# Graph Neural Networks for Tabular Data Learning: A Survey with Taxonomy & Directions

Cheng-Te Li *Member, IEEE*, Yu-Che Tsai, Chih-Yao Chen, Jay Chiehen Liao

**Abstract**—In this survey, we dive into Tabular Data Learning (TDL) using Graph Neural Networks (GNNs), a domain where deep learning-based approaches have increasingly shown superior performance in both classification and regression tasks compared to traditional methods. The survey highlights a critical gap in deep neural TDL methods: the underrepresentation of latent correlations among data instances and feature values. GNNs, with their innate capability to model intricate relationships and interactions between diverse elements of tabular data, have garnered significant interest and application across various TDL domains. Our survey provides a systematic review of the methods involved in designing and implementing GNNs for TDL (GNN4TDL). It encompasses a detailed investigation into the foundational aspects and an overview of GNN-based TDL methods, offering insights into their evolving landscape. We present a comprehensive taxonomy focused on constructing graph structures and representation learning within GNN-based TDL methods. In addition, the survey examines various training plans, emphasizing the integration of auxiliary tasks to enhance the effectiveness of instance representations. A critical part of our discussion is dedicated to the practical application of GNNs across a spectrum of GNN4TDL scenarios, demonstrating their versatility and impact. Lastly, we discuss the limitations and propose future research directions, aiming to spur advancements in GNN4TDL. This survey serves as a resource for researchers and practitioners, offering a thorough understanding of GNNs' role in revolutionizing TDL and pointing towards future innovations in this promising area.

**Index Terms**—graph neural networks, tabular data learning, graph representation learning, graph structure learning, survey paper



## 1 Introduction

The deep learning-based approaches to Tabular Data Learning (TDL), e.g., classification and regression, have exhibited promising performance in recent years [8]. However, despite the great ability to learn effective feature representations from raw tabular records, the deep neural TDL provides weak modeling on the latent correlation among data instances and feature values. The prediction performance of TDL has been shown to improve by modeling high-order instance-feature relationships [157], high-order feature interactions [83], and multi-relational correlation between data instances [51]. As a natural antidote to model relations and interactions between different data entities, graph neural networks (GNNs) have recently received tremendous attention [145]. By properly constructing the graph structures from the input tabular data, GNNs can learn latent correlations between data elements and generate effective feature representations for prediction tasks. Inspired by the success of GNNs on natural language processing [140] and recommender systems [144], there is also an increasing trend towards developing **Graph Neural Networks for Tabular Data Learning** (GNN4TDL).

Currently, a number of early research efforts have attempted to apply existing GNN methods for tabular data learning, e.g., [25], [44], [93], [110], [112]. Some very recent studies [33], [51], [62], [83], [91], [157] also have started to explore TDL-specific GNNs. These studies almost span all TDL topics and applications, setting off a wave of research enthusiasm in this field. Some essential questions also arise with this research progress. (a) *What are the differences between GNN-based TDL and conventional TDL?* (b) *What are the proper ways to construct graph structures under different TDL scenarios and tasks?* (c) *What is the principle behind GNN-based tabular data representation learning?* (d) *What are the TDL tasks and application domains that can benefit from GNNs?* (e) *What are the limitations of current research and potential opportunities for future research?* Although encouraging results are reported in recent studies of GNN4TDL, these questions are not systematically investigated or even neglected. There is an urgent need for this GNN4TDL survey to reveal the answers to these questions in order to promote this line of research further.

We believe this GNN4TDL survey will be placed at a high value because of the high demand and low support on this topic. (a) *High Demand*: since tabular data is ubiquitous in many domains and applications, and people have gradually shifted their focus to model the relationships between data instances and their correlation with feature values, we believe graph neural networks for tabular data learning shall have not only high research impact but also practical values. It shall be able to receive attention from both academia and industry. (b) *Low Support*: Our GNN4TDL falls in a niche yet crucial area largely over-


- *Cheng-Te Li, Department of Computer Science and Information Engineering, National Cheng Kung University, Tainan, Taiwan. Email: chengte@ncku.edu.tw*
- *Yu-Che Tsai, Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan. Email: roytsai27@gmail.com*
- *Chih-Yao Chen, University of North Carolina at Chapel Hill, NC, USA. Email: cychen@cs.unc.edu*
- *Jay Chiehen Liao, Department of Computer Science and Information Engineering, National Cheng Kung University, Tainan, Taiwan. Email: jay.chiehen@gmail.com*


*Github page: https://github.com/Roytsai27/awesome-GNN4TDL*
*Version date: January 1, 2024*

TABLE 1: A comparison between existing surveys on graph neural networks-based application domains. *TDP*: tabular data prediction; *GRL*: graph representation learning; *GSL*: graph structure learning; *SSL*: self-supervised learning; *TS*: training strategy; *AT*: auxiliary task; App: Applications.

| | Year | Domain | Data | TDP | GRL | GSL | SSL | TS | AT | App |
|---|---|---|---|---|---|---|---|---|---|---|
| Wu et al. [145] | 2021 | Graph Machine Learning | Graph | | ✓ | | | | | ✓ |
| Zhang et al. [166] | 2021 | CTR Prediction | Tabular | ✓ | | | | | | ✓ |
| Wang et al. [133] | 2021 | Recommender Systems | User-Item | | ✓ | | | | | ✓ |
| Borisov et al. [8] | 2022 | Tabular Data Learning | Tabular | ✓ | | | ✓ | | | ✓ |
| Wu et al. [144] | 2022 | Recommender Systems | User-Item | | ✓ | | | | | ✓ |
| Zhu et al. [174] | 2022 | Graph Machine Learning | Graph | | ✓ | ✓ | | ✓ | ✓ | ✓ |
| Zhang et al. [168] | 2022 | Graph Machine Learning | Graph | | ✓ | | | ✓ | ✓ | ✓ |
| Wu et al. [140] | 2023 | Natural Language Processing | Text | | ✓ | ✓ | | | | ✓ |
| Wu et al. [139] | 2023 | Recommender Systems | User-Item | | ✓ | | ✓ | | | ✓ |
| Ma et al. [98] | 2023 | Anomaly Detection | Graph | | ✓ | | | ✓ | | ✓ |
| Gao et al. [40] | 2023 | Recommender Systems | User-Item | | ✓ | | | | | ✓ |
| Liu et al. [90] | 2023 | Graph Machine Learning | Graph | | ✓ | | ✓ | ✓ | ✓ | ✓ |
| Dong et al. [24] | 2023 | Internet of Things | Sensor | | ✓ | | | | | ✓ |
| Jiao et al. [64] | 2023 | Computer Vision | Image | | ✓ | | | | | ✓ |
| Jin et al. [65] | 2023 | Time Series Analysis | Sensor | | ✓ | ✓ | | | | ✓ |
| This survey (GNN4TDL) | 2024 | Tabular Data Learning | Tabular | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |

looked in previous surveys, according to a summary of comparisons in Table 1. Unlike other works that concentrate on broader GNN applications across various domains and data types, this survey not only highlights the potential of GNNs in tabular data prediction, representation learning, and graph structure learning, but also pioneers in discussing self-supervised learning, various training strategies, and auxiliary tasks specifically in GNN4TDL.

This survey paper presents an in-depth exploration of applying GNNs in tabular data learning. It first establishes the fundamental problem statement and introduces various graph types used to represent tabular data. The survey is structured around a detailed GNN-based learning pipeline, encompassing phases include *Graph Formulation*, where tabular elements are converted into graph nodes; *Graph Construction*, focusing on establishing connections within these elements; *Representation Learning*, highlighting how GNNs process these structures to learn data instance features; and *Training Plans*, discussing the integration of auxiliary tasks and training strategies for enhanced predictive outcomes. Beyond the review of GNN4TDL techniques, the survey further illustrates GNN applications in diverse fields, such as fraud detection and precision medicine, alongside a critical discussion on the current research limitations and future directions in the field of GNN4TDL.

We summarize the contributions of this survey as below.

- We provide a broad picture of the current development of graph neural networks for tabular data learning. A timely and comprehensive literature review is presented to help readers quickly grasp the basic concepts and step into this research field.
- We organize existing arts of applying GNNs to tabular data learning. Particularly, we dive into why and how GNNs can better model tabular data and demystify the performance gains of tabular data classification and regression brought by GNNs. In practice, we highlight the basic guidelines for constructing various graph structures for tabular data modeling.
- We demonstrate how GNN can be utilized in many tabular data application domains, such as fraud detection, precision medicine, click-through rate prediction, and handling missing data. We also provide academia and industry with an insightful discussion of the limitations in current research and future research directions on GNN4TDL.

We organize this paper as follows. Section 2 defines the related concepts used in the remaining sections. Section 3 describes the framework of GNN4TDL and provides categorization from multiple perspectives. Section 4 systematically reviews existing GNN4TDL methods based on our categorization. Section 5 surveys the real-world applications of GNN4TDL in various domains. Section 6 discusses the remaining challenges and possible future directions. Section 7 concludes this survey in the end.

## 2 Preliminaries

This section covers the common and necessary elements of GNN4TDL. We first provide the general problem statement of tabular data learning (Sec. 2.1). Then, we give the definitions and notations of various graphs commonly adopted to depict tabular data (Sec. 2.2). We also describe the basic ideas of representation learning with GNNs (Sec. 2.3), followed by different learning tasks (Sec. 2.4) that can be used to design the training objectives in GNN4TDL. Besides, we further discuss why GNNs are required for TDL (Sec. 2.5).

### 2.1 Tabular Data Learning

We consider the supervised learning setting. Let $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$ denote a tabular dataset with $n$ data instances, where $\mathbf{x}_i = (\mathbf{x}_i^{(\text{num})}, \mathbf{x}_i^{(\text{cat})}) \in \mathcal{X} \subseteq \mathbb{R}^d$ represents numerical feature $x_{ij}^{(\text{num})}$ and categorical features $x_{ij}^{(\text{cat})}$, and $y_i \in \mathcal{Y}$ denotes the label of data instance $i$. The total number of features is denoted as $d$. The dataset can be split into three disjoint subsets, $D = D_{train} \cup D_{val} \cup D_{test}$, where $D_{train}$ is used for training, $D_{val}$ is used for early stopping and hyperparameter tuning, and $D_{test}$ is used for testing. The common tabular data learning tasks have three types: binary classification $\mathcal{Y} = \{0, 1\}$, multi-class classification $\mathcal{Y} = \{1, 2, ..., C\}$, and regression $\mathcal{Y} = \mathbb{R}$, where $C$ is the number of class labels.

We assume that every input feature vector $\mathbf{x}_i$ in $D$ is sampled i.i.d. from a feature distribution $p_X$, and the labeled data pairs $(\mathbf{x}_i, y_i)$ in $D$ are drawn from a joint distribution $p_{X,Y}$. When some labeled samples from $p_{X,Y}$ are available, we can train a predictive model $f : \mathcal{X} \to \mathcal{Y}$ by minimizing the empirical supervised loss $\sum_{i=1}^{|D_{train}|} \mathcal{L}(f(\mathbf{x}_i), y_i)$, where $\mathcal{L}$ is a standard supervised loss function, e.g., cross-entropy (CE) for classification and mean squared error (MSE) for regression.

### 2.2 Notations on Graphs

We provide the definitions of different types of graphs with notations used in this paper.

**Homogeneous Graph.** A homogeneous graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, v_2, ..., v_n\}$ ($|\mathcal{V}| = n$) is the set of nodes, $\mathcal{E}$ is the set of edges, and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. The set of neighbors of node $v_i$ is denoted as: $\mathcal{N}(v_i) = \{v_j \in \mathcal{V} | e_{i,j} \in \mathcal{E}\}$. The graph structure can also be depicted by an adjacency matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$, where $\mathbf{A}_{i,j} = 1$ indicates $e_{i,j} \in \mathcal{E}$ and $\mathbf{A}_{i,j} = 0$ means $e_{i,j} \notin \mathcal{E}$.

**Attributed Graph.** An attributed graph contains features (e.g., attributes) associated with nodes or edges or both. In TDL, it is common to consider that only nodes have features and use $\mathbf{X} \in \mathbb{R}^{n \times d}$ to denote the node feature matrix. The attributed graph can be depicted by $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$

**Heterogeneous Graph.** A heterogeneous graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a graph that contains different types of nodes or edges or both. Each node is assocaited with a mapping function $\phi_v(v_i)$ that maps $v_i \in \mathcal{V}$ into the corresponding type, i.e., $\phi_v(v_i) : \mathcal{V} \to \mathcal{S}_v$, where $\mathcal{S}_v$ is the set of node types. Each edge is also assocaited with a mapping function $\phi_e(e_{i,j})$ that maps $e_{i,j} \in \mathcal{E}$ into the corresponding type, i.e., $\phi_e(e_{i,j}) : \mathcal{E} \to \mathcal{S}_e$, where $\mathcal{S}_e$ is the set of edge types. $|\mathcal{S}_v| + |\mathcal{S}_e| > 2$.

Note that bipartite, multi-partite, and multiplex graphs can be treated as three special types of heterogeneous graphs. For example, a bipartite graph is a heterogeneous graph with two types of nodes ($|\mathcal{S}_v| = 2$) and a single type of edge ($|\mathcal{S}_e| = 1$). A multi-partite graph is with multiple types of nodes ($|\mathcal{S}_v| > 1$) and multiple types of edges ($|\mathcal{S}_e| = |\mathcal{S}_v| - 1$). A multiplex graph has only one type of nodes ($|\mathcal{S}_v| = 1$) and multiple types of edges ($|\mathcal{S}_e| > 1$).

### 2.3 Graph Neural Networks (GNNs)

**Node Representation Learning in GNNs.** Given an attributed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$, where $\mathbf{x}_i$ is the $d$-dimensional feature vector of node $v_i$, a GNN algorithm can learn to generate the node representation $\mathbf{h}_i$ for every node $v_i \in \mathcal{V}$ by implementing two functions. Suppose we are training a $K$-layer GNN, the node embedding at the $k$-th layer, i.e., $\mathbf{h}_i^{(k)}$, can be obtained via:

$$\mathbf{a}_i^{(k)} = \text{aggregate}^{(k)}\left(\mathbf{h}_j^{(k-1)} : v_j \in \mathcal{N}(v_i)\right),$$
$$\mathbf{h}_i^{(k)} = \text{combine}^{(k)}\left(\mathbf{h}_i^{(k-1)}, \mathbf{a}_i^{(k)}\right),$$

where $\mathbf{h}_i^{(0)} = \mathbf{x}_i$, $\mathbf{h}_i = \mathbf{h}_i^{(K)}$, and $\text{aggregate}^{(k)}(\cdot)$ and $\text{combine}^{(k)}(\cdot)$ are the aggregation and combination functions at the $k$-th layer, respectively. The derived node representation $\mathbf{h}_i$ can be directly utilized for downstream tasks. Note that there are a variety of designs for aggregation and combination functions in different GNNs. One can refer to the general GNN survey paper [145] to find how such two functions can be fulfilled.

**Graph Representation Learning in GNNs.** Given an attributed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{X})$, along with the derived node representations $\mathbf{h}_i$ for every node $v_i \in \mathcal{V}$, we can generate the representation of the entire graph $\mathbf{h}_\mathcal{G}$ based on a *readout* layer, given by:

$$\mathbf{h}_\mathcal{G} = \mathcal{R}\left(\{\mathbf{h}_i | v_i \in \mathcal{V}\}\right),$$

where $\mathcal{R}(\cdot)$ is the readout function that maps the embeddings of nodes into a graph-level representation. The implementation of $\mathcal{R}(\cdot)$ can be a simple permutation-invariant function like summation or pooling methods (e.g., DiffPool [154] and TopKPool [41]).

### 2.4 Downstream Tasks

In this section, we discuss how to formulate tabular data representation learning as the problem of graph representation learning, and treat the tabular predictions as different graph-related downstream tasks. The common downstream tasks include node-level, edge-level, and graph-level variants.

- **Node-level Tasks** refer to node classification and node regression. The task aims at predicting the label $y_i \in \mathcal{Y}$ for every node $v_i \in \mathcal{V}$. Given the obtained node representation $\mathbf{h}_i$, by taking node classification as an example, the typical approach is to feed $\mathbf{h}_i$ into a multilayer perceptron (MLP) layer with Softmax function to produce prediction outcomes. The cross-entropy loss is typically adopted for model training.
- **Link-level Tasks** refer to link prediction, edge classification, and edge regression. Link prediction, a binary classification task $y_{i,j} \in \mathcal{Y} = \{0, 1\}$, predicts whether one node $v_i$ will connect to the other node $v_j$ given their corresponding embeddings $\mathbf{h}_i$ and $\mathbf{h}_j$. Edge classification aims to classify edges into multiple classes, and edge regression aims to regress an edge into a real value. For link prediction, a typical approach is to feed the vector concatenation of $\mathbf{h}_i$ and $\mathbf{h}_j$ into an MLP layer with the Softmax function to produce the outcome.
- **Graph-level Tasks** refer to graph classification and graph regression, in which the input is graph representation $\mathbf{h}_\mathcal{G}$. Taking graph classification as an example, each graph $\mathcal{G}_i$ is associated with a target label $y_i \in \mathcal{Y}$. The goal is to train a model to predict its label. A typical approach is to feed $\mathbf{h}_\mathcal{G}$ into an MLP classifier with the Softmax function to generate the prediction results.

### 2.5 Why are GNNs required for TDL?

Recent advances in tabular data learning have developed a rapidly growing branch: leveraging GNNs to obtain better instance representations to improve performance on downstream tasks. GNN-based tabular data learning methods have also achieved the state-of-the-art in various applications, such as click-through rate prediction [73], medical risk prediction with electronic health records [54], anomaly detection [44], and missing data imputation [157]. We summarize why GNNs can benefit tabular data learning in the

following five perspectives: (a) instance correlation, (b) feature interaction, (c) high-order connectivity, (d) supervision signal, and (e) inductive capability.

**Instance Correlation.** Data instances can be correlated with each other in terms of their features. For example, users with similar profiles or online behaviors tend to have similar preferences for ads or items [32], [50], [108]. Patients with similar clinical data or symptoms have a higher potential to suffer from similar diseases [10], [20], [54]. To better represent instances for downstream tasks, rather than solely employing each instance's self-features, it is crucial to model the correlation among instances. The key idea is to exploit such correlation to learn higher-quality feature representations of instances. Specifically, instances with similar downstream labels are close to one another, while those with different labels are pushed away from each other in the embedding space. With a proper construction of the graph structure that depicts the relationships between instances, GNN can be a good fit to let instances learn to represent each other. That said, we require GNNs to model instance correlation.

**Feature Interaction** refers to the effect of feature combination on the tabular prediction label. The positive contribution of one feature to the prediction task may depend on other features. Therefore, learning feature interactions can bring potential performance improvement. The conventional approach to obtain feature interactions is handcrafted by enumerating some possible combinations of features, such as $\{\text{gender} = male \ \& \ \text{age} > 20\}$ and $\{\text{gender} = female \ \& \ \text{age} > 20 \ \& \ \text{job} = lawyer\}$, which are second-order and third-order interactions, respectively. However, hand-crafting is time-consuming and requires domain knowledge. Deep neural network-based methods, such as Wide&Deep [18], DeepFM [49], Deep&Cross [132], xDeepFM [84], and Cross-GCN [36], can automatically learn feature interactions in an implicit fashion. Nevertheless, these methods merely combine the learned feature embeddings via simple concatenation. The *structured correlation* among different features cannot be modeled. By considering features or instances as nodes in a graph, edges can depict the potential interactions among features or instances. GNN can naturally learn sophisticated feature interactions and produce a single embedding that captures the structured correlation among features.

**High-order Connectivity.** Tabular data learning aims at making predictions based on feature values. Instances possessing similar features tend to obtain the same prediction labels. In scenarios of utilizing deep neural networks in tabular data learning [8], likewise, deep models (e.g., VIME [156], TabNet [51], SubTab [125], and SCARF [6]) generate similar feature representations for instances with similar raw features in the learned embedding space such that they tend to be predicted as the same or close labels. These typical approaches implicitly realize such an idea based on the direct interactions between instances and their features in the training data. This utilization can be termed as the *first-order connectivity*. To better learn feature representations of instances and to improve the prediction performance, the *high-order connectivity* among instances [51], among features [83], and among instances and features [157] can be further considered. GNN-based methods can essentially model high-order interactions among data elements effectively. The utilization of high-order connectivity can be learned by the message passing and aggregation mechanisms in GNNs that receive embedding propagation from multi-hop neighbors in the graph.

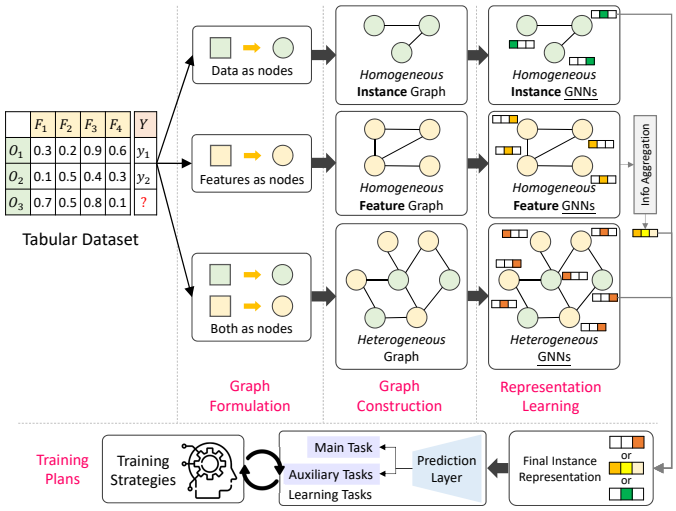


Fig. 1: The general pipeline of a GNN-based tabular data learning model. Given the tabular data as the model input, we first formulate the graph structure considering data instances and features, construct the graph structure, learn feature representations using GNNs, and train to generate the prediction outcomes, along with some auxiliary tasks.

**Supervision Signal.** In real-world applications, such as fraud detection, medical prediction, and precision marketing, collecting a sufficient number of labeled tabular data is usually challenging. We face limited supervision signals when performing tabular data learning. One of the essential characteristics of graph neural networks is *semi-supervised learning*, which can propagate the supervised information from the labeled instances to unlabeled ones via the graph structure so that better feature representations can be obtained and the supervision sparsity issue can be mitigated. Furthermore, recent advances in tabular data learning have attempted to incorporate *self-supervised singals* into the representation learning process of tabular instances [6], [125], [156]. Developing auxiliary learning tasks on the features and having them trained together with the main task can further improve the performance of downstream tasks. A potential direction is to jointly design the self-supervised tasks based on both features and the graph structure and leverage the semi-supervised learning of GNNs.

**Inductive Capability.** Graph neural networks exhibit the ability and *inductive* power to generalize the message-passing mechanism to unseen nodes or even entirely new graphs [52], [126], [161]. A GNN model possesses inductive capability by adjusting its weights based on the supervised examples in the training data, and the model can be applied to new testing instances without further accessing the training set. Hence, when instances and features in a tabular dataset are represented as nodes, and their various interactions are treated as edges in a constructed graph, learning representations based on GNNs can make tabular

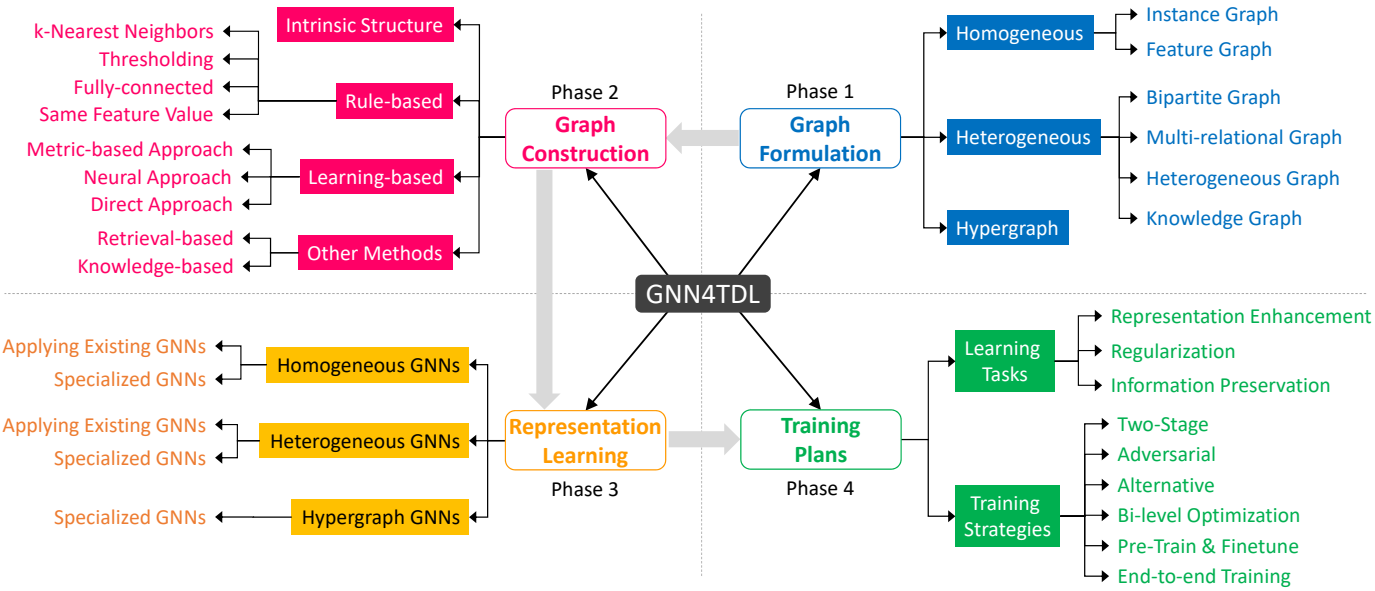


Fig. 2: The taxonomy systematically organizes GNNs for tabular data learning along four axes: graph formulation, graph construction, graph representation learning, and tabular training.

data predictions capable of inductivity power. For example, the GNN-based tabular data learning will be able to deal with additional new features at the testing stage (i.e., feature extrapolation) [142], produce the initialized embeddings for new data instances [105], generalize to unseen tasks (not learned at the training stage) [11]. Elements in tabular data will be endowed with inductive capability thanks to GNNs.

## 3 Pipeline & Categorization

In this section, we present the overarching pipeline of GNNs for tabular data learning, followed by a detailed categorization of each key phase, delineating how diverse methodologies fulfill these stages. Accompanying this categorization, we provide descriptions of select representative frameworks within each category. These studies exemplify the intricate interconnections among various phases or categories in the pipeline, highlighting their tight coupling and collaborative function in the overall GNN4TDL process. The detailed elaborations on every categorization are presented in Section 4.

**Pipeline.** The general pipeline of GNN-based tabular data learning is provided in Figure 1. The pipeline begins with the *Graph Formulation* phase, where the structure of the graph is defined using elements from the tabular dataset. This phase involves deciding which elements to use as nodes, with three common approaches: (1) representing data instances as nodes, (2) using features as nodes, or (3) a combination of both, forming different kinds of graph types. Following this, the *Graph Construction* phase aims to create connections among these elements, transforming the tabular data into a graph structure. This structure is determined by the initial formulation, leading to either a homogeneous graph (e.g., instance graph or feature graph) or a heterogeneous graph (e.g., bipartite graph, multi-relational graph, or hypergraph). Next, the *Representation Learning* phase involves applying different types of GNNs based on the graph's nature. Various homogeneous instance GNNs, homogeneous feature GNNs, or heterogeneous GNNs are employed to learn feature representations of data instances. This phase is crucial as it determines how messages are propagated through the graph, modeling interactions among features and instances and influencing the quality of the learned embeddings. If the feature graph is used, an additional information aggregation layer is required to produce the final instance representation based on the learned embeddings of features. Finally, the *Training Plans* phase receives the final instance representations. In this phase, different learning tasks and training strategies are employed, including the use of auxiliary tasks alongside the main task. The outcome is then processed through a prediction layer to produce the final prediction outcomes. This comprehensive pipeline highlights the versatility of GNNs in handling various graph formulations and learning tasks, ultimately leading to effective tabular data learning and prediction.

**Categorization.** The taxonomy of building graph neural networks for tabular data learning can be established according to the pipeline. We give the taxonomy in Figure 2. Below, we accordingly describe the categorizations in the taxonomy, in which some representative studies in each category are mentioned and summarized in Table 2.

1) The formulations of graphs from tabular data contain three main types: *homogeneous graphs*, *heterogeneous graphs*, and *hypergraphs*. Based on data instances as nodes or features as nodes, in homogeneous graphs, we can formulate instance graphs (e.g., [85], [91], [112]) and feature graphs (e.g., [83], [152], [173]), respectively. On the other hand, a heterogeneous graph can connect data instances to their corresponding features (and further to other metadata). The formulation of heterogeneous graphs can be either bipartite or multi-partite graphs [27], [63], [142], [157]. One can also consider different feature values as various edge types, which depict different

TABLE 2: List of representative GNN4TDL studies and their graph formulation settings. In column "TDL Task": Classification (Cla), Regression (Reg), Anomaly Detection (AD), Table Understanding (TU). Intrinsic (Sec. 4.2.1), Rule (Sec. 4.2.2), Learned (Sec. 4.2.3). In column "Graph Type", Homogeneous (Homo) (Sec. 4.1.1), Heterogeneous (Hete) (Sec. 4.1.2), and Hypergraph (Sec. 4.1.3) graphs.

| | Venue 'Year | Graph Type | Node | Edge | Node Initial Feature | Graph Task | TDL Task |
|---|---|---|---|---|---|---|---|
| FI-GNN [83] | CIKM '19 | Homo | Feature | Rule | One-hot | Graph | Cla |
| GCT [20] | AAAI '20 | Hete | Instance, Feature value | Learned | Random | Graph | Cla |
| GNNDB [22] | ICLR '20 | Hete | Instance | Foreign key reference | Raw feat. | Node | Cla |
| IDGL [16] | NeurIPS '20 | Homo | Instance | Learned | Raw feat. | Node | Cla |
| GRAPE [157] | NeurIPS '20 | Hete-Bipartite | Instance, Feature | Intrinsic | **1** (inst.), One-hot (feat.) | Node, Edge | Cla, Reg |
| HSGNN [93] | BigData '20 | Hete | Instance, Feature value | Intrinsic | One-hot | Node | Cla |
| GME [105] | SIGIR '21 | Hete | Instance, Feature | Intrinsic | Random | Node | Cla |
| XFraud [110] | VLDB '21 | Hete | Instance, Feature value | Intrinsic | Raw feat. | Node | Cla |
| TabGNN [51] | DLP-KDD '21 | Hete-Multiplex | Instance | Rule | Raw feat. | Node | Cla, Reg |
| FIVES [149] | KDD '21 | Homo | Feature | Search | One-hot | Graph | Cla |
| TabularNet [28] | KDD'21 | Homo | Cell | WordNet | Pretrained BERT | Node | Cla |
| SLAPS [33] | NeurIPS '21 | Homo | Instance | Learned | Raw feat. | Node | Cla |
| FATE [142] | NeurIPS '21 | Hete-Bipartite | Instance, Feature | Intrinsic | **0** (inst.), One-hot (feat.) | Node | Cla |
| DGM [69] | TPAMI '22 | Homo | Instance | Learned | Raw feat. | Node | Cla |
| LUNAR [44] | AAAI '22 | Homo | Instance | Rule | Raw feat. | Node | AD |
| SUBLIME [91] | WWW '22 | Homo | Instance | Learned | Raw feat. | Node | Cla |
| MetaLink [11] | ICLR '22 | Hete-Bipartite | Instance, Task | Intrinsic | Raw Feat. (inst.), Random (task) | Node | Cla, Reg |
| PET [27] | NeurIPS '22 | Hete-Bipartite | Instance, Feature value | Intrinsic | **0** (inst.), Random (others) | Node | Cla |
| HCL [10] | SDM '22 | Hypergraph | Feature value | Instance | One-hot | Edge | Cla |
| Table2Graph [173] | IJCAI '22 | Homo | Feature | Learned | One-hot | Graph | Cla |
| T2G-Former [152] | AAAI '23 | Homo | Feature | Learned | Tokenized Feature vector [46] | Graph | Cla, Reg |
| IGRM [63] | AAAI '23 | Hete-Bipartite | Instance, Feature | Intrinsic | **1** (inst.), One-hot (feat.) | Edge | Reg |
| EGG-GAE [123] | AAAI '23 | Homo | Instance | Learned | Raw feat. | Node | Cls, Reg |
| DRSA-Net [170] | TKDE '23 | Homo | Feature | Learned | Feature value | Graph | Cla, Reg |
| HES-GSL [141] | TNNLS '23 | Homo | Instance | Learned | Raw feat. | Node | Cla |
| GNN4MV [59] | TNNLS '23 | Homo | Instance | Rule | Raw feat. | Node | Cla |
| GraphFC [118] | CIKM '23 | Hete | Instance, Feature | Intrinsic | One-hot | Node | Cla, Reg |
| HyTrel [15] | NeurIPS '23 | Hypergraph | Cell | Intrinsic | Tokenized Feature vector [46] | Edge | TU |
| PALTO [114] | NeurIPS '23 | Homo | Feature | Other | Raw feat. | Graph | Cla |
| CCNS [31] | NeurIPS '23 | Homo | Instance | Rule | Raw feat. | Node | Cla |
| TabGSL [85] | Arxiv '23 | Homo | Instance | Learned | Raw feat. | Node | Cla |
| RelBench [37] | Arxiv '23 | Hete | Entry | Primary-foreign key | Raw feat. | Node, Edge | Cla, Reg |

relationships between data instances, and thus formulate multiplex/multi-relational graphs [51], [60], [89]. If a formulation allows data instances and all possible values of features to appear in a graph, one can construct heterogeneous graphs to represent the complicated information interdependency [22], [37], [93], [118]. As for the formulation of hypergraphs [10], [15], [27], tabular elements sharing the same attributes are linked by an edge. An edge in a hypergraph can join any number of tabular elements. For example, instances sharing the same feature values can be linked by an edge in a hypergraph.

2) Given a certain graph formulation, where nodes have already been determined, the second phase aims at constructing a graph by creating edge connections between nodes to substantiate that formulation. According to the criterion of edge creation, in general, there are four types of approaches, including **intrinsic structure**, **rule-based**, **learning-based**, and **other approaches**, where the first two types are widely adopted. The intuitive way to create links is to utilize the inherent relationships among tabular data elements, e.g., an instance contains feature values [142], [157], two instances share the same values of a specific feature [51], [95], a data tables is related to another via the primary-foreign key relationships [22], [37]. To define the edges between data instances and/or features, the rule-based approach relies on some manually-specified heuristics, such as k-nearest neighbors [44], [59], [112], fully-connected structure [56], [83], [119], and thresholding [21], [29]. The learning-based approach automatically generates edges between nodes. It can be divided into three sub-categories: The *metric-based method* uses kernel functions to compute edge weights based on node similarities [69], [123]. The *neural method* employs deep neural networks for adaptive graph construction [85], [91], [152]. The *direct method* views the adjacency matrix as learnable [39], [97]. Other approaches belong to either retrieval-based or knowledge-based. The retrieval-based approach resorts to either discover relevant and similar data instances to build the edges based on *information retrieval* techniques [27], or perform *neural architecture search* to find a better graph topology for representation learning [149]. The knowledge-based approach requires domain experts to provide either the knowledge on the correlation between data instances [28] or the knowledge graph that depicts the relationships among features [114], so that the graph can be constructed in a fine-grained manner.
3) Once the graph that depicts tabular data is derived, no matter how data instances and their corresponding features are depicted via the graph structure, the next phase is to learn the final representation of each instance. Based on the type of the obtained graph, e.g., homogeneous or heterogeneous graphs, we can utilize *homogeneous* GNN models (e.g., GCN [77], GraphSAGE [52], GAT [126], and GIN [151]) and *heterogeneous* GNN models (e.g., RGCN [115], HGAT [134], and HGT [58]) to produce the embedding of every instance. In addition to simply applying existing GNN models, some existing efforts have developed *GNNs specialized* to better capture the various complicated interactions between instances and features (e.g., [20], [44], [59], [83], [152]).
4) Designing a proper training plan based on the learned feature representation of instance is the last mile. The

training plan can be discussed from two aspects, *learning tasks* and *training strategy*. While the main task is to predict the target label, a variety of supervision variants are developed to enhance the learning, and thus, different auxiliary tasks can be constructed. For example, leveraging the contrastive learning to better refine the graph structure learning [85], [91], introducing self-supervised learning with autoencoder to produce the denoised features [33], and imposing various graph regularizations to stabilize the graph learning and avoid overfitting [16], [97]. Since the data is inherently in the tabular form, additional learning tasks can preserve the properties in the input tabular data, such as global statistics of features [119], domain-knowledge preservation [54], and spatial information encoding [28]. A range of training strategies is employed to optimize GNN4TDL performance. Two-stage methods (e.g., [91]) sequentially learn graph structures and then train prediction models. Adversarial techniques (e.g., [119]) enhance the realism of feature reconstruction. Alternative approaches (e.g., [14]) dynamically adjust feature reconstruction weights for improved task relevance. Bi-level optimization (e.g., [142]) concurrently tunes GCN parameters and graph generation. Pretrain-Finetune strategies (e.g., [118]) leverage self-supervised learning for robust initial data understanding, followed by targeted finetuning, albeit with potential stage mismatches. End-to-end training (e.g., [51]) is the widest adopted strategy, offering streamlined learning-to-prediction processes, directly enhancing improved performance.

## 4 METHODS

In this section, by following the pipeline and taxonomy presented in Section 3, we describe the main ideas of various graph neural network-based methods for tabular data learning. We also highlight the weaknesses and discuss the advantages of each approach.

### 4.1 Graph Formulation

#### 4.1.1 Homogeneous Graphs

Tabular data can be represented by a homogeneous graph so that the correlation between instances or the interactions between features can be captured. *Instance graphs* are created to depict the relationships between data samples while *feature graphs* can be constructed to model the associations between various features.

**Instance Graphs.** Most of existing methods formulate tabular data samples and their correlation as instance graphs for graph neural networks (e.g., [16], [31], [33], [39], [44], [59], [85], [91], [112], [119], [141], [146]). The construction of instance graphs relies on all features, and thus the graphs can been seen as modeling the *global* relationships among data samples. That said, the instance graph is hard to capture the relationships between instances from multiple different aspects, i.e., *local* persepctives in terms of feature subsets. In addition, we find that when creating instances, the numbers of features in the tabular datasets are usually small, e.g., $\leq 20$ in LSTM-GNN [112], $\leq 64$ in SUBLIME [91], LUNAR [44], SLAPS [33] and IDGL [16], $\leq 36$ in GINN [119], and $\leq 40$ in TabGSL [85]. The potential reason is three-fold. The first is simplicity and interpretability. With a limited number of features, the relationships between nodes in the graph, in which each feature can be represented as an attribute of a node, can be more easily understood. The second is dimensionality issue. High-dimensional data can lead to overly complex graphs, which can be difficult to analyze and can lead to overfitting or the "curse of dimensionality" in graph structure learning [100]. The third is avoiding noise: In datasets with many features, not all features are necessarily meaningful or beneficial for the task at hand [131]. Creating the instance graph with a large number of features can introduce noise that hinders the performance of graph neural networks.

**Feature Graphs.** Modeling the interactions between features of a data instance as a homogeneous graph, in which nodes are features and edges are their correlation, is also widely adopted in the literature of tabular data learning (e.g., [3], [53], [71], [83], [92], [114], [127], [149], [152], [162], [170], [173]). Most aim at learning high-order feature interactions based on the constructed feature graphs through stacking multi-layer GNNs and aggregate the feature embeddings to the instance representation for final predictions. A recent method, Casual-GNN [162], builds a causal feature graph to capture causal feature relationships based on causal discovery among field features. Two common ways are required to preprocess numerical and categorical features. One is transforming one-hot vectors of categorical features into low-dimensional dense vectors through field-wise embedding while directly feeding numerical features into the model (e.g., FI-GNN [83], GCN-INT [92], Table2Graph [173], and Causal-GNN [162]). The other relies on feature tokenizer [46], such as T2G-Former [152] and FT-GAT [71]. Factorization machines [111] can also be used to give initial embeddings of features [53]. Although the relationships between features can be learned in an automatic way, we may require significant domain knowledge to define the dependencies of some features, such as clinical variables in the medical [45] and financial domains [171]. Incorrect or oversimplified dependencies could lead to poor performance.

#### 4.1.2 Heterogeneous Graphs

Heterogeneous graphs can depict complex and diverse relationships. They can depict instance-instance relationships, feature-feature relationships, as well as instance-feature relationships, providing a richer and more comprehensive representation of the tabular data. Considering nodes and edges can possess different information, the relevant methods can be divided into five categories in formulating a tabular dataset, including general heterogeneous graphs, bipartite graphs, multi-relational graphs, semantic graphs, multiplex graphs, and knowledge graphs.

**General Heterogeneous Graphs.** Various information in a tabular data can be considered as different types of nodes to form a general heterogene graph. The basic idea is to take categorical feature values as nodes, along with the connections to nodes of data instances. Typical examples include product brands and item categories in click-through rate prediction [105], diagnosis and treatment codes in electronic health records [20], [93], importer companies and categories

of goods in customs import declarations [118], payment token and shipping address in credit card transactions [110], and products and devices in online product reviews [130]. The relational databases can be also coverted into a heterogeneous graph by treating rows as nodes, data tables as node types, and foreign key columns as edge types [22], [37], [163]. The formulation into heterogeneous graphs can model the complex relationships between different types of entites. Nevertheless, not all categorical attributes can be utilized as nodes. A strategy to select categorical attributes as nodes is to conduct the *homophilic tests* [5]. While GNNs work well due to the homophily assumption [113], [138], i.e., most connections happen among nodes in the same class or with alike features [102], attributes with strong homophilic effects are proper candidates to be used as nodes in the formulation of heterogeneous graphs.

**Bipartite Graphs.** A bipartite graph is a type of graph where nodes are divided into two distinct groups, and connections (edges) only exist between nodes from different groups, not within a group. For a tabular dataset, one common formulation adopted by existing studies [42], [54], [61], [121], [142], [157] as a bipartite graph is to treat the instances (rows) as one set of nodes, the features (columns) as the other set of nodes, and feature values as edge weights. The other formulation is to take feature values as one set of nodes [27], which is designed to enhance the interactions between features. Using a bipartite graph to depict tabular data offers several advantages. (a) By representing instances and features separately, the bipartite graph can maintain the original structure of the tabular data. (b) Bipartite graphs can handle diverse types of features (numerical, categorical, etc.) as different edge properties [157]. (c) Bipartite graphs offer a flexible way to incorporate new instances or features without requiring a restructuring of the entire dataset, and thus support incremental feature learning [142]. (d) Bipartite graphs can naturally tackle the issue of missing feature values by not creating the corresponding instance-feature links. (e) Link prediction (i.e., predicting whether a link should exist between instance and feature nodes) can be used to impute missing feature values [157]. (f) Bipartite graphs provide an efficient way to calculate the instance proximity, i.e., similarities between instances [61]. However, it is worth mentioning that transforming a large tabular dataset into a bipartite graph can be computationally expensive, and processing the resulting graph also requires specific techniques, such as incremental construction [9], locality-sensitive hashing [30], and FPGA-based Accelerator [87].

**Multi-Relational Graphs.** A tabular dataset can be depicted as a multi-relational graph that contain same-typed nodes and multi-typed edges (or termed "relations"). We can connect data instances with one another according to the different kinds of relationships between instances. Consturcting the multi-relational graph is common in representing user-review tabular data [25], [60], [89], [117]. Users are regarded as nodes, and edges depict various ways of user-user interactions, such as co-reviewing the same product [25], [60], [117], similar rating behaviors [25], [89], [117], one user trades to another [89], co-purchasing the same product [60], having the same symptom (symptom as the relation) [107], and having social relationship like family or workmate [89]. A special variant of multi-relational graph is *multiplex graph*. While a multi-relational graph put all relations that connects instances in the same graph structure, a multiplex graph is a layered structure, in which all graph layers share the same set of nodes and every layer is a homogeneous graph whose edges are created by a certain relation. By considering sharing common or similar feature values as a kind of relation, TabGNN [51], AMG [57], and GCondNet [101] utilize multiplex graphs to formulate tabular datasets. An obvious benefit of multi-relational graph formulation is better Data Integration. Multi-relational graphs can be an effective way to integrate data from multiple sources or of different types. Different types of relationships from different sources can be represented with different types of edges, allowing for more comprehensive modeling of the data. Nevertheless, multi-relational graph formulation have the same challenge as general heterogeneous graph formulation – requiring careful investigation on choosing feature or attribute values for relation creation.

**Knowledge Graphs.** A knowledge graph (KG) offers powerful auxiliary information to model the relationships between features of a tabular dataset. In KG formulation of tabular data, each feature corresponds to a node in the graph, and there are also non-feature nodes. One can access or construct the relevant KG from external resources and domain knowledge, such as human protein map [96] and DrugBank [137] for bio-medical tabular data. PLATO [114] incorporates auxiliary domain information structured as a knowledge graph to regularize a multilayer perceptron for tabular data prediction. PLATO had shown that when working with tabular datasets with extremely high-dimensional features and limited samples, KG can mitigate the overfitting issue [114]. The challenge of adopting KG lies in finding the correspondences between tabular features and KG nodes. By tackling the scarcity of context and the ambiguity and noisiness of the available conten, JenTab [1] learns to match tabular data to a knowledge graph.

#### 4.1.3 Hypergraphs

A hypergraph is a generalization of a graph in which an edge (or a hyperedge) can connect more than two nodes. When applying this to a tabular dataset, each feature value can be considered as a node, and each instance (or row of the dataset) is used to create a hyperedge that connects the nodes corresponding to the feature values of that instance [10], [15], [27], [147]. This representation can offer a more accurate depiction of complex relationships among feature values across instances. By treating each instance as a hyperedge, we effectively encode the simultaneous co-occurrence of multiple feature values. This can be particularly beneficial in cases where interactions among feature values are of interest. In addition to utilize every instance to create a hyperedge, HYTREL [15] further leverages each feature column and the entire tabular dataset to be two additional types of hyperedges. As pointed out by HYTREL [15], hypergraphs with three types of hyperedges can preserve four structural properties in tabular data: (1) capture the permutation invariance of tables, maintaining relationships regardless of row or column order; (2) highlight interdependencies within rows and structural similarities within columns; (3) high-order multilateral relations within cells,

rows, or columns are effectively represented; (4) depict the hierarchical organization of information in tables from cell-level to table-level. There are still limitations in hypergarph formulation. The first is sparse data issue. In high-dimensional datasets with many unique feature values, the hypergraph can become very large and sparse, which could pose challenges in analysis and computation. The second is scalability. As the size of the dataset grows, so does the complexity and size of the hypergraph. Handling large hypergraphs can be computationally challenging.

### 4.2 Graph Construction

Graph construction is a critical step when transforming tabular data into graph data, and the chosen methodology can greatly affect the quality of graph structure. This in turn can have a significant impact on downstream prediction tasks. The key consideration is selecting or learning a proper criteria for edge creation, which can significantly impact the sparsity and connectivity of the graph. Too many edges can make the graph overly complex and computationally intensive to work with, while too few can lead to a highly disconnected graph.

#### *4.2.1 Intrinsic Structure*

A tabular dataset typically consists of instances (rows) and features (columns). The intrinsic structure of a tabular dataset emerges from the relationships between these instances and features. It is intuitive and common to utilize the intrinsic structure of tabular data to construct the graph. The formulation of bipartite graphs directly employs the intrinsic structure to create edges between instance nodes and feature nodes. Typical bipartite graph formulations, including GRAPE [157], MedGraph [54], FATE [142], PET [27], GNNDP [121], MGNN [42], and IGRM [63], leverage this approach to construct the graph. In addition, the constructions of general heterogeneous graphs and multi-relational graphs also rely on the intrinsic structure of tabular data. If an instance possess a specific categorical feature value, which is used as a certain node type, we can create an edge between them.

In general heterogeneous grapohs, for example, edges encapsulate the interactions between user nodes, product brands, and item categories in click-through rate prediction [105]. This approach is mirrored in electronic health records, where edges are created between patient nodes, and diagnosis and treatment codes [20], [93]. Customs import declarations are depicted as connections between declaration nodes, importer companies, and goods categories [118]. Similarly, in credit card transactions, edges signify the relationships between transaction nodes, payment tokens, and shipping addresses [110]. In real estate valuation, edges depict the relationships, including location, facility, or floorplan, between houses entities [106]. In predicting stock price movements, various lead-lag relationships between companies, such as historical prices and media event, can be represented as edges [17]. For supply chain mining, the ownerships of enterprises and their owners, the transfers/transactions between enterprises and between consumers, and the social relations between the owners of enterprises, are used to build the graph [153]. In online product reviews, edges connect user nodes with corresponding products and device nodes, encapsulating user interactions [130]. Last, in the modeling of relational databases, while instances or entries are used as multi-typed nodes, the primary-foreign key relations between data tables are used to define edges [22], [37].

In multi-relational graphs, for example, two user instances can be connected with each other because both had ever purchased or reviewed that product, referring to user-user purchase or review relationships [25], [60], [117]. One user may trade something to another, indicating the trade relationship between them [89]. A patient has a common symptom with another, employing the specific symptom as the relationship [107]. In financial domains, companies can have explicit stock relations (e.g., supplier-consumer relations), which reflect the potential influence between stocks [19], [35], [74]. Users can be connected by multiple relationships, including social connections, capital transactions, and device dependence, for loan fraud detection [150]. Various relations like transaction, transfer and social can be used to build the multiplex graph among users for loan default prediction [57].

The intrinsic structure among data instances, feature columns, feature values, and data cells can be naturally used to constuct a hypergraph. In HCL [10], each data instance is viewed as a hyperedge, and each categorical feature value is viewed as a node. HyTrel [15] treats each cell as a node, and each data instance, each feature column, and the entire data table as various hyperedges, respectively. That said, each cell node is connected to three hyperedges. In PET [27], each distinct feature value is considered as a node, and each data instance constructs a hyperedge.

Utilizing the intrinsic structure of tabular data for graph construction has its limitations, largely due to the assumption that relationships between instances and features are clearly discernible. However, this may not always be the case due to various data challenges. For example, data noise, such as erroneous entries or outliers, can obscure true relationships. Imagine a tabular dataset of sensor readings where occasional malfunctioning sensors contribute incorrect data – these noisy entries could disrupt the accurate capture of relationships in a graph structure. Another challenge is missing values, which can leave gaps in the connections. For instance, in a retail dataset, if certain transactions do not record the customer's age or the purchased product category, these missing values would make it difficult to form complete instance-feature connections in the graph. Furthermore, the heterogeneity of features, i.e., different types and scales of features, can complicate the relationships. In a real estate dataset, features might range from categorical (e.g., house style) to continuous (e.g., house size), and from ordinal (e.g., condition rating) to binary (e.g., has pool or not). The different nature of these features could pose challenges in defining clear and meaningful connections for graph construction. Moreover, the intrinsic structure may not capture latent relationships, which are indirect or hidden relationships that are not explicitly represented in the data. For example, in a movie rating dataset, there might be a latent relationship between two users who haven't rated the same movies but have similar tastes inferred through other shared features or behaviors. These latent

TABLE 3: Summary of rule-based methods for tabular graph construction. In the column of Tabular Domain, "multi" means multiple domains.

| | Formulation | Similarity | Edge Criteria | Tabular Domain |
|---|---|---|---|---|
| LSTM-GNN [112] | Ho-Inst | TF-IDF | kNN | medical |
| LUNAR [44] | Ho-Inst | Euclidean | kNN | multi |
| GNNBFD [146] | Ho-Inst | Euclidean | kNN | manufacture |
| MST-GRA [2] | Ho-Inst | Heat Kernel | kNN | multi |
| GCIL [26] | Ho-Inst | RBF Kernel | kNN | multi |
| SSGNet [48] | Ho-Inst | Euclidean | kNN | medical |
| GNN4MV [59] | Ho-Inst | Gaussian Kernel | kNN | multi |
| GRIN [21] | Ho-Inst | Correntropy [88] | kNN | time-series |
| GINN [119] | Ho-Inst | Euclidean | Threshold | multi |
| GAEOD [29] | Ho-Inst | Cosine | Threshold | multi |
| GEDI [14] | Ho-Inst | Cosine | Threshold | time-series |
| LatentG [78] | Ho-Inst | Cosine | Threshold | medical |
| CCNS [31] | Ho-Inst | Cross-class sim [99] | kNN | multi |
| GATE [120] | He | PMI | Threshold | medical |
| WPN [95] | Ho-Inst | - | Same Feat Val | medical |
| TabGNN [51] | He-MultiRel | - | Same Feat Val | multi |
| SGANM [94] | Ho-Inst | - | Fully-connected | manufacture |
| IAGNN [13] | Ho-Inst | - | Fully-connected | manufacture |
| GCN-Int [92] | Ho-Feat | - | Fully-connected | CTR |
| FinGAT [56] | Ho-Feat | - | Fully-connected | financial |
| Fi-GNN [83] | Ho-Feat | - | Fully-connected | CTR |
| INCE [127] | Ho-Feat | - | Fully-connected | multi |
| IGNNet [3] | Ho-Feat | Pearson correlation | Fully-connected | multi |
| GatFM [53] | Ho-Feat | Inner Product | Threshold | CTR |

relationships can be crucial to uncover underlying patterns in the data. Hence, although the intrinsic structure provides a basis for graph construction, it is important to consider these potential issues and complement this approach with additional techniques for handling them effectively.

#### 4.2.2 Rule-based Methods

Rule-based methods for constructing graphs from tabular data involve the use of predefined criteria or heuristics to determine the nodes and edges of the resulting graph. Typically, instances (rows) and features (columns) of the table serve as the potential nodes, while relationships, often determined by logical conditions or thresholds, form the edges. For instance, an edge might be created between two instance nodes if they share a certain categorical feature value or if the simiarlity between their feature vectors exceeds a specific threshold. The rules can also be based on domain knowledge, correlations, or statistical properties of the data. The advantage of rule-based methods is that they offer explicit control over the graph construction process, ensuring that the resultant structure aligns well with domain knowledge or specific analytical objectives. However, crafting effective rules requires a deep understanding of both the data and the desired outcomes, and a poorly defined set of rules might lead to unrepresentative or overly sparse graph structures.

The basic steps of rule-based method include utilizing a certain simiarlity measure between data and determining some criteria for edge creation. By identifying the strategy to have the similarity measure and determine the edge criteria, we summarize the rule-based methods in Table 3. We can see that while different similarity measures can be used, there are four mainstream criteria of edge creation, including kNN, thresholding, fully-connected, and same feature value.

- **k-Nearest Neighbors (kNN)** (e.g., [31], [44], [59], [112], [146]) connects a node to its $k$ closest nodes based on some similarity metric in the feature space. kNN primarily focuses on local structure, ensuring that nodes with similar characteristics are connected, and preserving the local nuances of the dataset. Hence, kNN is often useful for datasets where instances have clear clusters or groupings. For large datasets, finding the $k$ nearest neighbors can be computationally expensive. It is also sensitive to the choice of the distance metric and the value of $k$. An inappropriate distance measure might not capture true data similarities.
- **Thresholding** (e.g., [14], [21], [29], [120]) forms edges between nodes when their similarity/distance value exceeds or falls below a set threshold. By setting a threshold, one can reduce the number of edges to eliminate weaker or spurious connections, resulting in a sparser and potentially more manageable graph. However, the choice of threshold can be arbitrary and might not always reflect inherent data structures. Overly aggressive thresholding can result in loss of important relationships.
- **Fully-connected** (e.g., [13], [56], [92], [94]) makes every node in the graph connected to every other node, creating a dense web of relationships. This method can maximize the potential information flow, and capture global properties of the dataset. However, this can lead to a very dense graph, which might be computationally expensive to process. The presence of all possible connections can dilute the importance of truly significant relationships.
- **Same Feature Value.** (e.g., [51], [95]) Edges are drawn between nodes that share an identical value for a particular feature. The reason for the connections is clear and easily interpretable. It is effective for categorical or discrete data. However, this can lead to isolated clusters in the graph, especially if many unique feature values exist. This is not always effective for continuous features without discretization.

TABLE 4: Comparison of learning-based methods for tabular graph construction. All of these methods are for the construction of instance graphs.

| Method | Strategy | Initialization | Modeling | Training |
|---|---|---|---|---|
| IDGL [16] | Metric | - | Cosine | End-to-end |
| RGSL [159] | Metric | - | Inner product | End-to-end |
| DGM [69] | Metric | kNN | Euclidean/Hyperbolic | End-to-end |
| EGG-GAE [123] | Metric | - | Gaussian Kernel | End-to-end |
| DRSA-Net [170] | Metric | - | Low-rank Approx. | End-to-end |
| HES-GSL [141] | Metric | kNN | Cosine | End-to-end |
| kNNGNN [68] | Metric | kNN | Mahalanobis | End-to-end |
| SLAPS [33] | Neural | kNN | Multilayer perceptron | End-to-end |
| SUBLIME [91] | Neural | kNN | Contrastive Learner | Unsupervised |
| TabGSL [85] | Neural | kNN | Contrastive Learner | End-to-end |
| T2G-Former [152] | Neural | Threshold | Multilayer perceptron | End-to-end |
| LDS [39] | Direct | kNN | Free variables | Alternative |
| ALLC [97] | Direct | kNN | Free variables | Unsupervised Active |
| Causal-GNN [162] | Direct | Random | Free variables | End-to-end |
| Table2Graph [173] | Direct | Random | Self Attention [116] | End-to-end |

#### 4.2.3 Learning-based Methods

Given a tabular dataset, learning-based methods seek to derive the optimal structure of a graph, aiming to capture underlying relationships among data instances effectively. Instead of relying solely on pre-defined rules or domain-specific knowledge, this approach focuses on automatically determining the relationships between instances based on patterns present in the data. We can categorize existing studies into the following three groups, and accordingly create a summary of comparison in Table 4.

- **Metric-based Approach** (e.g., [16], [69], [123], [141], [159], [170]) leverages kernel functions to compute the similarity between node features or embeddings, and use these similarities as edge weights. Built on the network homophily

principle, which posits that similar nodes are more likely to connect [104], metric-based techniques reinforce intra-class connections, resulting in denser and more concise graph representations. A significant advantage is that they often allow end-to-end training due to the differentiable nature of kernel functions. However, selecting an appropriate kernel function and its parameters can be non-trivial and can greatly influence results. In addition, the assumption of network homophily might not always hold for every dataset.

- **Neural Approach** (e.g., [33], [85], [91], [152]) employs sophisticated deep neural networks to determine edge weights based on node features and their representations. This results in a more dynamic and adaptive graph construction mechanism as it can adaptively model complex relationships, potentially capturing non-linearities and intricate patterns. However, they may overfit to noise in the data if not properly regularized or if the dataset is small.
- **Direct Approach** (e.g., [39], [97], [162], [173]) view the adjacency matrix of the desired graph as a set of learnable parameters. Unlike the previous methods that depend on node representations to define edge connections, direct approaches offer greater adaptability in graph construction due to no reliance on node representations. However, this increased flexibility often comes at the cost of difficulty in efficiently learning the matrix parameters. Without reliance on explicit features or patterns, these methods might sometimes incorporate noise or irrelevant relationships.

#### 4.2.4 Other Methods

**Retrieval-based.** The retrieval-based graph construction method [27], [149] specifically captures the relationships within tabular data. In PET [27], for each data row (target), relevant neighboring rows are identified from a data pool. These rows then form a hypergraph where sets of feature columns become hyperedges and individual feature values become nodes. This structure uniquely captures relationships, such as how two rows might share a particular feature value (a node). One significant advantage is the nuanced representation of data: whereas traditional tabular data might just list values, the hypergraph visually and structurally links related data points, offering a richer context and potentially unveiling hidden patterns and connections. Feature Interaction Via Edge Search (FIVES) [149] automatically generates high-order interactive features from tabular data. It formulates the interactive feature generation as an edge search task on a feature graph. Utilizing a dedicated GNN and an adjacency tensor, FIVES inductively searches for optimal higher-order features based on interactions between existing features. This method transforms feature space traversal into an efficient GNN training process.

**Knowledge-based.** For tabular data in domains like medicine or genomic, rows might represent patients while columns are genes. However, without sufficient context, each value – perhaps indicating the expression of a gene in a patient's tumor – can be cryptic. Here, knowledge graphs (KGs) become invaluable for tabular data, as highlighted in PLATO [114]. These KGs are constructed from auxiliary domain-specific information. Using an medical example, a KG might be sourced from scientific literature, databases of genetic interactions, or annotations about gene functions. Within the KG, each gene (a feature) is represented as a node. This node might be connected to other genes, indicating interactions such as "activates" or "inhibits." Additionally, it could link to nodes representing broader biological processes or functions, elucidating the gene's role. By integrating this structured knowledge from KGs, tabular data transforms from a mere matrix of values to a rich and interconnected web of insights, providing machine learning models with the depth they need to decode intricate patterns inherent in specialized domains.

TABLE 5: Summary of representative GNN models for graph representation learning on tabular data.

| Type | Model | References |
|---|---|---|
| Ho. | GCN | GINN [119], GEDI [14], IAGNN [13], GCN-Int [92], IDGL [16], RGSL [159], DGM [69], EGG-GAE [123], SLAPS [33], SUBLIME [91], TabGSL [85], LDS [39], Table2Graph [173], FIVES [149], IGN-Net [3], GNNBFD [146], HES-GSL [141], LatentG [78] |
| | GAT | GATE [120], WPN [95], FinGAT [56], FT-GAT [71], GNNDB [22], GatFM [53] |
| | GraphSAGE | LSTM-GNN [112], GRAPE [157], GN-NDP [121], IGRM [63] |
| | GIN [151] | DRSA-Net [170] |
| | GAE [76] | MST-GRA [2], GAEOD [29] |
| | DGN [81] | CCNS [31] |
| | ProGNN [67] | SSGNet [48] |
| | GGNN [82] | Fi-GNN [83], Causal-GNN [162] |
| | Specialized | LUNAR [44], PET [27], TabGNN [51], FATE [142], SGANM [94], MGNN [42], MedGraph [54], T2G-Former [152], GNN4MV [59], ALLG [97] |
| He. | GAT | GME [105], GraphFC [118] |
| | GCN | LUCE [106] |
| | GraphSAGE | AO-GNN [60] |
| | HAN [134] | HSGNN [93] |
| | HGT [58] | xFraud [110], RelBench [37] |
| | Specialized | GCT [20], MAGNN [17], ST-GNN [153], C-FATH [130], GRC [150], H2FD [117], PC-GNN [89], AMG-DP [57], CARE-GNN [25], RioGNN [107], RelBench [37] |
| Hy. | Specialized | HCL [10], HyTrel [15], PET [27] |

### 4.3 Representation Learning

With the constructed graph for the given tabular dataset, the next stage is to accordingly produce the representation of each data instance through graph neural networks. Different GNN architectures are designed to capture various types of information and patterns from graphs. The choice of GNN often hinges on the specific characteristics and requirements of the derived tabular graph. We summarize which of the GNN methods is utilized by which work in Table 5, which can be mainly divided into GNNs for the constructed homogeneous (Ho.) and heterogeneous (He.) graphs. In addition to directly employ or extend exsiting GNNs, some works develop GNNs specialized for tabular data with various key designs, summarized in Table 6.

No matter homogeneous or heterogeneous graphs are constructed to depict the tabular data, the typical GNN architectures, i.e., GCN [77], GAT [126], and GraphSAGE [52], are widely employed to learn instance representations. The reason is three-fold. The first is *Expressiveness and Adaptability*. They can effectively capture and process both local and

global information in a graph. This capability means they can be applied to a broad range of tabular datasets, making them adaptable to various tasks and scenarios [145], [168]. The second is *Proven Performance*. Given their foundational nature in the realm of GNNs, they have been extensively benchmarked, refined, and validated across various tasks. Their consistent performance in numerous studies provides a level of assurance in their efficacy [124], [145], [172]. The third is *Ease of Integration*. They can be easily integrated with other neural network components, allowing for the design of more complex models tailored to specific tasks. Furthermore, GAT [126] enables the model to weigh the importance of neighbors differently. For graphs converted from tabular data where certain relationships or nodes might be more relevant than others, GAT provides a more nuanced aggregation method. GraphSAGE [52] handles vast graphs by sampling neighbors, offering scalability without compromising much on representation quality. If the tabular dataset is vast, resulting in a big graph, GraphSAGE can efficiently handle and learn representations by considering a sampled subset of neighbors.

#### 4.3.1 GNNs for Homogeneous Graphs

ProGNN [67] is constructed to handle noisy graphs and can refine the graph structure iteratively as it learns. For tabular datasets that might have been imperfectly converted into graphs, ProGNN can iteratively enhance both the graph structure and node representations [48]. GGNN [82] utilizes gates to control the flow of information across nodes. If sequential aspects exist in the tabular dataset [162], or if there is a need to regulate the information flow in the graph more carefully [83], GGNN could be preferred. Due to the powerful structural discrimination ability, GIN [151] excel in instance graphs whose intricate relational patterns emerge based on tabular features. GIN can discern subtle differences in node relationships, capturing the interactions and combinations prevalent in tabular datasets [170]. To further capture high-order correlation between tabular elements, deeper models like DGN [81] can be used to overcome the oversmoothing issue. The unsupervised nature of Graph AutoEncoder (GAE) [76] allows it to distill the essence of a dataset by attempting to reconstruct its graph structure, making it suitable for capturing dense and interconnected relationships. In tabular datasets, the features are highly interconnected or exhibit patterns where capturing the joint distribution of multiple columns via GAE is crucial and effective [2], [29].

#### 4.3.2 GNNs for Heterogeneous Graphs

Heterogeneous Graph Attention Network (HAN) [134], which is inherently designed for heterogeneous structures, aligns perfectly with the multi-faceted relationships and diverse feature types present in tabular data transformed into heterogeneous graphs. Its attention mechanisms, both at node and semantic (meta-path) levels, ensure that instance representations capture the rich interactions and relationships inherent in such data [93]. Heterogeneous Graph Transformer (HGT) [58] excels in processing heterogeneous graphs from tabular data due to its dynamic meta-path aggregation that captures intricate feature interactions. Its type-aware self-attention respects diverse feature distinctions, and its hierarchical attention ensures both granular and broad relationship comprehension, making it adept for tabular data's complexity [110].

TABLE 6: Summary of key designs on specialized GNNs.

| Key Design | Homogeneous | Heterogeneous |
|---|---|---|
| Label Adjustment | PET [27], SGANM [94] | |
| Feature-relation Modeling | TabGNN [51], ALLG [97] | GCT [20], AMG [57], CARE-GNN [25], RioGNN [107] |
| Feature Selection | T2G-Former [152] | GRC [150] |
| Neighbor Sampling | | C-FATH [130], PCGNN [89], CARE-GNN [25], RioGNN [107] |
| Multi-modality | MGNN [42] | MAGNN [17] |
| Missing Values | GNN4MV [59] | |
| Temporal Dependency | MedGraph [54] | ST-GNN [153] |
| Homo-Heterophilic | HES-GSL [141] | H2FD [117] |
| Permutation Invariance | FATE [142] | |
| Distance Preservation | LUNAR [44] | |

#### 4.3.3 Specialized GNNs

Specialized GNNs for graphs derived from tabular data are crucial to tackle real data challenges, enhance performance, and meet various application purposes. Off-the-shelf GNNs may miss tabular intricacies, while tailored models effectively capture nuanced relationships and handle data/application issues, ensuring optimal instance representations.

**Label Adjustment.** Both PET [27] and SGANM [94] underscore the pivotal role of label-guided adjustments in refining tabular data graph representations. PET constructs a hypergraph where distinct feature values become nodes, facilitating message propagation for auxiliary label information transfer and feature space adjustments. SGANM employs a unique 2-D edge embedding, driven by label information, resulting in a self-adaptive graph structure. This adaptability, coupled with a multi-head masked attention mechanism, accentuates GNN's label-enhanced feature extraction capabilities.

**Feature-relation Modeling.** By taking each feature as a kind of relation to construct the instance graph, TabGNN [51] can model finer-grained instance correlation with attention learning applied to each relation's graph to find which features contribute most to the prediction task. To encode the dynamic instance correlation that evolves as the instance representations change in different neural network layers, ALLG [97] learns a series of relation propagated adjacent matrices so that more precise graph structures can be captured. GCT [20] delivers a modification to the Transformer to guide the self-attention to learn the hidden conditional dependency structure between features using prior knowledge in the form of attention masks. AMG-DP [57] presents a relation-specific receptive layer to leverage the local structure in the heterogeneous tabular graph, and to better incorporate rich semantics derived from multiplex feature relations. CARE-GNN [25] and RioGNN [107] formulate relation-aware neighbor aggregators

to combine neighborhood information from different relations and produce instance embeddings.

**Feature Selection.** By identifying and retaining only the most informative features, feature selection can reduce the graph's complexity, enhancing both computational efficiency and representation quality. This focused aspect mitigates noise, minimizes redundancy, and ensures that the graph structure effectively captures the underlying relationships in the data. T2G-Former [152] is a bespoke Transformer model for tabular learning with a Graph Estimator module for jointly selecting salient features and promoting heterogeneous feature interaction based on estimated relation graphs. GRC [150] characterizes the multiple types of relationships via self-attention mechanism and employs conditional random field to constrain nodes with the same type to select the focused features and produce similar representation.

**Neighbor Sampling.** Given the diverse types of nodes and relations in a heterogeneous graph, neighbor sampling selectively aggregates information from a subset of relevant neighbors. This ensures efficient computation and prioritizes essential contextual information, enabling the model to capture the intricate relationships and heterogeneity intrinsic to the data, thereby enhancing the quality of learned instance representations. By leveraging community tags, C-FATH [130] addresses structure-inconsistency, ensuring nodes interact primarily with similar-behavior neighbors. Additionally, using entity similarities, C-FATH also tackles content-inconsistency, ensuring neighbors share features resembling the central node's. In PCGNN [89], a label-balanced sampler is devised to pick nodes and edges for sub-graph training, and a neighborhood sampler is designed to choose neighbors for over-sampling the neighborhood of the minority class and under-sampling the neighborhood of the majority class. CARE-GNN [25] and RioGNN [107] introduce similarity-aware neighbor selection mechanisms to identify neighbors closely related to a central node within a given relation in a heterogeneous graph. Utilizing reinforcement learning, they dynamically determine the optimal threshold for neighbor selection in tandem with the GNN training process.

**Multi-modality.** GNNs are increasingly employed to process multimodal tabular data, harnessing the potential to capture intricate relationships and patterns across different data types. Through constructing appropriate graph representations, GNNs facilitate learning from disparate data modalities in an integrated manner. MAGNN [17] derives instance representations from a heterogeneous graph built on multimodal tabular inputs. Emphasizing model interpretability, MAGNN employs a dual-phase attention mechanism that jointly optimizes and elucidates the significance of both inner-modality and inter-modality connections. MGNN [42] harnesses a unified GNN framework to extract features from multimodal medical datasets. By creating multiple bipartite graphs that map relationships between patients and various data sources, MGNN uses a GNN to ascertain each patient's embedding, along with a subsequent fusion layer to integrate these multimodal embeddings.

**Missing Values.** GNN4MV [59] employs graph neural networks to classify instances in tabular data that have missing values, eliminating the need for imputation. By leveraging supervised information, it guides feature space construction, mitigating the influence of absent values when determining neighborhood relationships. The neighborhood graph convolutional mechanism within GNN4MV uses the graph's inherent structure to directly classify these incomplete instances.

**Temporal Dependency.** In domains like electronic medical records and financial user-item interactions, tabular data often contains temporal nuances. GNNs adeptly capture these temporal dependencies across instances and features. MedGraph [54] presents a method to seamlessly grasp both the structural co-location information between patient visits and codes, and the chronological sequencing of visits. Leveraging a visit-code bipartite graph, MedGraph integrates temporal point processes, offering an end-to-end solution for capturing medical history. ST-GNN [153] processes temporal tabular graphs, capturing structural embeddings for nodes across snapshots. Using a temporal-aware aggregator, it derives time-sensitive embeddings. Combining these via attention, ST-GNN delivers a node embedding integrated with the end-to-end learning objective.

**Homo-Heterophilic.** In tabular data prediction models, instances often exhibit complex correlation with others, leading to the establishment of both *homophilic* (similar nodes) and *heterophilic* (dissimilar nodes) connections. Most GNNs predominantly emphasize homophily in the data graph [141], using low-pass filters to maintain node feature commonalities among neighbors. This approach can inadvertently overlook the nuances in heterophilic connections. $H^2$-FD [117] distinguishes between homophilic and heterophilic connections with guidance from labeled nodes. Its information aggregation strategy ensures homophilic connections propagate analogous information, while heterophilic ones highlight distinct information.

**Permutation Invariance.** Permutation invariance with respect to the order of input features is a critical property for tabular data modeling. Essentially, it ensures that the model's output remains consistent regardless of the sequence in which input features are presented. This property is crucial because, in many real-world datasets, the order of features is arbitrary and should not influence the model's prediction. Moreover, permutation invariance inherently provides flexibility to handle variable-length input feature vectors [160]. As new features emerge, either from previously unseen values of existing features or entirely new raw features, the model remains robust and adaptable. FATE [142] presents a permutation-invariant aggregation in the tabular GNN model. The idea is to replace the embedding layer with an embedding lookup and a sum aggregation over indexed feature embeddings.

**Distance Preservation.** Distance between instances is pivotal in tabular data learning tasks, especially in anomaly detection and clustering where relative proximities among data points provide significant insights. LUNAR [44] incorporates these distances by representing them as edge features within a GNN framework. In the GNN's message aggregation phase, these edge features, which encapsulate the distance information, are encoded. As a result, the derived instance representations inherently embody the distance-based relationships, offering a richer, more nu-

TABLE 7: Summary of additional learning tasks (besides the main task of tabular data prediction).

| Learning Tasks | Representative studies |
|---|---|
| Feature Reconstruction | GINN [119], GEDI [14], EGG-GAE [123], IGRM [63], MST-GRA [2], GAEOD [29], ALLG [97], GRAPE [157] |
| Denoising Autoencoder | SLAPS [33], HES-GSL [141] |
| Contrastive Learning | SUBLIME [91], TabGSL [85], SSGNet [48] |
| Graph Regularization | IDGL [16], MST-GRA [2], GraphFC [118], ALLG [97] |
| Sparsity Regularization | Table2Graph [173] |
| Multi-level Regularization | ALLG [97] |
| Property Preservation | GINN [119], LUCE [106], GCT [20], MedGraph [54], TabularNet [28] |
| Explanation Preservation | xFraud [110] |

TABLE 8: Summary of learning strategies.

| Strategy | Representative studies |
|---|---|
| Two-stage | SUBLIME [91], GNNBFD [146], GRAPE [157], IGRM [63], GINN [119], MST-GRA [2], GAEOD [29], MedGraph [54] |
| Adversarial | GINN [119] |
| Alternative | GEDI [14] |
| Bi-level | LDS [39], FIVES [149], FATE [142] |
| Pretrain-finetune | ALLG [97], GraphFC [118] |
| End-to-end | Most of the rest methods (e.g., TabGSL [85], T2G-Former [152], LUNAR [44] TabGNN [51], PET [27], DGM [69], Fi-GNN [83]) |

anced understanding of the tabular data's structure. The distance-preserving tabular GNNs improve generalization due to their ability to recognize the relative positioning of instances, bolstering robustness against noisy data.

### 4.4 Training Plans

Crafting effective training plans, encompassing both learning tasks and strategies, is essential in GNN-based tabular data learing. The learning tasks, such as feature reconstruction or graph regularization, tailor GNNs to intricately capture the complex relationships within tabular data. Concurrently, the selection of learning strategies, like bi-level optimization or pretrain-finetune methods, critically influences the model's generalization, robustness, and adaptability. This deliberate orchestration of training plans is fundamental; it ensures that GNNs not only generate precise and interpretable embeddings but also enhance predictive accuracy and reliability. Such well-structured training plans are crucial for leveraging GNNs to their fullest potential in deriving meaningful insights from tabular datasets across diverse applications. We have created Table 7 and Table 8 to summarize various aspects of learning tasks and strategies, respectively.

#### 4.4.1 Learning Tasks

**Representation Enahacement** through learning to reconstruct in GNN-based tabular data learning involves training the model to not only learn effective representations but also to accurately reconstruct the original features. This process, mainly involving either feature reconstruction or contrastive learning, enhances model robustness by encouraging the GNN to focus on and preserve key features during representation learning.

- *Feature Reconstruction.* Reconstructing features from embeddings obtained through a GNN encoder essentially involves decoding the rich, condensed information captured in the embeddings back into the original feature space. It serves as a robustness check, ensuring that the GNN encoder captures all relevant information and patterns present in the data. This reconstruction acts as a form of regularization, preventing the model from overfitting by learning to preserve essential data characteristics. There are three approaches: leveraging complete feature sets for high-fidelity representation, handling missing values to enhance model robustness, and processing noisy data to improve resilience against data quality issues. (a) **Complete Features**: GNNs trained on complete feature vectors produce detailed representations, ensuring comprehensive data integrity [29], [97]. This approach is ideal for applications requiring exact feature preservation. (b) **Missing Values**: GNNs learn from incomplete data, making them adept at handling real-world datasets with gaps [14], [63], [123], [157]. The reconstructed features can effectively impute missing values, enhancing the model's robustness in imperfect data scenarios. (c) **Noisy Data**: Training on noisy features through denoising autoencoder allows GNNs to become resilient to data corruption, focusing on underlying data patterns [2], [33], [119], [141]. This is beneficial for maintaining consistent performance even with compromised data quality.
- *Contrastive Learning.* Contrastive learning with GNN encoder embeddings boosts tabular data learning by refining the model's ability to differentiate between similar and dissimilar instances. This approach enhances representation quality, and bolsters the model's generalizability, ensuring it focuses on fundamental data characteristics and avoids overfitting to specific instances. SUBLIME [91] optimizes graph topology using contrastive learning to create robust structures in unsupervised settings. TabGSL [85] applies contrastive learning to enhance instance correlation and feature interaction, improving classification tasks. SSGNet [48] employs a siamese GNN architecture with a contrastive objective for instance similarity learning in medical tabular data, addressing sparsity and missing values. Each method utilizes contrastive learning to refine graph structures and node embeddings

**Regularization.** Regularization on embeddings from GNN encoders is a critical aspect of GNN-based tabular data learning, ensuring the stability and effectiveness of the learned graph structures and the learned instace representations. There are three primary types of regularization. The first is *Graph Regularization*: maintaining balance in connectivity to avoid overly smooth or sparse graphs. Representative regularizers include reducing adjacent nodes' embeddings [16], [97], [118], and minimizing Minimum Spanning Tree (MST) distance [2] that preserves local neighborhood structure between instaces. The second is *Sparsity Regularization*, as employed by Table2Graph [173], controls the adjacency matrix's sparsity to highlight key feature interactions while avoiding overfitting. The third is *Multi-level Regularization*, like in ALLG [97], uses multiple levels of adjacency matrices, with each layer informed and regularized by its predecessor, allowing for smooth propagation of relationships among samples.

**Information Preservation.** The learning task of information preservation in embeddings focuses on retaining essential characteristics and relationships inherent in the

tabular dataset, ensuring the model's output accurately reflects these core aspects. There are two main types of information being preserved: *Property Preservation* and *Explanation Preservation*. Property Preservation, as seen in models like GINN [119], GCT [20], and LUCE [106], aims to maintain global dataset attributes, conditional probabilities between features, and geographical distances among instances, respectively, in the learned representations. In addition, MedGraph aims at preserving medical dependency between disease and diagnosis [54], and TabularNet [28] targets at preserving spatial information encoding on some features [28]. On the other hand, Explanation Preservation focuses on ensuring that explanations highlighted by domain experts are adequately represented in the GNN's subgraph explanations [110] produced by GNNExplainer [155]. These preservation tasks are crucial for ensuring the GNN's embeddings are not just accurate but also meaningful and interpretable in the context of the original tabular data.

#### 4.4.2 Training Strategies

The strategies range from structured approaches, prioritizing sequential learning and optimization, to unified methods, emphasizing comprehensive model development. Below we review six widely adopted strategies.

- **Two-stage** methods involve a sequential approach where the initial task is to learn either a graph structure or instance representations from the tabular data. Subsequently, these learned structures or embeddings are utilized in training downstream prediction models. For instance, SUBLIME [91] first learns an instance graph and representations from tabular data, then employs these in training a GNN model for predictions. Similarly, models like GRAPE [157], GINN [119], and IGRM [63] focus initially on imputing missing values using tabular GNNs, followed by leveraging these imputed features for downstream predictions. Other approaches, such as GNNBFD [146], MST-GRA [2], GAEOD [29], and MedGraph [54], generate instance representations through GNN-based encoders built on various graphs derived from tabular data, which are then used in training models for final predictions. This separation allows for targeted optimization at each stage, potentially leading to enhanced model performance. However, drawbacks include increased computational demands and a potential disconnect between stages, where improvements in initial representation learning may not directly translate to better prediction outcomes. Additionally, the need for effective integration between stages can add to the complexity of the process.
- **Adversarial** methods in GNN-based tabular data predictions, exemplified by approaches like GINN [119], incorporate adversarial training strategies to enhance the model's ability to discern between imputed and real data. This technique involves using an adversarial loss during feature reconstruction, compelling the reconstructed vectors to closely align with the natural distribution of the original data patterns. While this method improves the realism and accuracy of predictions, it also adds complexity and computational demands to the training process, potentially impacting model convergence.
- **Alternative** methods like that used in GEDI [14] involve adaptively optimizing the weights of feature reconstruction tasks based on the performance of the target task. It treats feature reconstruction weights as meta-parameters, allowing for a more dynamic and responsive training process. This approach enhances the reconstruction of crucial features under the guidance of downstream tasks. The adaptive weighting mechanism guards against negative transfer from multi-task learning, ensuring auxiliary tasks do not hinder the primary objective. However, this method adds complexity in managing meta-parameters, requiring careful balancing for optimal task performance.
- **Bi-levevl** optimization involves structuring training as two inner-outer linked optimization problems. Models like LDS [39] and FIVES [149] utilize this approach to learn a discrete and sparse dependency structure among data points and features, respectively, while concurrently training a GCN's parameters. This method treats GCN parameter training as the 'outer' optimization problem, guided by a generative probabilistic model for graphs. The dual focus on optimizing both GCN parameters and the graph generator aims to minimize classification errors on the dataset. Similarly, FATE [142] employs strategies like proxy training data and asynchronous updates to enhance model extrapolation for new features. This involves using partial features for updates and decoupling the training of the prediction and GNN networks, updating them at different intervals.
- **Pretrain-Finetune** methods involves a two-step approach, exemplified by models like GraphFC [118] and ALLG [97]. In the pre-training stage, the model undergoes self-supervised learning on local neighborhood preservation or feature reconstruction, utilizing both labeled and unlabeled data to understand feature distributions and instance correlations. Subsequently, in the finetuning stage, the model weights are refined using labeled data, focusing on target label predictions. This approach benefits from establishing a robust initial understanding of the data, leading to more accurate predictions after finetuning. However, a potential drawback is the possible mismatch between pre-training and finetuning stages, especially if the data distributions vary.
- **End-to-end** training is a prevalent method in GNN-based tabular data predictions, streamlining the process from representation learning to final prediction in a unified model. Models can be exampled (but not limited) by those listed in Table 8. This approach is favored for its efficiency and ability to optimize the entire GNN for specific prediction tasks, often leading to improved accuracy by capturing intricate data relationships. However, it can be prone to overfitting, particularly with limited data, and may require substantial computational resources. Despite these challenges, end-to-end training's comprehensive and cohesive nature makes it a popular choice for effectively harnessing GNNs in tabular data learning.

## 5 Applications

The versatility of GNN4TDL has been useful across various domains. GNNs excel in complex tasks like anomaly detection (Sec. 5.1), click-through rate prediction (Sec. 5.2),

and medical prediction (Sec. 5.3), showcasing their ability to untangle intricate data relationships. Their profound impact extends to the nuanced challenges of missing data imputation (Sec. 5.4) and relational database modeling (Sec. 5.6), redefining traditional approaches in these critical areas.

### 5.1 Anomaly Detection

On the task of anomaly detection for tabular data, GNNs have emerged as a powerful tool, addressing the limitations of traditional statistical, supervised learning, and distance-based methods. GNNs excel in capturing complex dependencies and handling relational data, a critical advantage over traditional methods which often make strong assumptions about data distribution and fail to capture intricate feature interactions. For example, LUNAR [44] integrates local outlier methods into GNNs, enhancing adaptability and expressivity in anomaly detection. Similarly, graph autoencoders [2], [29] have proven effective in altering data distribution patterns to identify embedded outliers. In fraud detection, closely related to anomaly detection, GNNs adeptly uncover fraudulent activities by exploiting data connections. Models such as CARE-GNN [25], PC-GNN [89], AO-GNN [60], H2-FDetector [117], IHGAT [86], xFraud [110], C-FATH [130], HMF-GNN [164], GraphBEAN [34], and GraphFC [118] demonstrate GNNs' capabilities in detecting fraud through both homogeneous and heterogeneous graph-based approaches. In fault detection, GNNs like IAGNN [13], SGANM [94], and GNNBFD [146] showcase their effectiveness in identifying faults or failures in systems, such as mechanical equipment and industrial processes. These GNNs leverage the interaction among sensor signals and sample relationships to enhance fault detection, exemplifying their superior capability in identifying deviations in complex systems.

### 5.2 CTR Prediction

In the application of Click-Through Rate (CTR) prediction within the GNN4TDL framework, GNNs address critical challenges that traditional models face, such as sparse data, complex feature interactions, and user-item heterogeneity. Traditional approaches like logistic regression and deep learning models (DeepFM [49], Wide&Deep [18], GCN-int [92]) struggle with these issues, which GNNs effectively overcome. GNNs leverage structural information in user-item interaction graphs to mitigate sparse data and cold-start problems, as demonstrated by GME [105], which constructs graphs connecting old and new ads. For complex feature interactions, GNNs like Fi-GNN [83] and CrossGCN [36] model interactions between features as node attributes in the graph, capturing high-order interactions more efficiently. In addressing user and item heterogeneity, GNNs employ varied node and edge types to represent different user and item categories, as seen in DisenCTR [136] and DE-GNN [50], which utilize time-evolving graphs and dual graph structures, respectively. Additionally, in the context of sequential recommendation, RetaGNN [55] leverages a user-item-attribute tripartite graph, enhancing the capability of recommendation systems in diverse settings. This multifaceted approach by GNNs in CTR prediction showcases their superiority in capturing the nuanced dynamics of user-item interactions, significantly improving prediction accuracy in online services.

### 5.3 Medical Prediction

For medical prediction, GNNs have significantly advanced the analysis and prediction capabilities within healthcare, particularly in handling Electronic Medical Records (EMRs). EMRs, rich with patient visit details, diagnostic codes, and treatment information, present challenges such as structural and temporal complexity, data sparsity, and heterogeneity. GNNs address these challenges effectively, as demonstrated in various applications including disease prediction [95], [107], [121], patient representation learning [10], [20], [48], [54], [93], [107], medication recommendation [120], and survival prediction [42], [107], [112]. They enhance the understanding of the intrinsic structure and temporal dynamics in EMRs by treating medical codes as nodes, thus capturing the complex interrelations and temporal effects. In tackling data sparsity, particularly for rare diseases, GNNs excel by leveraging external knowledge and interconnected nodes, allowing for more accurate predictions and generalization. Additionally, the inherent heterogeneity of EMRs, with diverse patient information and medical histories, is adeptly managed by GNNs, which can interpret and utilize this complex, graph-like data structure. The application of GNNs in medical prediction thus marks a significant leap in exploiting the nuanced and intricate nature of healthcare data, leading to more informed and precise medical outcomes.

### 5.4 Missing Data Imputation

GNNs have emerged as a transformative solution in the field of missing data imputation for tabular data, addressing key challenges that traditional imputation methods face. Traditional techniques, ranging from simple mean or median imputation to more complex machine learning models like autoencoders [119], [123], often struggle with issues such as non-random missingness, multivariate missingness, and the disconnect between imputation and target prediction. GNNs, with their ability to model intricate data dependencies, provide a more nuanced approach to these challenges. For instance, GNN4MV [59] tackles non-random missingness by using a graph-based classification approach, leveraging neighbor information to guide the construction of a reliable graph structure. In addressing multivariate missingness, GNNs like GEDI [14] and GRIN [21] effectively capture inter-feature relationships, with GEDI projecting features into a common space using an attention mechanism, and GRIN learning spatio-temporal representations in a multivariate context. Furthermore, GRAPE [157] and IGRM [63] innovatively integrate imputation with target prediction, treating imputation as an edge weight prediction task and employing a friend network to enhance the differentiation of sample relationships. This comprehensive approach by GNNs in missing data imputation significantly enhances the accuracy and applicability of imputation methods across various domains, from healthcare and finance to environmental science and social sciences, ensuring more robust and reliable predictive modeling in datasets with incomplete information.

### 5.5 Financial Technology

GNNs are revolutionizing the handling and analysis of tabular financial data, such as transaction records, credit histories, and stock prices. GNNs can model complex interdependencies that traditional tabular data processing techniques overlook. For instance, in asset price prediction [17], [35], [106], [159], GNNs provide a nuanced understanding of the stock market by recognizing the interconnected nature of stocks, sectors, and market indicators, offering more context-rich predictions than traditional methods. This approach is critical in scenarios where the movement of one stock, like a major tech company, can significantly influence related stocks in the sector. In credit scoring [57], GNNs transcend the limitations of treating each client as an isolated entity by modeling the shared transactions and business relationships, thus uncovering hidden patterns in borrower networks that can inform credit risk assessments and management more accurately [19], [153]. Similarly, in investment decision-making [56], [74], GNNs utilize their ability to interpret complex relationships between various financial entities, enhancing decision-making processes with deeper insights. Moreover, GNNs are effective in detecting financial fraud [118], [150], where they address the challenge of coordinated activities among multiple accounts. By modeling the interactions among different accounts, GNNs are capable of identifying complex fraudulent activities that traditional methods might miss. This ability to capture and analyze the intricate web of financial transactions and relationships demonstrates the impact of GNNs in FinTech, paving the way for more sophisticated, accurate, and reliable financial analyses and decision-making processes.

### 5.6 Modeling Relational Databases

The modeling of relational databases can be enhanced through GNNs by capitalizing on the inherent structure and relationships within database schemas. This approach marks a significant departure from traditional database handling, which often involves flattening data tables and extensive feature engineering. As exemplified by RDBToGraph [22], relational databases are interpreted as directed multigraphs, where nodes represent database rows and edges denote relationships defined by keys and references. This structure effectively preserves the relational context that is typically lost in conventional database processing methods. Siamese Graph Convolutional Networks (S-GCNs) [79] integrate both structured and unstructured data sources, such as relational databases and free text. By constructing graphs where records are nodes and their relations are edges, and linking these with graphs derived from unstructured text, S-GCNs facilitate a comprehensive understanding of data relationships, enhancing tasks like record linkage. Combining language models (LMs) with GNNs can beeter exploit the full relational structure of databases [128]. This approach involves using LMs to encode individual rows and GNNs to model relationships within and across tables. Lastly, RelBench [37] introduces Relational Deep Learning, an end-to-end approach that views relational tables as heterogeneous graphs. This method allows GNNs to learn across multiple tables, automatically extracting representations that leverage all available data without manual feature engineering.

## 6 Open Problems & Future Directions

**Obtaining the Ability of Tree-based Models.** A recent study [47] had discussed the potential reasons that tree-based models still outperform deep learning on typical tabular data. Key findings highlight that neural networks struggle to create best-fit functions for non-smooth decision boundaries, while tree-based methods do much better with irregular patterns. In addition, irrelevant features can significantly degrade the performance of neural network models. But tree-based models have the ability to stay insulated from the effects of worse features. While developing the GNN models for tabular data learning, the performance can be boosted if we can incorporate the abilities to learn weird patterns and better select useful features like tree-based methods. Recent advances have effectively attempted to transfer such abilities from tree-based models to neural networks (e.g., NBDT [129], DeepGBM [70], and ANTs [122]). How to have such designs in GNNs for tabular data learning would be a useful direction.

**Incorporating Graph Transformers.** Standard transformers with graph-specific modifications can learn effective representations of node and edge tokens [72]. We have seen that various elements in tabular data can be treated as nodes, and their correlation can be utilized as edges. An appropriate arrangement of tabular elements as the input of transformers will not only bring the advantages of GNNs discussed in Section 2.5, but also impose the strength of transformers. The representation quality of data instances can be further improved because transformers are capable of learning complex representations and interactions between features in the data. Besides, we will have a natural way to handle missing values since transformers can implicitly learn to handle missing values through the self-attention mechanism, which assigns low attention scores to the missing data points. Even one can change the models of graph representation learning, described in Section 4.3, to graph-enhanced transformers like Structure-Aware Graph Transformer [12] and GPS Graph Transformer [109], so as to enjoy the advantages of transformers in tabular data learning.

**Scaling GNNs to Large Tabular Data.** Tabular data in real-world applications, such as click-through rate prediction [135] and fraud detection [75], may contain millions of instances and thousands of features. Formulating and constructing various graphs with their representation learning for large-scale tabular data requires huge computational costs, which make existing approaches infeasible. Three different strategies can bring scalability into GNN4TDL. The first is to choose a compact graph formulation that requires a relatively small number of nodes and/or edges constructed for tabular data. Hypergraphs can be a computationally efficient formulation [43]. The second is to apply sparse learning techniques to produce graph sparsification that samples a subgraph to reduce the amount of data aggregation and to achieve model sparsification that prunes the neural network to reduce the number of trainable weights [165]. The third is to adopt scalable GNN models, such as PPRGo [7], NDLS [167], GraphAutoScale [38], and GraphFM [158].

**Non-homogeneous Graph Structure Learning for Tabular Data.** We have seen a variety of learning-based approaches to construct the graph structures that depict tab-

TABLE 9: Comparison of three ways to use features in terms of their advantages and disadvantages.

| | Pros | Cons |
|---|---|---|
| Used as **feature nodes** | • Explicitly model the relationship between instances and their features, making it easier for GNNs to capture feature-level interactions.<br>• Preserve the original tabular data structure, which can be advantageous for certain learning needs, such as handling feature heterogeneity (e.g., continuous, categorical, ordinal) and integrating metadata or external knowledge. | • May not efficiently capture complex relationships between instances, as the graph structure only models relationships through shared features.<br>• Heterogeneous information might require tailored message-passing mechanisms to handle different types of nodes and edges. |
| Used to **create edges** | • Provide a richer representation of relationships between instances based on features, enabling GNNs to effectively learn complex patterns.<br>• Can capture higher-order relationships between instances by considering feature interactions. | • These features cannot be used for neighborhood aggregation to produce instances' representations.<br>• Need to choose manually determine the similarity metrics for different features to connect instances. |
| Used as **initial vectors** | • Integrate feature information directly into node attributes, making it accessible for GNN learning.<br>• Compatible with various GNN architectures, allowing for more flexible model design. | • May not capture feature-level relationships and dependencies between instances as effectively.<br>• Can result in limited interpretability, as harder to reason about feature importance or their influence on predictions due to the implicit nature of features in the node vectors. |

ular data in Section 4.2. However, most of them focus on learning homogeneous graphs, in which nodes are all data instances. The learning of graph structures for bipartite graphs, heterogeneous graphs, and hypergraphs constructed from tabular data is fully unexplored. While edges in homogeneous graphs depict the correlation between instances or between features, the learning of non-homogeneous graph structures of tabular data involves connecting data instances with various features. The underlying meanings can be elaborated from multiple aspects. For example, a learned edge can be viewed as associating an instance with an additional feature, which is essentially the effect of data augmentation. Since some instances can contain missing feature values, connecting them to features is a kind of data imputation, which is also the self-supervised learning task on tabular data. Furthermore, since graph structure learning considers not only edge addition but also edge removal, eliminating the association between instances and features, which is equivalent to creating missing values, can be regarded as adversarial learning, which can improve the generalization ability and the model robustness for tabular data prediction.

**Better Strategies to Utilize Features.** When representing tabular data as a graph, the common formulations are instance graphs and bipartite graphs, which require features to create edges and nodes, respectively. When applying graph neural networks, we need the initial vectors associated with nodes for information propagation. Features of an instance have multi-way usages – used as feature nodes (in bipartite graphs), used to create edges between instance nodes (in instance graphs), and used as the initial vectors of instance nodes (in both bipartite and instance graphs). We summarize the advantages and disadvantages of such three usages of features in Table 9. It is still unexplored to investigate which kinds of tabular features better fit which types of usages for GNNs. For example, which features are better used in initial node vectors? Also, which features should be used to create edges? To unleash the power of GNNs for tabular data, one can also seek to optimize the feature usage. The task is to select subsets of features for different usages in either a disjoint or overlapping manner so that the prediction performance can be boosted. This direction is also related to feature selection on graph neural networks, in which not utilizing irrelevant features as initial node vectors can improve the performance of node classification [148]. The selected features as initial vectors also cannot be redundant as highly correlated features would cause the performance degradation of deep GNNs [66].

**Graph-based SSL for Tabular Data.** Self-supervised learning (SSL) has been proven to be useful in deep learning-based tabular data prediction. Typical auxiliary SSL tasks applied to tabular data include contrastive learning [6], [125], feature reconstruction [4], [156], data imputation [157], and column prediction [103]. While tabular data is represented by a graph, one can further leverage the structure knowledge depicting the correlation between instances (and/or features), together with original features, to design proper graph-based SSL tasks [90]. Here are some potential SSL tasks based on tabular graphs. (a) *Missing Feature Imputation*: Predict missing feature values by training the GNN to reconstruct the known features for each instance. This helps the model learn feature-level relationships and dependencies. (b) *Graph Clustering*: Learn node embeddings that group similar instances together by optimizing a clustering objective (e.g., maximizing intra-cluster similarity and minimizing inter-cluster similarity). (c) *Graph Completion*: Train the GNN to predict missing edges in the graph based on the existing edges and node attributes, learning to capture higher-order relationships between instances. (d) *Neighborhood Prediction*: Train the GNN to predict the neighbors of a given node based on the node's attributes and graph structure, learning to recognize local patterns and relationships. (e) *Denoising or De-corrupting Graphs*: Train the GNN to reconstruct the original graph from a noisy or corrupted version by optimizing a graph reconstruction loss, learning to capture robust and clean representations of instances. (f) *Contrastive learning*: Create positive and negative pairs of instances (e.g., based on feature similarity or other criteria) and train the GNN to distinguish between them, learning informative and discriminative representations. The SSL tasks help GNNs learn effective and expressive representations from tabular data,

which can then be fine-tuned for downstream tasks such as classification, regression, and recommendation.

**Dealing with Robustness Issues.** Applying GNNs for tabular data prediction introduces robustness issues that arise from different factors. Below, we discuss these robustness issues that require further exploration in designing GNNs for tabular data. (a) *Noise in Graph Structure*: Spurious edges or incomplete connections, which result from noisy features and missing values, can hinder the model's ability to learn and generalize effectively [23], as they can lead to the incorrect propagation and aggregation of information in the GNN. (b) *Data Distribution Shifts*: GNNs may struggle with shifts in the data distribution, such as changes in feature distributions or relationships between instances [143]. While GNNs can capture complex patterns in the training data, they might not generalize well to unseen data with different characteristics. (c) *Overfitting and Oversmoothing*: GNNs can suffer from overfitting, especially when learning from small tabular datasets. Oversmoothing, a phenomenon where node representations become excessively similar after multiple layers of aggregation, can further exacerbate this issue, reducing the model's ability to discriminate between instances [169]. The way graphs are constructed can affect the degree of overfitting and oversmoothing. (d) *Adversarial Attacks*: GNNs can be vulnerable to adversarial attacks, where small perturbations to the graph structure or node features are introduced to mislead the model [80]. Such attacks can exploit the model's sensitivity to the graph structure and feature noise, potentially causing significant performance degradation. A tabular GNN model needs to be robust to structural perturbations that come from maliciously crafted feature values on instances.

## 7 Conclusions & Discussion

This survey paper provides a comprehensive examination of Graph Neural Networks (GNNs) for tabular data learning, contributing significantly to the understanding and advancement of this critical research area. Its primary contribution lies in its extensive exploration of the underlying processes – graph formulation, graph construction, graph representation learning, and training plans – which serves as a detailed guide for both novice and experienced researchers. In addition, by showcasing a broad range of practical applications of tabular GNNs, this paper substantiates the versatility and adaptability of GNNs in real-world scenarios. Furthermore, our delineation of potential future research directions fills a crucial gap in the existing literature, providing a roadmap for those keen on advancing this technology. By shedding light on these aspects, the paper serves as an invaluable resource, fostering understanding, accelerating innovations, and shaping the future trajectory of research in GNNs for tabular data learning.

## Acknowledgments

This work is supported by the National Science and Technology Council (NSTC) of Taiwan under grants 112-2628-E-006-012-MY3, 110-2221-E-006-136-MY3, and 112-2634-F-002-006. This work is also supported by the Institute of Information Science (IIS), Academia Sinica, Taiwan.

## References

[1] Nora Abdelmageed and Sirko Schindler. Jentab: Matching tabular data to knowledge graphs. In *SemTab@ ISWC*, pages 40–49, 2020.

[2] Imtiaz Ahmed, Travis Galoppo, Xia Hu, and Yu Ding. Graph regularized autoencoder and its application in unsupervised anomaly detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(8):4110–4124, 2022.

[3] Amr Alkhatib, Sofiane Ennadir, Henrik Boström, and Michalis Vazirgiannis. Interpretable graph neural networks for tabular data. *arXiv preprint arXiv:2308.08945*, 2023.

[4] Sercan Ö. Arik and Tomas Pfister. Tabnet: Attentive interpretable tabular learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 6679–6687, 2021.

[5] Bart Baesens, Veronique Van Vlasselaer, and Wouter Verbeke. *Fraud analytics using descriptive, predictive, and social network techniques: a guide to data science for fraud detection*. John Wiley & Sons, 2015.

[6] Dara Bahri, Heinrich Jiang, Yi Tay, and Donald Metzler. Scarf: Self-supervised contrastive learning using random feature corruption. In *International Conference on Learning Representations*, 2022.

[7] Aleksandar Bojchevski, Johannes Gasteiger, Bryan Perozzi, Amol Kapoor, Martin Blais, Benedek Rózemberczki, Michal Lukasik, and Stephan Günnemann. Scaling graph neural networks with approximate pagerank. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD'20, page 2464–2473, 2020.

[8] Vadim Borisov, Tobias Leemann, Kathrin Seßler, Johannes Haug, Martin Pawelczyk, and Gjergji Kasneci. Deep neural networks and tabular data: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[9] Alireza Bosaghzadeh and Fadi Dornaika. Incremental and dynamic graph construction with application to image classification. *Expert Systems with Applications*, 144:113117, 2020.

[10] Derun Cai, Chenxi Sun, Moxian Song, Baofeng Zhang, Shenda Hong, and Hongyan Li. Hypergraph contrastive learning for electronic health records. In *Proceedings of the 2022 SIAM International Conference on Data Mining (SDM)*, pages 127–135, 2022.

[11] Kaidi Cao, Jiaxuan You, and Jure Leskovec. Relational multi-task learning: Modeling relations between data and tasks. In *International Conference on Learning Representations*, 2022.

[12] Dexiong Chen, Leslie O'Bray, and Karsten Borgwardt. Structure-aware transformer for graph representation learning. In *Proceedings of the 39th International Conference on Machine Learning*, pages 3469–3489, 2022.

[13] Dongyue Chen, Ruonan Liu, Qinghua Hu, and Steven X Ding. Interaction-aware graph neural networks for fault diagnosis of complex industrial processes. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[14] Katrina Chen, Xiuqin Liang, Zhibin Zhang, and Zheng Ma. Gedi: A graph-based end-to-end data imputation framework. *arXiv preprint arXiv:2208.06573*, 2022.

[15] Pei Chen, Soumajyoti Sarkar, Leonard Lausen, Balasubramaniam Srinivasan, Sheng Zha, Ruihong Huang, and George Karypis. Hytrel: Hypergraph-enhanced tabular data representation learning. *Advances in Neural Information Processing Systems*, 2023.

[16] Yu Chen, Lingfei Wu, and Mohammed Zaki. Iterative deep graph learning for graph neural networks: Better and robust node embeddings. In *Advances in Neural Information Processing Systems*, pages 19314–19326, 2020.

[17] Dawei Cheng, Fangzhou Yang, Sheng Xiang, and Jin Liu. Financial time series forecasting with multi-modality graph neural network. *Pattern Recognition*, 121:108218, 2022.

[18] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, Rohan Anil, Zakaria Haque, Lichan Hong, Vihan Jain, Xiaobing Liu, and Hemal Shah. Wide & deep learning for recommender systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, DLRS 2016, pages 7–10, 2016.

[19] Rui Cheng and Qing Li. Subsequence-based graph routing network for capturing multiple risk propagation processes. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence (IJCAI)*, pages 3810–3816, 2022.

[20] Edward Choi, Zhen Xu, Yujia Li, Michael Dusenberry, Gerardo Flores, Emily Xue, and Andrew Dai. Learning the graphical structure of electronic health records with graph convolutional transformer. *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 606–613, 2020.

[21] Andrea Cini, Ivan Marisca, and Cesare Alippi. Filling the g_ap_s: Multivariate time series imputation by graph neural networks. In *International Conference on Learning Representations*, 2022.

[22] Milan Cvitkovic. Supervised learning on relational databases with graph neural networks. In *International Conference on Learning Representations*, 2020.

[23] Enyan Dai, Wei Jin, Hui Liu, and Suhang Wang. Towards robust graph neural networks for noisy graphs with sparse labels. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, WSDM'22, page 181–191, 2022.

[24] Guimin Dong, Mingyue Tang, Zhiyuan Wang, Jiechao Gao, Sikun Guo, Lihua Cai, Robert Gutierrez, Bradford Campbel, Laura E. Barnes, and Mehdi Boukhechba. Graph neural networks in iot: A survey. *ACM Trans. Sen. Netw.*, 19(2), apr 2023.

[25] Yingtong Dou, Zhiwei Liu, Li Sun, Yutong Deng, Hao Peng, and Philip S. Yu. Enhancing graph neural network-based fraud detectors against camouflaged fraudsters. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, CIKM '20, pages 315–324, 2020.

[26] Guodong Du, Jia Zhang, Min Jiang, Jinyi Long, Yaojin Lin, Shaozi Li, and Kay Chen Tan. Graph-based class-imbalance learning with label enhancement. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–15, 2021.

[27] Kounianhua Du, Weinan Zhang, Ruiwen Zhou, Yangkun Wang, Xilong Zhao, Jiarui Jin, Quan Gan, Zheng Zhang, and David P Wipf. Learning enhanced representation for tabular data via neighborhood propagation. In *Advances in Neural Information Processing Systems*, volume 35, pages 16373–16384, 2022.

[28] Lun Du, Fei Gao, Xu Chen, Ran Jia, Junshan Wang, Jiang Zhang, Shi Han, and Dongmei Zhang. Tabularnet: A neural network architecture for understanding semantic structures of tabular data. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, KDD '21, pages 322–331, 2021.

[29] Xusheng Du, Jiong Yu, Zheng Chu, Lina Jin, and Jiaying Chen. Graph autoencoder-based unsupervised outlier detection. *Information Sciences*, 608:532–550, 2022.

[30] Carlos Eiras-Franco, David Martínez-Rego, Leslie Kanthan, César Piñeiro, Antonio Bahamonde, Bertha Guijarro-Berdiñas, and Amparo Alonso-Betanzos. Fast distributed knn graph construction using auto-tuned locality-sensitive hashing. *ACM Trans. Intell. Syst. Technol.*, 11(6), oct 2020.

[31] Federico Errica. On class distributions induced by nearest neighbor graphs for node classification of tabular data. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[32] Wenqi Fan, Yao Ma, Qing Li, Jianping Wang, Guoyong Cai, Jiliang Tang, and Dawei Yin. A graph neural network framework for social recommendations. *IEEE Transactions on Knowledge and Data Engineering*, 34(5):2033–2047, 2022.

[33] Bahare Fatemi, Layla El Asri, and Seyed Mehran Kazemi. Slaps: Self-supervision improves structure learning for graph neural networks. In *Advances in Neural Information Processing Systems*, pages 22667–22681, 2021.

[34] Rizal Fathony, Jenn Ng, and Jia Chen. Interaction-focused anomaly detection on bipartite node-and-edge-attributed graphs. In *2023 International Joint Conference on Neural Networks (IJCNN)*, 2023.

[35] Fuli Feng, Xiangnan He, Xiang Wang, Cheng Luo, Yiqun Liu, and Tat-Seng Chua. Temporal relational ranking for stock prediction. *ACM Trans. Inf. Syst.*, 37(2), mar 2019.

[36] Fuli Feng, Xiangnan He, Hanwang Zhang, and Tat-Seng Chua. Cross-gcn: Enhancing graph convolutional network with k-order feature interactions. *IEEE Transactions on Knowledge and Data Engineering*, 35(1):225–236, 2023.

[37] Matthias Fey, Weihua Hu, Kexin Huang, Jan Eric Lenssen, Rishabh Ranjan, Joshua Robinson, Rex Ying, Jiaxuan You, and Jure Leskovec. Relational deep learning: Graph representation learning on relational tables. *arXiv preprint arXiv:2312.04615*, 2023.

[38] Matthias Fey, Jan E Lenssen, Frank Weichert, and Jure Leskovec. Gnnautoscale: Scalable and expressive graph neural networks via historical embeddings. In *International Conference on Machine Learning*, pages 3294–3304, 2021.

[39] Luca Franceschi, Mathias Niepert, Massimiliano Pontil, and Xiao He. Learning discrete structures for graph neural networks. In *International conference on machine learning*, pages 1972–1982, 2019.

[40] Chen Gao, Yu Zheng, Nian Li, Yinfeng Li, Yingrong Qin, Jinghua Piao, Yuhan Quan, Jianxin Chang, Depeng Jin, Xiangnan He, and Yong Li. A survey of graph neural networks for recommender systems: Challenges, methods, and directions. *ACM Trans. Recomm. Syst.*, 1(1), mar 2023.

[41] Hongyang Gao and Shuiwang Ji. Graph u-nets. In *Proceedings of the 36th International Conference on Machine Learning*, pages 2083–2092, 2019.

[42] Jianliang Gao, Tengfei Lyu, Fan Xiong, Jianxin Wang, Weimao Ke, and Zhao Li. Predicting the survival of cancer patients with multimodal graph neural network. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 19(2):699–709, 2022.

[43] Y. Gao, Y. Feng, S. Ji, and R. Ji. Hgnn+: General hypergraph neural networks. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 45(03):3181–3199, 2023.

[44] Adam Goodge, Bryan Hooi, See Kiong Ng, and Wee Siong Ng. Lunar: Unifying local outlier detection methods via graph neural networks. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2022.

[45] Shivapratap Gopakumar, Tu Dinh Nguyen, Truyen Tran, Dinh Phung, and Svetha Venkatesh. Stabilizing sparse cox model using statistic and semantic structures in electronic medical records. In *Advances in Knowledge Discovery and Data Mining: 19th Pacific-Asia Conference, PAKDD 2015, Ho Chi Minh City, Vietnam, May 19-22, 2015, Proceedings, Part II 19*, pages 331–343, 2015.

[46] Yury Gorishniy, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. Revisiting deep learning models for tabular data. In *Advances in Neural Information Processing Systems*, pages 18932–18943, 2021.

[47] Leo Grinsztajn, Edouard Oyallon, and Gael Varoquaux. Why do tree-based models still outperform deep learning on typical tabular data? In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022.

[48] Yifan Gu, Xuebing Yang, Lei Tian, Hongyu Yang, Jicheng Lv, Chao Yang, Jinwei Wang, Jianing Xi, Guilan Kong, and Wensheng Zhang. Structure-aware siamese graph neural networks for encounter-level patient similarity learning. *Journal of Biomedical Informatics*, 127:104027, 2022.

[49] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. Deepfm: A factorization-machine based neural network for ctr prediction. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, IJCAI'17, pages 1725–1731, 2017.

[50] Wei Guo, Rong Su, Renhao Tan, Huifeng Guo, Yingxue Zhang, Zhirong Liu, Ruiming Tang, and Xiuqiang He. Dual graph enhanced embedding neural network for ctr prediction. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, KDD '21, pages 496–504, 2021.

[51] Xiawei Guo, Yuhan Quan, Huan Zhao, Quanming Yao, Yong Li, and Weiwei Tu. Tabgnn: Multiplex graph neural network for tabular data prediction. *The 3rd Workshop on Deep Learning Practice for High-Dimensional Sparse Data with KDD (DLP-KDD)*, 2021.

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

[53] Qianlong He, Feng Zhou, Linyan Gu, and Zhibin Yuan. A novel graph-based feature interaction model for click-through rate prediction. *Information Sciences*, 649:119615, 2023.

[54] Bhagya Hettige, Yuan-Fang Li, Weiqing Wang, Suong Le, and Wray L. Buntine. Medgraph: Structural and temporal representation learning of electronic medical records. In *The 24th European Conference on Artificial Intelligence (ECAI)*, pages 1810–1817, 2020.

[55] Cheng Hsu and Cheng-Te Li. Retagnn: Relational temporal attentive graph neural networks for holistic sequential recommendation. In *Proceedings of the Web Conference 2021*, WWW '21, pages 2968–2979, 2021.

[56] Yi-Ling Hsu, Yu-Che Tsai, and Cheng-Te Li. Fingat: Financial graph attention networks for recommending top-k profitable stocks. *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[57] Binbin Hu, Zhiqiang Zhang, Jun Zhou, Jingli Fang, Quanhui Jia, Yanming Fang, Quan Yu, and Yuan Qi. Loan default analysis

with multiplex graph learning. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, CIKM '20, pages 2525–2532, 2020.

[58] Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. Heterogeneous graph transformer. In *Proceedings of The Web Conference 2020*, pages 2704–2710, 2020.

[59] Buliao Huang, Yunhui Zhu, Muhammad Usman, Xiren Zhou, and Huanhuan Chen. Graph neural networks for missing value classification in a task-driven metric space. *IEEE Transactions on Knowledge and Data Engineering*, 35(8):8073–8084, 2023.

[60] Mengda Huang, Yang Liu, Xiang Ao, Kuan Li, Jianfeng Chi, Jinghua Feng, Hao Yang, and Qing He. Auc-oriented graph neural network for fraud detection. In *Proceedings of the ACM Web Conference 2022*, WWW '22, pages 1311–1321, 2022.

[61] Xiao Huang, Qingquan Song, Fan Yang, and Xia Hu. Large-scale heterogeneous feature embedding. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence*, AAAI, 2019.

[62] Sergei Ivanov and Liudmila Prokhorenkova. Boost then convolve: Gradient boosting meets graph neural networks. In *International Conference on Learning Representations*, 2021.

[63] Zhong Jiajun, Ye Weiwei, and Gui Ning. Data imputation with iterative graph reconstruction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2023.

[64] Licheng Jiao, Jie Chen, Fang Liu, Shuyuan Yang, Chao You, Xu Liu, Lingling Li, and Biao Hou. Graph representation learning meets computer vision: A survey. *IEEE Transactions on Artificial Intelligence*, 4(1):2–22, 2023.

[65] Ming Jin, Huan Yee Koh, Qingsong Wen, Daniele Zambon, Cesare Alippi, Geoffrey I Webb, Irwin King, and Shirui Pan. A survey on graph neural networks for time series: Forecasting, classification, imputation, and anomaly detection. *arXiv preprint arXiv:2307.03759*, 2023.

[66] Wei Jin, Xiaorui Liu, Yao Ma, Charu Aggarwal, and Jiliang Tang. Feature overcorrelation in deep graph neural networks: A new perspective. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD'22, page 709–719, 2022.

[67] Wei Jin, Yao Ma, Xiaorui Liu, Xianfeng Tang, Suhang Wang, and Jiliang Tang. Graph structure learning for robust graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '20, pages 66–74, 2020.

[68] Seokho Kang. K-nearest neighbor learning with graph neural networks. *Mathematics*, 9(8):830, 2021.

[69] Anees Kazi, Luca Cosmo, Seyed-Ahmad Ahmadi, Nassir Navab, and Michael M Bronstein. Differentiable graph module (dgm) for graph convolutional networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(2):1606–1617, 2022.

[70] Guolin Ke, Zhenhui Xu, Jia Zhang, Jiang Bian, and Tie-Yan Liu. Deepgbm: A deep learning framework distilled by gbdt for online prediction tasks. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, page 384–394, 2019.

[71] Geun-Hyeong Kim, Jae-Woo Kim, Ka Hyun Kim, Hyeran Kang, Jae Young Moon, Yoon Mi Shin, and Seung Park. Ft-gat: Graph neural network for predicting spontaneous breathing trial success in patients with mechanical ventilation. *Computer Methods and Programs in Biomedicine*, page 107673, 2023.

[72] Jinwoo Kim, Dat Tien Nguyen, Seonwoo Min, Sungjun Cho, Moontae Lee, Honglak Lee, and Seunghoon Hong. Pure transformers are powerful graph learners. In *Advances in Neural Information Processing Systems*, 2022.

[73] Minkyu Kim, Hyun-Soo Choi, and Jinho Kim. Explicit feature interaction-aware graph neural networks. *arXiv preprint arXiv:2204.03225*, 2022.

[74] Raehyun Kim, Chan Ho So, Minbyul Jeong, Sanghoon Lee, Jinkyu Kim, and Jaewoo Kang. Hats: A hierarchical graph attention network for stock movement prediction. *arXiv preprint arXiv:1908.07999*, 2019.

[75] Sundong Kim, Yu-Che Tsai, Karandeep Singh, Yeonsoo Choi, Etim Ibok, Cheng-Te Li, and Meeyoung Cha. Date: Dual attentive tree-aware embedding for customs fraud detection. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD'20, pages 2880–2890, 2020.

[76] Thomas N Kipf and Max Welling. Variational graph auto-encoders. *Bayesian Deep Learning Workshop (NIPS 2016)*, 2016.

[77] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

[78] Boshko Koloski, Blaž Škrlj, Senja Pollak, and Nada Lavrač. Latent graph powered semi-supervised learning on biomedical tabular data. *arXiv preprint arXiv:2309.15757*, 2023.

[79] Evgeny Krivosheev, Mattia Atzeni, Katsiaryna Mirylenka, Paolo Scotton, and Fabio Casati. Siamese graph neural networks for data integration. *arXiv preprint arXiv:2001.06543*, 2020.

[80] Kuan Li, Yang Liu, Xiang Ao, and Qing He. Revisiting graph adversarial attack and defense from a data distribution perspective. In *The Eleventh International Conference on Learning Representations*, 2023.

[81] Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[82] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. In *International Conference on Learning Representations*, 2016.

[83] Zekun Li, Zeyu Cui, Shu Wu, Xiaoyu Zhang, and Liang Wang. Fi-gnn: Modeling feature interactions via graph neural networks for ctr prediction. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, CIKM '19, pages 539–548, 2019.

[84] Jianxun Lian, Xiaohuan Zhou, Fuzheng Zhang, Zhongxia Chen, Xing Xie, and Guangzhong Sun. Xdeepfm: Combining explicit and implicit feature interactions for recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '18, pages 1754–1763, 2018.

[85] Jay Chiehen Liao and Cheng-Te Li. Tabgsl: Graph structure learning for tabular data prediction. *arXiv preprint arXiv:2305.15843*, 2023.

[86] Can Liu, Li Sun, Xiang Ao, Jinghua Feng, Qing He, and Hao Yang. Intention-aware heterogeneous graph attention networks for fraud transactions detection. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, KDD '21, page 3280–3288, 2021.

[87] Chaoqiang Liu, Haifeng Liu, Long Zheng, Yu Huang, Xiangyu Ye, Xiaofei Liao, and Hai Jin. Fnng: A high-performance fpga-based accelerator for k-nearest neighbor graph construction. In *Proceedings of the 2023 ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, FPGA '23, pages 67–77, 2023.

[88] Weifeng Liu, Puskal P Pokharel, and Jose C Principe. Correntropy: Properties and applications in non-gaussian signal processing. *IEEE Transactions on signal processing*, 55(11):5286–5298, 2007.

[89] Yang Liu, Xiang Ao, Zidi Qin, Jianfeng Chi, Jinghua Feng, Hao Yang, and Qing He. Pick and choose: A gnn-based imbalanced learning approach for fraud detection. In *Proceedings of the Web Conference 2021*, WWW '21, pages 3168–3177, 2021.

[90] Yixin Liu, Ming Jin, Shirui Pan, Chuan Zhou, Yu Zheng, Feng Xia, and Philip S. Yu. Graph self-supervised learning: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 35(6):5879–5900, 2023.

[91] Yixin Liu, Yu Zheng, Daokun Zhang, Hongxu Chen, Hao Peng, and Shirui Pan. Towards unsupervised deep graph structure learning. In *Proceedings of the ACM Web Conference 2022*, WWW '22, pages 1392–1403, 2022.

[92] Yuchen Liu, Chuanzhen Li, Han Xiao, and Juanjuan Cai. Gcn-int: A click-through rate prediction model based on graph convolutional network interaction. *IEEE Access*, 9:140022–140030, 2021.

[93] Zheng Liu, Xiaohan Li, Hao Peng, Lifang He, and Philip S. Yu. Heterogeneous similarity graph neural network on electronic health records. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 1196–1205, 2020.

[94] Jianyu Long, Rongxin Zhang, Zhe Yang, Yunwei Huang, Yu Liu, and Chuan Li. Self-adaptation graph attention network via meta-learning for machinery fault diagnosis with few labeled data. *IEEE Transactions on Instrumentation and Measurement*, 71:1–11, 2022.

[95] Haohui Lu and Shahadat Uddin. A weighted patient network-based framework for predicting chronic diseases using graph neural networks. *Scientific reports*, 11(1):1–12, 2021.

[96] Katja Luck, Dae-Kyum Kim, Luke Lambourne, Kerstin Spirohn, Bridget E Begg, Wenting Bian, Ruth Brignall, Tiziana Cafarelli,

Francisco J Campos-Laborie, Benoit Charloteaux, et al. A reference map of the human binary protein interactome. *Nature*, 580(7803):402–408, 2020.

[97] Handong Ma, Changsheng Li, Xinchu Shi, Ye Yuan, and Guoren Wang. Deep unsupervised active learning on learnable graphs. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–7, 2022.

[98] Xiaoxiao Ma, Jia Wu, Shan Xue, Jian Yang, Chuan Zhou, Quan Z. Sheng, Hui Xiong, and Leman Akoglu. A comprehensive survey on graph anomaly detection with deep learning. *IEEE Transactions on Knowledge and Data Engineering*, 35(12):12012–12038, 2023.

[99] Yao Ma, Xiaorui Liu, Neil Shah, and Jiliang Tang. Is homophily a necessity for graph neural networks? In *International Conference on Learning Representations*, 2022.

[100] Qi Mao, Li Wang, Steve Goodison, and Yijun Sun. Dimensionality reduction via graph structure learning. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, pages 765–774, 2015.

[101] Andrei Margeloiu, Nikola Simidjievski, Pietro Lio, and Mateja Jamnik. GCondnet: A novel method for improving neural networks on small high-dimensional tabular data. In *NeurIPS 2023 Second Table Representation Learning Workshop*, 2023.

[102] Miller McPherson, Lynn Smith-Lovin, and James M Cook. Birds of a feather: Homophily in social networks. *Annual review of sociology*, 27(1):415–444, 2001.

[103] Jaehyun Nam, Jihoon Tack, Kyungmin Lee, Hankook Lee, and Jinwoo Shin. STUNT: Few-shot tabular learning with self-generated tasks from unlabeled tables. In *The Eleventh International Conference on Learning Representations*, 2023.

[104] Mark Newman. *Networks*. Oxford university press, 2018.

[105] Wentao Ouyang, Xiuwu Zhang, Shukui Ren, Li Li, Kun Zhang, Jinmei Luo, Zhaojie Liu, and Yanlong Du. Learning graph meta embeddings for cold-start ads in click-through rate prediction. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '21, pages 1157–1166, 2021.

[106] Hao Peng, Jianxin Li, Zheng Wang, Renyu Yang, Mingsheng Liu, Mingming Zhang, Philip S. Yu, and Lifang He. Lifelong property price prediction: A case study for the toronto real estate market. *IEEE Transactions on Knowledge and Data Engineering*, 35(3):2765–2780, 2023.

[107] Hao Peng, Ruitong Zhang, Yingtong Dou, Renyu Yang, Jingyi Zhang, and Philip S. Yu. Reinforced neighborhood selection guided multi-relational graph neural networks. *ACM Trans. Inf. Syst.*, 40, 2021.

[108] Jiarui Qin, Weinan Zhang, Rong Su, Zhirong Liu, Weiwen Liu, Ruiming Tang, Xiuqiang He, and Yong Yu. Retrieval & interaction machine for tabular data prediction. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, KDD '21, pages 1379–1389, 2021.

[109] Ladislav Rampasek, Mikhail Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. Recipe for a general, powerful, scalable graph transformer. In *Advances in Neural Information Processing Systems*, 2022.

[110] Susie Xi Rao, Shuai Zhang, Zhichao Han, Zitao Zhang, Wei Min, Zhiyao Chen, Yinan Shan, Yang Zhao, and Ce Zhang. Xfraud: Explainable fraud transaction detection. *Proc. VLDB Endow.*, 15(3):427–436, 2021.

[111] Steffen Rendle. Factorization machines. In *2010 IEEE International conference on data mining*, pages 995–1000. IEEE, 2010.

[112] Emma Rocheteau, Catherine Tong, Petar Velickovic, Nicholas D. Lane, and Pietro Liò. Predicting patient outcomes with graph representation learning. *International Workshop on Deep Learning on Graphs: Method and Applications (DLG-AAAI)*, 2021.

[113] Ryan A Rossi, Di Jin, Sungchul Kim, Nesreen K Ahmed, Danai Koutra, and John Boaz Lee. On proximity and structural role-based embeddings in networks: Misconceptions, techniques, and applications. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 14(5):1–37, 2020.

[114] Camilo Ruiz, Hongyu Ren, Kexin Huang, and Jure Leskovec. High dimensional, tabular deep learning with an auxiliary knowledge graph. In *Thirty-seventh Conference on Neural Information Processing Systems*.

[115] Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne vanden Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *The Semantic Web*, 2018.

[116] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 464–468, 2018.

[117] Fengzhao Shi, Yanan Cao, Yanmin Shang, Yuchen Zhou, Chuan Zhou, and Jia Wu. H2-fdetector: A gnn-based fraud detector with homophilic and heterophilic connections. In *Proceedings of the ACM Web Conference 2022*, WWW '22, pages 1486–1494, 2022.

[118] Karandeep Singh, Yu-Che Tsai, Cheng-Te Li, Meeyoug Cha, and Shou-De Lin. Graphfc: Customs fraud detection with label scarcity. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, CIKM '23, 2023.

[119] Indro Spinelli, Simone Scardapane, and Aurelio Uncini. Missing data imputation with adversarially-trained graph convolutional networks. *Neural Networks*, 129:249–260, 2020.

[120] Chenhao Su, Sheng Gao, and Si Li. Gate: Graph-attention augmented temporal neural network for medication recommendation. *IEEE Access*, 8:125447–125458, 2020.

[121] Zhenchao Sun, Hongzhi Yin, Hongxu Chen, Tong Chen, Lizhen Cui, and Fan Yang. Disease prediction via graph neural networks. *IEEE Journal of Biomedical and Health Informatics*, 25(3):818–826, 2021.

[122] Ryutaro Tanno, Kai Arulkumaran, Daniel Alexander, Antonio Criminisi, and Aditya Nori. Adaptive neural trees. In *Proceedings of the 36th International Conference on Machine Learning*, pages 6166–6175, 2019.

[123] Lev Telyatnikov and Simone Scardapane. Egg-gae: scalable graph neural networks for tabular data imputation. In *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, pages 2661–2676, 2023.

[124] Josephine Thomas, Alice Moallemy-Oureh, Silvia Beddar-Wiesing, and Clara Holzhüter. Graph neural networks designed for different graph types: A survey. *Transactions on Machine Learning Research*, 2023.

[125] Talip Ucar, Ehsan Hajiramezanali, and Lindsay Edwards. Subtab: Subsetting features of tabular data for self-supervised representation learning. In *Advances in Neural Information Processing Systems*, 2021.

[126] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

[127] Mario Villaizán-Vallelado, Matteo Salvatori, Belén Carro Martinez, and Antonio Javier Sanchez Esguevillas. Graph neural network contextual embedding for deep learning on tabular data. *arXiv preprint arXiv:2303.06455*, 2023.

[128] Liane Vogel, Benjamin Hilprecht, and Carsten Binnig. Towards foundation models for relational databases [vision paper]. In *NeurIPS 2022 First Table Representation Workshop*, 2022.

[129] Alvin Wan, Lisa Dunlap, Daniel Ho, Jihan Yin, Scott Lee, Suzanne Petryk, Sarah Adel Bargal, and Joseph E. Gonzalez. {NBDT}: Neural-backed decision tree. In *International Conference on Learning Representations*, 2021.

[130] Li Wang, Peipei Li, Kai Xiong, Jiashu Zhao, and Rui Lin. Modeling heterogeneous graph network on fraud detection: A community-based framework with attention mechanism. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 1959–1968, 2021.

[131] Li Wang and Ren-cang Li. Learning low-dimensional latent graph structures: A density estimation approach. *IEEE Transactions on Neural Networks and Learning Systems*, 31(4):1098–1112, 2019.

[132] Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. Deep & cross network for ad click predictions. In *Proceedings of the ADKDD'17*, ADKDD'17, 2017.

[133] Shoujin Wang, Liang Hu, Yan Wang, Xiangnan He, Quan Z. Sheng, Mehmet A. Orgun, Longbing Cao, Francesco Ricci, and Philip S. Yu. Graph learning based recommender systems: A review. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 4644–4652. International Joint Conferences on Artificial Intelligence Organization, 2021.

[134] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. Heterogeneous graph attention network. In *The World Wide Web Conference*, WWW '19, pages 2022–2032, 2019.

[135] Xin Wang, Peng Yang, Shaopeng Chen, Lin Liu, Lian Zhao, Jiacheng Guo, Mingming Sun, and Ping Li. Efficient learning

to learn a robust ctr model for web-scale online sponsored search advertising. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, CIKM'21, pages 4203—–4213, 2021.

[136] Yifan Wang, Yifang Qin, Fang Sun, Bo Zhang, Xuyang Hou, Ke Hu, Jia Cheng, Jun Lei, and Ming Zhang. Disenctr: Dynamic graph-based disentangled representation for click-through rate prediction. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '22, page 2314–2318, 2022.

[137] David S Wishart, Yannick D Feunang, An C Guo, Elvis J Lo, Ana Marcu, Jason R Grant, Tanvir Sajed, Daniel Johnson, Carin Li, Zinat Sayeeda, et al. Drugbank 5.0: a major update to the drugbank database for 2018. *Nucleic acids research*, 46(D1):D1074–D1082, 2018.

[138] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *International conference on machine learning*, pages 6861–6871. PMLR, 2019.

[139] Le Wu, Xiangnan He, Xiang Wang, Kun Zhang, and Meng Wang. A survey on accuracy-oriented neural recommendation: From collaborative filtering to information-rich recommendation. *IEEE Trans. on Knowl. and Data Eng.*, 35(5):4425–4445, may 2023.

[140] Lingfei Wu, Yu Chen, Kai Shen, Xiaojie Guo, Hanning Gao, Shucheng Li, Jian Pei, and Bo Long. Graph neural networks for natural language processing: A survey. *Foundations and Trends in Machine Learning*, 16(2):119–328, 2023.

[141] Lirong Wu, Haitao Lin, Zihan Liu, Zicheng Liu, Yufei Huang, and Stan Z Li. Homophily-enhanced self-supervision for graph structure learning: Insights and directions. *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

[142] Qitian Wu, Chenxiao Yang, and Junchi Yan. Towards open-world feature extrapolation: An inductive graph learning approach. In *Advances in Neural Information Processing Systems*, 2021.

[143] Qitian Wu, Hengrui Zhang, Junchi Yan, and David Wipf. Handling distribution shifts on graphs: An invariance perspective. In *International Conference on Learning Representations*, 2022.

[144] Shiwen Wu, Fei Sun, Wentao Zhang, Xu Xie, and Bin Cui. Graph neural networks in recommender systems: A survey. *ACM Comput. Surv.*, 55(5), dec 2022.

[145] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):4–24, 2021.

[146] Lu Xiao, Xiaoxin Yang, and Xiaodong Yang. A graph neural network-based bearing fault detection method. *Scientific Reports*, 13(1):5286, 2023.

[147] Cong Xie, Wen Zhong, Wei Xu, and Klaus Mueller. Visual analytics of heterogeneous data using hypergraph learning. *ACM Trans. Intell. Syst. Technol.*, 10(1), dec 2018.

[148] Jiaqing Xie and Rex Ying. Fea2fea: Exploring structural feature correlations via graph neural networks. In *Machine Learning and Principles and Practice of Knowledge Discovery in Databases: International Workshops of ECML PKDD 2021*, pages 238–257, 2022.

[149] Yuexiang Xie, Zhen Wang, Yaliang Li, Bolin Ding, Nezihe Merve Gürel, Ce Zhang, Minlie Huang, Wei Lin, and Jingren Zhou. Fives: Feature interaction via edge search for large-scale tabular data. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, KDD '21, pages 3795–3805, 2021.

[150] Bingbing Xu, Huawei Shen, Bingjie Sun, Rong An, Qi Cao, and Xueqi Cheng. Towards consumer loan fraud detection: Graph neural networks with role-constrained conditional random field. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 4537–4545, 2021.

[151] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019.

[152] Jiahuan Yan, Jintai Chen, Yixuan Wu, Danny Z Chen, and Jian Wu. T2g-former: Organizing tabular features into relation graphs promotes heterogeneous feature interaction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2023.

[153] Shuo Yang, Zhiqiang Zhang, Jun Zhou, Yang Wang, Wang Sun, Xingyu Zhong, Yanming Fang, Quan Yu, and Yuan Qi. Financial risk analysis for smes with graph-based supply chain mining. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, pages 4661–4667, 2020.

[154] Rex Ying, Jiaxuan You, Christopher Morris, Xiang Ren, William L. Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, pages 4805–4815, 2018.

[155] Zhitao Ying, Dylan Bourgeois, Jiaxuan You, Marinka Zitnik, and Jure Leskovec. Gnnexplainer: Generating explanations for graph neural networks. *Advances in neural information processing systems*, 32, 2019.

[156] Jinsung Yoon, Yao Zhang, James Jordon, and Mihaela van der Schaar. Vime: Extending the success of self- and semi-supervised learning to tabular domain. In *Advances in Neural Information Processing Systems*, pages 11033–11043, 2020.

[157] Jiaxuan You, Xiaobai Ma, Yi Ding, Mykel J Kochenderfer, and Jure Leskovec. Handling missing data with graph representation learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 19075–19087, 2020.

[158] Haiyang Yu, Limei Wang, Bokun Wang, Meng Liu, Tianbao Yang, and Shuiwang Ji. Graphfm: Improving large-scale gnn training via feature momentum. In *International Conference on Machine Learning*, pages 25684–25701, 2022.

[159] Hongyuan Yu, Ting Li, Weichen Yu, Jianguo Li, Yan Huang, Liang Wang, and Alex Liu. Regularized graph structure learning with semantic knowledge for multi-variates time-series forecasting. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pages 2362–2368, 2022.

[160] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. *Advances in neural information processing systems*, 30, 2017.

[161] Hanqing Zeng, Hongkuan Zhou, Ajitesh Srivastava, Rajgopal Kannan, and Viktor Prasanna. Graphsaint: Graph sampling based inductive learning method. In *International Conference on Learning Representations*, 2020.

[162] Panyu Zhai, Yanwu Yang, and Chunjie Zhang. Causality-based ctr prediction using graph neural networks. *Information Processing & Management*, 60(1):103137, 2023.

[163] Han Zhang, Quan Gan, David Wipf, and Weinan Zhang. Gfs: Graph-based feature synthesis for prediction over relational databases. *arXiv preprint arXiv:2312.02037*, 2023.

[164] Jian Zhang, Fan Yang, Kaibiao Lin, and Yongxuan Lai. Hierarchical multi-modal fusion on dynamic heterogeneous graph for health insurance fraud detection. In *2022 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6, 2022.

[165] Shuai Zhang, Meng Wang, Pin-Yu Chen, Sijia Liu, Songtao Lu, and Miao Liu. Joint edge-model sparse learning is provably efficient for graph neural networks. In *The Eleventh International Conference on Learning Representations*, 2023.

[166] Weinan Zhang, Jiarui Qin, Wei Guo, Ruiming Tang, and Xiuqiang He. Deep learning for click-through rate estimation. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 4695–4703, 2021.

[167] Wentao Zhang, Mingyu Yang, Zeang Sheng, Yang Li, Wen Ouyang, Yangyu Tao, Zhi Yang, and Bin Cui. Node dependent local smoothing for scalable graph learning. *Advances in Neural Information Processing Systems*, 34:20321–20332, 2021.

[168] Ziwei Zhang, Peng Cui, and Wenwu Zhu. Deep learning on graphs: A survey. *IEEE Trans. on Knowl. and Data Eng.*, 34(1):249–270, jan 2022.

[169] Lingxiao Zhao and Leman Akoglu. Pairnorm: Tackling oversmoothing in gnns. In *International Conference on Learning Representations*, 2020.

[170] Qinghua Zheng, Zhen Peng, Zhuohang Dang, Linchao Zhu, Ziqi Liu, Zhiqiang Zhang, and Jun Zhou. Deep tabular data modeling with dual-route structure-adaptive graph networks. *IEEE Transactions on Knowledge and Data Engineering*, 35(9):9715–9727, 2023.

[171] Yu Zheng, Yongxin Yang, and Bowei Chen. Incorporating prior financial domain knowledge into neural networks for implied volatility surface prediction. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, KDD '21, pages 3968–3975, 2021.

[172] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI open*, 1:57–81, 2020.

[173] Kaixiong Zhou, Zirui Liu, Rui Chen, Li Li, and Xia Hu Soo-Hyun Choi. Table2graph: Transforming tabular data to unified weighted graph. In *Proceedings of the 31st International Joint Conference on Artificial Intelligence*, IJCAI '22, 2022.

[174] Yanqiao Zhu, Weizhi Xu, Jinghao Zhang, Yuanqi Du, Jieyu Zhang, Qiang Liu, Carl Yang, and Shu Wu. A survey on graph structure learning: Progress and opportunities. In *The 31st International Joint Conference on Artificial Intelligence (IJCAI)*, 2022.