USER: Unsupervised Structural Entropy-Based Robust Graph Neural Network

Yifei Wang[1], **Yupan Wang**[1], **Zeyu Zhang**[1], **Song Yang**[1], **Kaiqi Zhao**[1], **Jiamou Liu** [1*]

[1] School of Computer Science, The University of Auckland, New Zealand
{wany107, ywan980, zzha669, syan382}@aucklanduni.ac.nz, {kaiqi.zhao, jiamou.liu}@auckland.ac.nz

Abstract

Unsupervised/self-supervised graph neural networks (GNN) are susceptible to the inherent randomness in the input graph data, which adversely affects the model's performance in downstream tasks. In this paper, we propose USER, an unsupervised and robust version of GNN based on structural entropy, to alleviate the interference of graph perturbations and learn appropriate representations of nodes without label information. To mitigate the effects of undesirable perturbations, we analyze the property of intrinsic connectivity and define the intrinsic connectivity graph. We also identify the rank of the adjacency matrix as a crucial factor in revealing a graph that provides the same embeddings as the intrinsic connectivity graph. To capture such a graph, we introduce structural entropy in the objective function. Extensive experiments conducted on clustering and link prediction tasks under random-perturbation and meta-attack over three datasets show that USER outperforms benchmarks and is robust to heavier perturbations. [1]

Introduction

Neural-based methods have become crucial for processing complex graph data in various application areas such as social media mining, recommender systems, biological data analysis, and traffic prediction. Graph representation learning (GRL) plays a central role in these methods by providing vectorized graph encodings that are essential for downstream tasks like community detection, link prediction, node classification, and network visualization (Hamilton, Ying, and Leskovec 2017a). Among the many GRL methods that have emerged in recent years, graph neural networks (GNNs) (Hamilton, Ying, and Leskovec 2017b; Kipf and Welling 2017; Veličković et al. 2018) provide a powerful paradigm that extracts graph encodings through a recursive aggregation scheme (Kipf and Welling 2017). This aggregation scheme learns a node's embedding using both the node's feature and the aggregated feature of its neighbors, capturing the structural information of the graph. The outstanding performance of GNN-based models on many tasks (Kipf and Welling 2016; Wang et al. 2017; Pan et al. 2018; Gao and Huang 2018; Veličković et al. 2019; Zhang et al. 2019; Wang et al. 2019a; Pan et al. 2019; Cui et al. 2020; Mavromatis and Karypis 2021) attests to the advantage of these models. Despite the successes above, the performance of a GNN-based model hinges on reliable input graph data (Jin et al. 2020b; Chen et al. 2020; Luo et al. 2021). Even small perturbations in the input graph can lead to drastically different encodings after the recursive aggregation scheme .However, as (Wang et al. 2019b) discussed, the randomness of input graph is inevitable. The edges in the graph are formed randomly, following an underlying intrinsic connectivity distribution of nodes. This distribution is highly inhomogeneous, with edges concentrated within some conglomerates of nodes, resulting in a community structure, as asserted by (Fortunato 2010). Each of these conglomerates is called a community, and they have dense inner-connections but sparse inter-connections. The community structure of a network can be interpreted probabilistically as a fixed but unknown edge distribution between any pair of nodes that determines the community structure, yet we only observe a sample from this distribution. This inherent randomness is responsible for perturbations in the input graph that can interfere with the recursive aggregation scheme. To develop a robust GNN that captures the intrinsic connectivity graph, we need to overcome the challenges posed by the inherent randomness of the input graph. The **first challenge** is to define an operational criterion that can alleviate the interference of graph randomness, specifically perturbations in the edges. To address this challenge, we need to identify the properties of a graph that can help us mitigate the effects of undesirable perturbations. One approach to achieving this is to generate a graph that reflects the underlying intrinsic connectivity in the dataset, even in the absence of ground truth labels. **The second challenge** involves developing an unsupervised approach for learning a model that can generate such a graph. To do this, we need to define an objective function that guides the model to reveal the intrinsic connectivity graph based on its properties. By addressing these challenges, we can create an unsupervised/self-supervised GNN that is robust to perturbations and can capture the intrinsic connectivity graph. **For the first challenge**, we will address in **Section** that multiple "innocuous graphs" exist, for which GNN can produce the same embeddings as the de-

*Corresponding author

[1]Full proof, experimental details, and code of our work are available at https://github.com/wangyifeibeijing/USER.

sired intrinsic connectivity graph. We refer to these graphs as *GNN-indistinguishable* to the intrinsic connectivity graph. Therefore, any of these innocuous graphs can help GNN alleviate the interference of randomness. As many datasets have a known number of groups in the intrinsic connectivity graph (the number of communities c), we can make two assertions about the innocuous graphs: First, the rank of the corresponding adjacency matrix of innocuous graphs is no less than the number of groups in the intrinsic connectivity graph. Second, if we group the innocuous graphs into sets with high concentrations of edges inside each set, and low concentrations of edges between them, the features of two nodes that belong to the same group should be relatively similar. These assertions guide us in our pursuit of innocuous graphs. **For the second challenge**, we propose a tool to learn a graph that satisfies the aforementioned conditions. In Section , we leverage structural information theory (Li and Pan 2016; Li et al. 2016; Liu et al. 2021) and introduce a class of structural entropy measures that can capture the intrinsic information contained within a graph structure. These measures have been increasingly applied to graph learning. Specifically, we connect the notion of network partition structural information (NPSI), a type of structural entropy, with the rank of the adjacency matrix and demonstrate that minimizing the structural entropy and the widely-used Davies-Bouldin index (DBI) (Davies and Bouldin 1979) would facilitate the search for an innocuous graph. By combining the tools developed above, we introduce a novel framework, named USER (**U**nsupervised **S**tructural **E**ntropy-based **R**obust graph neural network), which supports GNNs with a trainable matrix to learn the adjacency matrix of an innocuous graph, as detailed in Section . This method enables the simultaneous learning of the innocuous graph structure and node embeddings. As the embeddings are derived from the innocuous graph, instead of the input graph, they become resilient to randomness. We conducted a series of experiments to validate the effectiveness of our USER framework. The results demonstrate that even the traditional GAE model outperforms state-of-the-art baselines on both clustering and link prediction tasks when supported by USER. To test the robustness of our approach to graph randomness, we injected 50% random noise into the graph. Our experiments show that on the well-known Cora dataset, USER achieves up to a 14.14% improvement in clustering accuracy and a 13.12% improvement in link prediction accuracy over the state-of-the-art baseline. Additionally, our approach exhibits a significant advantage in the presence of adversarial attacks. When subjected to a 20% meta-attack (Zügner and Günnemann 2019), USER's improvements over the state-of-the-art are up to 190.01% for clustering on the Citeseer dataset. Our contributions can be summarized as follows:

- we introduce the notion of innocuous graphs and utilize them to mitigate the interference of graph randomness;
- we propose a structural entropy-based objective function that is suitable for learning the innocuous graph; and
- we conduct extensive experiments that demonstrate the effectiveness of USER in confronting randomness.

Related Work

GRL and GNN. Graph representation learning (GRL) generates vectorized encoding from graph data. Nowadays GRL is instrumental in many tasks that involves the analysis of graphs (Hamilton, Ying, and Leskovec 2017a). As a mainstream GRL paradigm, graph neural network (GNN) captures a node's structural information and node features by recursively aggregating its neighborhood information using an *information aggregation scheme*. Based on this idea, Graph Autoendoer (GAE) and Variational GAE (VGAE) (Kipf and Welling 2016) are developed to use GCN (Kipf and Welling 2017) as an encoder to learn node embeddings, and an inner product decoder to reconstruct the graph structure. As a variant of GAE, ARGA (Pan et al. 2018) trains an adversarial network to learn more robust node embeddings. To alleviate the high-frequency randomness in the node features, Adaptive Graph Encoder (AGE) (Cui et al. 2020) utilizes a Laplacian smoothing filter to prepossess the node features. Different from reconstructing the graph structure, maximizing the *mutual information* (MI) (Hjelm et al. 2018) is another well-studied approach for GRL. For example, the model DGI (Veličković et al. 2019) employs GNN to learn node embeddings and a graph-level embedding, then maximize MI between them to improve the representations' quality. The model GIC (Mavromatis and Karypis 2020) follows this idea and seeks to additionally capture community-level information of the graph structure. Despite their outstanding performance, GNNs have been shown to be vulnerable to small perturbations on the input graph, which can be particularly challenging for unsupervised models (Li et al. 2018; Zhu et al. 2021a; Wan and Kokel 2021; Wu et al. 2019b). To address this issue, new learning paradigms have been proposed to enhance the robustness of GNNs. For example, the Cross-graph model (Wang et al. 2020) uses two autoencoders, where each encoder learns node embeddings and reconstructs the adjacency matrix to be passed to the peer-autoencoder as the input for the next iteration. Graph contrastive learning (G-CL) models, such as GCA (Zhu et al. 2021b), improve the robustness of GNNs by constructing data augmentation and negative pairs through modifications of the input graph structure. However, none of these approaches explains how these perturbations arise. In this paper, we introduce the notion of an innocuous graph, inspired by previous works (Wang et al. 2019b; Fortunato 2010; Zhang et al. 2019; Wu et al. 2019a; Zhu and Koniusz 2020), to learn embeddings that are equivalent to those corresponding to the intrinsic connectivity graph. This approach helps GNN models mitigate the impact of randomness.

Structural entropy. In our paper, we utilize structural entropy as a key tool. The search for an entropy measure that can analyze the intrinsic information contained in structures has been a long-standing challenge in computer science (Brooks Jr 2003). A number of classical entropy measures have been developed for this purpose (Dehmer 2008; Anand and Bianconi 2009). One notable example is infomap (Rosvall, Axelsson, and Bergstrom 2009), which uses a form of entropy defined on random walks to analyze graphs. More recently, in (Li and Pan 2016; Li et al. 2016), the authors introduced a hierarchy of *structural entropy* measures to an-

alyze networks. These measures have since been applied in several works, such as (Liu et al. 2019, 2022; Chen and Liu 2019), for adversarial graph learning. However, to the best of our knowledge, no previous study has explored the integration of structural entropy to enhance GNN's resilience to randomness in graph data.

Criteria to Mitigate Randomness

To clarify our notation, we use $\vec{x}, \vec{y}, \ldots$ to denote vectors where x_i denotes the ith entry of $\vec{x}$. We use capital letters $X, Y, A, \ldots$ to denote real-valued matrices. For any matrix M, M_i denotes the ith row vector and M_{ij} denotes the (i,j)th entry of M. In this paper, we focus on undirected, unweighted graphs where every node is associated with a d-dimensional feature vector. Formally, such a graph can be denoted by $\mathcal{G} = (\mathcal{V}, \mathcal{E}, X)$ where $\mathcal{V}$ is a set of n nodes $\{v_1, \ldots, v_n\}$, $\mathcal{E}$ is a set of edges $\{v_i, v_j\}$, and $X \in \mathcal{M}_{n,d}(\mathbb{R})$ denotes the *feature matrix* where X_i is the feature vector of node v_i. The pair $(\mathcal{V}, \mathcal{E})$ is represented by an *adjacent matrix* $A \in \mathcal{M}_n(\{0,1\})$, where $A_{ij} = 1$ if $\{v_i, v_j\} \in \mathcal{E}$. Here we assume the graph does not contain any isolated node. Indeed, most studies on GNN omit isolated nodes before training (Kipf and Welling 2016; Mavromatis and Karypis 2021). At last, we use $\mathcal{C}_0, \mathcal{C}_1, \ldots \mathcal{C}_{c-1}$ to denote c sets of nodes. If they satisfy: $k \neq m \Rightarrow \mathcal{C}_k \cap \mathcal{C}_m = \varnothing$ and $\forall k < c$: $\mathcal{C}_k \neq \varnothing$, we call them partitions.

Taking input graph $\mathcal{G}$, a *graph neural network (GNN)* is a function denoted by:

$$\mathsf{GNN}\left(A, X, \left\{W^{(\ell)}\right\}\right) = H^{(t)} \tag{1}$$

with $H^{(0)} = X$ and $H^{(\ell)} = \sigma(\mathrm{agg}(AH^{(\ell-1)}W^{(\ell)}))$ for all $\ell \in (0, t]$, $H^{(\ell)} \in \mathbb{R}^{n \times d^{\ell}}$, $W^{(\ell)} \in \mathbb{R}^{d^{(\ell-1)} \times d^{(\ell)}}$, and $d^{(0)} = d$. where, $\sigma(\cdot)$ is the activation function; the function $\mathrm{agg}(\cdot)$ is responsible for aggregating information from neighboring nodes. The input to this function is typically of the form $AH^{(\ell-1)}W^{(\ell)}$, where $H^{(\ell-1)}$ is the feature matrix of the neighboring nodes ($H^{(0)} = X$, taking the original features as the input to the 1st layer;), $W^{(\ell)}$ is the weight matrix for the ℓ-th layer, and A is the adjacency matrix that encodes the graph structure. The function $\mathrm{agg}(\cdot)$ then processes this input to produce a new feature vector for each node. This new feature vector includes information from the central node's own features and its neighbors' features. The precise way in which $\mathrm{agg}(\cdot)$ combines these features depends on the specific GNN architecture being used. According to (Xu et al. 2019), GNN models that use non-injective $\sigma(\cdot)$ and $\mathrm{agg}(\cdot)$ functions can be inefficient when learning graph structures. Therefore, we will focus only on GNN models that use injective versions of these functions. As shown in Equation (1), the vector representation of a node in GNN models is computed recursively by aggregating not only its own features but also the features of its neighboring nodes.

As previously mentioned, input graph datasets in the real world are inherently random and unstable (Jin et al. 2020b). However, these datasets reflect certain hidden but stable underlying distribution of *intrinsic connectivity* (Wang et al. 2019b; Fortunato 2010). According to (Fortunato 2010), intrinsic connectivity for a dataset that can be naturally divided into c partitions (or *classes* in a node classification task) dictates that nodes within the same partition are more likely to be connected than nodes in different partitions. We formalize this notion with the following definition.

Definition 1 (Intrinsic connectivity graph)**.** *For a dataset that contains c partitions, suppose $\mathcal{G}_I = (\mathcal{V}, \mathcal{E}_I)$ satisfies: For any two nodes v_i and v_j, there exists an edge $(v_i, v_j) \in \mathcal{E}_I$ iff v_i and v_j belong to the same partition. We call $\mathcal{G}_I$ the* intrinsic connectivity graph*.*

Let $Rank(M)$ denote the rank of a matrix M.

Theorem 1 (Rank of $\mathcal{G}_I$'s adjacency matrix A_I)**.** *For a dataset that contains c partitions, we have $Rank(A_I) = c$ where A_I is $\mathcal{G}_I$'s adjacency matrix.*

Our goal is to extract a new graph from a real-world dataset to mitigate the interference of graph randomness. Finding the intrinsic connectivity graph without a ground truth label is impractical. However, we observe that GNN can learn the same embeddings from different input graphs:

Definition 2 (GNN-indistinguishability)**.** *Let $\mathcal{G}_0 = (\mathcal{V}, \mathcal{E}_0, X)$ and $\mathcal{G}_1 = (\mathcal{V}, \mathcal{E}_1, X)$ be two graphs with the same set of nodes and adjacency matrices A_0 and A_1, respectively. We say that $\mathcal{G}_1$ is* GNN-indistinguishable *from $\mathcal{G}_0$ if, for any feature matrix X, for any weight matrices $W_0^{(1)}, \ldots, W_0^{(L)}$, there exist $W_1^{(1)}, \ldots, W_1^{(L)}$ such that the GNNs ran on $\mathcal{G}_0$ and $\mathcal{G}_1$ with these weight matrices have the same activations in each layer, i.e., for all $1 \leq \ell \leq L$, we have: $\sigma(\mathrm{agg}(A_0 H^{(\ell-1)} W_0^{(\ell)})) = \sigma(\mathrm{agg}(A_1 H^{(\ell-1)} W_1^{(\ell)}))$, where, $H^{(\ell)}$ denotes node representations in the ℓth layer.*

According to Definition 2, if $\mathcal{G}_1$ is GNN-indistinguishable from $\mathcal{G}_0$, then a GNN with $\mathcal{G}_1$ as input can learn the same embeddings as if $\mathcal{G}_0$ were the input. Therefore, using graphs that are GNN-indistinguishable from the intrinsic connectivity graph $\mathcal{G}_I$ allows the GNN to learn the same embeddings as if $\mathcal{G}_I$ were the input. We refer to such graphs as *innocuous*.

Definition 3 (innocuous graph)**.** *Suppose $\mathcal{G}_I$ is the intrinsic connectivity graph for a dataset. An* innocuous graph *$\mathcal{G}'$ is a graph that is GNN-indistinguishable from $\mathcal{G}_I$.*

Next, we introduce the necessary condition for a graph to be GNN-indistinguishable from a specific graph:

Theorem 2 (necessary condition for being GNN-indistinguishable)**.** *$\mathcal{G}_1$ is GNN-indistinguishable to $\mathcal{G}_0$ only if $Rank(A_1) \geq Rank(A_0)$.*

Corollary 1 (necessary condition of innocuous graph)**.** *$\mathcal{G}'$ is a innocuous graph only if $Rank(A') \geq Rank(A_I)$.*

From Theorem 1 and Corollary 1, it follows that A' of an innocuous graph $\mathcal{G}'$ must satisfy $Rank(A') \geq c$.

In addition to the property mentioned above, another commonly used assumption in the literature (Wu et al. 2019b; Jin et al. 2020b) is that in a graph where a GNN can extract semantically useful node embeddings, adjacent nodes are likely to have similar features compared to non-adjacent nodes. However, this formulation only considers information aggregation of GNN along a single edge. To extend feature smoothness to the group level, we introduce a function

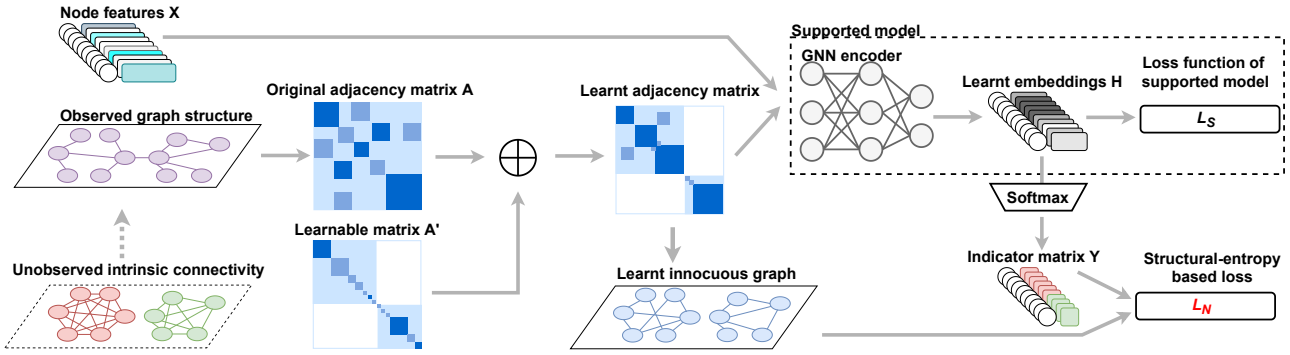


Figure 1: The USER framework.

$f(\vec{x}, \vec{y})$ that evaluates the similarity between learned node embeddings, where a smaller $f(\vec{x}, \vec{y})$ indicates a higher similarity between the two embedding vectors $\vec{x}$ and $\vec{y}$. We then formulate the group-level feature smoothness:

Assumption 1 (group-level feature smoothness)**.** *Suppose $k \neq m$. Then for any three nodes v_a, v_b, v_c that satisfy $v_a \in \mathcal{C}_k$, $v_b \in \mathcal{C}_k$ and $v_c \in \mathcal{C}_m$, we have $f(X_a, X_b) \leq f(X_a, X_c)$.*

We present a criterion for identifying an innocuous graph, which combines a necessary condition (Corollary 1) with an additional assumption (Assumptions 1).

Learning An Innocuous Graph

To obtain an innocuous graph that satisfies the necessary conditions (Corollary 1) and the auxiliary assumption (Assumption 1), we formulate an optimization problem in this section. We interpret Corollary 1 in the context of structural information theory and use the Davies-Bouldin index (DBI) to achieve Assumption 1.

Drawing on recent advances in structural information theory (Li and Pan 2016), we introduce the concept of *network partition structural information (NPSI)*, which has not been utilized in GNN models previously. To explain NPSI, we first introduce the following notations: $P(\mathcal{G}) = \{\mathcal{C}_0, \mathcal{C}_1, \dots \mathcal{C}_{r-1}\}$ is a *partition* of $\mathcal{G}$. Then, $P(\mathcal{G})$ can be denoted by a matrix $Y \in \{0,1\}^{n \times r}$, where $Y_{ik} = 1$ if $v_i \in \mathcal{C}_k$ otherwise $Y_{ik} = 0$. We call Y the *indicator matrix*. Since $\mathcal{C}_k \neq \varnothing$, $(Y^T Y)_{kk} > 0$, and since $\forall k \neq m$, $\mathcal{C}_k \cap \mathcal{V}_m = \varnothing$, if $k \neq m$, $(Y^T Y)_{km} = 0$. For a graph $\mathcal{G}$ and partition $P(\mathcal{G})$, let vol_k be the number of edges with at least one node in $\mathcal{C}_k$ and g_k be the number of edges with only one node in $\mathcal{C}_k$. Then by (Liu et al. 2019), NPSI is:

$$NPSI_{\mathcal{G}P(\mathcal{G})} = \sum_{k<r} \left(\frac{vol_k - g_k}{2|\mathcal{E}|} \log_2 \frac{vol_k}{2|\mathcal{E}|} \right) \tag{2}$$

To utilize NPSI in GNN models, we represent it in a matrix form. Note that $vol_k - g_k$ is the number of edges with both nodes in $\mathcal{C}_k$, which equals the k-th diagonal element in $Y^T AY$, while the k-th value in the sum of column in (AY) equals to vol_k and can be computed by the k-th diagonal element in $\{1\}^{r \times n} AY$. Let $trace(\cdot)$ be trace of input matrix,

$$\begin{aligned} NPSI(A,Y) &= NPSI_{\mathcal{G}P(\mathcal{G})} \\ &= \sum_{k<r} \left(\frac{vol_k - g_k}{2|\mathcal{E}|} \log_2 \frac{vol_k}{2|\mathcal{E}|} \right) \\ &= \sum_{k<r} \left(\frac{(Y^T AY)_{kk}}{2sum(A)} \times \log_2 \left(\frac{(\{1\}^{r \times n} AY)_{kk}}{2sum(A)} \right) \right) \\ &= \text{trace} \left(\frac{Y^T AY}{2sum(A)} \otimes \log_2 \left(\frac{\{1\}^{r \times n} AY}{2sum(A)} \right) \right) \end{aligned}$$

We can incorporate $NPSI(A,Y)$ into GNN using the definition above. To understand the relation between NPSI and the learned graph, we need to understand the relationship between the rank of the adjacency matrix and the structure:

Lemma 1 (Rank and connected components)**.** *Suppose $\mathcal{G}$ is a graph without isolated nodes. If there are r connected components in $\mathcal{G}$, the corresponding adjacency matrix A satisfies $Rank(A) \geq r$.*

Then we find that with a fixed $Y \in \{0,1\}^{n \times r}$ which satisfies $(Y^T Y)_{kk} > 0$ and $(Y^T Y)_{km} = 0$ for $k \neq m$, we can learn a graph $\mathcal{G}'$ with corresponding adjacency matrix A' satisfying $Rank(A') \geq r$:

Theorem 3 (minimize $NPSI$ with learnable A')**.**

$$\begin{aligned} &\textit{Suppose } A' = \arg\min_{A'} \left(NPSI(A', Y) \right), \\ &s.t. A'_{ij} \geq 0 \textit{ and } A' = A'^T, \end{aligned} \tag{3}$$

A' satisfies: $Rank(A') \geq r$

Therefore, based on NPSI, if we set $r = c$, we construct an objective function to learn an adjacency A' which meets the necessary condition Corollary 1. Simultaneously, with the partition indicator Y, we utilize the well-known Davies-Bouldin index (DBI) to analyze the similarity of node features inside the same partition (Davies and Bouldin 1979):

$$\begin{aligned} DBI(X,Y) &= \frac{1}{r} \sum_{k<r} DI_k \\ \text{where: } DI_k &= max_{m \neq k}(R_{km}), \; R_{km} = \frac{S_k + S_m}{M_{km}} \\ S_k &= \left(\frac{1}{|\mathcal{C}_k|} \sum_{Y_{ik}=1} (|X_i - \overline{X}_k|^2) \right)^{\frac{1}{2}}, \\ M_{km} &= (|\overline{X}_k - \overline{X}_m|^2)^{\frac{1}{2}}, \; \overline{X}_k = \frac{\sum_{Y_{ik}=1} X_i}{|\mathcal{C}_k|}. \end{aligned} \tag{4}$$

An adjacency matrix A' satisfies Assumptions 1 would make $DBI(X,Y)$ small. Based on the necessary conditions (Corollary 1) and assumptions (Assumptions 1) for obtaining an innocuous graph, we construct an objective function using the network partition structural information (NPSI) and Davies-Bouldin index (DBI) to learn an adjacency matrix A that meets these conditions. Let β be a hyperparameter. The objective function is:

$$\begin{aligned} &\mathcal{L}_N = NPSI(A',Y) + \beta DBI(X,Y) \\ &s.t.\ A'_{ij} \geq 0,\ A' = A'^T, \\ &\quad Y \in \{0,1\}^{n\times c},\ Y_{km} \begin{cases} >0 & \text{if } k=m, \\ =0 & \text{otherwise.} \end{cases} \end{aligned} \tag{5}$$

Then our overall criterion for finding an innocuous graph is formulated into an optimization problem of minimizing $\mathcal{L}_N$ in (5), where Y and A' are elements to be optimized.

Unsupervised Structural Entropy-based Robust Graph Neural Network

In this section, we propose a novel framework that enables GNN models to learn embeddings and innocuous graphs simultaneously, achieving robust learning by optimizing the loss in Equation (5). Our framework supports various GNN models, and we take the classical GAE (Kipf and Welling 2016) as an example. There are two key components in our framework that distinguish it from previous work:

Learned Matrices. Let A denote the adjacency matrix of the original input graph. To remove the effect of randomness, we construct an innocuous graph and use it as the input of the supported model instead of the original graph. We thus construct a learnable matrix $A' \in \mathbb{R}^{n\times n}$, and use it as the input of the supported GNN model:

$$H = \mathsf{GNN}\left(A', X, \left\{W^{(1)}, W^{(2)}\right\}\right) \tag{6}$$

H is the learnt node embeddings. Besides the node embeddings, an indicator matrix Y that divide the graph into partitions with high inner-connectivity and sparse inter-connectivity is desired (Li and Pan 2016):

$$\begin{aligned} &\mathcal{C}_k = \{v_i | Y_{ik} \neq 0\} \text{ where,} \\ &Y = \arg\min_{Y} \left(NPSI(A',Y)\right) \\ &s.t.\ Y \in \{0,1\}^{n\times r},\ (Y^TY)_{km} \begin{cases} >0 & \text{if } k=m, \\ =0 & \text{otherwise.} \end{cases} \end{aligned}$$

To learn such an indicator matrix Y, we add a softmax layer with learnable parameter matrix $W^Y \in \mathbb{R}^{d^{(2)}\times c}$:

$$Y = softmax(HW^Y) \tag{7}$$

Overall Optimization. Let $\mathcal{L}_S$ be the loss function of the supported model, e.g., for GAE:

$$\mathcal{L}_S = ||\hat{A} - A||_F^2, \tag{8}$$

where $\hat{A}$ is reconstructed from learnt node embeddings by $\hat{A} = sigmoid(HH^T)$. Besides $\mathcal{L}_S$, $\mathcal{L}_N$ in (5) is employed to alleviate the interference of randomness. Thus, let α be hyper-parameter, model is trained by minimizing $\mathcal{L}$:

$$\mathcal{L} = \mathcal{L}_N + \alpha \mathcal{L}_S. \tag{9}$$

Although unsupervised, with structural entropy based $\mathcal{L}_N$, this framework mitigates randomness-interference, making the supported model more capable. We call it an Unsupervised Structural Entropy-based Robust Graph Neural Network (USER). The detailed structure is shown in Figure 1.

Dataset	# Nodes	# Edges	# Features	# Classes
Cora	2,708	5,429	1,433	7
Citeseer	3,327	4,732	3,703	6
Wiki	2,405	17,981	4,973	17

Table 1: Dataset statistics.

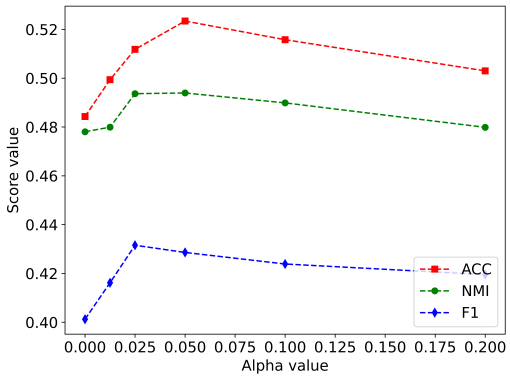


Figure 2: Parameter analysis on Wiki (α).

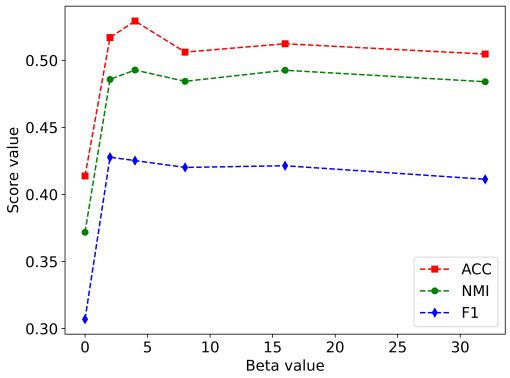


Figure 3: Parameter analysis on Wiki (β).

Experimental Results

Experiments

In this section, we compare USER-supported GAE (USER) with state-of-the-art methods and conduct some analyses.

Dataset	Ptb	deepwalk	GAE	VGAE	ARGA	AGE	DGI	GIC	GCA	G_CG	A_CG	USER
cora	0	39.58	44.32	43.42	44.12	56.4	**57.32**	52.16	32.76	44.3	45.18	56.24
	0.1	35.2	41.28	39.65	40.05	42.23	53.45	50.24	35.1	42.47	43.11	**54.38**
	0.2	28.61	33.0	35.16	34.6	35.1	50.47	48.54	32.18	38.31	37.96	**52.17**
	0.3	26.55	29.07	32.31	29.49	36.05	48.22	44.75	32.93	35.19	36.59	**52.39**
	0.4	21.21	27.08	27.35	26.86	32.68	44.02	40.97	33.4	33.47	34.46	**46.7**
	0.5	21.5	25.03	24.99	23.71	36.79	43.22	40.73	31.81	31.93	31.53	**49.33**
citeseer	0	13.89	21.66	20.84	20.72	35.82	**44.02**	43.56	28.1	21.3	19.75	35.52
	0.1	11.58	18.3	17.52	17.94	29.47	**41.31**	41.29	10.75	20.41	18.26	37.04
	0.2	8.77	16.16	15.07	15.6	21.65	36.66	**36.72**	7.0	16.84	16.93	34.42
	0.3	7.58	12.86	13.59	13.01	18.06	33.39	33.58	6.16	15.17	14.41	**34.5**
	0.4	6.85	9.81	10.34	10.03	15.57	32.26	31.91	4.22	12.71	12.03	**34.58**
	0.5	5.49	10.41	10.33	9.63	14.13	29.58	30.96	2.98	12.59	13.66	**34.5**
wiki	0	35.85	23.52	24.49	22.8	**51.2**	43.14	27.45	37.54	22.72	23.04	48.99
	0.1	33.15	22.57	16.59	18.01	48.85	40.13	24.65	30.77	19.69	19.63	**48.97**
	0.2	30.74	13.66	14.22	14.05	46.92	36.15	22.9	30.74	11.62	16.54	**48.71**
	0.3	27.79	14.7	15.97	15.1	47.43	34.84	38.29	31.42	16.5	10.66	**48.55**
	0.4	26.56	8.26	9.8	15.0	46.7	31.28	36.33	32.38	9.99	11.41	**48.54**
	0.5	25.52	9.52	7.4	12.01	46.83	29.29	33.26	34.24	9.26	6.6	**48.68**

Table 2: Node clustering performance (NMI) under random-noises

Dataset	Ptb	deepwalk	GAE	VGAE	ARGA	AGE	DGI	GIC	GCA	G_CG	A_CG	USER
cora	0.05	41.73	43.37	43.06	43.33	48.6	50.33	46.89	38.12	43.64	43.0	**50.64**
	0.10	37.68	34.1	33.6	34.5	39.35	37.73	36.58	34.07	35.47	35.94	**41.71**
	0.15	21.99	19.96	19.56	20.04	25.39	23.13	23.19	21.54	22.59	22.92	**29.27**
	0.20	7.31	7.26	7.22	7.88	9.65	10.17	10.96	9.97	10.34	10.31	**18.82**
citeseer	0.05	16.97	22.5	22.73	20.85	34.06	**40.22**	39.91	20.78	22.67	21.69	35.72
	0.10	23.52	22.25	22.59	22.02	25.13	29.71	29.45	18.92	22.6	22.06	**31.86**
	0.15	17.33	13.73	13.6	13.94	15.71	17.68	17.81	13.61	15.6	15.61	**27.77**
	0.20	8.3	5.64	5.71	5.63	9.11	9.11	9.08	7.08	7.68	7.61	**26.42**
wiki	0.05	34.06	19.59	19.22	20.8	41.76	32.94	35.03	27.24	18.24	16.27	**48.44**
	0.10	22.96	13.09	11.14	12.48	38.72	22.59	23.64	25.86	13.34	10.98	**47.71**
	0.15	14.35	4.59	4.99	6.82	40.9	12.27	15.19	20.14	4.62	7.04	**47.54**
	0.20	9.3	2.22	1.52	3.61	42.71	8.85	9.24	15.39	3.1	2.49	**47.48**

Table 3: Node clustering performance (NMI) under meta-attack

Dataset	Ptb	GAE	ARGA	G_CG	A_CG	USER
citeseer	0	94.09	94.87	94.08	94.0	**95.38**
	0.1	94.09	94.25	93.96	94.04	**95.59**
	0.2	94.12	93.69	94.01	94.05	**94.99**
	0.3	91.72	92.31	93.15	93.27	**95.15**
	0.4	90.29	90.81	93.07	92.89	**94.41**
	0.5	90.05	90.91	91.85	91.6	**94.54**
wiki	0	86.75	82.14	83.25	79.81	**88.72**
	0.1	80.12	83.66	68.04	77.24	88.07
	0.2	79.5	80.86	70.62	74.57	**87.82**
	0.3	73.02	80.06	66.27	68.97	**87.41**
	0.4	78.37	79.44	61.58	64.06	**87.45**
	0.5	67.78	76.79	64.48	71.51	**86.87**

Table 4: Link prediction (AUC) under random-noises

Dataset	Ptb	GAE	ARGA	G_CG	A_CG	USER
citeseer	0	94.22	94.89	94.5	94.14	**95.84**
	0.1	94.08	94.47	94.21	94.38	**95.98**
	0.2	94.57	94.05	94.5	94.42	**95.46**
	0.3	92.13	92.77	93.83	93.94	**95.66**
	0.4	90.95	91.38	93.62	93.58	**94.92**
	0.5	90.81	91.66	92.73	92.44	**95.05**
wiki	0	88.18	83.38	85.65	83.01	**89.9**
	0.1	81.95	86.02	69.8	80.78	**89.48**
	0.2	82.62	82.44	73.09	77.22	**89.07**
	0.3	76.89	82.98	69.06	72.13	**88.9**
	0.4	81.53	82.22	63.15	65.91	**88.61**
	0.5	70.36	80.29	67.09	73.6	**88.28**

Table 5: Link prediction (AP) under random-noises

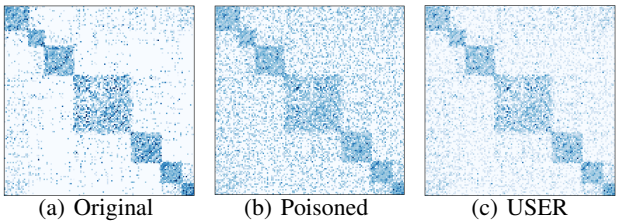
Experimental Settings

Datasets. We evaluate all models on three benchmark datasets: *Cora*, *Citeseer*, *Wiki* (Kipf and Welling 2017; Yang et al. 2015; Jin et al. 2020b). They have different structures and node features. Dataset statistics are in Table 1.

Randomness. We inject noises into the original graph to promote graph randomness. In particular, we develop two types of noises: *random noise* and *meta-attack* (Zügner and Günnemann 2019). Random noise "randomly flips" the state of a pair of nodes (i.e., if there is an edge between them, we

Dataset	USER	w.o. NPSI	w.o. DBI	Fix A'
cora	54.38	14.82	52.54	40.11
citeseer	37.04	28.95	12.82	30.94
wiki	48.97	48.44	37.28	39.77

Table 6: NMI of USER's variants with 10% random-noise



(a) Original (b) Poisoned (c) USER

Figure 4: Case study: the graph heat maps of Cora

remove it; otherwise we add an edge between them). Random noises are not very effective, so we create several poisoned graphs with a noise ratio from 0% to 50% with a step of 10%. Meta-attack can promote randomness significantly (Zügner and Günnemann 2019; Jin et al. 2020a). Even for supervised models, meta-attack is hardly applied with a perturbation rate higher than 20% (Jin et al. 2020a). Thus, we create several poisoned graphs with meta-attack ratio from 0% to 20% with a step of 5%.

Baselines. For **USER**, the classical GAE (Kipf and Welling 2016) is utilized as supported model. We compare it with 10 baselines retaining the default parameter settings in their original papers. **DeepWalk** (Perozzi, Al-Rfou, and Skiena 2014) utilizes random walks to learn embeddings. **GAE** and **VGAE** (Kipf and Welling 2016) firstly leverage GCN (Kipf and Welling 2017)for GRL. **ARGA** (Pan et al. 2018) is an adversarial GNN model. **AGE** (Cui et al. 2020) applies Laplacian smoothing to GNN. **DGI** (Veličković et al. 2019) trains GNN with MI. **GIC** (Mavromatis and Karypis 2020) captures cluster-level information. **GCA** (Zhu et al. 2021b) is a Graph Contrastive learning GNN. **G_CG** and **A_CG** are Cross-Graph (Wang et al. 2020) models. G_CG is the GAE version while A_CG maintains ARGA encoders.

Parameter Settings We train USER for 400 epochs using Adam optimizer with a learning rate η. The two hyperparameters α and β, are selected through a grid search. A detailed analysis could be found in Parameter Analysis. The dimension $d^{(1)}$, learning rate η, α and β are selected accordingly based on the parameter analysis.

Evaluation Metrics For node clustering, we employ popular normalized mutual information (NMI) and clustering accuracy (ACC) (Aggarwal and Reddy 2014). For link prediction, we report area under the ROC (AUC) (Bradley 1997), and average precision (AP) (Su, Yuan, and Zhu 2015).

Clustering. Performance of all models are listed in Table 2 and Table 3. From Table 2 and Table 3, we observe that: When the input graph is the original graph, the USER's improvement from GAE is significant. The performances of USER are always close to the best. When graph randomness is promoted by random noises, USER outperforms others. Even under large Ptb e.g., 50%, the performance of USER only drops 12%, 3%, 0.6% on Cora, Citeseer and Wiki. Meta-attack makes the effect of baselines drop rapidly. However, USER is still effective.

Link prediction. To compare the performances on link prediction tasks, all the experiments are run 10 times and we report the AUC and AP in Table 4 and Table 5. The best performance is in bold. From the results, we observe that for link prediction, USER also outperforms other models. Classical models are rather unstable towards promoted randomness. Even robust model Cross-graph's performance drop drastically under large ratio noises (e.g. the A_CG dropped 3.617% and 10.400% on citeseer and wiki when noise rate is 50%). USER demonstrates stability w.r.t. different noise levels (only 0.881% and 2.085% drop with 50% noise).

Case Study To show the graph learned by USER. We illustrate the normalized adjacency matrix of Cora dataset in Figure 4(a). It is clearly observable that most edges are in one of seven groups with few edges between them. However, the adjacency matrix with 50%-ratio random-noises of Cora (as shown in Figure 4(b)) have more inter-group edges and the boundaries of classes get visibly blurred. The learned graph structure by USER is shown in Figure 4(c). We observe that the group boundaries are much clearer. This demonstrates that USER can capture ideal innocuous graphs.

Ablation Study To understand the importance of different components, we conduct ablation studies with 10% random noise. From Table 6, USER without NPSI component loses its effectiveness on all three datasets. The performance after removing DBI drops slightly on Cora but it is significantly affected on Wiki and Citeseer. This implies for these two datasets, feature information is more important. If we fix A' the same as the original, the model tends to be disturbed by the graph randomness. By incorporating all these components, USER can explore for innocuous graphs.

Parameter Analysis We explore the sensitivity of the two hyperparameters. α controls the influence of the objective from the supported model and β is used to adjust the influence of Assumption 1. We report the experimental results on Wiki with 10% random-noise in Figure 2 and Figure 3 as similar observations are made in other settings. USER's performance can be boosted when choosing appropriate values for all the hyper-parameters, but performance under other values drops slightly. This is consistent with our analysis.

Conclusion

We aim to mitigate the impact of graph randomness and learn accurate node representations without label information by proposing USER, an unsupervised robust framework. During its development, we discovered that GNNs can learn suitable embeddings with multiple innocuous graphs and that the adjacency matrix's rank plays a critical role in identifying such graphs. We introduced structural entropy as a tool to construct objective functions for capturing innocuous graphs. In the future, we plan to explore the intrinsic connectivities of graph data further.

Acknowledgements

This research was supported by NSFC (Grant No. 61932002) and Marsden Fund (21-UOA-219). The first author and third authors are supported by a PhD scholarship from China Scholarship Council.

References

Aggarwal, C. C.; and Reddy, C. K. 2014. Data clustering. *Algorithms and applications. Chapman&Hall/CRC Data mining and Knowledge Discovery series, Londra*.

Anand, K.; and Bianconi, G. 2009. Entropy measures for networks: Toward an information theory of complex topologies. *Physical Review E*.

Bradley, A. P. 1997. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern recognition*.

Brooks Jr, F. P. 2003. Three great challenges for half-century-old computer science. *JACM*.

Chen, L.; Li, J.; Peng, J.; Xie, T.; Cao, Z.; Xu, K.; He, X.; and Zheng, Z. 2020. A survey of adversarial learning on graphs. *CoRR,abs:2003.05730*.

Chen, Y.; and Liu, J. 2019. Distributed community detection over blockchain networks based on structural entropy. In *Proceedings of the 2019 ACM International Symposium on Blockchain and Secure Critical Infrastructure*, 3–12.

Cui, G.; Zhou, J.; Yang, C.; and Liu, Z. 2020. Adaptive graph encoder for attributed graph embedding. In *KDD*.

Davies, D. L.; and Bouldin, D. W. 1979. A cluster separation measure. *IEEE transactions on pattern analysis and machine intelligence*.

Dehmer, M. 2008. Information processing in complex networks: Graph entropy and information functionals. *Applied Mathematics and Computation*.

Fortunato, S. 2010. Community detection in graphs. *Physics reports*, 486(3-5): 75–174.

Gao, H.; and Huang, H. 2018. Deep Attributed Network Embedding. In *IJCAI*.

Hamilton, W. L.; Ying, R.; and Leskovec, J. 2017a. Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584*.

Hamilton, W. L.; Ying, Z.; and Leskovec, J. 2017b. Inductive Representation Learning on Large Graphs. In *NIPS*.

Hjelm, R. D.; Fedorov, A.; Lavoie-Marchildon, S.; Grewal, K.; Bachman, P.; Trischler, A.; and Bengio, Y. 2018. Learning deep representations by mutual information estimation and maximization. In *ICLR*.

Jin, W.; Li, Y.; Xu, H.; Wang, Y.; Ji, S.; Aggarwal, C.; and Tang, J. 2020a. Adversarial Attacks and Defenses on Graphs: A Review, A Tool and Empirical Studies. *KDD Explorations*.

Jin, W.; Ma, Y.; Liu, X.; Tang, X.; Wang, S.; and Tang, J. 2020b. Graph structure learning for robust graph neural networks. In *KDD*.

Kipf, T. N.; and Welling, M. 2016. Variational Graph Auto-Encoders. *CoRR*.

Kipf, T. N.; and Welling, M. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR (Poster)*.

Li, A.; Hu, Q.; Liu, J.; and Pan, Y. 2016. Resistance and Security Index of Networks: Structural Information Perspective of Network Security. *Scientific Reports*.

Li, A.; and Pan, Y. 2016. Structural information and dynamical complexity of networks. *IEEE Trans. Inf. Theory*.

Li, R.; Wang, S.; Zhu, F.; and Huang, J. 2018. Adaptive graph convolutional neural networks. In *AAAI*, volume 32.

Liu, W.; Liu, J.; Zhang, Z.; Liu, Y.; and Zhu, L. 2022. Residual Entropy-based Graph Generative Algorithms. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*, 816–824.

Liu, Y.; Liu, J.; Wan, K.; Qin, Z.; Zhang, Z.; Khoussainov, B.; and Zhu, L. 2021. From local to global norm emergence: dissolving self-reinforcing substructures with incremental social instruments. In *International Conference on Machine Learning*, 6871–6881. PMLR.

Liu, Y.; Liu, J.; Zhang, Z.; Zhu, L.; and Li, A. 2019. REM: From Structural Entropy To Community Structure Deception. In *NeurIPS*.

Luo, D.; Cheng, W.; Yu, W.; Zong, B.; Ni, J.; Chen, H.; and Zhang, X. 2021. Learning to drop: Robust graph neural network via topological denoising. In *Proceedings of the 14th ACM international conference on web search and data mining*, 779–787.

Mavromatis, C.; and Karypis, G. 2020. Graph InfoClust: Leveraging cluster-level node information for unsupervised graph representation learning. *PA-KDD*.

Mavromatis, C.; and Karypis, G. 2021. Graph InfoClust: Maximizing Coarse-Grain Mutual Information in Graphs. In *KDD*.

Pan, S.; Hu, R.; Fung, S.-f.; Long, G.; Jiang, J.; and Zhang, C. 2019. Learning graph embedding with adversarial training methods. *IEEE transactions on cybernetics*.

Pan, S.; Hu, R.; Long, G.; Jiang, J.; Yao, L.; and Zhang, C. 2018. Adversarially Regularized Graph Autoencoder for Graph Embedding. In *IJCAI*.

Perozzi, B.; Al-Rfou, R.; and Skiena, S. 2014. Deepwalk: Online learning of social representations. In *KDD*.

Rosvall, M.; Axelsson, D.; and Bergstrom, C. T. 2009. The map equation. *The European Physical Journal Special Topics*.

Su, W.; Yuan, Y.; and Zhu, M. 2015. A relationship between the average precision and the area under the ROC curve. In *ICTIR*.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2018. Graph Attention Networks. In *ICLR (Poster)*.

Veličković, P.; Fedus, W.; Hamilton, W. L.; Liò, P.; Bengio, Y.; and Hjelm, R. D. 2019. Deep Graph Infomax. *ICLR (Poster)*.

Wan, G.; and Kokel, H. 2021. Graph Sparsification via Meta-Learning. *DLG@ AAAI*.

Wang, C.; Han, B.; Pan, S.; Jiang, J.; Niu, G.; and Long, G. 2020. Cross-Graph: Robust and Unsupervised Embedding for Attributed Graphs with Corrupted Structure. In *ICDM*.

Wang, C.; Pan, S.; Hu, R.; Long, G.; Jiang, J.; and Zhang, C. 2019a. Attributed Graph Clustering: A Deep Attentional Embedding Approach. In *IJCAI*.

Wang, C.; Pan, S.; Long, G.; Zhu, X.; and Jiang, J. 2017. MGAE: Marginalized Graph Autoencoder for Graph Clustering. In *CIKM*.

Wang, H.; Wang, J.; Wang, J.; Zhao, M.; Zhang, W.; Zhang, F.; Li, W.; Xie, X.; and Guo, M. 2019b. Learning graph representation with generative adversarial nets. *IEEE Transactions on Knowledge and Data Engineering*, 33(8): 3090–3103.

Wu, F.; Souza, A.; Zhang, T.; Fifty, C.; Yu, T.; and Weinberger, K. 2019a. Simplifying graph convolutional networks. In *International conference on machine learning*, 6861–6871. PMLR.

Wu, H.; Wang, C.; Tyshetskiy, Y.; Docherty, A.; Lu, K.; and Zhu, L. 2019b. Adversarial examples on graph data: Deep insights into attack and defense. *IJCAI*.

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2019. How Powerful are Graph Neural Networks? In *ICLR*.

Yang, C.; Liu, Z.; Zhao, D.; Sun, M.; and Chang, E. 2015. Network representation learning with rich text information. In *IJCAI*.

Zhang, X.; Liu, H.; Li, Q.; and Wu, X. M. 2019. Attributed graph clustering via adaptive graph convolution. In *IJCAI*.

Zhu, H.; and Koniusz, P. 2020. Simple spectral graph convolution. In *International Conference on Learning Representations*.

Zhu, Y.; Xu, W.; Zhang, J.; Liu, Q.; Wu, S.; and Wang, L. 2021a. Deep Graph Structure Learning for Robust Representations: A Survey. *arXiv preprint arXiv:2103.03036*.

Zhu, Y.; Xu, Y.; Yu, F.; Liu, Q.; Wu, S.; and Wang, L. 2021b. Graph contrastive learning with adaptive augmentation. In *WWW*.

Zügner, D.; and Günnemann, S. 2019. Adversarial attacks on graph neural networks via meta learning. *ICLR*.