Deep Graph Structure Learning for Robust Representations: A Survey

Yanqiao Zhu* , **Weizhi Xu*** , **Jinghao Zhang*** , **Qiang Liu** , **Shu Wu†** and **Liang Wang**
[1]Center for Research on Intelligent Perception and Computing
Institute of Automation, Chinese Academy of Sciences
[2]School of Artificial Intelligence, University of Chinese Academy of Sciences
{yanqiao.zhu,weizhi.xu,jinghao.zhang}@cripac.ia.ac.cn {qiang.liu,shu.wu,wangliang}@nlpr.ia.ac.cn

Abstract

Graph Neural Networks (GNNs) are widely used for analyzing graph-structured data. Most GNN methods are highly sensitive to the quality of graph structures and usually require a perfect graph structure for learning informative embeddings. However, the pervasiveness of noise in graphs necessitates learning robust representations for real-world problems. To improve the robustness of GNN models, many studies have been proposed around the central concept of Graph Structure Learning (GSL), which aims to jointly learn an optimized graph structure and corresponding representations. Towards this end, in the presented survey, we broadly review recent progress of GSL methods for learning robust representations. Specifically, we first formulate a general paradigm of GSL, and then review state-of-the-art methods classified by how they model graph structures, followed by applications that incorporate the idea of GSL in other graph tasks. Finally, we point out some issues in current studies and discuss future directions.

1 Introduction

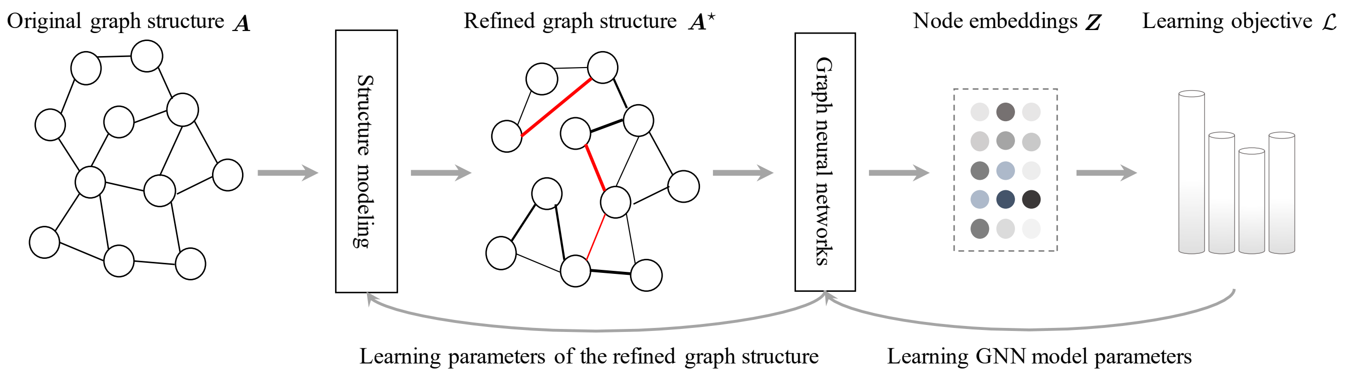
Graphs are ubiquitous in representing objects and their complex interactions. As a powerful tool of analyzing graph-structured data, Graph Neural Networks (GNNs) have been widely employed for graph analytical tasks across a variety of domains, including node classification [Kipf and Welling, 2017], link prediction [Kipf and Welling, 2016; Zhang and Chen, 2018], recommendation [Wu *et al.*, 2019; Yu *et al.*, 2020b; He *et al.*, 2020], information retrieval [Zhang *et al.*, 2021; Yu *et al.*, 2021], etc.

It is known that deep neural nets are error-prone to noise [Szegedy *et al.*, 2014]; noise even exacerbates the quality of representations produced by deep GNN models, which poses a great challenge for applying GNNs to real-world problems and especially in some risk-critical scenarios, for instance medical analysis. Since GNNs compute node embeddings by recursively aggregating information from neighborhoods, such an iterative mechanism has cascading effects — small noise in a graph will be propagated to neighboring nodes, affecting the embeddings of many others. Consider a social network as an example, where nodes correspond to users and edges indicate friend relationship. Fraudulent accounts establish false links to real accounts and they can easily inject wrong information to the entire network, which leads to difficulty in estimating account creditability. In addition, recent research suggests that unnoticeable, deliberate perturbation (aka., adversarial attacks) in graph structure can easily result in wrong predictions [Dai *et al.*, 2018; Zhu *et al.*, 2019; Zhang and Zitnik, 2020]. Therefore, GNNs are highly sensitive to the quality of the given data and usually require a perfect graph structure for learning informative representations [Luo *et al.*, 2021].

A primary concern of improving robustness of GNN models is to produce a denoised graph structure for learning representations [Jin *et al.*, 2020]. Recently, considerable literature has arisen around the central theme of Graph Structure Learning (GSL), which targets at jointly learning an optimized graph structure and corresponding representations [Yu *et al.*, 2020a]. Unfortunately, many GSL techniques have been proposed across separate research communities and they have never been systematically reviewed. In order to establish connections among different researches and ease the understanding of them, in this paper, we survey recent work on deep GSL for robust representations. Specifically, we formulate a general paradigm of GSL, categorize existing work according to the way they model the graph structure, and highlight critical merits of each model. Additionally, as GSL benefits GNNs in other aspects, we discuss applications that incorporate the idea of GSL in other domains such as explainability and graph pooling. To the best of our knowledge, this is the first work that has comprehensively review recent progress of GSL studies for robust representations.

The remaining part of this survey is structured as follows. In Section 2, we introduce preliminary knowledge of graph structure learning and present a general paradigm for characterizing different studies. Following that, in Section 3, we classify existing work into three categories and closely examine representative work in each category. Section 4 further discusses applications that incorporate the idea of GSL in other domains. Finally, we identify several challenges in existing work and outline some potential research directions in Section 5 and conclude the paper in Section 6.

*These authors contributed equally to this work.
†To whom correspondence should be addressed.



Original graph structure $\boldsymbol{A}$
Structure modeling
Refined graph structure $\boldsymbol{A}^\star$
Graph neural networks
Node embeddings $\boldsymbol{Z}$
Learning objective $\mathcal{L}$
Learning parameters of the refined graph structure
Learning GNN model parameters

Figure 1: A general paradigm of Graph Structure Learning (GSL). GSL methods start with an original graph structure. The graph structure is then refined via a structure modeling module and fed into GNNs, where node embeddings are obtained to compute the learning objective function. The parameters in GNNs and the structure modeling module are updated alternatively (or jointly) until a preset stopping condition is satisfied.

2 Preliminaries

2.1 Problem Formulation

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a graph, where $\mathcal{V} = \{v_i\}_{i=1}^N$ is the vertex set and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the edge set. We denote the node feature matrix as $\boldsymbol{X} \in \mathbb{R}^{N \times F}$, where $\boldsymbol{x}_i \in \mathbb{R}^F$ is the attribute of node v_i. The initial graph structure can be represented in an adjacency matrix $\boldsymbol{A} \in \{0,1\}^{N \times N}$ for binary graphs or $\boldsymbol{A} \in \mathbb{R}^{N \times N}$ for weighted graphs, where $\boldsymbol{A}_{ij} > 0$ indicates $(v_i, v_j) \in \mathcal{E}$. A GNN encoder $f(\boldsymbol{X}, \boldsymbol{A})$, parameterized by $\boldsymbol{\Theta}$, receives the graph structure and node features as input, and produces node embeddings $\boldsymbol{Z} \in \mathbb{R}^{N \times F'}$ in low dimensionality, i.e. $F' \ll F$, for downstream tasks.

Given an input graph $\mathcal{G}$[1], the goal of GSL is to learn a clean adjacency matrix $\boldsymbol{A}^\star$ as well as corresponding node embeddings $\boldsymbol{Z}^\star = f(\boldsymbol{X}, \boldsymbol{A}^\star)$. This survey primarily focuses on the work that *modifies the edge set* for refining the graph structure, involving adding new edges, removing existing edges, and changing edge weights.

2.2 A General Paradigm for Structure Learning

Most existing deep GSL work could be regarded as a *plug-in module* on top of off-the-shelf GNN models that learn representations for graphs. We characterize previous studies with a general paradigm that consists of three components: structure modeling, message propagation, and learning objectives.

Structure modeling. The core of GSL is an encoding function that models the optimal graph structure $\boldsymbol{A}^\star$ represented in edge weights. The encoding function may define the edge weights via pairwise distances or by learnable parameters. In this work, we categorize existing studies into the following three groups:

- **Metric learning approaches**, where edge weights are derived from learning a metric function between pairwise representations.

[1]Note that sometimes the initial graph structure is not available. Then, we may manually construct a proximity k-nearest-neighbor graph (kNN graph) [Chen *et al.*, 2009] or ϵ-nearest-neighborhood graph (ϵNN graph) [Bentley *et al.*, 1977] based on the given attributes.

Table 1: Summary of trainable parameters in different methods.

Methods	Main learnable parameters
Metric learning	Metric function
Probabilistic modeling	Probability distributions
Direct optimization	Adjacency matrix

- **Probabilistic modeling approaches**, which assume the graph is generated via a sampling process from certain distributions and model the probability of sampling edges using learnable parameters.
- **Direct optimization approaches**, which refine the graph structure based on the original adjacency matrix by incorporating various graph priors.

These three groups of methods can also be characterized by main learnable parameters, as summarized in Table 1. Moreover, several studies consider additional unlearnable postprocessing steps, such as hard thresholding, to further regularize the modeled graph structure.

Message propagation. After obtained an optimized graph structure $\boldsymbol{A}^\star$, node features will be propagated to the refined neighborhoods. Using GNN models, each node aggregates the information from neighboring nodes and produces an embedding $\boldsymbol{Z}^\star = f(\boldsymbol{A}^\star, \boldsymbol{X})$. It is worth pointing out that due to the difficulty of optimizing the graph structure, many approaches employ structure modeling and message propagation in an iterative manner until several stopping criteria are satisfied.

Learning objective. In order to train the model with the refined graph structures, existing work typically defines a learning objective containing two parts:

$$\mathcal{L} = \mathcal{L}_{\text{task}}(\boldsymbol{Z}^\star, \boldsymbol{Y}) + \lambda \mathcal{L}_{\text{reg}}(\boldsymbol{A}^\star, \boldsymbol{A}). \tag{1}$$

The first term $\mathcal{L}_{\text{task}}$ refers to a task-specific objective with respect to the ground truth $\boldsymbol{Y}$, such as cross entropy for node classification and Bayesian personalized ranking loss for link prediction. The second term $\mathcal{L}_{\text{reg}}$ regularizes the learned graph structure to meet some prior topology constraints, such as sparsity and low-rank property. $\lambda \in \mathbb{R}$ is a hyperparameter that balances the two terms.

Table 2: Summary of different metric learning methods.

Method	Metric function	Update function	Post-processing
AGCN	Gaussian kernel	Interpolation	—
PG-LEARN	Gaussian kernel	—	kNN graph
GAUG-M	Inner product	Interpolation	kNN graph
GRCN	Inner product	Interpolation	kNN graph
GNN-Guard	Cosine similarity	Interpolation	ϵNN graph
AM-GCN	Cosine similarity	Attention	—
SLAPS	Attentive network	—	kNN graph
HGSL	Attentive network	Attention	ϵNN graph
IDGL	Attentive network	Interpolation	ϵNN graph
GPNL	Attentive network	—	—
Grale	Hybrid similarity	—	kNN graph

3 Algorithm Taxonomy

In the following section, we examine existing representative GSL methods in three categories and elaborate on how these work fit into the above paradigm.

3.1 Metric Learning Approaches

Intuitively, the weight of an edge between two nodes could be represented as a distance measure between two end nodes. Metric learning approaches refine the graph structure by learning a metric function $\phi(\cdot,\cdot)$ of a pair of representations:

$$\widetilde{\boldsymbol{A}}_{ij} = \phi(\boldsymbol{z}_i, \boldsymbol{z}_j), \tag{2}$$

where $\boldsymbol{z}_i$ is the learned embedding of node v_i produced by GNNs. At the beginning, we may also use raw features $\boldsymbol{x}_i$ instead. $\widetilde{\boldsymbol{A}}_{ij}$ denotes the learned edge weight between node v_i and v_j.

Then, the learned edge weights are combined with the original adjacency matrix using an update function $g(\cdot,\cdot)$:

$$\boldsymbol{A}^\star = g(\boldsymbol{A}, \widetilde{\boldsymbol{A}}). \tag{3}$$

Among methods in this category, two update functions are often used: interpolation and attentive combination. The former one combines the original structure $\boldsymbol{A}$ and learned adjacency matrix $\widetilde{\boldsymbol{A}}$ with a hyperparameter α mediating the influence of the learned structure:

$$\boldsymbol{A}^\star = \alpha\widetilde{\boldsymbol{A}} + (1-\alpha)\boldsymbol{A}. \tag{4}$$

The latter employs a channel attention mechanism to fuse the two structures, where g is usually implemented in multilayer perceptrons (MLPs).

Additionally, many methods impose a sparsity regularization, since dense graphs not only bring heavy computational burden but also might contain noise. Hence, it is common to prune edges according to edge weights as a post-processing operation, which results in either a kNN graph (i.e. each node has up to k neighbors) or a ϵNN graph (i.e. edges whose weights are less than ϵ will be discarded).

In general, approaches falling in this category mainly differ in the actual implementation of the functions ϕ and g and post-processing steps. According to different realization of ϕ, we further categorize them into two subgroups: kernel-based and attention-based approaches. We summarize these surveyed work in Table 2.

3.1.1 Kernel-based Approaches

Kernel-based approaches employ traditional kernel functions as the metric ϕ to model edge weights between two nodes.

AGCN [Li *et al.*, 2018] first computes the generalized Mahalanobis distance between each pair of nodes in the latent space. Thereafter, it models the edge weight using a Gaussian kernel given the distance, which can be formulated as

$$\phi(\boldsymbol{z}_i, \boldsymbol{z}_j) = \sqrt{(\boldsymbol{z}_i - \boldsymbol{z}_j)^\top \boldsymbol{M}(\boldsymbol{z}_i - \boldsymbol{z}_j)}, \tag{5}$$

$$\widetilde{\boldsymbol{A}}_{ij} = \exp\left(-\frac{\phi(\boldsymbol{z}_i, \boldsymbol{z}_j)}{2\sigma^2}\right), \tag{6}$$

where $\boldsymbol{M} = \boldsymbol{W}_d\boldsymbol{W}_d^\top$ and $\boldsymbol{W}_d \in \mathbb{R}^{N\times M}$ is a trainable matrix that projects node embeddings into a latent space for measuring the generalized Euclidean distance.

PG-LEARN [Wu *et al.*, 2018] takes a similar approach as AGCN for point-cloud data by utilizing a Gaussian kernel to measure the similarity between nodes. It also introduces a regularizer $\mathcal{L}_{\text{feat}}$ to ensure feature smoothness, formulated by

$$\mathcal{L}_{\text{feat}} = \operatorname{tr}(\boldsymbol{Z}^\top \boldsymbol{L}^\star \boldsymbol{Z}), \tag{7}$$

where $\boldsymbol{L}^\star = \boldsymbol{I} - (\boldsymbol{D}^\star)^{-1/2}\boldsymbol{A}^\star(\boldsymbol{D}^\star)^{-1/2}$ is the normalized graph Laplacian matrix.

Moreover, recent work GRCN [Yu *et al.*, 2020a] and GAUG-M [Zhao *et al.*, 2021b] model edge weights by taking inner product of embeddings of two end nodes with no additional parameters introduced:

$$\widetilde{\boldsymbol{A}} = \sigma(\boldsymbol{Z}\boldsymbol{Z}^\top), \tag{8}$$

where $\sigma(x) = 1/(1+e^{-x})$ is the sigmoid function to normalize edge weights.

Besides, GNN-Guard [Zhang and Zitnik, 2020] and AM-GCN [Wang *et al.*, 2020b] utilize cosine similarity to model the edge weights:

$$\phi(\boldsymbol{z}_i, \boldsymbol{z}_j) = \frac{\boldsymbol{z}_i\boldsymbol{z}_j^\top}{\|\boldsymbol{z}_i\|_2\|\boldsymbol{z}_j\|_2}. \tag{9}$$

Note that AM-GCN takes a different approach to obtain the final node embeddings. It propagates node features over the original graph and combines the embeddings from both the generated feature graph and the original graph via an attention network.

To enhance the expressive capability, Grale [Halcrow *et al.*, 2020] fuses different potentially weak similarity metrics to learn informative and task-specific graphs. It also utilizes locality sensitive hashing [Andoni and Indyk, 2008] to increase scalability to deal with datasets with billions of nodes.

3.1.2 Attentive Approaches

Approaches in this subgroup usually utilize an attention network or more complicated neural networks to capture the interaction among nodes, based on which they generate the new graph topology. It is worth mentioning that although GAT [Veličković *et al.*, 2018] does not explicitly learn a new graph structure, it learns edge weights via a simple attention network, and could be regarded as a special case in this group. GAT assigns different weights to one-hop neighbors and does not create new links in the graph.

GLCN [Jiang *et al.*, 2019] implement a single-layer neural network to represent pairwise relationship between two nodes:

$$\widetilde{\boldsymbol{A}}_{ij} = \frac{\exp(\mathrm{ReLU}(\boldsymbol{w}^\top |\boldsymbol{z}_i - \boldsymbol{z}_j|))}{\sum_{j=1}^{N} \exp(\mathrm{ReLU}(\boldsymbol{w}^\top |\boldsymbol{z}_i - \boldsymbol{z}_j|))}, \quad (10)$$

where $\mathrm{ReLU}(\cdot) = \max(0, \cdot)$ and $\boldsymbol{w}$ is the weight vector.

Follow-up work IDGL [Chen *et al.*, 2020b] constructs the graph via a multi-head self-attention network, which has inductive capability that it does not need to retraining when new nodes are added. Specifically, it leverages weighted cosine similarity as the metric for each head and sums up all H attention scores to compute the final edge weights, which can be formulated as

$$a_{ij}^h = \cos(\boldsymbol{w}_h \odot \boldsymbol{z}_i, \boldsymbol{w}_h \odot \boldsymbol{z}_j), \quad (11)$$

$$\widetilde{\boldsymbol{A}}_{ij} = \frac{1}{H} \sum_{h=1}^{H} a_{ij}^h, \quad (12)$$

where $\boldsymbol{w}_h$ is a trainable parameter for the h-th attention head. In addition, on the learned graph, IDGL imposes constraints of feature smoothness similar to Eq. (7) as well as a structure sparsity regularizer via ℓ_1 norms:

$$\mathcal{L}_{\text{sp}} = \|\boldsymbol{A}^\star\|_1. \quad (13)$$

Recently, HGSL [Zhao *et al.*, 2021a] extends the idea of GSL to heterogeneous graphs. It first designs a feature similarity graph for each edge type via a multi-head attention mechanism like IDGL. Then, it constructs two feature propagation graphs and fuses these two graphs and the original graph altogether in an attentive manner.

Besides simple attention networks, many methods use more complex neural networks to parameterize the metric function. Representative work includes GPNL [Cosmo *et al.*, 2020] and SLAPS [Fatemi *et al.*, 2021]. GPNL employs a MLP with non-linear activation functions to obtain edge weights. SLAPS introduces a novel self-supervised auxiliary task to improve structure learning, which masks some input features (or adds noise to them) and trains a separate denoising Graph AutoEncoder (GAE) to recover the masked (or noisy) features by the learned structure. The intuition behind this auxiliary task is that a graph structure suitable for predicting node features should also benefit predicting node labels. SLAPS trains the denoised GAE as follows: it receives a noisy version $\tilde{\boldsymbol{X}}$ and the learned structure $\boldsymbol{A}^\star$ as input and produces the denoised features $\boldsymbol{X}$. Its objective can be concretely represented as

$$\mathcal{L} = \mathcal{L}_{\text{task}} + \lambda \mathcal{L}_{\text{DAE}}, \quad (14)$$

$$\mathcal{L}_{\text{DAE}} = \ell\left(\boldsymbol{X}_{\text{idx}}, \mathrm{GNN}_{\text{DAE}}(\tilde{\boldsymbol{X}}, \boldsymbol{A}^\star)_{\text{idx}}\right), \quad (15)$$

where idx represents the indices corresponding to elements of $\boldsymbol{X}$ with noise, $\mathcal{L}_{\text{DAE}}$ denotes the reconstruction loss used in the denoising GAE $\mathrm{GNN}_{\text{DAE}}$, and ℓ is a regression loss such as mean square error.

3.2 Probabilistic Modeling Approaches

Previous metric learning methods directly derive edge weights from pairwise relationship. Instead of modifying the graph structure in such a deterministic manner, probabilistic modeling approaches assume the graph is generated via a sampling process from certain distributions and model the probability of sampling edges with learnable parameters. Considering that the graph structure is often represented in a discrete matrix, sampling from a discrete distribution is not differentiable. Thus, one main obstacle to these approaches is how to make these sampling operation differentiable and enabling optimizing with conventional gradient descent methods. Inspired from variational approaches [Kingma and Welling, 2014; Blei *et al.*, 2017], most work adopts reparameterization tricks to allows backpropagation to flow through the sampling process.

The first work LDS-GNN [Franceschi *et al.*, 2019] models the edges between each pair of nodes by sampling from a Bernoulli distribution with learnable parameters and frame GSL as a bilevel programming problem. It proposes a practical algorithm using hypergradient estimation to approximate the solution to the proposed bilevel problem [Franceschi *et al.*, 2018].

NeuralSparse [Zheng *et al.*, 2020] considers the graph sparsification task by removing task-irrelevant edges. It utilizes a deep neural network to parameterize the sparsification process and then selects one neighbor u for node v from a categorical distribution:

$$p_{uv} = \mathrm{MLP}_\phi(\boldsymbol{z}_u, \boldsymbol{z}_v), \quad (16)$$

$$\pi_{uv} = \frac{\exp(p_{uv})}{\sum_{w \in \mathcal{N}_u} \exp(p_{uw})}, \quad (17)$$

where $\mathcal{N}$ is the neighborhood set. Then, it utilizes the Gumbel-Softmax trick [Jang *et al.*, 2017; Maddison *et al.*, 2017] to approximate the categorical distribution:

$$x_{uv} = \frac{\exp((\log(\pi_{uv}) + \epsilon_v)/\tau)}{\sum_{w \in \mathcal{N}_u} \exp((\log(\pi_{uw}) + \epsilon_w)/\tau)}, \quad (18)$$

where $\epsilon_v = -\log(-\log(s))$ is drawn from a Gumbel distribution with noise $s \sim \mathrm{Uniform}(0, 1)$ and τ is a temperature hyperparameter. The above process is repeated without replacement for k times to generate a kNN graph. Similarly, GAUG-O [Zhao *et al.*, 2021b] proposes to utilize GNN instead of MLP to parameterize the categorical distribution.

Following NeuralSparse, PTDNet [Luo *et al.*, 2021] relaxes the kNN restriction and optimizes the sparsification process by imposing the sparsity and low-rank constraints on the learned graph structure. GIB [Wu *et al.*, 2020] treats the attention weights of GAT as the parameters of categorical or Bernoulli distributions to sample the refined graph structure which leverages the Gumbel-Softmax reparameterization trick as well.

DGM [Kazi *et al.*, 2020] makes use of the Gumbel-Top-k trick [Kool *et al.*, 2019] to sample edges from the probability distribution defined by a Gaussian kernel $\pi_{ij} = e^{-t\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}$ and exploits reinforcement learning, rewarding edges involved in a correct classification and penalizing edges leading to misclassification.

Different from the above methods which only consider the distributions of a pristine graph, DenNE [Wang *et al.*, 2020a] explicitly models noise on the graph structure. It proposes a generative model with a graph generator that incorporates

Table 3: Summary of direct optimization methods.

Method	Optimization	Regularization				
		Sparsity	Feature smoothness	Low-rank	Label smoothness	Connectivity
GCN-GT	Alternating optimization				✓	
GLNN	—	✓	✓			
ProGNN	Alternating optimization	✓	✓	✓		
GSML	Meta-gradients					✓

various graph priors and a noise generator that generates graph noise in an adaptive way to reconstruct the pristine graph and noise separately. Then, the two generators are optimized jointly via maximum likelihood estimation with real graphs as observations. The overall loss function can be written as

$$\mathcal{L} = \mathcal{L}_{\text{prox}}(\hat{\boldsymbol{A}}, \boldsymbol{Z}, \boldsymbol{E}) + \mathcal{L}(\boldsymbol{E}) + \mathcal{L}(\boldsymbol{Z}), \tag{19}$$

where $\boldsymbol{Z}, \boldsymbol{E}$ are node embeddings and structural noise respectively. $\mathcal{L}_{\text{prox}}$ is a modification of the structure-preserving objective proposed in DeepWalk, which treats $\hat{\boldsymbol{A}}_{ij}$ as a positive sample if v_j is the context of v_i in a random walk. $\mathcal{L}(\boldsymbol{Z})$ is imposed to regularize representations which enables the model to learn robust representations and $\mathcal{L}(\boldsymbol{E})$ is imposed to constrain the noises to eliminate the influence of noises from node representations.

BGCN [Zhang *et al.*, 2019a] proposes a Bayesian approach which regards the existing graph as a sample from a parametric family of random graphs. Specifically, it utilizes Monte Carlo dropout to sample the learnable parameters in the model for several times on each generated graph. Similarly, vGCN [Elinas *et al.*, 2020] also formulates GSL from a Bayesian perspective that considers a prior Bernoulli distribution (parameterized by the observed graph) along with a GNN-based likelihood. Since computation of the posterior distribution is analytically intractable, vGCN utilizes variational inference [Kingma and Welling, 2014; Blei *et al.*, 2017] to approximately infer the posterior. They define the approximated posterior as Bernoulli distribution and adopt the Gumbel-Softmax trick as well. The posterior parameters are estimated via gradient-based optimization of the evidence lower bound (ELBO), given by

$$\begin{aligned}\mathcal{L}_{\text{ELBO}}(\phi) = \\ \mathbb{E}_{q_\phi(\boldsymbol{A})} \log p_{\boldsymbol{\Theta}}(\boldsymbol{Y} \mid \boldsymbol{X}, \boldsymbol{A}) - \mathrm{KL}(q_\phi(\boldsymbol{A}) \parallel p(\boldsymbol{A})),\end{aligned} \tag{20}$$

where $p_{\boldsymbol{\Theta}}(\boldsymbol{Y} \mid \boldsymbol{X}, \boldsymbol{A})$ is the GNN-based likelihood with GNN parameters $\boldsymbol{\Theta}$ and $\mathrm{KL}(q_\phi(\boldsymbol{A}) \parallel p(\boldsymbol{A}))$ is the KL divergence between the approximated posterior and the prior.

3.3 Direct Optimization Approaches

Last, a group of approaches treat the graph adjacency matrix $\boldsymbol{A}^\star$ as learnable parameters, which are optimized directly along with GNN parameters $\boldsymbol{\Theta}$. Graph priors and optimization algorithms are the core of these direct optimization approaches. In order to obtain a rational graph structure, various graph priors are proposed to regularize adjacency matrix, which have been investigated in previous work such as sparsity (Eq. (13)) and feature smoothness (Eq. (7)). Additionally, jointly optimizing $\boldsymbol{A}^\star$ and $\boldsymbol{\Theta}$ is challenging and various constraints on $\boldsymbol{A}^\star$ such as the non-differentiable ℓ_1-norm and nuclear-norm make the optimization even harder. Therefore, special optimization algorithms are often needed.

Please note that since these methods directly optimize the adjacency matrix, they are inherently transductive and cannot generalize to graphs with unseen nodes. We summarize the reviewed direct optimization approaches in Table 3.

GCN-GT [Yang *et al.*, 2019] considers utilizing the given class labels to simultaneously refine the network topology and learn the GNN parameters. Assuming nodes belonging to the same class should be connected, GCN-GT further regularizes the label to be smooth:

$$\mathcal{L}_{\text{lb}} = \frac{1}{2} \sum_{i,j} \boldsymbol{A}^\star_{ij} \| y_i - y_j \|_2^2, \tag{21}$$

where $y_n \in \{0,1\}^{1\times K}$ is the predicted label of node v_n. When jointly optimizing $\boldsymbol{A}^\star$ and $\boldsymbol{\Theta}$, GCN-GT alternatively updates one at a time while fixing the other.

GLNN [Gao *et al.*, 2020] incorporates the initial graph structure, sparsity, and feature smoothness into a hybrid objective. Additionally, ProGNN [Jin *et al.*, 2020] incorporates a low-rank prior which is implemented using the nuclear norm $\|\boldsymbol{A}^\star\|_*$ and also uses an alternating optimization scheme to iteratively update $\boldsymbol{A}^\star$ and $\boldsymbol{\Theta}$.

Recent work GSML [Wan and Kokel, 2021] assumes the graph should be connected with no isolated nodes. It formulates GSL as a bilevel optimization problem and uses meta-gradients to solve the bilevel optimization, in which inner loop represents classification and the outer loop represents structure learning. The regularizer of GSML tries to decrease the classification error over unlabeled nodes:

$$\boldsymbol{A}^\star = \min_{\boldsymbol{A}\in\Phi(\mathcal{G})} \mathcal{L}_{\text{reg}}(f_{\boldsymbol{\Theta}^\star}(\boldsymbol{A}, \boldsymbol{X}), \boldsymbol{Y}_U), \tag{22}$$

$$\text{s.t.} \quad \boldsymbol{\Theta}^\star = \operatorname*{argmin}_{\boldsymbol{\Theta}} \mathcal{L}_{\text{train}}(f_{\boldsymbol{\Theta}}(\boldsymbol{A}, \boldsymbol{X}), \boldsymbol{Y}_L), \tag{23}$$

where $\Phi(\mathcal{G})$ specifies an admissible topology space that leads to no singleton nodes, and $\boldsymbol{Y}_U, \boldsymbol{Y}_L$ are the labels of nodes in the test and training set, respectively. Due to a lack of labels of test nodes, $\mathcal{L}_{\text{reg}}$ cannot be directly optimized and therefore, GSML provides several options to approximate it.

4 Applications

In addition to obtain robust representations, GSL benefits GNN models in other aspects, and many graph-based models which admit an imperfect graph structure also inherently consider the problem of structure learning. In this section, we briefly review recent studies that incorporate the idea of GSL for other applications.

Graph pooling generalizes the pooling operation to graph data, in which nodes are organized in several community structures. Therefore, graph pooling can be seen as to learn a *hierarchical graph structure* [Hu *et al.*, 2019] for better learning representations. DiffPool [Ying *et al.*, 2018] generates a coarsened adjacency matrix denoting the connectivity strength between each pair of clusters; SAGPool [Lee *et al.*, 2019] creates node hierarchy by learning to drop nodes via a self-attention network; HGP-SL [Zhang *et al.*, 2019b] drops nodes whose current embeddings are similar to those aggregated from neighbors to generate hierarchical representations.

Another natural application of GSL is to give *explainability* to GNN models, where GSL is usually employed as a post-processing step following conventional graph representation learning. For example, GNNExplainer [Ying *et al.*, 2019] and PGExplainer [Luo *et al.*, 2020] identify compact subgraph structures that are crucial in prediction as the explanation for the GNN models.

Moreover, GSL is also helpful for accelerating the computation of GNN models. Specifically, with GSL techniques, a subset of important nodes are firstly selected. Then, GNN models are employed over the simplified graph structure to reduce the computational burden. Representative methods in this domain include FastGCN [Chen *et al.*, 2018], ASGCN [Huang *et al.*, 2018], and FastGAT [Srinivasa *et al.*, 2020], where the former two models use importance sampling to sample a subset of nodes during training to speed up GNNs and the latter FastGAT speeds up attention-based GNNs by using spectral graph refinement.

In the domain of representation learning on knowledge graphs, SemanticGCN [Wu *et al.*, 2021] learns semantic-paths dynamically and exploits the implicit heterogeneous information from large amounts of graph data. In computer vision field, wDAE-GNN [Gidaris and Komodakis, 2019] creates graphs using cosine similarity of the class features to capture the co-dependencies between different classes for few-shot learning; DGCNN [Wang *et al.*, 2019b] recovers the topology of point-cloud data using GSL, and thus enriches the representation power for both classification and segmentation on point-cloud data.

Recently, the over-smoothing problem and the universality problem [Chien *et al.*, 2021; Yan *et al.*, 2021] in designing GNNs have gained a lot of research interests, where some proposals share the common idea with GSL. For example, DropEdge [Rong *et al.*, 2020] randomly removes a part of edges during training to relieve the over-smoothing problem. AdaEdge [Chen *et al.*, 2020a] firstly proposes a metric called mean average distance, which computes the mean average distance among node representations to measure the smoothness of the graph. Then, it adds this metric into training objective as a regularizer to choose which edges to drop at each iteration. GRAND [Feng *et al.*, 2020] drops nodes according to a pre-defined distribution for several times to obtain several augmented graphs. Then, features propagated on these graphs are encouraged to be consistent to make the final representations more robust.

5 Challenges and Future Directions

Though promising progress has been made, the development of GSL is still at a nascent stage with many challenges remained unsolved. In this section, we point out some critical problems in current studies and outline several research directions worthy for further investigation.

Beyond homogeneity. Most of the proposed work focuses on homogeneous graphs, where nodes and edges are of the same type. However, nodes and edges in real-world graphs are often associated with multiple types [Yang *et al.*, 2021]. Few attempts have been made to learn a clean structure on heterogeneous graphs till now. How to learn graph structure by considering complex relations between nodes remains widely open.

Beyond homophily. A large body of current work models the edge weights by measuring similarity of node embedding pairs, which is based on the homophily assumption where similar nodes are likely to be connected. However, in real world problems, there are many graphs exhibit strong *heterophily*, where connected nodes are dissimilar, e.g., chemical and molecular graphs [Zhu *et al.*, 2020]. There is abundant room for further progress in designing different strategies to learn heterophilous graph structures.

Learning structure in the absence of attributes. Most existing work modifies the graph structure based on node embeddings, where rich attributes are indispensable. However, real-world benchmarks carry sparse initial attributes or even do not carry features at all, e.g., in sequential recommendation only user-item interactions are available [Wang *et al.*, 2019a]. Thereby, the performance of existing methods may be degraded under such scenario. Further work is required to establish the viability of learning a reasonable graph structure when lacking attributes.

Increasing scalability. Most of existing work involves the computation of pairwise similarity of node embeddings, which suffers from high computational complexity and thereby hinders the applicability of large-scale datasets [Chen *et al.*, 2020b]. Further research should be undertaken to increase scalability of GSL methods.

Towards task-agnostic structure learning. Existing work most requires task-relevant supervision signal for training. However, it is often time-consuming to obtain high-quality labels and transferrable, task-agnostic knowledge is preferred [Jing and Tian, 2020]. Recently, self-supervised learning on graphs [You *et al.*, 2020; Zhu *et al.*, 2021] have been developed to tackle such problem, but how to combine those methods with GSL remains under explored.

6 Concluding Remarks

In this paper, we have broadly reviewed previous work of GSL for learning robust representations. We first elaborate the concept of GSL and formulate a general paradigm for structure learning. Then, we categorize recent work into three groups: metric learning, probabilistic modeling, and direct optimization approaches. We also summarize key characteristics in each work and demonstrate their similarities and differences. Furthermore, we also discuss the connections of related work in other graph-based tasks with GSL. Finally, we outline challenges in current studies and point out directions for further research. We envision this survey to be helpful in expanding our understanding of GSL and guiding future development of GSL algorithms.

References

[Andoni and Indyk, 2008] Alexandr Andoni and Piotr Indyk. Near-Optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions. *Commum. ACM*, 2008.

[Bentley *et al.*, 1977] Jon L. Bentley, Donald F. Stanat, and E. Hollins Williams Jr. The Complexity of Finding Fixed-Radius Near Neighbors. *Inf. Process. Lett.*, 1977.

[Blei *et al.*, 2017] David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. Variational Inference: A Review for Statisticians. *Journal of the American Statistical Association*, 2017.

[Chen *et al.*, 2009] Jie Chen, Haw-ren Fang, and Yousef Saad. Fast Approximate kNN Graph Construction for High Dimensional Data via Recursive Lanczos Bisection. *JMLR*, 2009.

[Chen *et al.*, 2018] Jie Chen, Tengfei Ma, and Cao Xiao. FastGCN: Fast Learning with Graph Convolutional Networks via Importance Sampling. In *ICLR*, 2018.

[Chen *et al.*, 2020a] Deli Chen, Yankai Lin, Wei Li, Peng Li, Jie Zhou, and Xu Sun. Measuring and Relieving the Over-Smoothing Problem for Graph Neural Networks from the Topological View. In *AAAI*, 2020.

[Chen *et al.*, 2020b] Yu Chen, Lingfei Wu, and Mohammed J. Zaki. Iterative Deep Graph Learning for Graph Neural Networks: Better and Robust Node Embeddings. In *NeurIPS*, 2020.

[Chien *et al.*, 2021] Eli Chien, Jianhao Peng, Pan Li, and Olgica Milenkovic. Adaptive Universal Generalized PageRank Graph Neural Network. In *ICLR*, 2021.

[Cosmo *et al.*, 2020] Luca Cosmo, Anees Kazi, Seyed-Ahmad Ahmadi, Nassir Navab, and Michael Bronstein. Latent Patient Network Learning for Automatic Diagnosis. In *MICCAI*, 2020.

[Dai *et al.*, 2018] Hanjun Dai, Hui Li, Tian Tian, Xin Huang, Lin Wang, Jun Zhu, and Le Song. Adversarial Attack on Graph Structured Data. In *ICML*, 2018.

[Elinas *et al.*, 2020] Pantelis Elinas, Edwin V. Bonilla, and Louis C. Tiao. Variational Inference for Graph Convolutional Networks in the Absence of Graph Data and Adversarial Settings. In *NeurIPS*, 2020.

[Fatemi *et al.*, 2021] Bahare Fatemi, Layla El Asri, and Seyed Mehran Kazemi. SLAPS: Self-Supervision Improves Structure Learning for Graph Neural Networks. *arXiv.org*, February 2021.

[Feng *et al.*, 2020] Wenzheng Feng, Jie Zhang, Yuxiao Dong, Yu Han, Huanbo Luan, Qian Xu, Qiang Yang, Evgeny Kharlamov, and Jie Tang. Graph Random Neural Networks for Semi-Supervised Learning on Graphs. In *NeurIPS*, 2020.

[Franceschi *et al.*, 2018] Luca Franceschi, Paolo Frasconi, Saverio Salzo, Riccardo Grazzi, and Massimiliano Pontil. Bilevel Programming for Hyperparameter Optimization and Meta-Learning. In *ICML*, 2018.

[Franceschi *et al.*, 2019] Luca Franceschi, Mathias Niepert, Massimiliano Pontil, and Xiao He. Learning Discrete Structures for Graph Neural Networks. In *ICML*, 2019.

[Gao *et al.*, 2020] Xiang Gao, Wei Hu, and Zongming Guo. Exploring Structure-Adaptive Graph Learning for Robust Semi-Supervised Classification. In *ICME*, 2020.

[Gidaris and Komodakis, 2019] Spyros Gidaris and Nikos Komodakis. Generating Classification Weights With GNN Denoising Autoencoders for Few-Shot Learning. In *CVPR*, 2019.

[Halcrow *et al.*, 2020] Jonathan Halcrow, Alexandru Mosoi, Sam Ruth, and Bryan Perozzi. Grale: Designing Networks for Graph Learning. In *KDD*, 2020.

[He *et al.*, 2020] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yong-Dong Zhang, and Meng Wang. LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation. In *SIGIR*, 2020.

[Hu *et al.*, 2019] Fenyu Hu, Yanqiao Zhu, Shu Wu, Liang Wang, and Tieniu Tan. Hierarchical Graph Convolutional Networks for Semi-supervised Node Classification. In *IJCAI*, 2019.

[Huang *et al.*, 2018] Wenbing Huang, Tong Zhang, Yu Rong, and Junzhou Huang. Adaptive Sampling Towards Fast Graph Representation Learning. In *NeurIPS*, 2018.

[Jang *et al.*, 2017] Eric Jang, Shixiang Gu, and Ben Poole. Categorical Reparameterization with Gumbel-Softmax. In *ICLR*, 2017.

[Jiang *et al.*, 2019] Bo Jiang, Ziyan Zhang, Doudou Lin, and Jin Tang. Semi-supervised Learning with Graph Learning-Convolutional Networks. In *CVPR*, 2019.

[Jin *et al.*, 2020] Wei Jin, Yao Ma, Xiaorui Liu, Xianfeng Tang, Suhang Wang, and Jiliang Tang. Graph Structure Learning for Robust Graph Neural Networks. In *KDD*, 2020.

[Jing and Tian, 2020] Longlong Jing and Yingli Tian. Self-supervised Visual Feature Learning with Deep Neural Networks: A Survey. *TPAMI*, 2020.

[Kazi *et al.*, 2020] Anees Kazi, Luca Cosmo, Nassir Navab, and Michael Bronstein. Differentiable Graph Module (DGM) for Graph Convolutional Networks. *arXiv.org*, February 2020.

[Kingma and Welling, 2014] Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. In *ICLR*, 2014.

[Kipf and Welling, 2016] Thomas N. Kipf and Max Welling. Variational Graph Auto-Encoders. In *BDL@NIPS*, 2016.

[Kipf and Welling, 2017] Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. In *ICLR*, 2017.

[Kool *et al.*, 2019] Wouter Kool, Herke van Hoof, and Max Welling. Stochastic Beams and Where To Find Them: The Gumbel-Top-k Trick for Sampling Sequences Without Replacement. In *ICML*, 2019.

[Lee *et al.*, 2019] Junhyun Lee, Inyeop Lee, and Jaewoo Kang. Self-Attention Graph Pooling. In *ICML*, 2019.

[Li *et al.*, 2018] Ruoyu Li, Sheng Wang, Feiyun Zhu, and Junzhou Huang. Adaptive Graph Convolutional Neural Networks. In *AAAI*, 2018.

[Luo *et al.*, 2020] Dongsheng Luo, Wei Cheng, Dongkuan Xu, Wenchao Yu, Bo Zong, Haifeng Chen, and Xiang Zhang. Parameterized Explainer for Graph Neural Network. In *NeurIPS*, 2020.

[Luo *et al.*, 2021] Dongsheng Luo, Wei Cheng, Wenchao Yu, Bo Zong, Jingchao Ni, Haifeng Chen, and Xiang Zhang. Learning to Drop: Robust Graph Neural Network via Topological Denoising. In *WSDM*, 2021.

[Maddison *et al.*, 2017] Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables. In *ICLR*, 2017.

[Rong *et al.*, 2020] Yu Rong, Wenbing Huang, Tingyang Xu, and Junzhou Huang. DropEdge: Towards Deep Graph Convolutional Networks on Node Classification. In *ICLR*, 2020.

[Srinivasa *et al.*, 2020] Rakshith S. Srinivasa, Cao Xiao, Lucas Glass, Justin Romberg, and Jimeng Sun. Fast Graph Attention Networks Using Effective Resistance Based Graph Sparsification. *arXiv.org*, June 2020.

[Szegedy *et al.*, 2014] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing Properties of Neural Networks. In *ICLR*, 2014.

[Veličković *et al.*, 2018] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. In *ICLR*, 2018.

[Wan and Kokel, 2021] Guihong Wan and Harsha Kokel. Graph Sparsification via Meta-Learning. In *DLG@AAAI*, 2021.

[Wang *et al.*, 2019a] Shoujin Wang, Liang Hu, Yan Wang, Longbing Cao, Quan Z. Sheng, and Mehmet A. Orgun. Sequential Recommender Systems: Challenges, Progress and Prospects. In *IJCAI*, 2019.

[Wang *et al.*, 2019b] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M Solomon. Dynamic Graph CNN for Learning on Point Clouds. *TOG*, 2019.

[Wang *et al.*, 2020a] Junshan Wang, Ziyao Li, Qingqing Long, Weiyu Zhang, Guojie Song, and Chuan Shi. Learning Node Representations from Noisy Graph Structures. In *ICDM*, 2020.

[Wang *et al.*, 2020b] Xiao Wang, Meiqi Zhu, Deyu Bo, Peng Cui, Chuan Shi, and Jian Pei. AM-GCN: Adaptive Multi-channel Graph Convolutional Networks. In *KDD*, 2020.

[Wu *et al.*, 2018] Xuan Wu, Lingxiao Zhao, and Leman Akoglu. A Quest for Structure: Jointly Learning the Graph Structure and Semi-Supervised Classification. In *CIKM*, 2018.

[Wu *et al.*, 2019] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. Session-based Recommendation with Graph Neural Networks. In *AAAI*, 2019.

[Wu *et al.*, 2020] Tailin Wu, Hongyu Ren, Pan Li, and Jure Leskovec. Graph Information Bottleneck. In *NeurIPS*, 2020.

[Wu *et al.*, 2021] Likang Wu, Zhi Li, Hongke Zhao, Qi Liu, Jun Wang, Mengdi Zhang, and Enhong Chen. Learning the Implicit Semantic Representation on Graph-Structured Data. In *DASFAA*, 2021.

[Yan *et al.*, 2021] Yujun Yan, Milad Hashemi, Kevin Swersky, Yaoqing Yang, and Danai Koutra. Two Sides of the Same Coin: Heterophily and Oversmoothing in Graph Convolutional Neural Networks. *arXiv.org*, February 2021.

[Yang *et al.*, 2019] Liang Yang, Zesheng Kang, Xiaochun Cao, Jin Di, Bo Yang, and Yuanfang Guo. Topology Optimization based Graph Convolutional Network. In *IJCAI*, 2019.

[Yang *et al.*, 2021] Carl Yang, Yuxin Xiao, Yu Zhang, Yizhou Sun, and Jiawei Han. Heterogeneous Network Representation Learning: A Unified Framework with Survey and Benchmark. *TKDE*, 2021.

[Ying *et al.*, 2018] Rex Ying, Jiaxuan You, Christopher Morris, Xiang Ren, William L. Hamilton, and Jure Leskovec. Hierarchical Graph Representation Learning with Differentiable Pooling. In *NeurIPS*, 2018.

[Ying *et al.*, 2019] Rex Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. GNN Explainer: A Tool for Post-hoc Explanation of Graph Neural Networks. In *NeurIPS*, 2019.

[You *et al.*, 2020] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. Graph Contrastive Learning with Augmentations. In *NeurIPS*, 2020.

[Yu *et al.*, 2020a] Donghan Yu, Ruohong Zhang, Zhengbao Jiang, Yuexin Wu, and Yiming Yang. Graph-Revised Convolutional Network. In *ECML PKDD*, 2020.

[Yu *et al.*, 2020b] Feng Yu, Yanqiao Zhu, Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. TAGNN: Target Attentive Graph Neural Networks for Session-based Recommendation. In *SIGIR*, 2020.

[Yu *et al.*, 2021] Xueli Yu, Weizhi Xu, Zeyu Cui, Shu Wu, and Liang Wang. Graph-based Hierarchical Relevance Matching Signals for Ad-hoc Retrieval. In *WWW*, 2021.

[Zhang and Chen, 2018] Muhan Zhang and Yixin Chen. Link Prediction Based on Graph Neural Networks. In *NeurIPS*, 2018.

[Zhang and Zitnik, 2020] Xiang Zhang and Marinka Zitnik. GNNGuard: Defending Graph Neural Networks against Adversarial Attacks. In *NeurIPS*, 2020.

[Zhang *et al.*, 2019a] Yingxue Zhang, Soumyasundar Pal, Mark Coates, and Deniz Üstebay. Bayesian Graph Convolutional Neural Networks for Semi-Supervised Classification. In *AAAI*, 2019.

[Zhang *et al.*, 2019b] Zhen Zhang, Jiajun Bu, Martin Ester, Jianfeng Zhang, Chengwei Yao, Zhi Yu, and Can Wang. Hierarchical Graph Pooling with Structure Learning. *arXiv.org*, November 2019.

[Zhang *et al.*, 2021] Yufeng Zhang, Jinghao Zhang, Zeyu Cui, Shu Wu, and Liang Wang. A Graph-based Relevance Matching Model for Ad-hoc Retrieval. In *AAAI*, 2021.

[Zhao *et al.*, 2021a] Jianan Zhao, Xiao Wang, Chuan Shi, Binbin Hu, Guojie Song, and Yanfang Ye. Heterogeneous Graph Structure Learning for Graph Neural Networks. In *AAAI*, 2021.

[Zhao *et al.*, 2021b] Tong Zhao, Yozen Liu, Leonardo Neves, Oliver Woodford, Meng Jiang, and Neil Shah. Data Augmentation for Graph Neural Networks. In *AAAI*, 2021.

[Zheng *et al.*, 2020] Cheng Zheng, Bo Zong, Wei Cheng, Dongjin Song, Jingchao Ni, Wenchao Yu, Haifeng Chen, and Wei Wang. Robust Graph Representation Learning via Neural Sparsification. In *ICML*, 2020.

[Zhu *et al.*, 2019] Dingyuan Zhu, Ziwei Zhang, Peng Cui, and Wenwu Zhu. Robust Graph Convolutional Networks Against Adversarial Attacks. In *KDD*, 2019.

[Zhu *et al.*, 2020] Jiong Zhu, Yujun Yan, Lingxiao Zhao, Mark Heimann, Leman Akoglu, and Danai Koutra. Beyond Homophily in Graph Neural Networks: Current Limitations and Effective Designs. In *NeurIPS*, 2020.

[Zhu *et al.*, 2021] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. Graph Contrastive Learning with Adaptive Augmentation. In *WWW*, 2021.