Graph Data Augmentation for Graph Machine Learning: A Survey

Tong Zhao[1,4], Wei Jin[2], Yozen Liu[1], Yingheng Wang[3], Gang Liu[4],
Stephan Günnemann[5], Neil Shah[1], Meng Jiang[4]
[1]Snap Inc., [2]Michigan State University, [3]Cornell University,
[4]University of Notre Dame, [5]Technical University of Munich
[1]{tzhao,yliu2,nshah}@snap.com,[2]jinwei2@msu.edu,[3]yw2349@cornell.edu,
[4]{gliu7,mjiang2}@nd.edu,[5]guennemann@in.tum.de

Abstract

Data augmentation has recently seen increased interest in graph machine learning given its demonstrated ability to improve model performance and generalization by added training data. Despite this recent surge, the area is still relatively under-explored, due to the challenges brought by complex, non-Euclidean structure of graph data, which limits the direct analogizing of traditional augmentation operations on other types of image, video, or text data. Our work aims to give a necessary and timely overview of existing graph data augmentation methods; notably, we present a comprehensive and systematic survey of graph data augmentation approaches, summarizing the literature in a structured manner. We first introduce three different taxonomies for categorizing graph data augmentation methods from the data, task, and learning perspectives, respectively. Next, we introduce recent advances in graph data augmentation, differentiated by their methodologies and applications. We conclude by outlining currently unsolved challenges and directions for future research. Overall, our work aims to clarify the landscape of existing literature in graph data augmentation and motivates additional work in this area, providing a helpful resource for researchers and practitioners in the broader graph machine learning domain. Additionally, we provide a continuously updated reading list at https://github.com/zhao-tong/graph-data-augmentation-papers.

1 Introduction

Data driven inference has received a significant boost in generalization capability and performance improvement in recent years from data augmentation (DA) techniques. DA techniques increase the amount of training data by creating plausible variations of existing data without additional ground-truth labeling efforts, and have seen widespread adoption in fields such as computer vision (CV) [15] and natural language processing (NLP) [26]. These techniques allow machine learning models to learn to generalize across those variations and attend to signal over noise. In recent years, with the rapid development of graph machine learning (GML) methods such as graph neural networks (GNNs) [57, 38], studies have shown that the effectiveness of GML approaches also largely depends on the data quality. Given the dependent nature of graph data and the message-passing design of most GNNs, GML faces unique challenges such as: structural data sparsity brought by power-law degree distributions in most graphs, noisy and even erroneous topology brought by imperfect construction of the graph structure from raw data under other formats, low quality and incomplete node attributes, adversarial attacks on structure and attributes, lack of labelled data due to costly human annotations, and over-smoothing caused by the message passing design in GNNs. As DA allows researchers to alleviate such challenges from a data perspective, there has been increased interest and demand for such techniques on graph data [140], and there has been a growing number of works on graph data augmentation (GDA).

With the irregular and non-Euclidean structure of graph data, it is non-trivial to directly analogize DA techniques from CV and NLP to the graph domain, except for the most basic operations such as random masking/dropping/cropping. To better promote the effectiveness of GML approaches and alleviate the unique challenges in GML, recent literature designed graph-specific augmentation techniques following methodologies such as graph structure learning, graph adversarial training, graph rationalization, etc. Creating a unified taxonomy for all GDA techniques is not intuitive as they can be categorized under different facets. For example, taking the data modelity that the augmentation methods work on, they can be separated into structure augmentations, feature augmentations, and label augmentations. On the other hand, the focusing downstream tasks (i.e., node-level, edge-level, and graph-level tasks) can also categorize the GDA techniques. Moreover, the GDA methods can also be separated by whether the methods involves learning during the augmentation process. That is, whether they are rule-based approaches or learned approaches.

This paper aims to sensitize the GML community towards this growing area of work, as DA has already drawn much attention in CV and NLP [15, 26]. As interest and work on this topic continue to increase, this is an opportune time for a comprehensive work to (i) introduce background and motivation of GDA, (ii) give a bird's-eye view of existing GDA techniques under different taxonomies, (iii) introduce representative GDA techniques with their usage and applications, and (iv) identify key challenges to effectively motivate and orient interest in this area. We hope this survey can serve as a guide for researchers and practitioners who are new to or interested in studying this topic, and also inspire future research in this area.

The text is structured as follows: Section 2 gives background and motivation on GNNs and GDA. It defines GDA and motivates its use in GML tasks. Section 3 categorizes GDA techniques based on three different taxonomies: the operated graph data, the downstream tasks, and whether the method involves learning. Section 4 describes rule-based GDA techniques for GML – which we partition into Data Removal (Section 4.1), Data Addition (Section 4.2), and Data Manipulation (Section 4.3) focuses. Similarly, Section 5 introduces learned GDA techniques, which are further categorized by their methodologies: Graph Structure Learning (Section 5.1), Graph Adversarial Training (Section 5.2), Graph Rationalization (Section 5.3), and Automated Augmentation (Section 5.4). Section 6 introduces GDA techniques that are used under three different self-supervised learning objectives: Contrastive Learning (Section 6.1), Non-contrastive Learning (Section 6.2), and Consistency Training (Section 6.3). Finally, Section 7 discusses challenges and future directions for GDA.

2 Preliminaries

2.1 Notations

Let $G = (\mathcal{V}, \mathcal{E})$ be a graph of N nodes, where $\mathcal{V} = \{v_1, v_2, \ldots, v_N\}$ is the set of N nodes and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of links. We denote the adjacency matrix as $\mathbf{A} \in \{0, 1\}^{N \times N}$, where $A_{i,j} = 1$ indicates nodes v_i and v_j are connected and vice versa. We denote the node feature matrix as $\mathbf{X} \in \mathbb{R}^{N \times F}$, where F is the number of raw node features and $\boldsymbol{x}_i$ indicates the feature vector of node v_i (the i-th row of $\mathbf{X}$). We use $\boldsymbol{y}$ to denote the label each sample, which can be node, edge, or graph depending on the task. We use symbol with tilde to denote the data generated by GDA methods. For example, $\tilde{\mathbf{A}}$ for the augmented adjacency matrix, $\tilde{\boldsymbol{x}}_i$ for the augmented feature vector of node v_i, etc.

2.2 Graph Neural Networks

Graph neural networks (GNNs) enjoy widespread use in modern graph-based machine learning due to their flexibility to incorporate node features, custom aggregations, and inductive operation, unlike earlier works which were based on embedding lookups [87, 34]. Following the initial idea of convolution based on spectral graph theory [6], many spectral GNNs have since been developed and improved by [19, 57, 67, 59, 80]. As spectral GNNs generally operate (expensively) on the full adjacency, spatial-based methods which perform

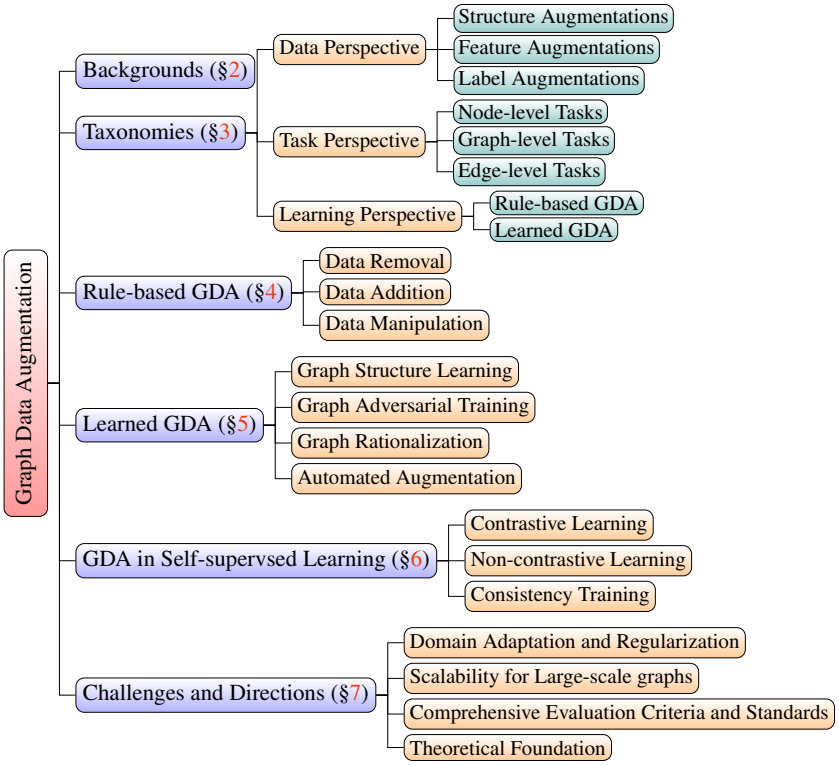

Figure 1: Structure of this survey.

graph convolution with neighborhood aggregation became prominent [38, 107], owing to their scalability and flexibility [128, 121].

Generally, the generic formulation of message passing-based GNNs can be defined by an aggregation function (AGGREGATE) and an update function (UPDATE). In each layer, AGGREGATE aggregates the embeddings from previous layer for each node from all its neighbors, and UPDATE updates each node's embedding by combining its own previous embedding and the aggregated neighbor embeddings [38]. Specifically,

$$\begin{aligned} \boldsymbol{h}_{\mathcal{N}(v)}^{l} &= \text{AGGREGATE}(\{\boldsymbol{h}_{u}^{l-1} | u \in \mathcal{N}(v)\}), \\ \boldsymbol{h}_{v}^{l} &= \text{UPDATE}(\boldsymbol{h}_{v}^{l-1}, \boldsymbol{h}_{\mathcal{N}(v)}^{l}), \end{aligned} \tag{48}$$

where $\boldsymbol{h}_{v}^{l}$ denotes the representation of node v at the l-th layer, and $\mathcal{N}(v)$ denotes the set of node v's neighbors.

In implementation, GNNs can usually be implemented with (sparse) matrix multiplications. Without the loss of generality, here we take the most commonly used Graph Convolutional Network (GCN) [57] as an example. One layer of GCN is defined as

$$\mathbf{H}^{l} = \sigma(\mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}\mathbf{W}^{l}\mathbf{H}^{l-1}), \tag{49}$$

where $\mathbf{D}$ is the diagonal degree matrix s.t. $\mathbf{D}_{i,i} = \sum_{j} \mathbf{A}_{i,j}$ (assuming $\mathbf{A}$ contains self-loops), $\sigma(\cdot)$ is the nonlinear activation function such as ReLU, and $\mathbf{W}^{l}$ denotes the learnable weight matrix at the l-th GNN layer. Furthermore, we use $g_{\Theta}(\cdot)$ to denote the mapping function of the whole GNN model parameterized by Θ.

2.3 Graph Data Augmentation

The DA area encompasses techniques of increasing/generating training data without directly collecting or labeling more data. Most DA techniques either add slightly modified copies of existing data, or generate synthetic data

based on existing data. The augmented data act as a regularizer and reduce overfitting when training data-driven models [93]. DA techniques has been commonly used in CV [15] and NLP [26], where augmentation operations such as cropping, flipping, and back-translation are usually used in machine learning model training.

In GML, in contrast to regular and Euclidean data such as grids (e.g., images) and sequences (e.g., sentences), the graph structure is encoded by node connectivity, which is non-Euclidean and irregular. Most structured augmentation operations used frequently in CV and NLP cannot be easily analogized to graph data. Therefore, how to design effective augmentations of graph data is less obvious. For example, the data objects for node-level and edge-level tasks are inter-connected and non-i.i.d, meaning that GDA techniques typically modify the entire dataset (graph) instead of a specific data object (nodes or edge) in isolation. Generally, a GDA method can be defined as a transformation function $f : G \rightarrow \tilde{G}$, where the the transformation function f can be either rule-based or learnable, and the augmented graph $\tilde{G}$ contains the augmented adjacency matrix $\tilde{\mathbf{A}}$ and node feature matrix $\tilde{\mathbf{X}}$ (and optionally augmented edge features, node or graph labels). Moreover, the augmentation function f is not necessarily deterministic. That is, the same f may generate multiple different versions of the augmented graph $\tilde{G}$, and the model may use one or multiple of these augmentations as required for training.

2.4 Motivation: Why Augment Graphs?

Graphs are often utilized to model or represent an underlying process of relationships or affinities; for example, "which individuals are friends with one another?" or "which movies do individuals like?" In some cases, these relationships are strictly defined and known, e.g. researchers jointly co-authoring articles, or atoms interacting in a chemical compound. However, in many other cases, an "observed" graph may be misaligned with the true process it intends to model for a variety of reasons [5]. In some cases, like in social interaction graphs, noise may be inadvertently or adversarially introduced by spammers who pollute underlying data about authentic interactions with inauthentic ones for nefarious purposes [90, 65]. In other cases, noise may be inherently created by limited or partial observation (e.g. a movie recommendation system never recommending a certain genre of movies to a group of users) caused by privacy reasons [14, 22], biased recommendation policies [55, 143], or other reasons. Noise can also occur by measurement or thresholding errors (e.g. discretizing continuous signals between brain voxels into discrete ones) [30], or human errors (e.g. a person forgetting to add a known contact to their phone's contact-book). All of these scenarios can introduce gaps between an intended and observed graph. Moreover, even if all relationships a graph intends to capture are observed properly, there is no guarantee that the graph is a particularly useful [5] for a particular downstream learning task, especially when utilized in a GML context, e.g. a graph connecting individuals by similar heights may be unhelpful in regressing income.

GDA methods offer an attractive solution in denoising, imputing, and generally enhancing graph structure to align better with an intended modeling processes, or objectives of a target learning task [140]. Adding or removing nodes and edges can help connect or disconnect a graph to facilitate its use towards targeted objectives. Moreover, utilizing heuristic graph modification strategies to increase model exposure in training may lead to better generalizing, more robust and higher performance models [114, 62, 147]. Both learned and rule-based DA techniques have shown immense potential in other domains like tabular ML (e.g. oversampling [2] and SMOTE [7]), CV (e.g. rotations, flips and translations of images [93] and random erasure [146]) and NLP (e.g. synonym replacement and random token additions/deletions [118] and back-translation [89]); however, as aforementioned, these techniques usually lack clear analogs in the graph domain due to unclear correspondence of their label-preserving transforms. This lack of clarity motivates work into understanding the limitations of graphs, suitable designs for augmentation techniques, and their breadth of impact.

3 Taxonomies

In this section, we introduce three different taxonomies that can be used to categorize GDA techniques. They come from different perspectives of data, task, and learnability, respectively. As these taxonomies are orthogonal to each other, and each of them can in some way categorize all GDA methods, we will only focus on one taxonomy (rule-based vs. learned augmentation) for the later sections.

3.1 Operated Data Modality

As GDA methods all operate on graph data, they can naturally be categorized by the data modality that they aim to manipulate. Therefore, one intuitive taxonomy for GDA methods would be classifying them into one or more of three categories: structure, feature, and label augmentations.

Structure Augmentations are the GDA operations that modify the graph connectivity via adding/removing edges or adding/removing nodes from the graph. The modifications can be either deterministic (e.g., GDC [60] and GAug-M [140] both modify the graph structure and used the modifed graph for training/inferencing) or stochastic (e.g., DropEdge[88] and DropNode [27] randomly drop edges/nodes from the observed training graph). **Feature Augmentations** are the GDA operations that modify or create raw node features. For example, You et al. [129] used Attribute Masking that randomly masked off node features; FLAG [62] augments node features with gradient-based adversarial perturbations. It's worth noting that stucture augmentations and feature augmentations are also sometimes combined in some GDA methods. For example, MoCL [99] substitutes subgraphs in molecular graphs with subgraphs of different functional groups. **Label Augmentations** are the GDA operations that involves modifying the labels. For example, Mixup-based methods [39, 37] interpolate existing training examples and assign new label for the generated example. Counterfactual data augmentation methods (e.g., CFLP [143]) generate counterfactual examples with corresponding new labels.

3.2 Downstream Tasks

Another straightforward taxonomy of categorizing GDA methods is by the downstream tasks that they tackle. Generally, most GML methods can be categorized into three high-level task types: **node-level**, **edge-level**, and **graph-level** tasks. Similarly, many GDA methods are designed toward one of these tasks, and cannot be easily generalized to other tasks. For example, CFLP [143] generates counterfactual links as augmented data specifically for training a neural link predictor, and these counterfactual links are useless to other tasks like node classification as they are counterfactual labels on node pairs under specific treatments. Moreover, certain GDA methods that are designed for molecular graphs (e.g., MoCL [99]) are not opeartable on the large graph datasets used in other tasks as they rely on the domain specific substructures of molecular graphs, e.g., functional groups. Nonetheless, the downside of categorizing by downstream tasks is that a fair number of GDA methods were designed more generically for various tasks; for example, DropEdge [88] simply conducts random edge dropping during training, and the method can naturally be applied on most GML methods.

3.3 Rule-based vs. Learned Augmentations

GDA methods can also be categorized by whether the augmentation process involved learning, namely rule-based GDA approaches and learned GDA approaches. More specifically, **rule-based GDA approaches** refer to the non-learnable methods that modify or manipulate the graph data following pre-defined rules, which can be stochastic, deterministic, or mixture of both. A rule-based GDA method can be as simple as randomly removing a given fraction of edges [88] or randomly cropping out part of the graph [129]; it can also be more complicated such as counterfactual augmentation [143] based on similarity matching rules and graph diffusion methods [60] that follows specific diffusion kernels. We also categorize Mixup-based augmentations [39] as rule-based approaches

since they usually only contain one non-learnable parameter (sampled from pre-defined distributions) when generating new data objects by interpolating two existing data objects.

On the other hand, **learned GDA approaches** refer to the augmentation methods that contains learnable parameters in the process of generating augmented examples. The augmentation module can either be trained independently or in an end-to-end style with the downstream classifier or regressor [140]. For example, graph structure learning methods [152, 50, 140] often assume the observed graph data is noisy, incomplete, or entirely missing, so they first try to learn the "clean" graph structure before using it in the training and inference of GNNs. Graph rationalization methods [120, 71] learn subgraphs that are likely to be causally related with the graph labels and use them for augmentation. Automated augmentation methods [144, 79] utilize reinforcement learning agents to learn the optimal augmentation strategy for the given data automatically.

In Sections 4 and 5, we will introduce GDA approaches in more detail based on this separation as it provides better differentiation of the methodologies and improved readability. Table 1 shows a summary of GDA techniques, categorized following this taxonomy and the methods' methodologies.

4 Rule-based Approaches for GDA

Owing to their simplicity and efficiency, rule-based graph data augmentation methods are the most commonly used augmentation techniques in graph machine learning. The rule-based GDA approaches can generally categorized into three categories, where the first category of methods would remove part of the data (e.g., Stochastic Masking) to create new graph data, the second category of methods augments the graph data by generating new graphs or adding components (e.g., Counterfactual Augmentation, Pseudo-labeling), and the third category includes methods that manipulate the data following rules can involve both removing and adding operations (e.g., Diffusion, etc.) In the following subsections, we summarize the representative approaches in each category and also discuss their applications on different tasks and domains.

4.1 Data Removal

Edge Dropping. Edge dropping methods stochastically remove a certain fraction of edges from the graph data. Aiming to alleviate the known over-smoothing problem of GNNs, Rong et al. [88] first proposed DropEdge which randomly dropped a fixed fraction of edges in each training epoch, resembling Dropout [96]. More specifically, at the beginning of each training epoch, the modified adjacency matrix $\tilde{\mathbf{A}}$ is defined by

$$\tilde{\mathbf{A}} = \mathbf{M} \odot \mathbf{A}, \tag{50}$$

where $\mathbf{M} \in \{0,1\}^{N \times N}$ is a binary mask on the adjacency matrix s.t. $M_{i,j} = Bernoulli(\varepsilon)$, $\varepsilon \in (0,1)$ is the drop rate hyper-parameter, and $\odot$ denotes the Hadamard product.

During GNN training, DropEdge adopts a newly sampled $\tilde{\mathbf{A}}$ instead of the original graph structure $\mathbf{A}$ for message passing (e.g., Equation equation 49) in each training epoch. By showing the GNN models different perturbations of the graph in each training epoch, DropEdge improves the model's generalization and shows significant performance improvements on deeper GNNs, indicating that the strategy mitigates over-smoothing. Several other methods [129, 102, 144] also adopt random edge masking in other learning schemes such as self-supervised learning, which conducts the same operation as DropEdge.

Node Dropping. Similar to edge dropping, node dropping methods stochastically remove nodes from the graph. Node dropping is typically implemented in two ways: removing all features of the target nodes from the feature matrix, or removing the target nodes along with all the edges connected with them from the graph structure. Feng et al. [27] proposed DropNode, which follows the first schema. Concurrently, You et al. [129] proposed NodeDropping following the latter.

Table 1: A summary of GDA techniques, categorized by whether they are learned augmentations and their methodologies.

	Methodology	Representative Works	Task Level			Augmented Data		
			Node	Graph	Edge	Structure	Feature	Label
Rule-based GDA	Stochastic Dropping/Masking	DropEdge [88]	✓			✓		
		DropNode [27]		✓			✓	
		NodeDropping [129]		✓		✓		
		Feature Masking [102]	✓				✓	
		Feature Shuffling [108]	✓				✓	
		DropMessage [23]	✓		✓		✓	
		Subgraph Masking [129]		✓		✓	✓	
	Subgraph Cropping/Substituting	GraphCrop [113]		✓		✓		
		M-Evolve [147]		✓		✓		
		MoCL [99]		✓		✓	✓	
	Virtual Node	Graphormer [127]		✓		✓		
		GNN-CM$^+$/CM [45]			✓	✓		
	Mixup	Graph Mixup [117]	✓	✓				✓
		ifMixup [37]		✓		✓	✓	✓
		Graph Transparent [86]		✓		✓	✓	✓
		G-Mixup [39]		✓		✓	✓	✓
	SMOTE	GraphSMOTE [142]	✓				✓	
		GATSMOTE [76]	✓			✓		
		GNN-CL [70]	✓			✓	✓	
	Diffusion	GDA [60]	✓			✓		
	Counterfactual Augmentation	CFLP [143]			✓	✓		✓
	Attribute Augmentation	LA-GNN [75]	✓				✓	
		SR+DR [94]	✓				✓	
	Pseudo-labeling	Label Propagation [149]	✓					✓
		PTA [21]	✓					✓
Learned GDA	Graph Structure Learning	GAug [140]	✓			✓		
		GLCN [47]	✓			✓		
		LDS [28]	✓			✓		
		ProGNN [50]	✓			✓		
		Eland [141]	✓			✓		
	Graph Adversarial Training	RobustTraining [125]	✓			✓		
		AdvT [18]	✓		✓	✓		
		FLAG [63]	✓	✓	✓		✓	
		GraphVAT [25]	✓				✓	
	Graph Rationalization	GREA [71]		✓		✓	✓	
		AdvCA [97]		✓		✓	✓	
	Automated Augmentation	AutoGDA [144]	✓			✓	✓	
		GraphAug [79]		✓		✓	✓	
		JOAO [130]		✓		✓	✓	
		MolCLE [116]		✓		✓	✓	

Both DropNode [27] and NodeDropping [129] aim to randomly remove a fraction of the nodes from the given graph, assuming that the missing nodes should not affect the semantic meanings of the remaining nodes, or the whole graph G. Feng et al. [27] focused on semi-supervised node classification, where a consistency loss is used on the predicted logits of different augmented versions of the graphs. On the other hand, You et al. [129] focused on self-supervised graph representation learning with contrastive targets.

Feature Masking. Other than the graph structure, i.e., nodes and edges, multiple works also adopted masking augmentations on the node features. For example, graph contrastive learning methods [102, 129, 130, 151] commonly utilize stochastic feature masking as an efficient way of augmenting or corrupting the graph. On top

of randomly masking feature values (i.e., random entries in $\mathbf{X}$) or feature signals (i.e., random columns in $\mathbf{X}$), Velickovic et al. [108] utilized row swapping as an effective way of corrupting the graph. Specifically, Velickovic et al. [108] randomly re-assigned the each node's feature vector to another node in the graph, which can be obtained by row-wise shuffling of $\mathbf{X}$.

More recently, Fang et al. [23] proposed DropMessage, which masks the features aggregated by message passing in GNNs. More specifically, denoting the aggregated neighbor feature of node v by the l-th layer as $\boldsymbol{h}^l_{\mathcal{N}(v)}$ (Equation equation 48), DropMessage randomly applies a binary mask on $\boldsymbol{h}^l_{\mathcal{N}(v)}$ for each node $v \in \mathcal{V}$ in every GNN layer. Similar to other dropping methods, the masks are sampled according to a Bernoulli distribution.

Subgraph Cropping. Another common data removal augmentation approach is cropping out part of the graph data. Such subgraph cropping can usually be achieved by either sampling the remaining subgraph or the subgraph that will be cropped out. For example, You et al. [129] first proposed the Subgraph augmentation, which samples the remaining subgraph via random walk. The method later learns the graph representations by contrasting the sampled subgraphs, with the assumption that the semantics of the whole graph can be preserved in part or its local structure. On the other hand, GraphCrop [113] crops a contiguous subgraph from each of the given graph object. GraphCrop adopts a graph diffusion-based node-centric strategy, performing graph diffusion on the randomly selected seed nodes, to maintain the topology characteristics of original graphs after the cropping.

4.2 Data Addition

Opposite to data removal methods, data addition methods augments the graph data by adding components to the existing/observed graph data or directly generating additional graphs. Note that although edge dropping is one of the most common techniques in data removal, rule-based edge addition is rather uncommon due to the huge search space for potential edge addition candidates (with a complexity of $O(N^2)$). While graph diffusion methods include adding edges, we discuss them later in Section 4.3 as they also include sparsification operations after edge addition.

Virtual Node. For graph classification, creating a virtual node that connect to all nodes in the graph is a commonly used GDA approach [31, 69, 46, 44, 127]. The idea of virtual node is to compute a graph representation in parallel with the node representations during the aggregation process. Therefore, instead of using an additional pooling layer, the virtual node's representation can directly be used as the graph representation, in a way similar to the `[CLS]` token in language modeling. Moreover, as the virtual node connects to all the nodes, it allows feature aggregation between previously unreachable nodes without adding additional GNN layers. Ying et al. [127] further show that it acts similar as self-attention in Transformers. Other than graph-level tasks, Hwang et al. [45] also studied virtual nodes for link prediction. As the graph data for link prediction is usually much larger for link prediction when compared with those in graph-level tasks, Hwang et al. [45] proposed to augment the graph data multiple virtual nodes, each connecting with a subset of all nodes in the graph with assignment decided by clustering methods.

Data Interpolation. With it's simplicity and effectiveness, Mixup [135] has been commonly used in image and language domains for augmenting new data samples. Specifically, Mixup constructs virtual training examples by interpolating two labeled training samples:

$$\begin{aligned} \tilde{\boldsymbol{x}} &= \lambda \boldsymbol{x}_i + (1-\lambda)\boldsymbol{x}_j, \\ \tilde{\boldsymbol{y}} &= \lambda \boldsymbol{y}_i + (1-\lambda)\boldsymbol{y}_j, \end{aligned} \tag{51}$$

where $(\boldsymbol{x}_i, \boldsymbol{y}_i)$ and $(\boldsymbol{x}_j, \boldsymbol{y}_j)$ are two randomly selected labeled training examples, and $\lambda \in [0, 1]$. By linearly interpolating the feature vectors and labels, Mixup incorporates the prior knowledge and extends the training distribution. Similarly, Manifold Mixup [109] performs Mixup on latent intermediate representations instead of raw features of the two training samples.

The direct analog of Mixup on graphs is not obvious, given the inter-dependent and irregular nature of graph data. Verma et al. [110] proposed GraphMix that augmented the training of a GNNs with a Fully-Connected Network, which is trained by interpolating the hidden states and labels. As GraphMix is more of a regularization method than the analog of Mixup on graphs, Wang et al. [117] proposed Graph Mixup, which analogized Manifold Mixup with a two-branch graph convolution module. Given a pair of nodes, Graph Mixup mixes their raw features, passes them into the two-branch GNN layer, and mixes the hidden representations of each layer. Notably, mixing up the nodes on features and hidden states avoids re-assembling the local neighborhoods of the two nodes. Graph Mixup also works for the task of graph classification. To avoid the node matching problem when mixing up two independent graphs, Graph Mixup mixes the latent representations of the pair of graphs.

On the other hand, ifMixup [37] directly applies Mixup on the graph data instead of the latent space for graph-level tasks. As the pair of graphs are irregular and the nodes from two graphs are not generally aligned, ifMixup arbitrarily assigns indices to the nodes in each graph and matches the nodes according to the indices. Empirically, ifMixup shows marginal performance improvements over Graph Mixup on the task of graph classification. Following ifMixup, Graph Transplant [86] also mixes graph in data space , but uses substructures as mixing units to preserve the local structural information. Graph Transplant employs the node saliency information to select one meaningful substructure from each graph, where the saliency information is defined as the l_2 norm of the gradient of the classification loss.

Different from the above Mixup-based methods which operate on instance level, Han et al. [39] proposed G-Mixup that performs Mixup on class-level. Instead of directly interpolating the individual graphs, G-Mixup interpolates the graph generators (graphons) for each class. Specifically, G-Mixup first estimates a graphon for each class of the training graphs, then mixes up the graphons of different classes, and finally generate synthetic graphs with the mixed graphons. Denoting the graphons of classes a and b as W_a and W_b, respectively, G-Mixup can be formulated as

$$\begin{aligned} \tilde{\boldsymbol{x}} \sim W_c, \text{ where } \quad W_c = \lambda W_a + (1-\lambda) W_b, \\ \tilde{\boldsymbol{y}} = \lambda \boldsymbol{y}_a + (1-\lambda)\boldsymbol{y}_b, \end{aligned} \tag{52}$$

where $\boldsymbol{y}_a$ and $\boldsymbol{y}_b$ are corresponding labels for graphs in classes a and b, respectively.

Besides Mixup, SMOTE [7] is also a classical data augmentation method that interpolates data instances. Different from Mixup which interpolates examples from different classes, SMOTE interpolates examples within the minority classes. Hence, SMOTE is especially effective when dealing with imbalanced data. On graph data, GraphSMOTE [142] augments the minority class by over-sampling synthetic nodes and then generating edges for them. GATSMOTE [76] and GNN-CL [70] further utilize attention designs to improve the edge generating process between the synthetic nodes and original nodes in the graph.

Counterfactual Augmentations. Counterfactual augmentation has been relatively under-explored in the field of graph machine learning. Zhao et al. [143] first proposed a counterfactual data augmentation method CFLP for the task of link prediction. To better understand the relationship between observed graph structure and link formation, CFLP asks the counterfactual question of "would the link still exist if the graph structure became different from observation?" To answer the question, Zhao et al. [143] proposed counterfactual links that approximates the unobserved outcome in the question. CFLP then trains a link prediction model with both the given training data and the generated counterfactual links (as augmented data). Similarly, CLBR Zhu et al. [148] proposed counterfactual data augmentation for bundle recommendation. CLBR generates the counterfactual example by answering the counterfactual question "what would a user interact with if the bundle-item affiliation relations change?".

Attribute Augmentation. Besides updating the graph topology, several works were also proposed to augment the graph data by generating additional node attributes. For example, LA-GNN [75] enhances the locality of node representations by generating node features based on the conditional distribution of the local structures and neighbor features. LA-GNN learns the new features of each node by the conditional distribution of its local neighborhood. The generated feature is directly used together with the raw node features as part of the input of

GNNs for both training and inference. Similarly, SR+DR [94] generates topology features with DeepWalk [87], and uses a dual GNN model with topology regularization to jointly train with both raw and topology features.

Pseudo-labeling. The training data in graph tasks is often only partially labeled due to the generally high cost of human labeling. With the large amount of unlabeled data, pseudo-labeling for the unlabeled data is often adopted under semi-supervised learning settings. Label Propagation [150, 149, 21] is one of the most classical methods for generating pseudo labels when only part of the nodes in the graph are labeled. Label propagation assumes that the two nodes are more likely to have the same label if they are connected, so it iteratively propagates node labels along the edges. With the propagated labels on the previously unlabeled nodes, the GNN model can then be trained with more labeled data.

4.3 Data Manipulation

Other than only adding or removing graph data, several rule-based methods also augment the graph data by combining both kind of operations. In order to separate them from the methods that purely conducts data removal or data addition, we introduce such augmentation methods in this subsection.

Diffusion. Klicpera et al. [60] first proposed generalized graph diffusion that modeled a "future" state of the graph where the signals were more spread out. Specifically, the generalized graph diffusion is formulated as

$$\tilde{\mathbf{A}} = \sum_{k=0}^{\infty} \theta_k \mathbf{T}^k, \tag{53}$$

where θ_k denote the global-local coefficient and $\mathbf{T} \in \mathbb{R}^{N \times N}$ represents the transition matrix derived from the adjacency matrix $\mathbf{A}$ (e.g., $\mathbf{A}\mathbf{D}^{-1}$ or $\mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$). θ_k is usually pre-defined by specific diffusion variants, e.g., heat kernel [61] ($\theta_k = e^{-t}\frac{t^k}{k!}$) or Personalized PageRank (PPR) [84] ($\theta_k = \alpha(1-\alpha)^k$), where α denotes the teleport probability in a random walk and t is diffusion time. The analytical solution of the heat kernel and PPR diffusions are defined as

$$\tilde{\mathbf{A}}^{\text{heat}} = e^{-(t\mathbf{T}-t)}; \quad \tilde{\mathbf{A}}^{\text{PPR}} = \alpha(\mathbf{I}_N - (1-\alpha)\mathbf{T})^{-1}, \tag{54}$$

where $\mathbf{I}_N$ is the N by N identity matrix. As the obtained adjacency matrix after diffusion $\tilde{\mathbf{A}}$ is often too dense as input for GNNs, graph sparsification is commonly conducted to filter out some trivial edges, e.g., setting a threshold to cut-off edges with small weights.

For (semi-)supervised learning on graphs, $\tilde{\mathbf{A}}$ can be directly used for both training and inferencing with GNNs [60]. While most message passing-based GNNs are only capable of aggregating one-hop information in each layer, the augmented graph after diffusion allows GNNs to learn from multi-hop (global) information without specifically re-designing the GNN models. In self-supervised graph representation learning, $\tilde{\mathbf{A}}$ is often used as the augmented view for self-supervised learning objectives such as contrastive learning [40, 132].

Subgraph Substituting. Several methods also make use of special substructures such as motifs and functional groups during subgraph augmentation. For example, M-Evolve [147] utilizes motifs to augment the graph data. M-Evolve first finds and selects the target motif in the graph, then adds or removes edges within the selected motifs based on a sampling weight calculated with Resource Allocation index. Similarly, MoCL [99] utilizes biomedical domain knowledge to augment the molecular graphs on the substructures such as functional groups. MoCL selects a substructure from each molecular graph and replaces it with another substructure.

4.4 Applications of Rule-based GDA

The rule-based augmentation techniques are mostly designed for improving general graph learning, and usually does not have constrains on specific tasks or domains. For example, although Rong et al. [88] only evaluated DropEdge for node classification task, the usage of it on other tasks is straightforward, and similar for most

stochastic data removal methods discussed in Section 4.1. Nonetheless, some rule-based GDA methods are more suitable for certain domains. For instance, subgraph substituting methods [147, 99] utilizes substructure information or even biomedical domain knowledge to augment the graphs, which makes them naturally more suitable for graph-level tasks on biomedical data. On the other hand, graph diffusion methods [60] are designed based on the spread of information along the relations in the graph, which makes such methods for suitable for larger graphs such as social networks or citation networks. Similarly, designed for exploring the formation of the links, counterfactual augmentation methods [143, 148] are tailored for link prediction or recommendation on larger graphs. We also specify the targeted tasks for each GDA method in Table 1.

Other than supervised graph representation learning schemes, the stochastic data removal methods (Section 4.1) are also commonly used in self-supervised graph representation learning methods as an efficient way of augmenting/corrupting graph data. For example, several methods [108, 129, 130, 102] use one or multiple of the above-mentioned techniques as augmentation methods for generating the augmented views of graph data. We further elaborate on the usage of data removing augmentations for self-supervised learning in Section 6.

5 Learned Approaches for GDA

In the previous section, we introduced rule-based GDA approaches where no learnable parameters are involved during data augmentation. However, these approaches could sometimes be suboptimal since the augmentations do not take advantage of the rich information from downstream tasks, especially in (semi-)supervised training. Indeed, some prior works from the vision [15] and natural language [83] learning domains show the promise of learned augmentation approaches. To address this concern, learned GDA approaches are proposed to learn augmentation strategies in a data-driven manner. The existing methods can be categorized into the following types: (1) structure learning, (2) adversarial training, (3) rationalization, and (4) automated augmentation.

5.1 Graph Structure Learning

In real-world scenarios, given graph structures are often incomplete [28], noisy [50, 78] or manipulated by adversarial attacks [49, 36]. Simply applying rule-based GDA approaches for training (semi-)supervised models on such graphs can lead to suboptimal performances, as they may not necessarily generate better graph structures for downstream tasks. To tackle these issues, several works propose graph structure learning approaches which aim to search for a better graph structure that augments the initial graph structure. Essentially, those methods treat the graph structure as learnable parameters and iteratively refine it while learning the model parameters [140, 50, 28, 12, 78, 145]. Numerous studies have demonstrated the effectiveness of graph structure learning methods in improving model generalization [140, 12] and robustness [50, 145]. In the following, we introduce several representative works that fall into the category of graph structure learning.

Improving Generalization. There are numerous methods for graph structure learning that target improving the generalization performance. Overall, they can be divided into two categories based on the adjacency matrix which they learn: learning continuous structure and learning discrete structure.

Although the original adjacency matrix is usually discrete (or binary), continuous structure methods do not assume the learned adjacency matrix to be discrete, as modeling discrete structure requires additional efforts in optimization. Typically, these methods either model the adjacency matrix as free parameters or use a parameterized neural network to model the structure. For instance, GLCN [47] is an early work which proposes a unified network architecture to learn an optimal graph structure and GNN. It incorporates the similarities of node features to learn a sparse and continuous graph structure. Formally, it defines a graph learning loss $\mathcal{L}_{\text{GL}}$ as:

$$\mathcal{L}_{\text{GL}} = \sum_{i,j=1}^{N} \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2 \, \tilde{\mathbf{A}}_{ij} + \gamma \|\tilde{\mathbf{A}}\|_F^2 + \beta \|\tilde{\mathbf{A}} - \mathbf{A}\|_F, \tag{55}$$

where the first two terms control the smoothness and sparsity of the augmented graph, respectively; the third term forces the augmented graph to be close to the original graph; γ and β are the hyper-parameters that balance the three terms. By minimizing $\mathcal{L}_{\text{GL}}$ together with the classification loss, GLCN is able to learn a graph structure that best serves the downstream task. TO-GCN [126] also considers the feature similarity, but it further employs label similarity to refine the graph topology. To handle the inductive learning setting, IDGL [12] casts the graph structure learning problem as similarity metric learning which will be jointly trained with the prediction model dedicated to a downstream task. To encourage learning graph structure invariant to task-irrelevant information, Sun et al. [100] utilized the Information Bottleneck [103] principle to solve the graph structure learning problem. Moreover, SLAPS [24] identifies a supervision starvation problem in previous structure learning approaches and proposes to incorporate additional self-supervision by designing a feature denoising task.

Despite the appeal of the first type of methods, continuous structures typically deviate from the original, sparse and discrete structure evident in many real-world graphs. To address this concern, some works focus on sampling the graph structures from a targeted distribution. For instance, by taking advantage of neural edge predictors like GAE [58], Zhao et al. [140] proposed GAug to generate plausible edge augmentations for an input graph. The output of the edge predictor can be formulated as

$$\mathbf{M} = \sigma_0\left(\mathbf{Z}\mathbf{Z}^T\right), \text{ with } \mathbf{Z} = \mathbf{H}^l, \tag{56}$$

where $\mathbf{M}$ is the edge probabilities matrix and σ_0 is an element-wise sigmoid function. Based on the edge probabilities matrix, two variants GAug-M and GAug-O are proposed to tackle augmentation in settings where edge manipulation is and is not feasible at inference time, respectively. Specifically, GAug-M deterministically adds edges with the highest edge probabilities to the graph at inference time; GAug-O optimizes the graph structure by minimizing the downstream classification loss together with the edge prediction loss and samples the adjacency matrix according to an element-wise Bernoulli distribution. Another representative work is LDS [28], which aims at learning discrete structure between data points while learning GNN parameters. It models the process as learning the edge probability matrix, which parameterizes the element-wise Bernoulli distribution from which the discrete structure is sampled. Then it formulates the learning process as a bi-level problem and updates the structure and model parameters in a differentiable way. Shang et al. [91] improves the efficiency of LDS by converting the bi-level problem to a uni-level problem and extends it to multivariate time series. In addition to Bernoulli distribution, recent studies have investigated other distributions to sample the discrete structure. For example, to account for the underlying generation of graphs, GEN [111] hypothesizes that the estimated graph is drawn from Stochastic Block Model (SBM) [42]. Similarly, BGCN [138] iteratively trains an assortative mixed membership stochastic block model with predictions of GCN to produce multiple denoised graphs, and ensembles results from multiple GCNs. To explicitly guarantee the strength and diversity of graph augmentation, MH-Aug [85] draws augmented graphs from an explicit target distribution through the Metropolis-Hastings algorithm, which can also be viewed as a graph structure learning process.

Instead of drawing discrete structures from targeted distributions, another line of works focus on dropping/adding edges from the original graph which can also lead to a discrete adjacency matrix. For instance, to improve the performance of GNNs under random noise, PTDNet [78] proposes to prune task-irrelevant edges by penalizing the number of edges in the sparsified graph and imposing the low-rank constraint with parameterized networks. Similarly, NeuralSparse [145] learns to drop task-irrelevant edges; it takes node/edge features as parts of input and jointly optimizes graph sparsification from the supervision of downstream task. Moreover, Gao et al. [29] proposed TADropEdge which leverages the graph spectrum to generate edge weights that represent the edges' criticality for the graph connectivity and drops edges by treating their weights as probabilities. Besides node classification tasks, Spinelli et al. [95] proposed FairDrop for the task of fair graph representation learning, which biasedly dropped edges with a sensitive attribute homophily mask to protect against unfairness. Later, Chen et al. [8] proposed AdaEdge, which iteratively adds/removes edges according to the node classification prediction. In each iteration, after the GNN model is sufficiently trained, AdaEdge adds edges between nodes that are predicted to be in the same class with high confidence, and vice versa. AdaEdge iteratively performs

GNN training and graph modification until convergence. Besides, Zhao et al. [141] proposed Eland for the task of anomaly detection on time-stamped user-item bipartite graphs. Eland first transforms the user-item graph into users' action sequences and adopts seq2seq model for future action prediction. The predicted user actions are added back into the graph to yield the augmented graph data. As the augmented graph contains richer user behavior information, Eland enhances the anomaly detection performance and detects anomalies at an early stage. It is worth mentioning that the aforementioned techniques are focused on one specific task such as node classification. To make graph structure learning benefit various downstream tasks, Liu et al. [77] proposed an unsupervised approach to learn graph structures with the aid of self-supervised contrastive learning [153].

While existing methods majorly focus on training-time augmentation, i.e., modifying the training graph data, a new line of work (e.g., GTrans [54]) introduces test-time augmentation by transforming the test graph through optimizing a self-supervised loss. It has been demonstrated to significantly improve the generalization performance of GNNs on out-of-distribution data.

Improving Robustness. Recent studies have demonstrated the vulnerability of GNNs under adversarial attacks, i.e., carefully-crafted small perturbation on the input graph leads GNNs into giving wrong predictions [157, 17, 154, 51]. A series of works are proposed to focus on enhancing the robustness of graph neural networks under adversarial attacks by learning clean graph structure. Jin et al. [50] observed that adversarial attacks violate important graph properties such as sparsity, low-rank, and feature smoothness; it then proposes the ProGNN framework to robustify GNNs by alternatively updating the graph structure by preserving these graph properties by adding penalizing regularization terms and training GNN parameters on the updated graph structure. Specifically, it defines the following graph learning loss:

$$\mathcal{L}_{\text{GL}} = \alpha\|\tilde{\mathbf{A}}\|_1 + \beta\|\tilde{\mathbf{A}}\|_* + \lambda \operatorname{tr}\left(\mathbf{X}^T\tilde{\mathbf{L}}\mathbf{X}\right) + \|\tilde{\mathbf{A}} - \mathbf{A}\|_F^2, \tag{57}$$

where $\|\cdot\|_1$ is the ℓ_1 norm, $\|\cdot\|_*$ is the nuclear norm, and $\tilde{\mathbf{L}}$ is the normalized Laplacian matrix of $\tilde{\mathbf{A}}$. The first three terms in Equation equation 57 force the learned graph to preserve the properties of sparsity, low-rank, and feature smoothness, respectively. Similar to GLCN [47], ProGNN also includes the downstream classification loss in the graph learning process. Despite the robustness of ProGNN, it is computationally expensive with $\mathrm{O}(N^3)$ time complexity and $\mathrm{O}(N^2)$ space complexity. To speed up ProGNN, LRGNN [124] decouples the adjacency matrix into a low-rank component and a sparse one, and learns the graph structure by minimizing the rank of the low-rank component and suppressing the sparse one. Furthermore, as robust GNNs tend to yield unsatisfying performance when trained with limited labeled nodes, Dai et al. [16] took advantage of self-supervision and uses node attributes to predict the links so as to boost robust performance, which also saves computational cost from direct structure learning. Also using a link predictor, DefenseVAE [134] employs variational graph autoencoder [58] to reconstruct graph structure that can reduce the effects of adversarial perturbations and boost the performance of GNNs under adversarial attacks. In addition, utilizing information theory, CoGSL [74] targets at learning the most compact structure relevant to downstream tasks in order to achieve a better balance between robustness and accuracy. Instead of explicitly learning the graph structure, GNNGuard [137] mitigates the negative effects of adversarial attacks by assigning higher weights to edges connecting similar nodes while pruning edges between dissimilar nodes, which can also be considered as implicit graph structure learning. While the aforementioned techniques have shown robustness in some specific settings, one recent work [82] revealed that their robustness decreases significantly under proper evaluation (in particular the adaptive attacks). This suggests that a more powerful and adaptive GSL method is needed for effective defense.

It is worth noting that there are some other graph structure learning works which aim at learning graphs to improve the scalability of graph machine learning models [53, 52, 73]. They do not target improving model performance or robustness of GNNs, and hence are not in the GDA scope tackled in this work.

5.2 Graph Adversarial Training

Adversarial training is a widely used countermeasure for adversarial attacks on computer vision [32], and has also been extended to graph domain [17, 25, 20, 43, 18, 10, 63]. Unlike graph structure learning, graph adversarial training does not seek to find an optimal graph structure. Instead, it augments input graphs with adversarial patterns during model training by perturbing node features or graph structure. The adversarially trained models are expected to tolerate adversarial perturbations in graph data and yield better generalization and robustness performance at test time. At the core of adversarial training is the injection of adversarial examples into the training set, with which the trained model can predict the test adversarial examples properly. Thus, we can adopt this strategy to enhance the robustness of GNNs as follows,

$$\min_{\Theta} \max_{\substack{\Delta_{\mathbf{A}} \in \mathcal{P}_{\mathbf{A}} \\ \Delta_{\mathbf{X}} \in \mathcal{P}_{\mathbf{X}}}} \mathcal{L}_{\text{train}} \left(g_{\Theta}(\mathbf{A} + \Delta_{\mathbf{A}}, \mathbf{X} + \Delta_{\mathbf{X}}) \right), \tag{58}$$

where $\mathcal{L}_{\text{train}}$ denotes the training loss for the downstream task; $\Delta_{\mathbf{A}}$ and $\Delta_{\mathbf{X}}$ stand for the perturbation on $\mathbf{A}, \mathbf{X}$, respectively; $\mathcal{P}_{\mathbf{A}}$ and $\mathcal{P}_{\mathbf{X}}$ denote the perturbation space. From the bi-level optimization problem in Equation equation 58, we can observe that adversarial training generates perturbations that maximize the prediction loss and updates model parameters to minimize the prediction loss. The process of generating perturbations (i.e., $\mathbf{A} + \Delta_{\mathbf{A}}, \mathbf{X} + \Delta_{\mathbf{X}}$) can be viewed as adversarial data augmentation and we can leverage such augmentations to improve the model robustness and generalization.

To augment the adjacency matrix, Dai et al. [17] proposed to randomly drop edges during adversarial training without any optimization on the graph data. While this strategy does not bring significant improvement, such cheap adversarial training still shows some improvement in robust classification accuracy. This finding is also in line with that from Zügner and Günnemann [156]. Instead of randomly dropping edges, Xu et al. [125] leveraged projected gradient descent (PGD) to optimize the bi-level problem and generate perturbations on the discrete structure, which achieves significant improvement in robust performance. Similarly, Chen et al. [9] and Dai et al. [18] also used existing adversarial attacks to modify the input graph structure during adversarial training, designed for network embedding methods. Furthermore, Suresh et al. [101] proposed to generate adversarial graph augmentation by learning to drop edges such that the augmentation can capture the minimal information that is sufficient to classify each graph.

On the other hand, there are some works focusing on perturbing the input features to serve as adversarial examples. For instance, Feng et al. [25] proposed an adversarial training strategy with dynamic regularization, which aims to reconstruct graph smoothness and constrains the divergence between the prediction of the target node and its connected nodes. Deng et al. [20] proposed batch virtual adversarial training to promote the smoothness of GNNs and thus defend against adversarial perturbations. Moreover, Kong et al. [63] proposed FLAG which utilizes adversarial training to iteratively augment the node features with gradient-based adversarial perturbations and improves the performances of GNNs on node classification, link prediction, and graph classification tasks. In addition, Zügner and Günnemann [155] studied certifiable robustness of GNNs w.r.t. perturbations of node attributes and propose a robust training scheme inspired by the certificates. Several other variants of adversarial training on perturbing node features are introduced in [112, 43].

5.3 Rationalization

A rationale is defined as a subset of input features that best represent, guide and support model prediction [71]. In the graph domain, rationales are subgraphs intrinsically learned by graph learning models. Rationales can be viewed as a form of augmented graph data that provide intrinsic explanations to the graph models' predictions, as opposed to the post-hoc explanation methods. Rationalization is commonly applied to graph-level property prediction or classification tasks [119, 71, 129, 13, 81, 68] for drug and material discovery on molecular and polymer datasets, etc.

Rationalization emerged in the graph domain as an approach to enhance both the interpretability and overall performance of graph classification and regression. Yu et al. [131] found similarity to the Information Bottleneck (IB) problem, and proposed the Graph Information bottleneck framework (GIB), which learns to generate the maximally informative and compressed subgraph (IB-graph) by leveraging a bi-level optimization scheme and a novel connectivity loss. Also rooted in the IB paradigm, Miao et al. [81] proposed GSAT to better learn and select task-relevant subgraphs that improve interpretation and prediction by injecting stochasticity into the attention weights in order to constrain information from task-irrelevant components. GREA, another rationalization work proposed by Liu et al. [71], proposed a novel environment replacement augmentation method, which separates the rationale and the environment subgraphs (the remaining and complementary subgraphs to the rationale ones) and optimized the separation (rationalization identification) with data augmentation by replacing the original environment subgraph with a different one in the latent space.

Rationalization models are also effective in addressing data bias and out-of-distribution (OOD) problems for graph property prediction tasks since rationales are both interpretable and generalizable [81]. Wu et al. [120] proposed DIR to generate distribution perturbation on training data with causal intervention. Based on the idea that causal patterns are stable to distribution shift, they created a rationale generator that separates causal and non-causal graphs, applies causal intervention to create perturbed distributions, and then jointly learn both the causal and non-causal representation to minimize invariant risk. Similarly, Chen et al. [13] also took a causal perspective to solve the OOD problem. They proposed CIGA to model the graph generation process and the interactions between invariant and spurious features with Structural Causal Models (SCM). The resulting subgraphs generated by CIGA maximally preserves the invariant intra-class information. Li et al. [68] also proposed to separate invariant and variant graphs. In their framework GIL, they proposed a GNN based subgraph generator to identify potentially invariant subgraphs, then infer latent environment labels for the variant subgraphs, before jointly optimizing all modules. To address the limited environments and unstable causal features in data augmentation methods for graph rationalization, AdvCA [97] was proposed to improve the generalization capacity against covariate shift through adversarial causal augmentation.

5.4 Automated Augmentation

GDA techniques mentioned in Section 4 take rule-based approaches to augment graph data, applying the same augmentation method to subgraphs and graphs which embody different attributes and characteristics like degree distribution and homophily. To tackle this issue, Automated GDA techniques [98, 79, 144, 130, 64, 41, 153] were recently explored to automatically learn tailored augmentations for different subgraphs or graphs. For example, Sun et al. [98] proposed AutoGRL for the task of node classification. Through the training process, AutoGRL learns the best combination of GDA operations, GNN architecture, and hyper-parameters. The searching space of AutoGRL includes four GDA operations implemented by random masking and GAug-M [140]: drop features, drop nodes, add edges, and remove edges.

Since automated GDA objectives are often complex to optimize, some recent works use reinforcement learning approaches as a solution. Zhao et al. [144] framed the AutoGDA as a bi-level optimization problem, aiming to find a different set of augmentation strategies for each community in the graph as they observed various characteristics in each community. They employ an RL-agent to generalize the learning and find localize augmentation strategies for node classification tasks. On graph classification tasks, Luo et al. [79] set out to learn an automated augmentation model with GraphAug, to provide label-invariant augmentations for each graph in the dataset. Applying reinforcement learning, they maximize the estimated label-invariance probability to learn the augmentation category and transformation selection.

Another group of works on automated augmentation focus on graph contrastive learning. You et al. [130] proposed to learn augmentations to replace ad hoc and handpicked augmentations for contrastive learning. They design an augmentation-aware projection head to avoid complicated augmentations, and formulate a bi-level optimization problem to learn both the augmentation strategy and graph representation. Hassani and Khasahmadi

Table 2: Representative self-supervised graph learning works that utilized graph data augmentation techniques. †Although the methods in this category are semi-supervised methods, they used GDA operations with only self-supervised learning objectives (i.e, consistency loss). Therefore, we categorize their GDA techniques as designed for self-supervised learning objectives.

	Representative Works	Task Level			Augmented Data		
		Node	Graph	Edge	Structure	Feature	Label
Contrastive Learning	DGI [108]	✓				✓	
	GRACE [151]	✓			✓	✓	
	MVGRL [40]	✓			✓		
	GraphCL [129]		✓		✓	✓	
	JOAO [130]		✓		✓	✓	
Non-contrastive Learning	CCA-SSG [136]	✓			✓	✓	
	GBT [3]	✓			✓	✓	
	BGRL [102]	✓			✓	✓	
	T-BGRL [92]	✓			✓	✓	
Consistency Training†	GRAND [27]	✓			✓	✓	
	NodeAug [114]	✓			✓	✓	
	MV-GCN [132]	✓			✓		
	NASA [4]	✓			✓		

[41] learned a probabilistic policy that contains a set of distributions over different augmentation operations in their method LG2AR, and samples an augmentation strategy from the policy in each training epoch. Zhu et al. [153] proposed GCA, which proposes adaptive augmentations based on node centrality measures. Unlike the aforementioned methods which find the best augmentation strategy for the dataset, GCA adaptively augments different nodes according to their importance. Wang et al. [116] proposed to use a generative probabilistic model and a learnable feature selector to automatically parameterize topological and attribute augmentations, which can also provide explanations for underlying patterns in molecular graphs. Lastly, Kose and Shen [64] proposed FairAug which utilizes adaptive augmentations for fair graph representation learning.

6 GDA for Self-supervised Learning

Other than directly using the augmented graph data in supervised learning, the most common use case for GDA is under self-supervised learning (SSL) schemes, e.g. contrastive learning. Self-supervised objectives learn representations that are robust to noise and perturbations by maximizing the (dis)agreements of learned representations. Therefore, unlike most of the previously mentioned learned GDA techniques (Section 5) which aim to enhance the task-relevant information in the data, most of the GDA techniques for self-supervised learning are rule-based augmentations (Section 4) which aim to corrupt or perturb the given graph data. Moreover, most self-supervised graph representation learning methods tend to use a combination of several simple GDA operations. In this section, we introduce three commonly used self-supervised graph learning schemes as well as the GDA approaches they utilize.

6.1 Contrastive Learning

In recent years, with the rapid development of contrastive learning in CV [11], many contrastive learning methods [151, 129, 105, 122, 71, 56] have been proposed for applications on graph data. Typically, a graph contrastive learning framework includes three main components: a GDA module that generates different views of the given graph data, a GNN-based encoder to compute the representations, and a contrastive learning objective to train the model. For each data example (nodes for node-level tasks and graphs for graph-level tasks), these methods consider augmented views or variants of itself as associated positive samples and other data examples in

the same batch as associated negative samples. Contrastive learning objectives then maximize the (dis)agreements of the representations between each data example with their (negative) positive examples.

To efficiently generate different augmented data for graph contrastive learning, rule-based data removal operations (Section 4.1) are the most commonly used GDA techniques, as they are fast and easy to apply. For example, multiple methods (GRACE [151], GraphCL [129], etc.) adopt stochastic edge dropping and/or feature masking due to their simplicity. DGI [108] adopts feature corruption by conducting a row-wise shuffling on the raw node feature matrix $\mathbf{X}$. In general, graph contrastive learning methods usually adopt a combination of multiple augmentation techniques to generate different augmented views. GraphCL [129] and InfoGCL [123] adopt four GDA operations: node dropping which randomly removes nodes along with its edges, edge perturbation which randomly adds or drops edges, attribute masking which randomly masks off certain node attributes, and subgraph sampling which samples connected subgraphs. SUBG-CON [48] utilizes a subgraph sampler to extract the context subgraph as a proxy of data augmentation. GRACE [151] uses only the basic random edge dropping and attribute masking for creating different views of the graph.

Other than data removal augmentations, graph diffusion is also commonly used in contrastive learning as it can naturally create a "future view" of the given graph where the information are more spread out. MVGRL [40] adopts the diffusion graph proposed by GDC [60] as the second view. Interestingly, Hassani and Khasahmadi [40] showed that using three views (original graph, PPR diffusion graph and heat kernel diffusion graph) would not result with better performance than using two views (original graph and one diffusion graph), and concluded "increasing the number of views does not improve the performance." However, Yuan et al. [132] later proposed MV-CGC which adopted a similar contrastive learning framework with three views: the original graph, diffusion graph, and a proposed feature similarity view. Empirically, the node representations learned by MV-CGC outperformed those learned by MVGRL on node classification, suggesting that additional well-designed GDA methods or views may be helpful to graph contrastive learning approaches.

More recently, several studies [101, 106, 72] pointed out that stochastic rule-based GDA operations may suffer from failing to induce useful task-relevant invariance on common benchmark datasets. Specifically, Trivedi et al. [106] analyzed that the generalization error of graph contrastive learning can be bounded under the assumptions of invariance to relevant augmentations, recoverability, and separability, which refer to *data-centric properties*, by instantiating rule-based GDA as a composition of graph edit operations. Such bound demonstrates conditions with low separability and recoverability during the usage of rule-based GDA, which motivates the necessity of inducing task-relevant invariance. Following the theoretical analysis, Zhang et al. [139] proposed a covariance-preserving feature augmentation technique, in which the augmented feature has bounded variance. Wang et al. [115] proposed to use different levels in hierarchical graphs as augmented views.

6.2 Non-contrastive Learning

While showing promising performance on various tasks, contrastive learning methods rely heavily on disagreement between data examples and their associated negative examples to avoid model collapse [33]. As sampling high quality negative examples is often costly, and random negative sampling usually requires large batch sizes, several works [33, 133, 1] propose non-contrastive self-supervised learning methods to learn representations in a self-supervised manner without needing negative examples. Instead of comparing across different samples, non-contrastive self-supervised methods compare only between different views of the same sample and use designs such as prediction heads and stop gradient to avoid model collapsing [33], or measure the cross-correlation matrix between the representations learned form different views [133].

As the non-contrastive methods are designed for more efficient self-supervised learning than the contrastive methods, the GDA techniques they adopt are all the most basic, stochastic ones (Section 4.1). Specifically, all the non-contrastive self-supervised graph representation learning methods (CCA-SSG [136], GBT [3], BGRL [102], and T-BGRL [92]) utilized only random edge dropping and node feature masking as the augmentation strategies. While the first three methods generates two augmented views for comparison, to further improve the performance

on link prediction under inductive settings, T-BGRL [92] also used the same augmentation strategies but with higher masking probability as an efficient corruption to create an third "negative" view to mitigate collapse, which is later used in a triplet loss.

6.3 Consistency Training

In real GML applications, semi-supervised learning usually plays an important role as only a small fraction of training data are labeled in most of the cases [121]. Due to such label scarcity, consistency training is commonly used to leverage the unlabeled data to improve the model quality. Similar to contrastive learning, consistency training itself is a self-supervised learning objective that aims to maximizes the agreement of representations learned from different views of the data. However, unlike (non-)contrastive learning that compares between data objects, the consistency loss compares the distributions of a batch of representations via metrics like KL-divergence. Therefore, the consistency loss is rarely used itself, but often used along with supervised losses in the semi-supervised learning settings. The final learning objective is usually a linear combination of the supervised loss (e.g., cross entropy for classification tasks) and the consistency loss.

NodeAug [114] uses three local structure-based augmentation operations: replacing attributes, removing and adding edges. NodeAug minimizes the KL-divergence between the node representations learned from the original graph and augmented graph. GRAND [27] creates multiple different augmented graphs with node dropping and feature masking. The consistency loss then minimizes the distances of the representations learned from the augmented graphs. NASA [4] proposes Neighbor Replace augmentation to randomly replace the 1-hop neighbors with 2-hop neighbors, and then use a neighbor-constrained consistency regularization during training. To further utilize the information given by different graph diffusions, MV-GCN [132] generates two complementary views with PPR and heat kernel and learns from both created views and the original graph. Then, it feeds three views of the graph into three GCNs, and uses a consistency regularization loss to reduce the distribution distance of the representations learned across the views, and derives the final node representations as a combination of the three.

7 Challenges and Directions

Despite substantial progress has been achieved in graph data augmentation research, several open problems remain to solve. In this section, we summarize several promising yet under-explored research directions.

7.1 Domain Adaptation and Regularization

Given the rapid development of GDA techniques in recent years, automated GDA methods have been proposed to automatically tune the augmentation strategy for different datasets and tasks. Nonetheless, the existing automated GDA methods for graph data (as introduced in Section 5.4) mainly focus on specific datasets and downstream tasks. Ideally, automated augmentation solutions should be transferable. That is, domain adaptation is a desired characteristic for automated GDA techniques. When the automated augmentation method trained on one dataset could only be used on that dataset, the method may be equivalent to automating the hyperparameter tuning process and lose the generalizability across datasets [144]. Therefore, for an ideal automated GDA method, it should be able to be trained on one dataset and used for many, ideally cross domain or under OOD settings. While OOD benchmarks are already available in the GML community [35], automated GDA methods that can be transferable across domains are still missing in the literature. Moreover, on certain types of graph data such as molecule graphs, most commonly used GDA operations would change the underlying semantics of the graph. For example, dropping a carbon atom from the phenyl ring of aspirin breaks the aromatic system and results in a alkene chain [66], which is an entirely different chemical compound. This motivates a need for domain-based regularization methods for such tasks. So far, only Sun et al. [99] proposed MoCL that

considers the semantic information brought by local substructures when augmenting the molecule graphs, leaving domain-based regularization GDA methods rather under-explored.

7.2 Scalability for Large-Scale Graphs

GDA techniques add additional complexity on top of the existing GNNs, and many GDA techniques use global information during the augmentation process, which might not be able to easily scale. For example, GAug-M [140] involves selecting the top K out of $O(N^2)$ logits for node pairs when selecting edges to add. Such high complexity operations can cause scalability issues in actual applications where the graph size can be very large, e.g., at billion scale. While complex GDA techniques bring significant performance improvements, the scalability of these methods are still worthy of attention. For example, in order to enable end-to-end training, GAug-O [140] requires back-propagating on the entire learned adjacency matrix, creating massive memory overheads. To improve the performance of DropEdge [88], TADropEdge [29] required the pre-calculation of a score for each edge in the graph prior to the training of GNNs. Therefore, to be applicable in practical applications, efficiency is also a necessity for GDA techniques. As mentioned in the previous subsections, automated solution which combine the fast and simple augmentation operations may be a promising direction. Nonetheless, how to design a scalable and efficient automated GDA framework is still an open line of research.

7.3 Comprehensive Evaluation Criteria and Standards

Similar to the DA research in other domains, a general concern for GDA research is that the evaluation only focuses on the prediction performance on specific datasets. Although this is likely the most important metric, other metrics such as additional time and resource consumption, transferability, or scalability are also important for researchers to more comprehensively understand the methods. For example, as aforementioned, while graph structure learning methods such as GAug [140] shows promising performances for node classification, the method's design inherently limits its ability to generalize on large-scale graphs. Furthermore, only few works discuss the additional time and resource requirement needed for applying their proposed GDA methods, especially for the learned augmentations which may require training of additional modules. Therefore, a set of comprehensive evaluation criteria and standards is desired for better understanding the benefits and costs of the newly proposed GDA methods. Ideally, such a benchmark could contain multiple datasets in different scales and domains, enabling researchers to better evaluate transferability and scalability tradeoffs.

7.4 Theoretical Foundation

GDA is a powerful technology to improve the performance of data-driven inference on graphs without the need of extra labeling effort or complex models. Empirically, GDA methods are also shown to improve the generalization of GML methods and alleviate the over-smoothing problem encountered by GNNs. Yet, there is little rigorous understanding of how and why GDA achieves those benefits, especially for (semi-)supervised learning. Although several works [140, 8] have analyzed the relation between graph homophily and classification performance or the over-smoothing problem, there is limited work showcasing rigorous proofs or theoretical bounds on these relationships.

Recently, several works provided theoretical insights of DA in the CV domain. For example, Wu et al. [119] theoretically analyzed the generalization effect of data augmentation on images. They interpreted the effect of data augmentation from a bias-variance perspective, where data augmentation adds new information to model training while also serving as a regularizer. Due to the irregular characteristics of graph data, these theoretical analysis cannot be directly applied for the GDA context. Besides the generalization perspective, several recent works have studied the certified robustness of GNNs [156]. Improved robustness bounds would be a desired property of GDA techniques. Recent studies [104] on the topology bottleneck and over-squashing of GNNs

provide theoretical guides for edge-based GDA techniques. Counterfactual augmentation methods on graphs such as CFLP [143] can also bring insights for analyzing GDA from the perspective of causality.

7.5 Data Augmentation for Complex Graph Types

Existing GDA approaches are mainly designed for homogeneous graphs, while not all of them can be easily generalized to other complex types of graphs such as heterogeneous graphs, dynamic graphs, hypergraphs, etc. These complex graphs have broader applications with their ability of modeling more complex relationship, nonetheless, the complexity of the data requires more sophisticated design of GDA methods. Taking heterogeneous graphs as an example, even the simplest edge dropping would require a drop rate hyper-parameter for each of the edge types in the graph, which could introduce significant computational overhead for hyper-parameter searching. Additionally, beyond direct analogous of GDA methods for homogeneous graph for complex graph types, specially designed GDA methods for different graph types could better utilize the rich information contained in them. Therefore, a comprehensive evaluation of the existing GDA methods on complex graphs is needed by the community to better understand the effectiveness of existing GDA methods and also better design principled augmentation approaches for each graph types.

8 Conclusion

Our work presents a comprehensive and structured survey of data augmentation techniques for graph machine learning (GML). We categorized existing graph data augmentation (GDA) techniques three taxonomies from different perspectives, introduced recent GDA approaches based on their core methodology, and introduced their applications in self-supervised learning. Finally, we outlined current challenges as well as directions for future research explorations in the GDA domain. We hope this survey serves as a guide for GML researchers and practitioners to study and use GDA techniques, and inspires additional interest and work on this topic.

References

[1] R. Balestriero and Y. LeCun. Contrastive and non-contrastive self-supervised learning recover global and local spectral embedding methods. arXiv:2205.11508, 2022.

[2] R. Barandela, R. M. Valdovinos, J. S. Sánchez, and F. J. Ferri. The imbalanced training sample problem: Under or over sampling? In Joint IAPR international workshops on SPR and SSPR, 2004.

[3] P. Bielak, T. Kajdanowicz, and N. V. Chawla. Graph barlow twins: A self-supervised representation learning framework for graphs. Knowledge-Based Systems, 2022.

[4] D. Bo, B. Hu, X. Wang, Z. Zhang, C. Shi, and J. Zhou. Regularizing graph neural networks via consistency-diversity graph augmentations. In AAAI, 2022.

[5] I. Brugere, B. Gallagher, and T. Y. Berger-Wolf. Network structure inference, a survey: Motivations, methods, and applications. CSUR, 2018.

[6] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun. Spectral networks and locally connected networks on graphs. arXiv:1312.6203, 2013.

[7] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. Smote: synthetic minority over-sampling technique. JAIR, 2002.

[8] D. Chen, Y. Lin, W. Li, P. Li, J. Zhou, and X. Sun. Measuring and relieving the over-smoothing problem for graph neural networks from the topological view. In AAAI, 2020.

[9] J. Chen, Y. Wu, X. Lin, and Q. Xuan. Can adversarial network attack be defended? arXiv:1903.05994, 2019.

[10] J. Chen, X. Lin, H. Xiong, Y. Wu, H. Zheng, and Q. Xuan. Smoothing adversarial training for gnn. IEEE Transactions on Computational Social Systems, 2020.

[11] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A simple framework for contrastive learning of visual representations. In ICML, 2020.

[12] Y. Chen, L. Wu, and M. Zaki. Iterative deep graph learning for graph neural networks: Better and robust node embeddings. In NeurIPS, 2020.
[13] Y. Chen, Y. Zhang, Y. Bian, H. Yang, M. KAILI, B. Xie, T. Liu, B. Han, and J. Cheng. Learning causally invariant representations for out-of-distribution generalization on graphs. In NeurIPS, 2022.
[14] F. Chierichetti, A. Epasto, R. Kumar, S. Lattanzi, and V. Mirrokni. Efficient algorithms for public-private social networks. In KDD, 2015.
[15] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le. Autoaugment: Learning augmentation strategies from data. In CVPR, 2019.
[16] E. Dai, W. Jin, H. Liu, and S. Wang. Towards robust graph neural networks for noisy graphs with sparse labels. In WSDM, 2022.
[17] H. Dai, H. Li, T. Tian, X. Huang, L. Wang, J. Zhu, and L. Song. Adversarial attack on graph structured data. In ICML, 2018.
[18] Q. Dai, X. Shen, L. Zhang, Q. Li, and D. Wang. Adversarial training methods for network embedding. In TheWebConf, 2019.
[19] M. Defferrard, X. Bresson, and P. Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In NeurIPS, 2016.
[20] Z. Deng, Y. Dong, and J. Zhu. Batch virtual adversarial training for graph convolutional networks. arXiv:1902.09192, 2019.
[21] H. Dong, J. Chen, F. Feng, X. He, S. Bi, Z. Ding, and P. Cui. On the equivalence of decoupled graph convolution network and label propagation. In TheWebConf, 2021.
[22] Q. Duong, M. P. Wellman, and S. Singh. Modeling information diffusion in networks with unobserved links. In IEEE PASSAT/SocialCom, 2011.
[23] T. Fang, Z. Xiao, C. Wang, J. Xu, X. Yang, and Y. Yang. Dropmessage: Unifying random dropping for graph neural networks. arXiv:2204.10037, 2022.
[24] B. Fatemi, L. El Asri, and S. M. Kazemi. Slaps: Self-supervision improves structure learning for graph neural networks. NeurIPS, 2021.
[25] F. Feng, X. He, J. Tang, and T.-S. Chua. Graph adversarial training: Dynamically regularizing based on graph structure. TKDE, 2019.
[26] S. Y. Feng, V. Gangal, J. Wei, S. Chandar, S. Vosoughi, T. Mitamura, and E. Hovy. A survey of data augmentation approaches for nlp. arXiv:2105.03075, 2021.
[27] W. Feng, J. Zhang, Y. Dong, Y. Han, H. Luan, Q. Xu, Q. Yang, E. Kharlamov, and J. Tang. Graph random neural networks for semi-supervised learning on graphs. NeurIPS, 2020.
[28] L. Franceschi, M. Niepert, M. Pontil, and X. He. Learning discrete structures for graph neural networks. In ICML, 2019.
[29] Z. Gao, S. Bhattacharya, L. Zhang, R. S. Blum, A. Ribeiro, and B. M. Sadler. Training robust graph neural networks with topology adaptive edge dropping. arXiv:2106.02892, 2021.
[30] K. A. Garrison, D. Scheinost, E. S. Finn, X. Shen, and R. T. Constable. The (in) stability of functional brain network measures across thresholds. Neuroimage, 2015.
[31] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl. Neural message passing for quantum chemistry. In ICML, 2017.
[32] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. arXiv:1412.6572, 2014.
[33] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. Richemond, E. Buchatskaya, C. Doersch, B. Avila Pires, Z. Guo, M. Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. NeurIPS, 2020.
[34] A. Grover and J. Leskovec. node2vec: Scalable feature learning for networks. In KDD, 2016.
[35] S. Gui, X. Li, L. Wang, and S. Ji. Good: A graph out-of-distribution benchmark. NeurIPS, 2022.
[36] S. Günnemann. Graph neural networks: Adversarial robustness. In Graph Neural Networks: Foundations, Frontiers, and Applications. 2022.
[37] H. Guo and Y. Mao. Intrusion-free graph mixup. arXiv:2110.09344, 2021.
[38] W. Hamilton, Z. Ying, and J. Leskovec. Inductive representation learning on large graphs. In NeurIPS, 2017.
[39] X. Han, Z. Jiang, N. Liu, and X. Hu. G-mixup: Graph data augmentation for graph classification. In ICML, 2022.
[40] K. Hassani and A. H. Khasahmadi. Contrastive multi-view representation learning on graphs. In ICML, 2020.

[41] K. Hassani and A. H. Khasahmadi. Learning graph augmentations to learn graph representations. arXiv:2201.09830, 2022.
[42] P. W. Holland, K. B. Laskey, and S. Leinhardt. Stochastic blockmodels: First steps. Social networks, 1983.
[43] W. Hu, C. Chen, Y. Chang, Z. Zheng, and Y. Du. Robust graph convolutional networks with directional graph adversarial training. Applied Intelligence, 2021.
[44] W. Hu, M. Fey, H. Ren, M. Nakata, Y. Dong, and J. Leskovec. Ogb-lsc: A large-scale challenge for machine learning on graphs. NeurIPS, 2021.
[45] E. Hwang, V. Thost, S. S. Dasgupta, and T. Ma. Revisiting virtual nodes in graph neural networks for link prediction. 2021.
[46] K. Ishiguro, S.-i. Maeda, and M. Koyama. Graph warp module: an auxiliary module for boosting the power of graph neural networks in molecular graph analysis. arXiv preprint arXiv:1902.01020, 2019.
[47] B. Jiang, Z. Zhang, D. Lin, J. Tang, and B. Luo. Semi-supervised learning with graph learning-convolutional networks. In CVPR, 2019.
[48] Y. Jiao, Y. Xiong, J. Zhang, Y. Zhang, T. Zhang, and Y. Zhu. Sub-graph contrast for scalable self-supervised graph representation learning. In ICDM, 2020.
[49] W. Jin, Y. Li, H. Xu, Y. Wang, S. Ji, C. Aggarwal, and J. Tang. Adversarial attacks and defenses on graphs: A review, a tool and empirical studies. arXiv:2003.00653, 2020.
[50] W. Jin, Y. Ma, X. Liu, X. Tang, S. Wang, and J. Tang. Graph structure learning for robust graph neural networks. In KDD, 2020.
[51] W. Jin, Y. Li, H. Xu, Y. Wang, S. Ji, C. Aggarwal, and J. Tang. Adversarial attacks and defenses on graphs. SIGKDD Explorations, 2021.
[52] W. Jin, X. Tang, H. Jiang, Z. Li, D. Zhang, J. Tang, and B. Yin. Condensing graphs via one-step gradient matching. In KDD, 2022.
[53] W. Jin, L. Zhao, S. Zhang, Y. Liu, J. Tang, and N. Shah. Graph condensation for graph neural networks. In ICLR, 2022.
[54] W. Jin, T. Zhao, J. Ding, Y. Liu, J. Tang, and N. Shah. Empowering graph representation learning with test-time graph transformation. ICLR, 2023.
[55] T. Joachims and A. Swaminathan. Counterfactual evaluation and learning for search, recommendation and ad placement. In SIGIR, 2016.
[56] M. Ju, T. Zhao, Q. Wen, W. Yu, N. Shah, Y. Ye, and C. Zhang. Multi-task self-supervised graph neural networks enable stronger task generalization. ICLR, 2023.
[57] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. arXiv:1609.02907, 2016.
[58] T. N. Kipf and M. Welling. Variational graph auto-encoders. arXiv:1611.07308, 2016.
[59] J. Klicpera, A. Bojchevski, and S. Günnemann. Predict then propagate: Graph neural networks meet personalized pagerank. arXiv:1810.05997, 2018.
[60] J. Klicpera, S. Weißenberger, and S. Günnemann. Diffusion improves graph learning. NeurIPS, 2019.
[61] R. I. Kondor and J. Lafferty. Diffusion kernels on graphs and other discrete structures. In ICML, 2002.
[62] K. Kong, G. Li, M. Ding, Z. Wu, C. Zhu, B. Ghanem, G. Taylor, and T. Goldstein. Flag: Adversarial data augmentation for graph neural networks. arXiv:2010.09891, 2020.
[63] K. Kong, G. Li, M. Ding, Z. Wu, C. Zhu, B. Ghanem, G. Taylor, and T. Goldstein. Robust optimization as data augmentation for large-scale graphs. In CVPR, 2022.
[64] O. D. Kose and Y. Shen. Fair node representation learning via adaptive data augmentation. arXiv:2201.08549, 2022.
[65] S. Kumar and N. Shah. False information on web and social media: A survey. arXiv:1804.08559, 2018.
[66] N. Lee, J. Lee, and C. Park. Augmentation-free self-supervised learning on graphs. In AAAI, 2021.
[67] R. Levie, F. Monti, X. Bresson, and M. M. Bronstein. Cayleynets: Graph convolutional neural networks with complex rational spectral filters. IEEE Transactions on Signal Processing, 2018.
[68] H. Li, Z. Zhang, X. Wang, and W. Zhu. Learning invariant graph representations for out-of-distribution generalization. In NeurIPS, 2022.
[69] J. Li, D. Cai, and X. He. Learning graph-level representation for drug discovery. arXiv preprint arXiv:1709.03741, 2017.

[70] X. Li, L. Wen, Y. Deng, F. Feng, X. Hu, L. Wang, and Z. Fan. Graph neural network with curriculum learning for imbalanced node classification. arXiv:2202.02529, 2022.
[71] G. Liu, T. Zhao, J. Xu, T. Luo, and M. Jiang. Graph rationalization with environment-based augmentations. In KDD, 2022.
[72] G. Liu, E. Inae, T. Zhao, J. Xu, T. Luo, and M. Jiang. Data-centric learning from unlabeled graphs with diffusion model. arXiv preprint arXiv:2303.10108, 2023.
[73] M. Liu, S. Li, X. Chen, and L. Song. Graph condensation via receptive field distribution matching. arXiv:2206.13697, 2022.
[74] N. Liu, X. Wang, L. Wu, Y. Chen, X. Guo, and C. Shi. Compact graph structure learning via mutual information compression. In TheWebConf, 2022.
[75] S. Liu, H. Dong, L. Li, T. Xu, Y. Rong, P. Zhao, J. Huang, and D. Wu. Local augmentation for graph neural networks. arXiv:2109.03856, 2021.
[76] Y. Liu, Z. Zhang, Y. Liu, and Y. Zhu. Gatsmote: Improving imbalanced node classification on graphs via attention and homophily. Mathematics, 2022.
[77] Y. Liu, Y. Zheng, D. Zhang, H. Chen, H. Peng, and S. Pan. Towards unsupervised deep graph structure learning. In TheWebConf, 2022.
[78] D. Luo, W. Cheng, W. Yu, B. Zong, J. Ni, H. Chen, and X. Zhang. Learning to drop: Robust graph neural network via topological denoising. In WSDM, 2021.
[79] Y. Luo, M. McThrow, W. Y. Au, T. Komikado, K. Uchino, K. Maruhash, and S. Ji. Automated data augmentations for graph classification. arXiv:2202.13248, 2022.
[80] Y. Ma, X. Liu, T. Zhao, Y. Liu, J. Tang, and N. Shah. A unified view on graph neural networks as graph signal denoising. In CIKM, 2021.
[81] S. Miao, M. Liu, and P. Li. Interpretable and generalizable graph learning via stochastic attention mechanism. In ICML, 2022.
[82] F. Mujkanovic, S. Geisler, S. Günnemann, and A. Bojchevski. Are defenses for graph neural networks robust? NeurIPS, 2022.
[83] T. Niu and M. Bansal. Automatically learning data augmentation policies for dialogue tasks. arXiv:1909.12868, 2019.
[84] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.
[85] H. Park, S. Lee, S. Kim, J. Park, J. Jeong, K.-M. Kim, J.-W. Ha, and H. J. Kim. Metropolis-hastings data augmentation for graph neural networks. NeurIPS, 2021.
[86] J. Park, H. Shim, and E. Yang. Graph transplant: Node saliency-guided graph mixup with local structure preservation. arXiv:2111.05639, 2021.
[87] B. Perozzi, R. Al-Rfou, and S. Skiena. Deepwalk: Online learning of social representations. In KDD, 2014.
[88] Y. Rong, W. Huang, T. Xu, and J. Huang. Dropedge: Towards deep graph convolutional networks on node classification. arXiv:1907.10903, 2019.
[89] R. Sennrich, B. Haddow, and A. Birch. Improving neural machine translation models with monolingual data. arXiv:1511.06709, 2015.
[90] N. Shah, A. Beutel, B. Gallagher, and C. Faloutsos. Spotting suspicious link behavior with fbox: An adversarial perspective. In ICDM, 2014.
[91] C. Shang, J. Chen, and J. Bi. Discrete graph structure learning for forecasting multiple time series. In ICLR, 2021.
[92] W. Shiao, Z. Guo, T. Zhao, E. E. Papalexakis, Y. Liu, and N. Shah. Link prediction with non-contrastive learning. arXiv:2211.14394, 2022.
[93] C. Shorten and T. M. Khoshgoftaar. A survey on image data augmentation for deep learning. Journal of Big Data, 2019.
[94] R. Song, F. Giunchiglia, K. Zhao, and H. Xu. Topological regularization for graph neural networks augmentation. arXiv:2104.02478, 2021.
[95] I. Spinelli, S. Scardapane, A. Hussain, and A. Uncini. Fairdrop: Biased edge dropout for enhancing fairness in graph representation learning. TAI, 2021.
[96] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. JMLR, 2014.

[97] Y. Sui, X. Wang, J. Wu, A. Zhang, and X. He. Adversarial causal augmentation for graph covariate shift. arXiv preprint arXiv:2211.02843, 2022.

[98] J. Sun, B. Wang, and B. Wu. Automated graph representation learning for node classification. In IJCNN, 2021.

[99] M. Sun, J. Xing, H. Wang, B. Chen, and J. Zhou. Mocl: data-driven molecular fingerprint via knowledge-aware contrastive learning from molecular graph. In KDD, 2021.

[100] Q. Sun, J. Li, H. Peng, J. Wu, X. Fu, C. Ji, and S. Y. Philip. Graph structure learning with variational information bottleneck. In AAAI, 2022.

[101] S. Suresh, P. Li, C. Hao, and J. Neville. Adversarial graph augmentation to improve graph contrastive learning. NeurIPS, 2021.

[102] S. Thakoor, C. Tallec, M. G. Azar, M. Azabou, E. L. Dyer, R. Munos, P. Veličković, and M. Valko. Large-scale representation learning on graphs via bootstrapping. In ICLR, 2022.

[103] N. Tishby, F. C. Pereira, and W. Bialek. The information bottleneck method. arXiv, 2000.

[104] J. Topping, F. Di Giovanni, B. P. Chamberlain, X. Dong, and M. M. Bronstein. Understanding over-squashing and bottlenecks on graphs via curvature. In ICLR, 2022.

[105] P. Trivedi, E. S. Lubana, Y. Yan, Y. Yang, and D. Koutra. Augmentations in graph contrastive learning: Current methodological flaws & towards better practices. arXiv:2111.03220, 2021.

[106] P. Trivedi, E. S. Lubana, M. Heimann, D. Koutra, and J. J. Thiagarajan. Analyzing data-centric properties for graph contrastive learning. In NeurIPS, 2022.

[107] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio. Graph attention networks. arXiv:1710.10903, 2017.

[108] P. Velickovic, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, and R. D. Hjelm. Deep graph infomax. ICLR, 2019.

[109] V. Verma, A. Lamb, C. Beckham, A. Najafi, I. Mitliagkas, D. Lopez-Paz, and Y. Bengio. Manifold mixup: Better representations by interpolating hidden states. In ICML, 2019.

[110] V. Verma, M. Qu, A. Lamb, Y. Bengio, J. Kannala, and J. Tang. Graphmix: Regularized training of graph neural networks for semi-supervised learning. arXiv:1909.11715, 2019.

[111] R. Wang, S. Mou, X. Wang, W. Xiao, Q. Ju, C. Shi, and X. Xie. Graph structure estimation neural networks. In TheWebConf, 2021.

[112] X. Wang, X. Liu, and C.-J. Hsieh. Graphdefense: Towards robust graph convolutional networks, 2019.

[113] Y. Wang, W. Wang, Y. Liang, Y. Cai, and B. Hooi. Graphcrop: Subgraph cropping for graph classification. arXiv:2009.10564, 2020.

[114] Y. Wang, W. Wang, Y. Liang, Y. Cai, J. Liu, and B. Hooi. Nodeaug: Semi-supervised node classification with data augmentation. In KDD, 2020.

[115] Y. Wang, Y. Min, X. Chen, and J. Wu. Multi-view graph contrastive representation learning for drug-drug interaction prediction. In TheWebConf, 2021.

[116] Y. Wang, Y. Min, E. Shao, and J. Wu. Molecular graph contrastive learning with parameterized explainable augmentations. In BIBM, 2021.

[117] Y. Wang, W. Wang, Y. Liang, Y. Cai, and B. Hooi. Mixup for node and graph classification. In TheWebConf, 2021.

[118] J. Wei and K. Zou. Eda: Easy data augmentation techniques for boosting performance on text classification tasks. arXiv:1901.11196, 2019.

[119] S. Wu, H. Zhang, G. Valiant, and C. Ré. On the generalization effects of linear transformations in data augmentation. In ICML, 2020.

[120] Y. Wu, X. Wang, A. Zhang, X. He, and T.-S. Chua. Discovering invariant rationales for graph neural networks. In ICLR, 2021.

[121] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip. A comprehensive survey on graph neural networks. TNNLS, 2020.

[122] Y. Xie, Z. Xu, J. Zhang, Z. Wang, and S. Ji. Self-supervised learning of graph neural networks: A unified review. TPAMI, 2022.

[123] D. Xu, W. Cheng, D. Luo, H. Chen, and X. Zhang. Infogcl: Information-aware graph contrastive learning. NeurIPS, 2021.

[124] H. Xu, L. Xiang, J. Yu, A. Cao, and X. Wang. Speedup robust graph structure learning with low-rank information. In CIKM, 2021.

[125] K. Xu, H. Chen, S. Liu, P.-Y. Chen, T.-W. Weng, M. Hong, and X. Lin. Topology attack and defense for graph neural networks: An optimization perspective. arXiv:1906.04214, 2019.

[126] L. Yang, Z. Kang, X. Cao, D. Jin, B. Yang, and Y. Guo. Topology optimization based graph convolutional network. In IJCAI, 2019.

[127] C. Ying, T. Cai, S. Luo, S. Zheng, G. Ke, D. He, Y. Shen, and T.-Y. Liu. Do transformers really perform badly for graph representation? NeurIPS, 2021.

[128] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec. Graph convolutional neural networks for web-scale recommender systems. In KDD, 2018.

[129] Y. You, T. Chen, Y. Sui, T. Chen, Z. Wang, and Y. Shen. Graph contrastive learning with augmentations. NeurIPS, 2020.

[130] Y. You, T. Chen, Y. Shen, and Z. Wang. Graph contrastive learning automated. In ICML, 2021.

[131] J. Yu, T. Xu, Y. Rong, Y. Bian, J. Huang, and R. He. Graph information bottleneck for subgraph recognition. arXiv:2010.05563, 2020.

[132] J. Yuan, H. Yu, M. Cao, M. Xu, J. Xie, and C. Wang. Semi-supervised and self-supervised classification with multi-view graph neural networks. In CIKM, 2021.

[133] J. Zbontar, L. Jing, I. Misra, Y. LeCun, and S. Deny. Barlow twins: Self-supervised learning via redundancy reduction. In ICML, 2021.

[134] A. Zhang and J. Ma. Defensevgae: Defending against adversarial attacks on graph data via a variational graph autoencoder. arXiv:2006.08900, 2020.

[135] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz. mixup: Beyond empirical risk minimization. arXiv:1710.09412, 2017.

[136] H. Zhang, Q. Wu, J. Yan, D. Wipf, and P. S. Yu. From canonical correlation analysis to self-supervised graph neural networks. NeurIPS, 2021.

[137] X. Zhang and M. Zitnik. Gnnguard: Defending graph neural networks against adversarial attacks. NeurIPS, 2020.

[138] Y. Zhang, S. Pal, M. Coates, and D. Ustebay. Bayesian graph convolutional neural networks for semi-supervised classification. In AAAI, 2019.

[139] Y. Zhang, H. Zhu, Z. Song, P. Koniusz, and I. King. Costa: Covariance-preserving feature augmentation for graph contrastive learning. In KDD, 2022.

[140] T. Zhao, Y. Liu, L. Neves, O. Woodford, M. Jiang, and N. Shah. Data augmentation for graph neural networks. In AAAI, 2021.

[141] T. Zhao, B. Ni, W. Yu, Z. Guo, N. Shah, and M. Jiang. Action sequence augmentation for early graph-based anomaly detection. In CIKM, 2021.

[142] T. Zhao, X. Zhang, and S. Wang. Graphsmote: Imbalanced node classification on graphs with graph neural networks. In WSDM, 2021.

[143] T. Zhao, G. Liu, D. Wang, W. Yu, and M. Jiang. Learning from counterfactual links for link prediction. In ICML, 2022.

[144] T. Zhao, X. Tang, D. Zhang, H. Jiang, N. Rao, Y. Song, P. Agrawal, K. Subbian, B. Yin, and M. Jiang. Autogda: Automated graph data augmentation for node classification. In LoG, 2022.

[145] C. Zheng, B. Zong, W. Cheng, D. Song, J. Ni, W. Yu, H. Chen, and W. Wang. Robust graph representation learning via neural sparsification. In ICML, 2020.

[146] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang. Random erasing data augmentation. In AAAI, 2020.

[147] J. Zhou, J. Shen, and Q. Xuan. Data augmentation for graph classification. In CIKM, 2020.

[148] S. Zhu, Q. Shen, Y. Zhang, Y. Pang, and Z. Wei. Data-augmented counterfactual learning for bundle recommendation. arXiv:2210.10555, 2022.

[149] X. Zhu. Semi-supervised learning with graphs. Carnegie Mellon University, 2005.

[150] X. Zhu and Z. Ghahramani. Learning from labeled and unlabeled data with label propagation. 2002.

[151] Y. Zhu, Y. Xu, F. Yu, Q. Liu, S. Wu, and L. Wang. Deep graph contrastive representation learning. arXiv:2006.04131, 2020.

[152] Y. Zhu, W. Xu, J. Zhang, Y. Du, J. Zhang, Q. Liu, C. Yang, and S. Wu. A survey on graph structure learning: Progress and opportunities. arXiv, 2021.

[153] Y. Zhu, Y. Xu, F. Yu, Q. Liu, S. Wu, and L. Wang. Graph contrastive learning with adaptive augmentation. In

TheWebConf, 2021.

[154] D. Zügner and S. Günnemann. Adversarial attacks on graph neural networks via meta learning. arXiv preprint arXiv:1902.08412, 2019.

[155] D. Zügner and S. Günnemann. Certifiable robustness and robust training for graph convolutional networks. In KDD, 2019.

[156] D. Zügner and S. Günnemann. Certifiable robustness of graph convolutional networks under structure perturbations. In KDD, 2020.

[157] D. Zügner, A. Akbarnejad, and S. Günnemann. Adversarial attacks on neural networks for graph data. In KDD, 2018.