Reversible Column Disentangled Augmentation Tricks for Graph Contrastive Learning

Yuntai Ding, Tao Ren, Yifan Wang, Chong Chen, Xian-Sheng Hua, *Fellow, IEEE*, and Wei Ju

***Abstract*—Graph contrastive learning (GCL) has garnered significant attention for its self-supervised graph representation learning without label information and excellent generalization to downstream tasks. However, data augmentation for graph-structured data is more challenging than that for images. We argue that simple data augmentations for GCL may risk damaging the intrinsic structure of the graph or creating views that are not diverse enough. Additionally, typical layer-by-layer feature propagation processes compress or discard pretext task-irrelevant feature information, resulting in unstable and suboptimal performance for unaligned downstream tasks. In this paper, we propose a novel framework termed RevGCL, which aims to maintain multi-level graph semantics without information loss via reversible column disentangled model augmentation tricks. Specifically, we propose a multi-column network with reversible connections as our encoder, where all columns share the same structure and receive a copy of the input graph. The reversible connections between columns ensure lossless transmission, allowing representations to be gradually disentangled from low-level to high-level semantics. Based on this, we introduce two model augmentation tricks, random propagation and asymmetric column, to construct different sibling encoders. These methods generate diverse graph views that can filter out high-frequency noise in contrastive learning, thereby yielding more generalizable node feature representations. Extensive experiments on eight commonly benchmark datasets demonstrate that RevGCL consistently outperforms existing state-of-the-art methods in node classification, clustering and link prediction tasks.**

***Index Terms*—Graph contrastive learning, disentangled representation learning, reversible networks.**

Received 16 July 2024; revised 10 February 2025; accepted 23 March 2025. Date of publication 9 October 2025; date of current version 17 December 2025. This work was supported in part by the National Natural Science Foundation of China under Grant 62276058 and Grant 41774063, in part by the Fundamental Research Funds for the Central Universities under Grant N25GFZ011, in part by the Fundamental Research Funds for the Central Universities in UIBE under Grant 23QN02, in part by the Natural Science Foundation of Liaoning Province under Grant 2023010411-JH3/101, and in part by the Key Laboratory of Optical Information and Simulation Technology of Liaoning Province. The associate editor coordinating the review of this article and approving it for publication was Prof. Antonio Pinheiro. *(Corresponding author: Yifan Wang.)*

Yuntai Ding and Tao Ren are with the Software College, Northeastern University, Shenyang 110169, China (e-mail: chinaytding@163.com; chinarentao@163.com).

Yifan Wang is with the School of Information Technology and Management, University of International Business and Economics, Beijing 100029, China (e-mail: yifanwang@uibe.edu.cn).

Chong Chen is with Terminus Group, Beijing 100027, China (e-mail: chenchong.cz@gmail.com).

Xian-Sheng Hua is with the Terminus Group, Beijing 100027, China, and also with the Institute of AI for Engineering, Tongji University, Shanghai 201210, China (e-mail: huaxiansheng@gmail.com).

Wei Ju is with the College of Computer Science, Sichuan University, Chengdu 610065, China (e-mail: juwei@scu.edu.cn).

Digital Object Identifier 10.1109/TMM.2025.3618580

I. Introduction

GRAPHS serve as a universal data representation for modeling complex relationships between objects in a wide range of real-world multimedia applications, e.g., molecule graphs [1], biology networks [2], citation networks [3], social networks [4] and recommender systems [5], [6], etc. For graph-structured data, Graph Neural Networks (GNNs) [7] are widely acknowledged for learning graph patterns by employing a message-passing scheme, where the node's representation is updated by leveraging information from neighboring nodes and connected edges. However, training high-quality GNNs usually requires a substantial amount of task-specific labels, which is highly expensive and time-consuming.

Contrastive learning (CL) has gained significant attention as a promising approach to address label dependence [8]. By pulling similar samples together and pushing dissimilar samples apart, the method is able to learn the discriminative representation within a self-supervised learning framework and achieves significant success across multiple downstream tasks. Motivated by the impact of contrastive learning in the visual domain, researchers have extensively explored its application to graph data. Despite technical variations, most graph contrastive learning (GCL) methods share the same high-level framework as those in the visual domain. Typically, GCL first generates different views of the source graphs, and then optimizes the GNN encoder by contrasting positive and negative views. Therefore, the effectiveness of GCL heavily relies on the quality of the augmented graph samples.

Generally, data augmentations for graphs are motivated by developments in computer vision. In image data augmentation, transformed samples are generated by applying random operations to the target image, e.g., resizing, cropping, rotating, and color distortion. Similar to image data augmentation, common techniques for graph data augmentation include node feature masking, node feature shuffling, edge modification, graph diffusion, and subgraph sampling [8]. For example, DGI [9] employs a corruption function to get the augmented negative graph by a row-wise shuffling of the node feature matrix. GRACE [10] and CCA-SSG [11] generate an augmented graph as a positive sample by randomly masking node features and dropping edges. MVGRL [12] leverages two different graph diffusion methods based on heat kernel [13] and Personalized PageRank [14] to capture global graph structural information. GraphCL [15] and JOVO [16] conduct a systematic analysis of the effects of four data augmentation methods and their combinations on downstream tasks. AD-GCL [17] and ARIEL [18] generate the

adversarial graph view by maximizing the graph contrastive loss in a learning-based manner.

Despite the encouraging performance of these GCL methods, we argue that existing graph augmentation methods continue to be inadequate due to the following constraints. *(1) Heavily rely on the human prior knowledge.* Unlike image data augmentation, where we have intuitive visual understanding of how to alter an image without changing its labels, graph data augmentation is far less intuitive for humans. This can result in graph augmentations that are either too similar to or drastically different from the original graph. *(2) Neglect the nuances between different levels of semantics within the graph.* The Information Bottleneck (IB) principle [19] posits that lower layers in GNNs capture more local information, while higher layers closer to the output contain richer semantic information. This means that irrelevant information is gradually filtered out through the layers. Although this approach has proven to be highly effective in many practical applications, it may not be the best choice when the learned semantic information does not align with the target tasks. This issue is particularly significant in self-supervised contrastive learning frameworks, where varying levels of semantic information can also impact downstream task performance.

Recently, disentangled representation learning has been increasingly investigated. Instead of the IB principle, which extracts the most relevant information while discarding the less relevant, disentangled representation learning aims at learning factorized representations that are able to uncover different task-relevant information in a few decoupled dimensions. For graph-structured data, disentangled representation learning aims to break down underlying relationships and primarily focuses on identifying the latent factors that create edges between nodes. For example, DisenGCN [20] learns to disentangle the latent factors behind graph data via a neighborhood partition. LGD-GCN [21] further considers the global factor-specific information of nodes via constructing different subspaces w.r.t. obtained different latent disentangled units. However, the extraction of multiple levels of relevant semantic representations in response to the demands of downstream tasks for each node remains largely unexplored.

In this work, we propose **Rev-GCL**, a novel Reversible column-based model augmentation trick to learn disentangled representation for Graph Contrastive Learning, which is greatly inspired by the family of Reversible Networks [22], [23], [24], [25], [26]. Different from previous GCL methods that construct different graph views and use identical GNNs as view encoders, our model focuses on model augmentation tricks for encoders to generate contrastive views. Specifically, our graph encoder consists of multiple subnetworks, referred to as columns, which share the same structure but with different weights. For each column, it receives a copy of the graph and generates embeddings at multiple GNN levels, ranging from low to highly semantic representations. Along with feature propagation among nodes, reversible connections are introduced to transfer features between columns in a bottom-up manner, ensuring a gradual learning of disentangled representations without any information loss. Based on the proposed reversible columns encoder, we introduce two data augmentation tricks: (1) *Random Propagation:* we treat each GNN layer as a composition of transformation and propagation operators and utilize a random number of propagation operators during each training epoch. (2) *Asymmetric Column:* we employ two view encoders with shared weights and select contrastive views from different column outputs between the two sibling encoders. Through reversible column disentangled augmentation tricks, we innovatively introduce the reversible network into graph contrastive learning, ensuring the lossless transmission of semantic information across different level while progressively achieving disentanglement. This combination cleverly realizes graph disentangled representation learning, which has been underexplored in previous research. Moreover, we also design a novel model augmentation strategy tailored for the multi-column reversible module.

In this paper, we summarize our Rev-GCL into the following contributions:

- *Conceptual:* We highlight the key limitations of current data augmentation GCL methods and present two model augmentation tricks based on two sibling disentangled graph encoders for more robust GCL.
- *Methodological:* We integrate reversible networks into graph contrastive learning for disentangled representations and propose a multi-column network with reversible connections between adjacent columns as our encoder, which could extract varying levels of semantic representations relevant to downstream tasks.
- *Experimental:* We conduct comprehensive experiments among public datasets to assess the performance of Rev-GCL. Experimental results demonstrate the effectiveness of our proposed framework on downstream tasks.

II. Related Work

We review existing works on three lines of fields: 1) Graph Contrastive Learning, 2) Disentangled Representation Learning and 3) Reversible Networks.

A. Graph Contrastive Learning

Generally, GCL methods follow a common paradigm that generates different graph views to maximize consistency. Based on how to generate contrastive views, we roughly categorized GCL methods into two groups. The first group generates views following pre-defined rules. For example, DGI [27] introduces row-wise shuffling as a corruption function to obtain the negative example from the original graph. GRACE [10] and GraphCL [15] utilize distinct graph augmentation methods (i.e., node/edge dropping, feature masking) to randomly perturb the graph structure. GCA [28] jointly performs adaptive edge dropping and feature masking as graph augmentations by identifying important edges and feature dimensions. MVGRL [12] employs graph diffusion to generate additional structural views of the input graph. BGRL [29] and CCA-SSG [11] employs simple graph augmentations (feature/edge masking) and eliminates the need for contrasting with negative samples. The second group aims to learn graph augmentation strategies in a data-driven manner. Info-GCL [30] utilizes task-relevant information to select optimized views according to the information bottleneck principle. AD-GCL [17] and ARIEL [18] develop an adversarial

way to generate augmented graph views for GCL and these views can be seen as hard training samples of GCL. RGCL [31] creates rationale-aware graph views to discover invariant structures for GCL. Besides, there are also some methods that consider different augmentation optimization. COLES [32] introduces Laplacian Eigenmaps into the negative sampling process of GCL. SimGRACE [33] utilizes an encoder perturbed by Gaussian noise to generate contrastive views without data augmentation. MA-GCL [34] introduces view encoders with structural random perturbations instead of perturbing the original graph and training parameters for GCL. Although these methods adopt different strategies to optimize the contrastive learning paradigm, they generally ignore the information loss under the principle of IB and disentanglement of semantic information, resulting in suboptimal and unstable performance results.

B. Disentangled Representation Learning

Disentangled representation learning aims to obtain factorized representations that could distinguish and separate the intrinsic explanatory factors underlying the observed data [35]. Existing disentangled representation learning efforts are mainly focus on computer vision [36], [37], natural language processing [3] and recommender systems [5], [38], [39], [40], [41], [42], [43]. For example, β-VAE [36] introduces the weight β for the KL divergence term in the variational autoencoders (VAEs) objective to control the dimensional independence. InfoGAN [37] decomposes the representation into noise and additional class codes, estimating the mutual information between the class code and the corresponding data to enable controllable image generation. DisenCite [3] leverages disentangled representation to separate the semantics of different sections in the paper for context-specific citation generation. MacridVAE [38] models hierarchical user intentions from user behavior data via macro and micro representation disentanglement. Recently, there has been growing interest in leveraging disentangled representation learning techniques for graph-structured data [5], [20], [44], [45], [46]. DisenGCN [20] assumes that nodes are connected for different reasons and proposes a neighborhood routing mechanism to partition the neighborhood into exclusive parts corresponding to these semantics. DisenHAN [5] utilizes the heterogeneous information network to learn disentangled representation by iteratively identifying the primary aspects of relations type and semantically propagating corresponding information. FactorGCN [44] processes the entire graph into multiple block-wise interpretable subgraphs to decouple the underlying relationships between entities. DGCL [45] introduces a self-supervised framework for disentangled graph representation using contrastive learning. Our work focuses on preserving multi-level graph semantics through disentangled representation learning in the GCL framework, which has been underexplored in previous research.

C. Reversible Networks

The design of reversible networks is inspired by NICE [47], which utilizes reversible transformations to map high-dimensional input features into mutually independent latent variable spaces without information loss. RevNet [22] firstly introduces a reversible architecture into residual networks to complete memory-efficient network training in computer vision tasks. Building upon this, a partially reversible U-Net [48] is proposed to construct deeper network architecture and utilize the entire medical image data instead of partitioning it into blocks for brain tumor segmentation task. RevCol [26] divides intermediate features into multiple groups, achieving inter-group lossless transmission while flexibly using nonlinear operations within each group to achieve different semantics feature extraction. Recently, Rev-ViT [25] and RevBiFPN [49] respectively extend reversible networks to Transformers and BiFPN [50]. Furthermore, reversible networks are also applied in natural language processing and other field. For example, reversible RNN [51] achieves lossless reversible computation by storing a minimal amount of information lost between hidden layer transmissions for recurrent network architecture. Particularly, RevSResNet [52] designs a spiking reversible block to construct the reversible spiking ResNet [53] and reversible spiking Transformer, aiming to potentially surpass artificial neural networks. Drawing inspiration from the successful advancements of reversible networks across various fields, we introduce reversible networks into graph contrastive learning, achieving semantic disentangled representation of nodes through lossless information transmission.

III. Notations and Preliminaries

In this section, we first give the notations of our paper. Then, we introduce some important concepts of the framework.

A. Notations

Let a tuple $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ represents an undirected graph, where $\mathcal{V} = \{v_1, \ldots, v_N\}$ is the node set with N nodes and $\mathcal{E}$ is the edge set with M edges. The node features are represented by the feature matrix $X = (x_1, \ldots, x_N)^{\mathrm{T}} \in \mathbb{R}^{N \times d'}$, where each node v is associated with a feature vector x_v and d' is the node feature dimension. We use adjacency matrix $A = (a_{uv}) \in \mathbb{R}^{N \times N}$ to describe the structure information of the graph, where $a_{uv} = 1$ if $(u, v) \in \mathcal{E}$, otherwise, $a_{uv} = 0$. And $D = \mathrm{diag}(d1, \ldots, d_N)$ is the degree matrix with a degree of each node $d_v = \sum_{u=1}^{N} a_{uv}$.

B. Graph Neural Networks

Graph neural networks (GNNs) have developed into a crucial backbone for learning graph representations. By leveraging information propagation from neighboring nodes to iteratively update representations, GNNs could embed the graph structure and corresponding feature matrix into node representations. Specifically, let $h_v^{(l)}$ represents the embedding of $v \in \mathcal{G}$ at l-th layer. For each node $v \in \mathcal{V}$, the neighborhood embeddings are firstly aggregated and then combined with v's previous layer embedding for the message passing. Formally, the propagation rule can be written as:

$$h_v^{(l)} = \mathcal{C}^{(l)}\left(h_v^{(l-1)}, \mathcal{A}^{(l)}\left(\left\{h_u^{(l-1)}\right\}_{u \in \mathcal{N}(v)}\right)\right), \tag{1}$$

where $\mathcal{N}(v)$ denotes the neighbors of node v, $\mathcal{A}^{(l)}$ and $\mathcal{C}^{(l)}$ denote the two basic functions that aggregate and combine embedding from the neighborhood. We can obtain the node-level

representation $H = \{\mathbf{h}_1^{(L)}, \ldots, \mathbf{h}_N^{(L)}\} \in \mathbb{R}^{N\times d}$ after L iterative propagations, where d is the representation dimension.

C. Graph Contrastive Learning

Contrastive learning (CL) has attracted considerable interest due to its remarkable ability to learn representations without manually labeled data. The core idea of CL is to pull similar objects (positive sample pairs) together while pushing dissimilar objects (negative sample pairs) apart in the embedding space. A typical CL model comprises three key modules, namely data augmentation, view encoder, and contrastive loss.

For graph-structured data, graph contrastive learning (GCL) [54], [55] adapts the CL techniques to learn node (graph) embeddings. The typical framework first generates graph views via two different data augmentation strategies (i.e., edge modification and feature masking). Then, a shared view encoder is utilized to map the augmented view pairs into embeddings in the representation space. Finally, the model can be optimized by minimizing a contrastive loss (i.e., InfoNCE [56] loss and NT-Xent [57] loss), formalized as:

$$\mathcal{L}_{con} = -\sum_{v=v_1}^{v_N} \log \frac{e^{\text{sim}(h_v,h_{v'})/\tau}}{e^{\text{sim}(h_v,h_{v'})/\tau} + \sum_{u\neq v} e^{\text{sim}(h_u,h_v)/\tau}}, \quad (2)$$

where $sim(\cdot,\cdot)$ denotes the similarity function (i.e., cosine similarity, dot product similarity, inverse of the distance) and τ denotes the temperature parameter that controls the sensitivity of penalties on hard negative samples.

IV. The Proposed Model

In this section, we first introduce the motivation and architecture of the proposed Rev-GCL, then we present the details of each component and the overall optimization.

A. Overview

Our framework is primarily designed to maintain different levels of semantic information while avoiding reliance on human prior knowledge for more robust GCL. Since disentangled representation learning has the ability to preserve multi-level embeddings from low-level to highly semantic representations, we propose a reversible column network to extract varying levels of semantic representations relevant to downstream tasks. Moreover, since manually designing augmentations for GCL cannot generate sufficiently diverse augmentations to filter out noise, we propose to perturb the neural architecture of the encoders as model augmentation tricks for GCL.

In particular, given the input graph, we propose a reversible column network with reversible connections between adjacent columns to serve as view encoders for extracting disentangled node features. Each column shares the same GNN structure, receives a copy of the input graph, and extracts both low-level and high-level information at different layers. Based on the proposed view encoders, we present two model augmentation strategies: (1) Random Propagation: we treat each GNN layer as a composition of transformation and propagation operators and vary the number of propagation operators at each epoch and (2) Asymmetric Column: we select different column outputs between the two sibling encoders as contrastive views. An overview of our Rev-GCL is depicted in Fig. 1 and we will delve into the details of each component.

B. Simple Graph Convolution

As one of the most common GNNs, graph convolution network (GCN) [58] performs neighborhood aggregation but always suffers from over-smoothing issues. A key factor that significantly compromises performance is the coupling of transformation and propagation operations of current graph convolution, which can be written as:

$$H = GCN(X) = g_L \circ s \circ g_{L-1} \circ s \circ \ldots g_1 \circ s(X), \quad (3)$$

where $\circ$ denotes the composition of transformation operator g and propagation operator s, g_l corresponds to the l-th transformation operator within GCN, H denotes the extracted node embeddings. The two operators on H can be defined as:

$$s(H;F) = FH, g(H;W) = \delta(HW), \quad (4)$$

where $F = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$ serves as the graph filter, which is the normalized adjacency matrix, $W \in \mathbb{R}^{d\times d}$ is the transformation parameters. Here $\tilde{A} = A + I$ is the adjacency matrix considering self-connections, $\tilde{D}$ is the corresponding degree matrix. Inspired by the recent deeper GNNs [59], [60], we disentangle the two operations, which can be written as:

$$H = SGC(X) = g \circ s^{[L]}(X), \quad (5)$$

where $s^{[L]}$ denotes the composition of L propagation operators. In practice, we define the propagation operator with the graph filter matrix $F = (1-\pi)I + \pi D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$, where $\pi \in (0,1)$ and I is the identity matrix.

C. Multi-Column Reversible Module

To uncover different levels of relevant information for downstream tasks, we propose a multi-column architecture with reversible connections to perform feature disentangling. The main body of the module is composed of M subnetworks denoted as columns, where each column shares an identical structure with K level of blocks and receives a copy of the input graph. Between each column, reversible connections are introduced to keep feature propagation between columns without information loss. The forward and inverse propagation functions can be defined as:

$$\begin{aligned} Forward: H_k^m &= R_k^m(H_{k-1}^m, H_{k+1}^{m-1}) + \gamma H_k^{m-1}, \\ Inverse: H_k^{m-1} &= \gamma^{-1}[H_k^m - R_k^m(H_{k-1}^m, H_{k+1}^{m-1})], \end{aligned} \quad (6)$$

where H_k^m denotes extracted node features of the k-th level in m-th column, R_k^m denotes a graph convolution function with reversible connections in k-th level, which is similar to the residual function in ResNets [53]. γ denotes the reversible operation (e.g., channel-wise scaling), with its inverse operation denoted as γ^{-1}. Given the node features within one column, the features in other columns can be computed recursively during forward

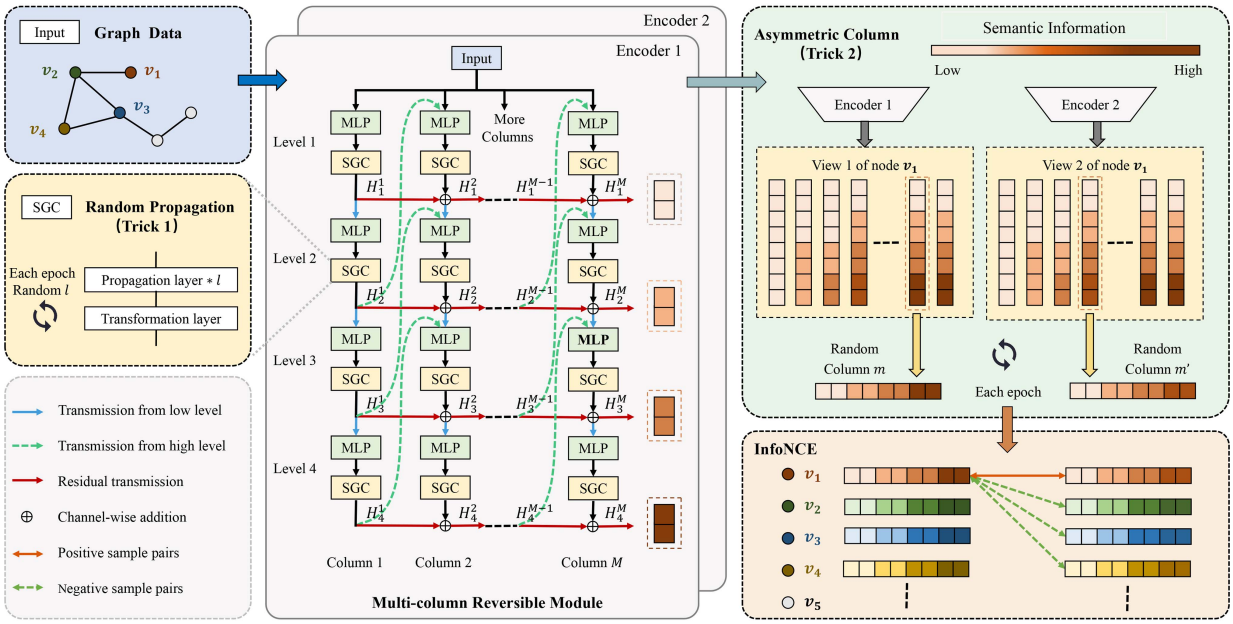

Fig. 1. The general framework of our proposed model. Given the graph data as input, we construct two multi-column sibling encoders, where each column of the encoder can be seen as SGCs with multiple levels and receives a copy of the input graph. Then, feature outputs from different levels are gradually disentangled into hierarchical semantics from low to high through the multi-column reversible module. Next, we introduce two model augmentation tricks for sibling encoders: random propagation and asymmetric column selection. Finally, InfoNCE loss is adopted as the contrastive loss for node-level contrast.

and backward propagation. Specifically, the k-th level takes the features from the lower level at the same column, namely, H_{k-1}^{m}, and the features from the upper level at the previous column, namely, H_{k+1}^{m-1} as input, where two features are fused together and passed to the corresponding graph convolution function. R_k^m can be defined as:

$$R_k^m(H_{k-1}^m, H_{k+1}^{m-1}) = SGC(MLP(H_{k-1}^m, H_{k+1}^{m-1})), \quad (7)$$

where $SGC(\cdot)$ and $MLP(\cdot)$ correspond to simple graph convolution and Multi-Layer Perceptron at the k-th level in m-th column. Then, the feature H_k^{m-1} is combined with the residual operation γ to get the final output.

Feature Disentangling: Note that the multiple columns in the module share an identical structure, and each level of the column can be seen as a different SGC block. The lowest level of each column preserves low-level features of the graph since it is closest to the input, while the highest level in the last column contains highly semantic information due to multiple layers of graph convolution. Therefore, reversible transformations between columns enable lossless information propagation. The extracted node representations H at different levels become gradually disentangled throughout this process. This property offers several potential advantages, such as increased flexibility for downstream tasks that depend on both high-level and low-level features.

D. Model Augmentation Tricks

For GCL, conventional data augmentation methods mainly rely on random data perturbations. However, selecting an appropriate method for generating views can be challenging, as it frequently involves tedious trial-and-error or relies on a limited set of pre-defined views, which results the loss of important information and restricts the applicability and potential of these methods [61], [62]. Instead of perturbing the graph inputs for GCL, we employ the multi-column reversible module as encoders and focus on manipulating the neural architecture of the module, which introduces two learnable and distinct view generators to obtain adaptive contrastive views. In this part, we present two model augmentation tricks and incorporate them into the multi-column reversible module.

Random Propagation: Given the input graph, we propose using two parameter-sharing encoders but with randomly varying numbers of propagation operators for two graph views. Formally, based on the SGC as each block of columns, we randomly select the numbers of propagation operators l_k, l_k' from a uniform distribution over $[1, L]$ for k-th levels of blocks of two views in every epoch. The graph convolution function with reversible connections R_k^m can be written as:

$$\begin{aligned} R_k^m(H_{k-1}^m, H_{k+1}^{m-1}) &= SGC(MLP(H_{k-1}^m, H_{k+1}^{m-1})) \\ &= g \circ s^{[l_k]}(MLP(H_{k-1}^m, H_{k+1}^{m-1})), l_k \sim \text{Uniform}[1, L]. \end{aligned} \quad (8)$$

Note that for the two graph view encoders, the selected l_k may vary among different levels of blocks, while within each graph view encoder, l_k remains consistent among columns. This method enables us to differentiate the two views using varying numbers of propagation operators, while the shared structure among columns and shared parameters between views ensure that the distance between them remains manageable.

Asymmetric Column: Traditional GCL methods always leverage typical data augmentation techniques (e.g., feature masking or edge modification) to generate two graph views. We argue that these methods fail to create sufficiently diverse augmentations

while preserving essential task-relevant information. Thus, we introduce an asymmetric view encoder in this paper. The main idea of the asymmetric column trick is to encode the input graph with a shared multi-column reversible module and maximize the consistency between different columns from two views. Specifically, for two graph view encoders, we randomly select m and m' from a uniform distribution over $[1, M]$ as two contrasting columns for the corresponding views. The extracted node embeddings from these two columns, namely H_k^m and $H_k^{m'}$, can then be considered as two graph view representations at different levels.

E. Optimization

By applying the two model augmentation tricks on the proposed reversible columns encoder, our framework maximizes the consistency of extracted node representations. Typically, for node v of the graph, its k-th level embedding generate from column m of one view $h_{v,k}^m$ is treated as anchor, where $H_k^m = \{h_{1,k}^m, \ldots, h_{N,k}^m\}$. We regard its k-th level embedding generated from column m' of another view $h_{v,k}^{m'}$ as a positive pair and the other embeddings in the two views as a negative pair. The InfoNCE [56] loss is adopted as the contrastive loss to enforce the consistency between positive pairs:

$$\mathcal{L} = -\sum_{v=v_1}^{v_N} \log \frac{e^{(-|h_v^m - h_v^{m'}|^2/2)}}{e^{(-|h_v^m - h_v^{m'}|^2/2)} + \sum_{u \neq v} e^{(-|h_u^m - h_v^m|^2/2)}}, \quad (9)$$

where $h_v^m = h_{v,1}^m \| \ldots \| h_{v,K}^m$ denotes the extracted multi-level representation of node v. Algorithm 1 illustrates the training procedure of our Rev-GCL.

F. Theoretical Analysis

In this part, we provide a theoretical analysis to demonstrate how Rev-GCL works to filter out high-frequency noises. We simplify the node features as one-hot identity matrix $X = I$ and non-linear transformation operation as a single-weight matrix W. And the multi-column reversible module of two views can be regarded as two encoders, namely, $R_k^m(X) = H_k^m = F^L XW$, $R_k^{m'}(X) = H_k^{m'} = F^{L'} XW$, where L and L' are the numbers of propagation operators of two views.

Here, we decompose the graph filter matrix as $F = U\Lambda U^{\mathrm{T}}$, where $U \in \mathbb{R}^{N \times N}$ comprises orthonormal eigenvectors and $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_N)$ is the diagonal matrix of eigenvalues with $\lambda_1 \geq \cdots \geq \lambda_N$. Notably, larger eigenvalues of the graph filter matrix F correspond to smaller eigenvalues of the graph Laplacian matrix, indicating more important information for downstream tasks [63]. Following the previous work [34], we concentrate on the numerator term and reformulate the contrastive loss objective in (9) as:

$$\begin{aligned} &\min_W \sum_{v=v_1}^{v_N} |h_v^m - h_v^{m'}|^2 \\ &= \min_W \mathrm{tr}((H_v^m - H_v^{m'})(H_v^m - H_v^{m'})^{\mathrm{T}}), \mathrm{s.t.} W^{\mathrm{T}} W = I, \end{aligned} \quad (10)$$

Algorithm 1: The Overall Training Process of Rev-GCL

Input: Graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ with adjacency matrix A and feature matrix X; Column Number M; Level Number K; Propagation Times L.
Output: Rev-GCL with learned parameters; Multi-scale node representation H_k^m for $k = 1 \ldots K$.

while *not convergence* **do**
  for $k \in [1, K]$ **do**
    /* Eq. 6-8 */
    Draw l_k, l_k' from Uniform$(1, L)$ for k-th level of blocks of two views ;
    Construct two multi-column Reversible modules with $\{l_k, l_k'\}$;
    Calculate the multi-level node representation H_k^m for $m = 1, \ldots, M$ of two views ;
  end
  Draw m, m' from Uniform$(1, M)$ to get m-th and m'-th column output h_v^m and $h_v^{m'}$ for $v \in \mathcal{V}$;
  /* Eq. 9 */
  Optimize framework via contrastive learning ;
end
return Learned parameters for Rev-GCL, H_k^m for $k = 1 \ldots K$;

where $\mathrm{tr}(\cdot)$ denotes the trace of a matrix. Note that here we add the constraints $W^{\mathrm{T}} W = I$ to prevent trivial solutions.

Theorem 1: The optimal parameter W of the multi-column reversible module for contrastive loss in (9) is:

$$W^* = U[c_1, \ldots, c_{d_O}], \quad (11)$$

where $U[c_1, \ldots, c_{d_O}]$ denotes the corresponding columns of U, $1 \leq c_1 < \ldots < c_{d_O} \leq N$ are the best column indexes c that could minimize $(\lambda_c^L - \lambda_c^{L'})^2$.

Proof: Since U is invertible, we alternatively optimize $P = U^{\mathrm{T}} W$ instead of W for the contrastive loss. Here we replace $H = U\Lambda^L U^{\mathrm{T}} W$, $H' = U\Lambda^{L'} U^{\mathrm{T}} W$ and (10) can be formulated as: $\min_W \mathrm{tr}(U(\Lambda^L - \Lambda^{L'})UW(U(\Lambda^L - \Lambda^{L'})UW)^{\mathrm{T}})$. Then the loss in (9) can be reformulated as:

$$\min_W \mathrm{tr}(ZZ^{\mathrm{T}}), \mathrm{s.t.} P^{\mathrm{T}} P = I, \quad (12)$$

where $Z = U(\Lambda^L - \Lambda^{L'})P$. We relax the constraint to $P_i^{\mathrm{T}} P_i = 1$ for $i = 1, \ldots, d_O$, where P_i denotes the i-th column of P. The Lagrangian function can be:

$$\mathcal{L}(P, \beta) = \mathrm{tr}(ZZ^{\mathrm{T}}) - \sum_{i=1}^{d_O} \beta_i (P_i^{\mathrm{T}} P_i - 1), \quad (13)$$

β_i is the Lagrangian multiplier. With Karush $-$ Kuhn$-$Tucker (KKT) conditions, the optimal parameter can be:

$$\frac{\partial \mathcal{L}(P, \beta)}{\partial P} = 0 \Rightarrow (\Lambda^L - \Lambda^{L'})^2 P = P\mathrm{diag}(\beta). \quad (14)$$

Here P denote eigenvectors of matrix $(\Lambda^L - \Lambda^{L'})^2$. Note that the constraints $P^{\mathrm{T}} P = I$, the optimal P^* is the eigenvectors

with minimal eigenvalues, which select the directions where the frequency differences between two views are smaller.

$$P^* = I[c_1, \ldots, c_{d_O}], \tag{15}$$

where $1 \leq c_1 < \ldots < c_{d_O} \leq N$ are the best columns of identity matrix I that minimize $(\lambda_c^L - \lambda_c^{L'})^2$. □

Remark: As shown in the previous work [34], [64], the popular datasets such as Cora and Citeseer have many eigenvalues close to 1. Hence, $U[c_1, \ldots, c_{d_O}]$ is likely to select the first d_O columns of U in practice and the extracted node representation can be represented as:

$$H_k^m = U\Lambda^L U^{\mathrm{T}} W = U\mathrm{diag}(\lambda_1^L, \ldots, \lambda_{d_O}^L, 0, \ldots, 0), \tag{16}$$

which could effectively eliminate high-frequency disturbances.

G. Complexity Analysis

The computational consumption of our Rev-GCL primarily consists of the three components: (i) SGC-based level unit; (ii) reversible connection module; (iii) disentangled consistency contrastive loss. Considering the graph that contains $|\mathcal{V}|$ nodes and $|\mathcal{E}|$ edges, the dimension of node representation is d, the number of level within multi-column reversible module is K. Noted that, for computational convenience, we take the average number of columns and propagation operators at each level as M and L within multi-column reversible module, instead of selecting them randomly. For (i), the time complexity is $O(|\mathcal{E}|Ld + |\mathcal{V}|d^2)$ with L propagation operators. For (ii), the time complexity is $O(2|\mathcal{V}|d^2 + d)$ with MLP from adjacent hierarchical levels and residual connection.The encoder constructed by the multi-column reversible module is $O((|\mathcal{E}|Ld + 2|\mathcal{V}|d^2 + d)MK)$. For (iii), we perform consistency regularization for node disentangled representations with the time complexity of $O(|\mathcal{V}|d)$. To sum up, the overall complexity of our Rev-GCL is $O((|\mathcal{E}|Ld + 2|\mathcal{V}|d^2 + d)MK + |\mathcal{V}|d)$, which exhibits a linear relationship with the number of nodes and edges in graph G and aligns with other self-supervised node classification methods (SSG [11] and MA-GCL [34]).

Moreover, our Rev-GCL requires K level SGC blocks and M column reversible connections in contrastive learning training phases. Due to employing the reversible connections, the actual model space complexity is $O(|\mathcal{E}| + |\mathcal{V}|d + K(|\mathcal{V}|d + 2M(d^2)) + d^2)$, which primarily influenced by the number of columns in the multi-column reversible module.

V. Experiment

In this section, we conduct comprehensive experiments on eight widely used benchmarks to evaluate the performance of our Rev-GCL. The following research questions are explored:

- *RQ1:* How does our proposed Rev-GCL perform compared with other baselines for downstream tasks, i.e., node classification, clustering and link prediction?
- *RQ2:* How does each key component of our proposed Rev-GCL affect the model's effectiveness?
- *RQ3:* How do different hyper-parameters in Rev-GCL alter the model's behavior?

TABLE I
Statistics of Datasets

Dataset	#Nodes	#Edges	#Features	#Classes
Cora	2,708	10,556	1,433	7
CiteSeer	3,327	9,104	3,703	6
PubMed	19,717	88,648	500	3
Coauthor-CS	18,333	163,788	6,805	15
Amazon-P (undirected)	7,650	238,162	745	8
Amazon-C (undirected)	13,752	491,722	767	10
Amazon-P (directed)	7,650	143,663	745	8
Amazon-C (directed)	13,752	287,209	767	10

- *RQ4:* Can we clearly illustrate the impact of our reversible column-based model augmentation trick on node representations?
- *RQ5:* How about computational efficiency of our model?

A. Experimental Setup

Datasets: Building upon previous work [34], we adopt eight real-world graph benchmark datasets that cover both undirected and directed graphs to verify the effectiveness of our method. Specifically, we use undirected graph datasets including Cora [65], CiteSeer [65], PubMed [65], Amazon-Photo [66], Amazon-Computers [66] and Coauthor-CS [66] from the Python package PyTorch Geometric. Node classification, clustering and link prediction tasks are performed on these undirected graphs. Additionally, we validate our method on directed graph datasets[1] for the node classification task, including Amazon-Photo [67] and Amazon-Computers [67]. Table I presents the details of these datasets.

Baselines: We compare our model against various approaches for three downstream tasks, namely, node classification, clustering and link prediction. Concretely, the node classification baselines can be categorized into two types. The first type adopts baselines trained with labels, such as GCN [68], GAT [69], and InfoGCL [30]. The second type consists of multiple self-supervised learning methods without training labels, including DGI [9], GRACE [10], MVGRL [12], BGRL [70], GCA [28], COLES [32], CCA-SSG [11], ARIEL [18] SimGRACE [33] and MA-GCL [34]. And these self-supervised learning methods are also chosen to be compared on node clustering tasks. Additionally, we select two representative methods CCA-SSG [11] and MA-GCL [34] for comparison on the link prediction task to demonstrate our model's applicability in the recommendation domain.

Evaluation Metrics: We adopt Accuracy, Normalized Mutual Information (NMI) and Normalized Discounted Cumulative Gain (NDCG) as evaluation metrics for node classification, node clustering and link prediction, respectively, following the previous work [34], [71]. For each dataset, we conduct 5 trials in different random seeds and report the mean and the standard deviation of Accuracy, NMI and NDCG. For node classification and clustering tasks, note that we compare model performance on the public splits of Cora, CiteSeer and PubMed datasets [68].

[1] https://github.com/EdisonLeeeee/GraphData

TABLE II
PERFORMANCE (%) OF NODE CLASSIFICATION. HERE WE USE PUBLIC SPLITS ON CORA, CITESEER AND PUBMED

Datasets	Cora	CiteSeer	PubMed	Coauthor-CS	Amazon-C	Amazon-P	Avg. Acc.	Avg. Rank
GCN	82.5 ± 0.4	71.2 ± 0.3	79.2 ± 0.3	93.03 ± 0.3	86.51 ± 0.5	92.42 ± 0.2		
GAT	83.0 ± 0.7	72.5 ± 0.7	79.0 ± 0.3	92.31 ± 0.2	86.93 ± 0.3	92.56 ± 0.4	-	-
InfoGCL	83.5 ± 0.3	73.5 ± 0.4	79.1 ± 0.2	-	-	-		
DGI	82.3 ± 0.6	71.8 ± 0.7	76.8 ± 0.3	92.15 ± 0.6	83.95 ± 0.5	91.61 ± 0.2	83.10	9.5
GRACE	81.7 ± 0.4	71.5 ± 0.5	80.7 ± 0.4	92.93 ± 0.0	87.46 ± 0.2	92.15 ± 0.2	84.44	7.5
MVGRL	83.4 ± 0.3	73.0 ± 0.3	80.1 ± 0.6	92.11 ± 0.1	87.52 ± 0.1	91.74 ± 0.0	84.63	7.2
BGRL	81.7 ± 0.5	72.1 ± 0.5	80.2 ± 0.4	93.01 ± 0.2	88.23 ± 0.3	92.57 ± 0.3	84.63	6.2
GCA	83.4 ± 0.3	72.3 ± 0.1	80.2 ± 0.4	93.10 ± 0.0	87.85 ± 0.3	92.53 ± 0.2	84.89	5.2
SimGRACE	77.3 ± 0.1	71.4 ± 0.1	78.3 ± 0.3	93.45 ± 0.4	86.04 ± 0.2	91.39 ± 0.4	82.98	9.5
COLES	81.2 ± 0.4	71.5 ± 0.2	80.4 ± 0.7	92.65 ± 0.1	79.64 ± 0.0	89.00 ± 0.5	82.40	10.0
ARIEL	82.5 ± 0.1	72.2 ± 0.2	80.5 ± 0.3	93.35 ± 0.0	88.27 ± 0.2	91.43 ± 0.2	84.71	5.8
CCA-SSG	83.9 ± 0.4	73.1 ± 0.3	81.3 ± 0.4	93.37 ± 0.2	88.42 ± 0.3	92.44 ± 0.1	85.42	3.3
MA-GCL	83.3 ± 0.4	73.6 ± 0.1	83.5 ± 0.4	94.19 ± 0.1	88.83 ± 0.3	93.80 ± 0.1	86.20	2.5
Base Model	81.1 ± 0.4	71.4 ± 0.1	79.1 ± 0.4	92.86 ± 0.3	87.65 ± 0.2	91.19 ± 0.3	83.88	9.8
Rev-GCL	**85.0 ± 0.1**	**74.5 ± 0.2**	**83.8 ± 0.2**	**94.34 ± 0.2**	**89.98 ± 0.2**	**94.08 ± 0.3**	**86.95**	**1.0**

For other datasets, We randomly split the dataset into training, validation, and testing sets in a ratio of 8:1:1 as previous work [11]. For link prediction task, we randomly split the graph edges of Amazon co-purchase dataset (Amazon-Photo [66] and Amazon-Computers [66]) into training, validation, and testing sets in a ratio of 8:1:1. For the test set, we follow the strategy [71], which randomly selects 100 items that the user has not interacted with and ranks the test item among them. The ranked list is truncated at 10, with NDCG@10 prioritizing higher-ranked hits by assigning greater scores to them.

Implementation: We implement our Rev-GCL model on a simple base model with reversible column disentangled augmentation tricks. The base model can be viewed as a typical graph contrastive learning model with perturbations such as edge dropping and feature masking, which incorporates SGC with the graph filter $F = \frac{1}{2}I + \frac{1}{2}D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$. For all datasets, we set the number of columns $M = 8$ with levels $K = 2$ to implement our basic reversible architecture, where each level of column denotes an SGC with the hidden size $d = 512$. We concatenate the 2 levels of the random column output as the node feature representation for contrastive learning. In our model augmentation tricks, the number of propagation operators of SGC, i.e., l, l', are respectively from the random range [1, L] for $L = 4$. During the evaluation phase, we set l, l' of two levels of SGC $l_1 = l_2 = l'_1 = l'_2 = 1$ for Coauthor-CS, and 2 for other datasets. For baseline methods, we generate the evaluation results according to their original settings. The source code of our Rev-GCL is available at https://github.com/BBDing-DYT/Rev-GCL.

B. Performance Comparison (RQ1)

We conduct extensive experiments on node classification, clustering and link prediction tasks on undirected graph. In addition, we also conduct node classification tasks on directed graphs to validate the adaptability of our method across different types of graphs. Tables II and III show the comparison results of the node classification and clustering tasks. We **bold** the best method and underline the runner-up method. Fig. 2 shows the performance comparison of undirected graph link prediction and directed graph node classification. From the result, we can get the following observations.

TABLE III
PERFORMANCE (%) OF NODE CLUSTERING

Datasets	Cora	PubMed	Amazon-P	Coauthor-CS
raw feature	14.4 ± 0.5	17.8 ± 0.2	25.7 ± 0.4	56.3 ± 0.2
GCA	47.4 ± 0.2	22.4 ± 0.2	58.4 ± 0.3	61.2 ± 0.2
COLES	47.0 ± 0.3	26.2 ± 0.2	54.6 ± 0.3	65.9 ± 0.1
ARIEL	48.7 ± 0.1	27.6 ± 0.2	56.9 ± 0.4	65.1 ± 0.2
CCA-SSG	51.7 ± 0.4	25.4 ± 0.2	59.1 ± 0.2	67.7 ± 0.3
MA-GCL	50.9 ± 0.2	29.7 ± 0.3	60.2 ± 0.2	69.6 ± 0.2
Rev-GCL	**60.6 ± 0.1**	**34.7 ± 0.2**	**65.6 ± 0.2**	**75.6 ± 0.2**

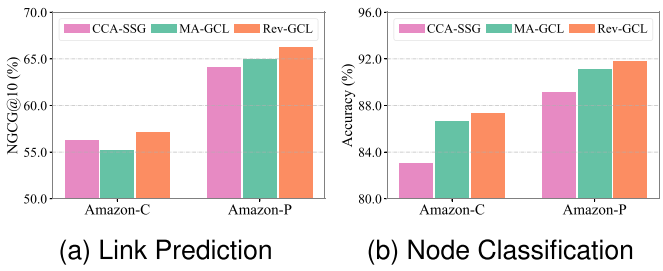

Fig. 2. Performance (%) of (a) link prediction on the undirected graph and (b) node classification on the directed graph.

- Compared to baselines trained with labels (i.e., GCN, GAT, InfoGCL), our Rev-GCL achieves better performance. For example, Rev-GCL generally obtains 3.03% improvement against InfoGCL on Cora, CiteSeer and PubMed. This indicates that introducing the reversible connections can make node representation preserve complete original information, including information relevant to downstream tasks.
- Compared with other baselines, our Rev-GCL consistently outperforms them across all six datasets. Specifically, our Rev-GCL achieves 2.04% and 1.29% increment over the runner-up baseline MA-GCL on Cora and Amazon-C. This demonstrates that model augmentation tricks effectively filter out extraneous noise, while feature disentanglement

TABLE IV
ABLATION STUDIES OF REV-GCL (%). OUR FULL MODEL ACHIEVES THE BEST PERFORMANCE CONSISTENTLY

Asymmetric	Random	Reversible	Cora	CiteSeer	PubMed	Coauthor-CS	Amazon-C	Amazon-P
✓	-	-	83.8 ± 0.2	72.8 ± 0.3	82.3 ± 0.2	93.03 ± 0.4	87.01 ± 0.2	93.19 ± 0.2
-	✓	-	83.7 ± 0.1	73.1 ± 0.1	81.9 ± 0.3	93.54 ± 0.2	88.86 ± 0.2	93.12 ± 0.1
-	-	✓	84.0 ± 0.2	73.0 ± 0.1	82.8 ± 0.2	93.71 ± 0.1	88.90 ± 0.2	93.01 ± 0.1
✓	✓	-	84.4 ± 0.1	73.3 ± 0.2	82.9 ± 0.2	93.65 ± 0.2	89.07 ± 0.0	93.71 ± 0.2
✓	-	✓	84.6 ± 0.1	73.5 ± 0.2	83.2 ± 0.5	94.38 ± 0.2	89.11 ± 0.1	93.51 ± 0.3
-	✓	✓	84.5 ± 0.2	73.6 ± 0.3	83.3 ± 0.2	93.84 ± 0.1	89.48 ± 0.2	94.00 ± 0.2
✓	✓	✓	85.0 ± 0.1	74.5 ± 0.2	83.8 ± 0.2	94.34 ± 0.2	89.98 ± 0.2	94.08 ± 0.3

enables the model to learn semantic information from different levels, thereby enhancing the downstream node classification performance.

- Our Rev-GCL also achieves the best performance on the node clustering task in comparison to other baselines across all datasets. Specifically, Rev-GCL obtains 17.21% and 16.84% improvement against the runner-up baseline on Cora and PubMed. The multi-level hierarchical disentanglement helps Rev-GCL extract node representations without information loss, thereby enhancing the constraints of node similarity evaluation and clarifying the boundaries of node clusters.
- In the link prediction task, our Rev-GCL achieves the best performance on the Amazon co-purchase dataset. Specifically, Rev-GCL obtains 1.51% and 2.03% improvement against the runner-up baseline methods on Amazon-C and Amazon-P. This indicates that hierarchical disentanglement effectively captures multi-level semantic similarity between nodes, enabling the model to distinguish different types of products more precisely and ultimately enhancing recommendation performance.
- Moreover, our Rev-GCL also achieves performance improvements in the node classification task on the directed graphs of the Amazon co-purchase dataset. Specifically, Rev-GCL obtains 0.75% and 0.67% improvement against the runner-up baseline on Amazon-C and Amazon-P. Disentangled semantic information from different levels effectively alleviates the imbalance in information propagation on directed graphs, ensuring that the dependencies between nodes are modeled more efficiently.

C. Ablation Study (RQ2)

We attribute the improvements in our Rev-GCL to sibling encoders with three key components: 1) **Asymmetric**: the number of columns being different in different encoders; 2) **Random**: SGC units with random numbers of propagation layers during each training epoch; 3) **Reversible**: reversible connections between adjacent columns within encoders. To evaluate the effectiveness of each component, we conduct ablation studies on various combinations of them. The experimental results on node classification are shown in Table IV, which brings the following conclusions:

- Each component contributes significantly to the model's performance, especially the reversible component. The performance notably drops when the reversible connections are removed. This indicates that reversible connections preserve the complete semantics through lossless information transfer, thereby greatly enhancing the model's capability of feature extraction.
- Removing any two components leads to a greater performance decrease, particularly when reversible connections are removed alongside either asymmetric or random encoders. This is likely because both asymmetric and random tricks contribute to model augmentation, enabling the extraction of more universal node representations while preserving rich feature information.
- Our Rev-GCL, which incorporates all components (the last row), exhibits the best performance over all datasets. This observation demonstrates the effectiveness of jointly utilizing reversible connections and model augmentation tricks on contrastive learning.

D. Parameter Sensitivity (RQ3)

We also examine the sensitivity of the proposed Rev-GCL to various hyper-parameters. Specifically, we explore the performance of our proposed Rev-GCL with varying numbers of columns and levels within the asymmetric encoder, as well as the different combinations of the propagation operator number for the sibling encoders.

1) Effect of the Number of Columns: To analyze whether Rev-GCL can benefit from the multi-column reversible module, we evaluated the model's performance with varying numbers of columns M in the range of $\{1, 2, 4, 6, 8, 10, 12\}$. As shown in Fig. 3, we investigate the influence of different columns within encoders across four datasets. From the results, we can discover that:

- when $M = 1$, the encoder can be viewed as simply concatenating the feature extraction results of each layer and performing contrastive learning. The poor performance suggests that the multi-column reversible module facilitates information interaction between different levels while ensuring lossless information transmission to gradually disentangle the learned feature.
- Generally, when $M = 6$ or 8, the model achieves best performance. However, with M continue to increase, there is a significant decrease in accuracy performance. Such a trend may be caused by overly large distribution ranges in the random column selection of the column asymmetric trick, leading to difficulties in model convergence.

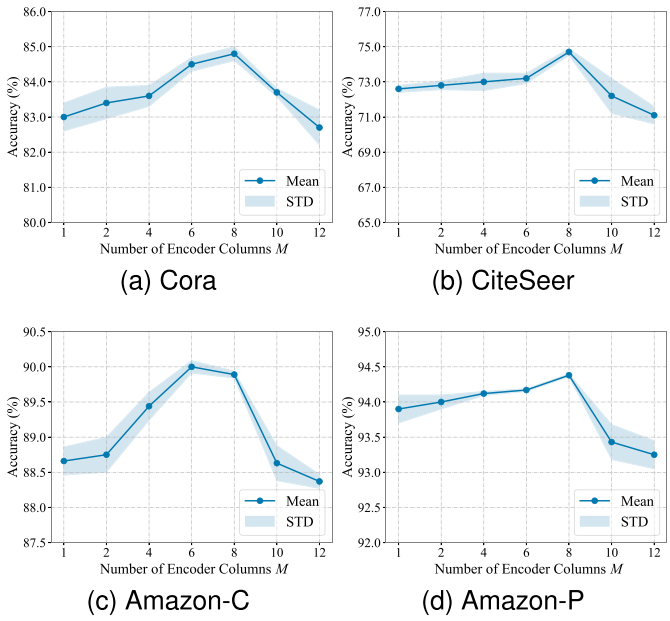

Fig. 3. Performance (%) comparison on node classification w.r.t. different numbers of columns within encoder.

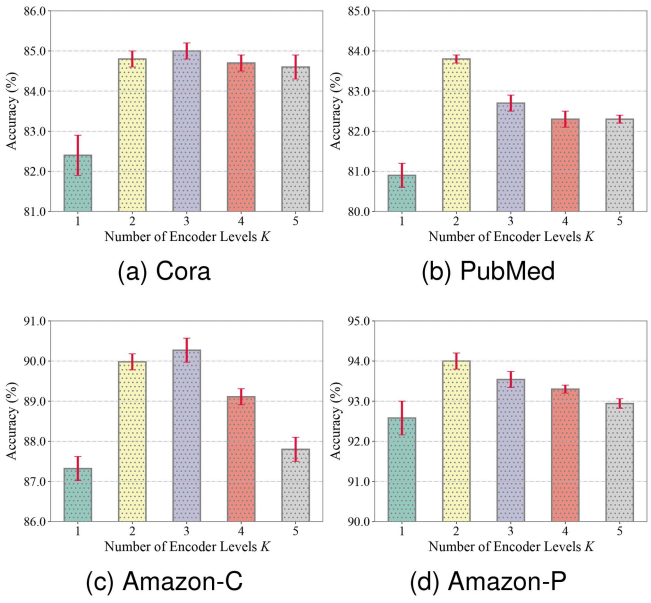

Fig. 4. Performance (%) comparison on node classification w.r.t. different numbers of levels within each column.

2) Effect of the Number of Levels: By facilitating information transfer between multiple columns within the encoder, we further explore the impact of different numbers of levels in each column on model performance. Fig. 4 shows the performance results of our Rev-GCL with varying numbers of levels K in a range of $\{1, 2, 3, 4, 5\}$ across four datasets. From the result, we draw the following conclusions:

- When $K = 1$, the encoder resembles a residual network structure that integrates information from the graph input with entangled representation as output. Due to the excessive compression of node features, the model can only learn semantic information relevant to the pretext tasks through contrastive learning, resulting in suboptimal performance in downstream tasks.
- With increasing K, node features begin to be hierarchically disentangled. When $K = 2$ or 3, the model generally achieves optimal performance, illustrating that multiple levels within the column can hierarchically separate node representations at different levels and keep the complete semantics for the downstream tasks.
- However, too many disentangled levels (i.e., $K > 3$) may weaken the model's performance, especially in PubMed and Amazon-C. This is likely due to an excessive number of layers in the feature extraction process, which causes over-smoothing and noise interference for each column during model training.

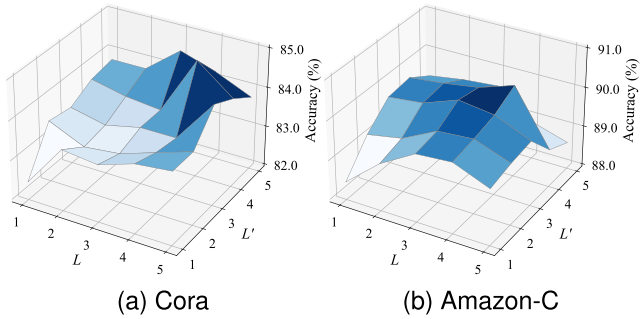

Fig. 5. Performance (%) comparison on node classification w.r.t. different combinations of the propagation operator number for the sibling encoders.

3) Effect of the Number of Propagation Operators: To validate the effectiveness of the propagation operator, we further investigate the impact of different combinations for the two sibling encoders. Specifically, we set different propagation operator numbers, which are randomly drawn from two distributions (i.e., $[0, L]$ and $[0, L']$) respectively in each level k. We set L and L' in a range of $\{0, 1, 2, 3, 4\}$, with the combination results shown in Fig. 5. We find that:

- The combination of excessively small distribution range (i.e., $L < 3$ while $L' < 3$) leads to few numbers of propagation operators with poor performance. On the one hand, few propagation operators will restrict graph nodes to learning only from their local neighboring nodes, resulting in poor node correlations (i.e., when $L = L' = 0$, the performance significantly decreases because only node self-features are considered). On the other hand, a narrow range of numerical distributions increases the likelihood that the two encoders share similar structures, consequently introducing noise into the model.
- With L and L' increase, a broader sampled distribution range results in a more distinct encoder structure between two views, reducing task-irrelevant noise and significantly boosting the model's expressive power. However, the performance begins to decline when $K = 5$ or $K' = 5$. This may be due to over-smoothing issues. Meanwhile, a significant disparity between sampled propagation operator numbers can also affect the consistency agreement between the two encoder outputs.

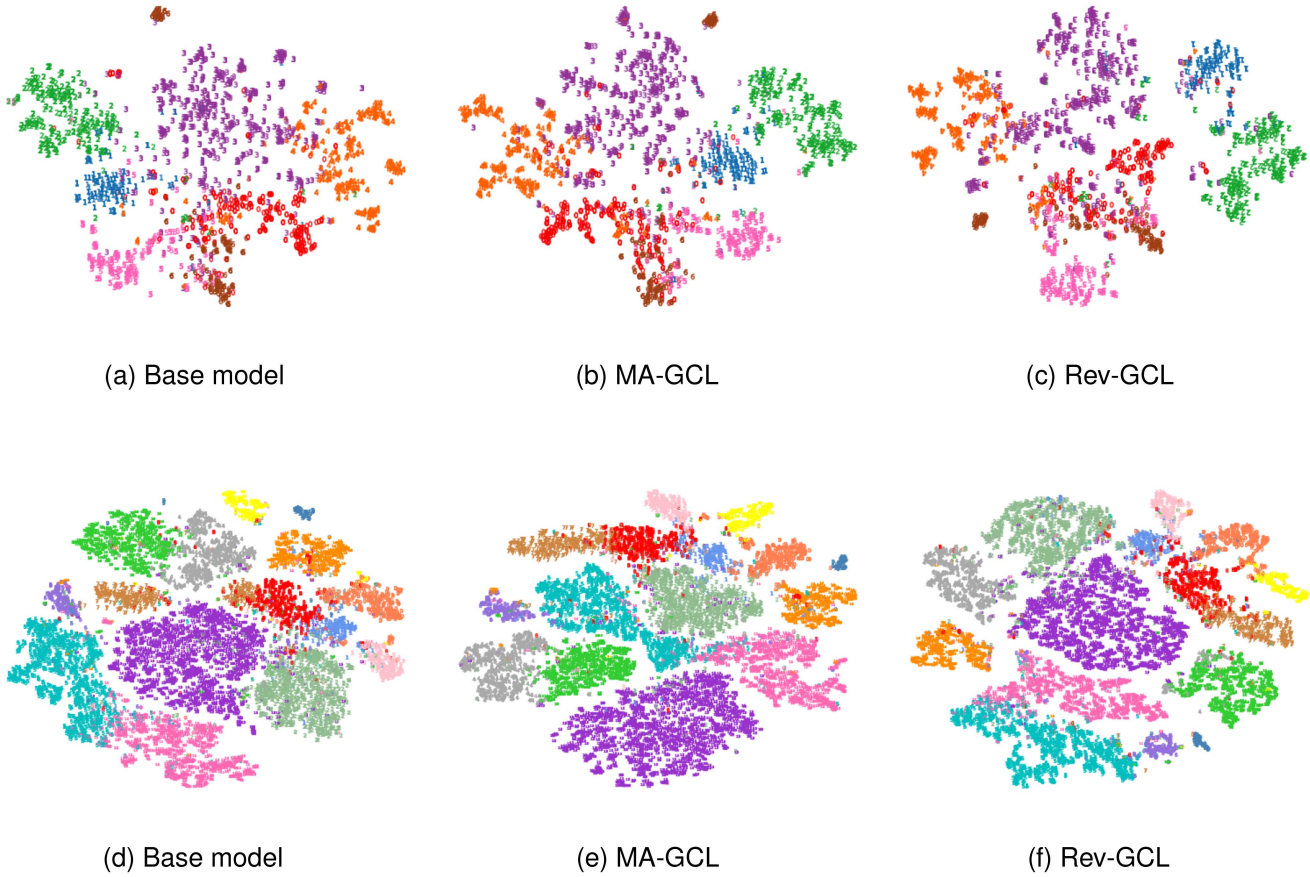

Fig. 6. 2D t-SNE visualization of Rev-GCL compared with the base model and MA-GCL on two benchmark datasets. The first row and second row correspond to Cora and Coauthor-CS.

E. Visualization (RQ4)

We conduct two visualizations to further investigate how our Rev-GCL facilitates node representation, including inter-node feature correlation and intra-node feature correlation.

1) Inter-Node Feature Correlation Analysis: To explore the influence of our Rev-GCL on node representations, we plot the distribution of extracted feature representation learned on Cora and Coauthor-CS. Through the t-SNE algorithm, we visualize the reduced-dimensional node features and compare them with those from the base model and MA-GCL. From Fig. 6(a) and (d), it is evident that the boundaries between the node features of different classes learned by the base model are blurry, with significant overlap among nodes from different classes and scattered distributions within the same class. In Fig. 6(b) and (e), node features learned by MA-GCL exhibit clear clustering. Moreover, our Rev-GCL not only effectively distinguishes nodes from different classes but also aggregates nodes within the same class more tightly. This indicates that our Rev-GCL could learn more discriminative and semantically rich node feature representations.

2) Intra-Node Feature Correlation Analysis: To further demonstrate the effectiveness of hierarchical disentanglement, we visualize the absolute correlation values between the elements of the extracted node representations on Cora. As shown in Fig. 7(a), node feature representations from MA-GCL exhibit high entanglement. Meanwhile, as depicted in Fig. 7(b), node representations of our Rev-GCL exhibit block-wise correlation patterns, with four red dashed diagonal blocks corresponding to the four levels of encoders used in our experiments. From the results, we observe that node features within each level are correlated. Additionally, node features across adjacent levels show weaker correlations due to reversible multi-column modules facilitating information transfer between levels. As the levels stack, higher-level node features exhibit higher correlations with deeper colors, indicating stronger semantic connections and

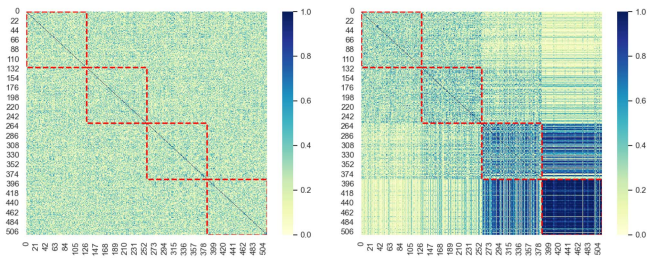

Fig. 7. Intra-correlation visualization of MA-GCL and Rev-GCL, where representations are obtained from Cora.

TABLE V
COMPARISON OF PARAMETER SPACE AND TRAINING TIME FOR MODELS

Model	Params (M)	Training Time (s)
CCA-SSG	0.66	1.33
MA-GCL	0.39	1.08
Rev-GCL	2.23	1.58

gradually disentangled representations. This preserves complete multi-scale information for downstream tasks, demonstrating that multi-column reversible modules achieve hierarchical disentanglement and enhance the model's feature representation capability.

F. Space and Runtime Analysis (RQ5)

We further evaluate the parameter space and the running time of the selection tricks for each epoch of graph contrastive learning to further analyze the efficiency of our proposed reversible column disentangled augmentation tricks. Specifically, we compare our proposed Rev-GCL with two latest self-supervised methods, including CCA-SSG [11] and MA-GCL [34] on the Amazon-Computer dataset. For our model, we select the multi-column reversible module with 2 levels and 2 columns as the encoder. As shown in Table V, due to the multi-column disentangled feature in Rev-GCL, the number of parameters increases within the method. However, we still observe that the running time of Rev-GCL is comparable to the latest self-supervised methods due to the reversible connections, which makes it more suitable for practical applications.

VI. CONCLUSION

In this paper, we propose a novel framework termed Rev-GCL for graph contrastive learning, which aims at learning multiple levels of semantic representations in a disentangled manner for downstream tasks (e.g., node classification and clustering tasks). Specifically, we propose a reversible multi-column module as the graph encoder to obtain hierarchical disentangled node representations while preserving the complete input information. Based on the proposed encoder, we leverage two model augmentation tricks, namely random propagation and asymmetric column, to ensure a distinct structure between the sibling encoders and obtain adaptive contrastive views. In this way, our Rev-GCL can filter out high-frequency noise while maintaining node representations from low-level to high-level semantics and gradually disentangle them. We conduct extensive experiments on eight widely-used graph network datasets to validate the effectiveness of our approach.

We hope our reversible column disentangled augmentation tricks could provide a new insight for GCL. In future work, we will attempt to apply this graph network architecture directly to other applications, such as drug discovery and recommender systems. In addition to graph representation learning, we are also interested in replacing the level units of the multi-column reversible module with other neural networks (e.g., TPNet [72]) to handle more complex non-regular grid data in other domains, such as computer-aided geometric design [73] and computer-aided design/engineering [74].

REFERENCES

[1] W. Ju et al., "Hypergraph-enhanced dual semi-supervised graph classification," in *Proc. Int. Conf. Mach. Learn.*, 2024, pp. 22594–22604.
[2] K. M. Borgwardt et al., "Protein function prediction via graph kernels," *Bioinformatics*, vol. 21, no. suppl_1, pp. i47–i56, 2005.
[3] Y. Wang et al., "Disencite: Graph-based disentangled representation learning for context-specific citation generation," in *Proc. AAAI Conf. Artif. Intell.*, 2022, pp. 11449–11458.
[4] J. Luo et al., "Rank and align: Towards effective source-free graph domain adaptation," in *Proc. 33rd Int. Joint Conf. Artif. Intell.*, 2024, pp. 4706–4714.
[5] Y. Wang et al., "Disenhan: Disentangled heterogeneous graph attention network for recommendation," in *Proc. Int. Conf. Inf. Knowl. Manage.*, 2020, pp. 1605–1614.
[6] W. Ju et al., "Kernel-based substructure exploration for next POI recommendation," in *Proc. IEEE Int. Conf. Data Mining*, 2022, pp. 221–230.
[7] W. Ju et al., "A survey of data-efficient graph learning," in *Proc. 33rd Int. Joint Conf. Artif. Intell.*, 2024, pp. 8104–8113.
[8] W. Ju et al., "Towards graph contrastive learning: A survey and beyond," 2024, *arXiv:2405.11868*.
[9] P. Veličković et al., "Deep graph infomax," in *Proc. Int. Conf. Learn. Representations*, 2019.
[10] Y. Zhu et al., "Deep graph contrastive representation learning," 2020, *arXiv:2006.04131*.
[11] H. Zhang, Q. Wu, J. Yan, D. Wipf, and P. S. Yu, "From canonical correlation analysis to self-supervised graph neural networks," in *Proc. Conf. Neural Inf. Process. Syst.*, 2021, pp. 76–89.
[12] K. Hassani and A. H. Khasahmadi, "Contrastive multi-view representation learning on graphs," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 4116–4126.
[13] R. I. Kondor and J. Lafferty, "Diffusion kernels on graphs and other discrete structures," in *Proc. Int. Conf. Mach. Learn.*, 2002, pp. 315–322.
[14] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bring order to the web," Stanford University, Stanford, CA, USA, Tech. Rep. 1999-66, 1998.
[15] Y. You et al., "Graph contrastive learning with augmentations," in *Proc. Conf. Neural Inf. Process. Syst.*, 2020, pp. 5812–5823.
[16] Y. You, T. Chen, Y. Shen, and Z. Wang, "Graph contrastive learning automated," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 12121–12 132.
[17] S. Suresh, P. Li, C. Hao, and J. Neville, "Adversarial graph augmentation to improve graph contrastive learning," in *Proc. Conf. Neural Inf. Process. Syst.*, 2021, pp. 15920–15 933.
[18] S. Feng, B. Jing, Y. Zhu, and H. Tong, "Adversarial graph contrastive learning with information regularization," in *Proc. Web Conf.*, 2022, pp. 1362–1371.
[19] N. Tishby and N. Zaslavsky, "Deep learning and the information bottleneck principle," in *Proc. IEEE Inf. Theory Workshop*, 2015, pp. 1–5.
[20] J. Ma, P. Cui, K. Kuang, X. Wang, and W. Zhu, "Disentangled graph convolutional networks," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 4212–4221.
[21] J. Guo, K. Huang, X. Yi, and R. Zhang, "Learning disentangled graph convolutional networks locally and globally," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 3, pp. 3640–3651, Mar. 2024.
[22] A. N. Gomez, M. Ren, R. Urtasun, and R. B. Grosse, "The reversible residual network: Backpropagation without storing activations," in *Proc. Conf. Neural Inf. Process. Syst.*, 2017, pp. 2214–2224.
[23] B. Chang et al., "Reversible architectures for arbitrarily deep residual neural networks," in *Proc. AAAI Conf. Artif. Intell.*, 2018, pp. 2811–2818.
[24] J.-H. Jacobsen, A. Smeulders, and E. Oyallon, "i-RevNet: Deep invertible networks," in *Proc. Int. Conf. Learn. Representations*, 2018.
[25] K. Mangalam et al., "Reversible vision transformers," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 10830–10 840.
[26] Y. Cai et al., "Reversible column networks," in *Proc. Int. Conf. Learn. Representations*, 2023.
[27] P. Velickovic et al., "Deep graph infomax," in *Proc. Int. Conf. Learn. Representations*, 2019.
[28] Y. Zhu et al., "Graph contrastive learning with adaptive augmentation," in *Proc. Web Conf.*, 2021, pp. 2069–2080.

[29] S. Thakoor et al., "Large-scale representation learning on graphs via bootstrapping," in *Proc. Int. Conf. Learn. Representations*, 2022.
[30] D. Xu, W. Cheng, D. Luo, H. Chen, and X. Zhang, "InfoGCL: Information-aware graph contrastive learning," in *Proc. Conf. Neural Inf. Process. Syst.*, 2021, pp. 30414–30 425.
[31] S. Li et al., "Let invariant rationale discovery inspire graph contrastive learning," in *Proc. Int. Conf. Mach. Learn.*, 2022, pp. 13052–130 65.
[32] H. Zhu, K. Sun, and P. Koniusz, "Contrastive Laplacian eigenmaps," in *Proc. Conf. Neural Inf. Process. Syst.*, 2021, pp. 5682–5695.
[33] J. Xia, L. Wu, J. Chen, B. Hu, and S. Z. Li, "Simgrace: A simple framework for graph contrastive learning without data augmentation," in *Proc. Web Conf.*, 2022, pp. 1070–1079.
[34] X. Gong, C. Yang, and C. Shi, "Ma-gcl: Model augmentation tricks for graph contrastive learning," in *Proc. AAAI Conf. Artif. Intell.*, 2023, pp. 4284–4292.
[35] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1798–1828, Aug. 2013.
[36] I. Higgins et al., "beta-VAE: Learning basic visual concepts with a constrained variational framework," in *Proc. Int. Conf. Learn. Representations*, 2016.
[37] X. Chen et al., "InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets," in *Proc. Conf. Neural Inf. Process. Syst.*, 2016, pp. 2172–2180.
[38] J. Ma, C. Zhou, P. Cui, H. Yang, and W. Zhu, "Learning disentangled representations for recommendation," in *Proc. Conf. Neural Inf. Process. Syst.*, 2019, pp. 5712–5723.
[39] Y. Wang et al., "Deep graph mutual learning for cross-domain recommendation," in *Proc. Int. Conf. Database Syst. Adv. Appl.*, 2022, pp. 298–305.
[40] Y. Wang et al., "DisenCTR: Dynamic graph-based disentangled representation for click-through rate prediction," in *Proc. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2022, pp. 2314–2318.
[41] Y. Qin et al., "DisenPOI: Disentangling sequential and geographical influence for point-of-interest recommendation," in *Proc. Int. ACM Conf. Web Search Data Mining*, 2023, pp. 508–516.
[42] H. Li et al., "Disco: Graph-based disentangled contrastive learning for cold-start cross-domain recommendation," in *Proc. AAAI Conf. Artif. Intell.*, vol. 39, no. 11, 2025, pp. 12049–12057.
[43] Y. Wang et al., "GMR-Rec: Graph mutual regularization learning for multi-domain recommendation," *Inf. Sci.*, vol. 703, 2025, Art. no. 121946.
[44] Y. Yang, Z. Feng, M. Song, and X. Wang, "Factorizable graph convolutional networks," in *Proc. Conf. Neural Inf. Process. Syst.*, 2020, pp. 20286–20296.
[45] H. Li et al., "Disentangled contrastive learning on graphs," in *Proc. Conf. Neural Inf. Process. Syst.*, 2021, pp. 21872–21884.
[46] Y. Wang et al., "DisenSemi: Semi-supervised graph classification via disentangled representation learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 36, no. 5, pp. 8192–8204, May 2025.
[47] L. Dinh, D. Krueger, and Y. Bengio, "Nice: Non-linear independent components estimation," 2014, *arXiv:1410.8516*.
[48] R. Brügger, C. Baumgartner, and E. Konukoglu, "A partially reversible u-net for memory-efficient volumetric image segmentation," *Med. Image Comput. Comput. Assist. Intervention*, vol. 11766, pp. 429–437, 2019.
[49] V. Chiley et al., "RevBiFPN: The fully reversible bidirectional feature pyramid network," *Proc. Mach. Learn. Syst.*, vol. 5, pp. 625–645, 2023.
[50] M. Tan, R. Pang, and Q. V. Le, "Efficientdet: Scalable and efficient object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 10781–10 790.
[51] M. MacKay, P. Vicol, J. Ba, and R. Grosse, "Reversible recurrent neural networks," in *Proc. Conf. Neural Inf. Process. Syst.*, 2018, pp. 9043–9054.
[52] H. Zhang and Y. Zhang, "Memory-efficient reversible spiking neural networks," in *Proc. AAAI Conf. Artif. Intell.*, 2024, pp. 16759–16 767.
[53] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.
[54] W. Ju et al., "Zero-shot node classification with graph contrastive embedding network," *Trans. Mach. Learn. Res.*, 2023.
[55] Y. Gu et al., "DEER: Distribution divergence-based graph contrast for partial label learning on graphs," *IEEE Trans. Multimedia*, early access, May 31, 2024, doi: 10.1109/TMM.2024.3408038.
[56] A. v. d. Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," 2018, *arXiv:1807.03748*.
[57] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 1597–1607.
[58] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. Int. Conf. Learn. Representations*, 2017.
[59] F. Wu et al., "Simplifying graph convolutional networks," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 6861–6871.
[60] M. Liu, H. Gao, and S. Ji, "Towards deeper graph neural networks," in *Proc. Int. ACM SIGKDD Conf. Knowl. Discov. Data Mining*, 2020, pp. 338–348.
[61] Y. Jiang, C. Huang, and L. Huang, "Adaptive graph contrastive learning for recommendation," in *Proc. Int. ACM SIGKDD Conf. Knowl. Discov. Data Mining*, 2023, pp. 4252–4261.
[62] W. He, G. Sun, J. Lu, and X. S. Fang, "Candidate-aware graph contrastive learning for recommendation," in *Proc. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2023, pp. 1670–1679.
[63] H. Nt and T. Maehara, "Revisiting graph neural networks: All we have is low-pass filters," 2019, *arXiv:1905.09550*.
[64] G. Cui, J. Zhou, C. Yang, and Z. Liu, "Adaptive graph encoder for attributed graph embedding," in *Proc. Int. ACM SIGKDD Conf. Knowl. Discov. Data Mining*, 2020, pp. 976–985.
[65] P. Sen et al., "Collective classification in network data," *AI Mag.*, vol. 29, no. 3, pp. 93–93, 2008.
[66] O. Shchur, M. Mumme, A. Bojchevski, and S. Günnemann, "Pitfalls of graph neural network evaluation," 2018, *arXiv:1811.05868*.
[67] J. McAuley, C. Targett, Q. Shi, and A. Van Den Hengel, "Image-based recommendations on styles and substitutes," in *Proc. Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2015, pp. 43–52.
[68] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. Int. Conf. Learn. Representations*, 2022.
[69] P. Veličković et al., "Graph attention networks," in *Proc. Int. Conf. Learn. Representations*, 2017.
[70] S. Thakoor et al., "Bootstrapped representation learning on graphs," in *Proc. ICLR 2021 Workshop Geometrical Topological Representation Learn*, 2021.
[71] X. He et al., "Neural collaborative filtering," in *Proc. 26th Int. Conf. World Wide Web*, 2017, pp. 173–182.
[72] P. Li, F. He, B. Fan, and Y. Song, "TPNet: A novel mesh analysis method via topology preservation and perception enhancement," *Comput. Aided Geometric Des.*, vol. 104, 2023, Art. no. 102219.
[73] H. Xu, F. He, L. Fan, and J. Bai, "D3AdvM: A direct 3D adversarial sample attack inside mesh data," *Comput. Aided Geometric Des.*, vol. 97, 2022, Art. no. 102122.
[74] L. Li, F. He, R. Fan, B. Fan, and X. Yan, "3D reconstruction based on hierarchical reinforcement learning with transferability," *Integr. Comput.-Aided Eng.*, vol. 30, no. 4, pp. 327–339, 2023.

Yuntai Ding is currently working toward the Ph.D. degree in software engineering with Software College, Northeastern University, Shenyang, China. His research interests include graph representation learning and sequential recommendation.

Tao Ren received the B.S., M.S., and Ph.D. degrees in engineering from Northeastern University, Shenyang, China, in 2003, 2005, and 2007, respectively. He held a Postdoctoral position in computer science, from 2009 to 2013. He is currently a Professor with Northeastern University. He is in charge of 20 projects, such as the National Natural Science Foundation of China. He has authored or coauthored more than 50 high-qualified academic papers in several high-ranking journals or conferences. He has also authored or coauthored four books and authorized more than 20 Chinese patents. His main research interests include graph representation learning, machine learning and its applications.

Yifan Wang received the B.S. and M.S. degrees in software engineering from Northeastern University, Liaoning, China, in 2014 and 2017, respectively, and the Ph.D. degree in computer science from Peking University, Beijing, China, in 2023. He is currently an Assistant Professor with the School of Information Technology & Management, University of International Business and Economics, Beijing. His research interests include graph representation learning, graph neural networks, disentangled representation learning, and corresponding applications such as drug discovery and recommender systems.

Chong Chen received the B.S. degree in mathematics and the Ph.D. degree in statistics from Peking University, Beijing, China, in 2013 and 2019, respectively under the supervision of Prof. Ruibin Xi. He is currently a research scientist with Terminus Group. His research interests include image understanding, self-supervised learning, and data mining.

Xian-Sheng Hua (Fellow, IEEE) received the B.S. and Ph.D. degrees in applied mathematics from Peking University, Beijing, China, in 1996 and 2001, respectively. In 2001, he joined Microsoft Research Asia, as a Researcher, and has been a Senior Researcher with Microsoft Research Redmond since 2013. He was a Researcher and the Senior Director of Alibaba Group in 2015. He is currently a Full Professor with the Institute of AI for Engineering, Tongji University, Shanghai, China, and a Principal Researcher with Terminus Group. He has authored or coauthored more than 250 research articles and has filed more than 90 patents. His research interests include multimedia search, advertising, understanding, and mining, pattern recognition, and machine learning. He was honored as one of the recipients of MIT35. He was a Program Co-Chair for the IEEE ICME 2013, the ACM Multimedia 2012, and the IEEE ICME 2012, and on the Technical Directions Board for the IEEE Signal Processing Society. He is an ACM Distinguished Scientist.

Wei Ju received the B.S. degree in mathematics from Sichuan University, Sichuan, China, in 2017, and the Ph.D. degree from the School of Computer Science, Peking University, Beijing, China, in 2022. He was a Postdoc Research Fellow. He is currently an Associate Professor with the College of Computer Science, Sichuan University. He has authored or coauthored more than 60 papers in top-tier venues. His current research interests include machine learning on graphs including graph representation learning and graph neural networks, and interdisciplinary applications such as recommender systems, bioinformatics, drug discovery, and spatio-temporal analysis. He won the best paper finalist in IEEE ICDM 2022.