemberczki et al., "PyTorch geometric temporal: Spatiotemporal signal processing with neural machine learning models," in *Proc. 30th ACM Int. Conf. Inf. Knowl. Manag.*, Oct. 2021, pp. 4564–4573.
[56] D. Hendrycks and K. Gimpel, "A baseline for detecting misclassified and out-of-distribution examples in neural networks," 2016, *arXiv:1610.02136*.
[57] K. Lee, K. Lee, H. Lee, and J. Shin, "A simple unified framework for detecting out-of-distribution samples and adversarial attacks," in *Proc. Adv. Neural Inf. Process. Syst.*, Jan. 2018, pp. 7167–7177.
[58] D. Hendrycks, M. Mazeika, and T. G. Dietterich, "Deep anomaly detection with outlier exposure," in *Proc. 7th Int. Conf. Learn. Represent. (ICLR)*, Jan. 2018, pp. 1–15.
[59] M. Belkin, P. Niyogi, and V. Sindhwani, "Manifold regularization: A geometric framework for learning from labeled and unlabeled examples," *J. Mach. Learn. Res.*, vol. 7, no. 85, pp. 2399–2434, Dec. 2006.
[60] S. Pan, R. Hu, G. Long, J. Jiang, L. Yao, and C. Zhang, "Adversarially regularized graph autoencoder for graph embedding," 2018, *arXiv:1802.04407*.
[61] F. Wu, T. Zhang, C. Fifty, T. Yu, and K. Q. Weinberger, "Simplifying graph convolutional networks," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2019, pp. 6861–6871.
[62] B. Jiang, Z. Zhang, D. Lin, J. Tang, and B. Luo, "Semi-supervised learning with graph learning-convolutional networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 11313–11320.

Jiaqiang Zhang received the B.S. degree in information and computing science from Nanjing XiaoZhuang University, Nanjing, China, in 2020. He is currently pursuing the Ph.D. degree with the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing.

His research interest includes self-supervised learning and graph representation learning.

Xinrui Wang received the B.S. degree in computer science from the College of Computer Science, Northwestern Polytechnical University, Xi'an, China, in 2021. He is currently pursuing the Ph.D. degree with the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, China, supervised by Prof. Songcan Chen.

His research interests include weakly supervised learning, continual learning, and deep learning safety.

Songcan Chen (Senior Member, IEEE) received the B.S. degree in mathematics from Hangzhou University (now merged into Zhejiang University), Hangzhou, China, in 1983, the M.S. degree in computer applications from Shanghai Jiao Tong University, Shanghai, China, in 1985, and the Ph.D. degree in communication and information systems from the Nanjing University of Aeronautics and Astronautics, Nanjing, China, in 1997.

In January 1986, he joined NUAA, where he has been a full-time Professor with the College of Computer Science and Technology since 1998. His research interests include pattern recognition, machine learning, and neural computing.

Dr. Chen is also a fellow of IAPR.