Graph Linear Convolution Pooling for Learning in Incomplete High-Dimensional Data

Fanghui Bi, Tiantian He, *Member, IEEE*, Yew-Soon Ong, *Fellow, IEEE*, and Xin Luo, *Fellow, IEEE*

***Abstract*—High-dimensional and incomplete (HDI) data are frequently encountered in diverse real-world applications involving complex interactions among numerous nodes. Approaches based on latent feature analysis (LFA) have proven effective in performing representation learning in HDI data. Nevertheless, they cannot handle the high-order connectivity among nodes in HDI data well, resulting in severe accuracy loss. To address the previously mentioned issue, we present a novel model in this paper, namely Graph Linear Convolution Pooling Network (GLCPN). The proposed GLCPN adopts the three-fold ideas. First, it leverages simplified graph convolutions to efficiently capture high-order connectivity among nodes for learning representations of matrix factorization. Second, a simple yet effective priori convolution operator is adopted by each graph neural layer to capture node-node collaboration for aggregation. Third, a locality-enhanced pooling scheme is designed to holistically utilize multi-layer representations of the neighborhood. Therefore, GLCPN can effectively acquire the hidden information in HDI data with high efficiency. In addition, we have conducted a theoretical analysis demonstrating that the proposed GLCPN is more expressive compared with existing graph neural networks for HDI data. Extensive experiments have been further conducted on ten well-established HDI datasets from various applications. The experimental results demonstrate that the proposed GLCPN significantly outperforms state-of-the-art models for learning representations in HDI data evaluated by accuracy and efficiency metrics.**

***Index Terms*—Data Science, Knowledge Acquisition, High-Dimensional and Incomplete Data, Graph Neural Networks, Non-Euclidean Data, Network Representation Learning, Missing Data Estimation, Latent Feature Analysis**

I. Introduction

WEIGHTED networks widely exist in various real industrial applications in the information-exploding era, e.g., protein networks [1], sensor networks [2], social networks [3], and intelligent transportation networks [4]. They contain complex interactions among numerous nodes, and full interaction mapping is impossible to achieve owing to the explosively increasing number of nodes involved. For instance, it is unrealistic to build full scientific collaboration networks even in an exclusive research field [5]. Consequently, such interaction data are always incomplete and commonly quantized as a High-Dimensional and Incomplete (HDI) matrix [1]-[5]. An HDI matrix, in spite of its extreme incompleteness, consists of a large number of valuable patterns and knowledge used in various real-world tasks such as Protein-Protein Interaction (PPI) prediction in biological processes [1], Quality-of-Service (QoS) estimation in cloud computing systems [6], sensor calibration in wireless sensor networks [2], community detection in social networks [3], and feature association analysis in transportation networks [4]. Effectively learning representations from HDI data for diverse downstream tasks has drawn much interest recently.

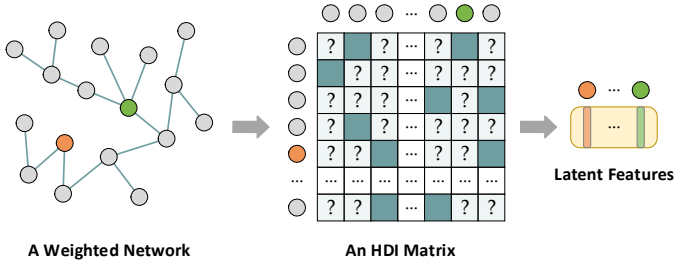


Fig. 1. Example of latent feature analysis on a weighted network. In a weighted network, the nodes could be proteins, sensors, drugs, users, services, and scientists, while the weights may quantify the protein-protein interactions, quality of services, co-authorships among scientists, or drug-target binding affinity. It is typically first characterized by an HDI matrix to preserve its inherent properties. Consequently, approaches based on latent feature analysis learn node representations for various downstream tasks.

Great research efforts have been devoted to tackling this challenging issue, and a pyramid of cutting-edge representation learning methods has been developed for HDI data. Among them, the approaches based on Latent Feature Analysis (LFA) [2], [4], [6]-[9] become dominant for their effectiveness in performing efficient representation learning to HDI data, whose processing flow is depicted in Fig. 1. It operates solely on a target HDI matrix, subsequently learning informative latent features for nodes through various techniques. Among these techniques, Matrix Factorization (MF) is commonly utilized, which maps each involved node into a specific low-dimensional latent feature space representing the node representations [2], [4], [6]-[9]. Based on the learned representations of each node, an MF-based LFA model can perform necessary tasks, such as PPI prediction, among others.

In the realm of representation learning in an HDI matrix, a prominent approach that most MF-based LFA models adopt is performing the inner product of node representations to

F. Bi and X. Luo are with the College of Computer and Information Science, Southwest University, Chongqing 400715, China (e-mail: bifanghui@cigit.ac.cn, luoxin21@gmail.com).

T. He is with the Center for Frontier AI Research, Institute of High Performance Computing, Agency for Science, Technology and Research (A*STAR), 699010, Singapore (e-mail: he_tiantian@cfar.a-star.edu.sg).

Y.S. Ong is with the Center for Frontier AI Research, Institute of High Performance Computing, Agency for Science, Technology and Research (A*STAR), and also with the College of Computing and Data Science, Nanyang Technological University, 699010, Singapore (e-mail: asysong@ntu.edu.sg).

Corresponding Author: X. Luo.

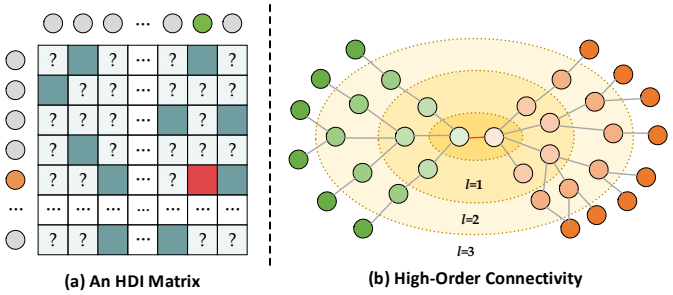


Fig. 2. An illustrative example of an HDI matrix and its graph structure. Note that for each interaction of two nodes, their multi-hop neighbors are implicitly involved through the intermediate connections, which emphasizes the importance of high-order connectivity information for obtaining more accurate estimations and node representations.

estimate missing interactions [6], [9]-[12]. Conversely, several studies have investigated the incorporation of Deep Neural Networks (DNNs) into the MF process for learning nonlinear interactions [7], [13]-[18]. Compared with linear approaches, these DNN-enhanced LFA models have demonstrated the ability to learn more complex interactions in the HDI data. However, they always suffer from training difficulties, resulting in suboptimal representations that are inferior to linear counterparts in certain contexts [6], [10]-[11]. A fundamental limitation inherent in these models is that they cannot handle the non-Euclidean structure in HDI data well. Specifically, they are not able to capture the high-order connectivity, as illustrated in Fig. 2, leading to sparse collaborative signals and a significant loss in accuracy.

Graph Neural Networks (GNNs) have recently become popular for their superiority in exploring graph-structured data to learn representations [19]-[29]. Naturally, they have been integrated into LFA to extract high-order connectivity patterns. However, when performing latent feature analysis in the HDI matrix, the following critical challenges still hinder existing GNN-based models:

(1)First, the majority of existing methods focus on developing more intricate model architectures, such as preserving feature transformation and nonlinearity [24]-[25], introducing self-attention mechanisms [20], [30], using contrastive learning [28], [31], constructing hypergraphs [23], and augmenting normalization variations [32]-[33]. They have been proven to be highly effective in specific scenarios, and the model size mainly depends on the feature dimension. However, such relatively complex structures easily cause training difficulties [27], [34]-[35], resulting in degraded generalization accuracy and efficiency.

(2)Second, existing approaches cannot well model the node collaboration in HDI data, i.e., they commonly ignore the fact that some nodes may have stronger interactions than others and thus should be assigned higher weights when aggregating neighborhood information. Applying attention-based methods [20], [30] is a natural choice, but the joint learning of attention scores and latent features may have negative consequences if the node features are uncertain and might result in efficiency loss due to increased computational demands [27], [36].

(3)Third, the long-range neighborhood information is overly aggregated by existing GNN-based models, i.e., the salience of each node's distinctive representation and the local similarities between involved nodes are distorted by the excessive message aggregation. While vanilla layer pooling mechanisms [27], [34]-[35] can somewhat alleviate this issue, they do not enhance the perception of locality. Hence, they still encounter the over-smoothing problem and node independence loss even in a shallow network, resulting in suboptimal representations.

To address the previously mentioned challenges, in this paper, we investigate whether a graph neural architecture based on a succinct yet effective learning paradigm can accurately learn representations in HDI data. To this end, we propose a novel model, namely Graph Linear Convolution Pooling Network (GLCPN). In contrast to existing approaches, GLCPN encompasses three novel modules. First, a simplified GNN structure is designed as the core building block of GLCPN, thereby effectively capturing the high-order connectivity in the linear form. Second, a simple yet effective graph convolution operator is additionally adopted by each layer of GLCPN to differentially aggregate the neighboring information according to the connection strength. Third, a locality-enhanced pooling scheme is further integrated into GLCPN to holistically fuse the multi-layer representations to alleviate the over-smoothing problem. The representations learned by the proposed GLCPN can effectively capture the features carried by the HDI matrix and demonstrate robust predictive performances in downstream tasks. Overall, this study makes the following major contributions:

(1)We propose a novel model, namely GLCPN, for learning latent features in HDI data describing weighted networks. Different from existing LFA methods, GLCPN is capable of capturing higher-order connectivity and holistically pooling the multi-layer representations, thereby learning high-quality representations for downstream tasks.

(2)For the first time, we conducted a theoretical analysis regarding the expressivity of methods for representation learning in HDI data. We demonstrate that GLCPN is more expressive than existing methods, indicating that GLCPN can theoretically perform better than most existing GNN-based approaches to learning representations in HDI data.

(3)Extensive empirical studies have been conducted on ten large-scale real HDI datasets raised from four distinct applications to evaluate the performance of GLCPN. The experimental results demonstrate that it can effectively capture the high-order connectivity from HDI data, thereby significantly outperforming state-of-the-art linear and GNN-based models in terms of accuracy for predicting missing HDI data and computational efficiency.

The rest of this paper is organized as follows. Section II presents the preliminaries. Section III describes GLCPN. Section IV theoretically analyzes the learning capability of GLCPN. Section V presents and analyzes the experimental results. Section VI summarizes the related work. Section VII concludes this paper and suggests future research directions.

II. Preliminaries

A. Notations

This paper follows the notation conventions given below: 1) Sets are indicated using uppercase Roman letters, e.g., Ω; 2)

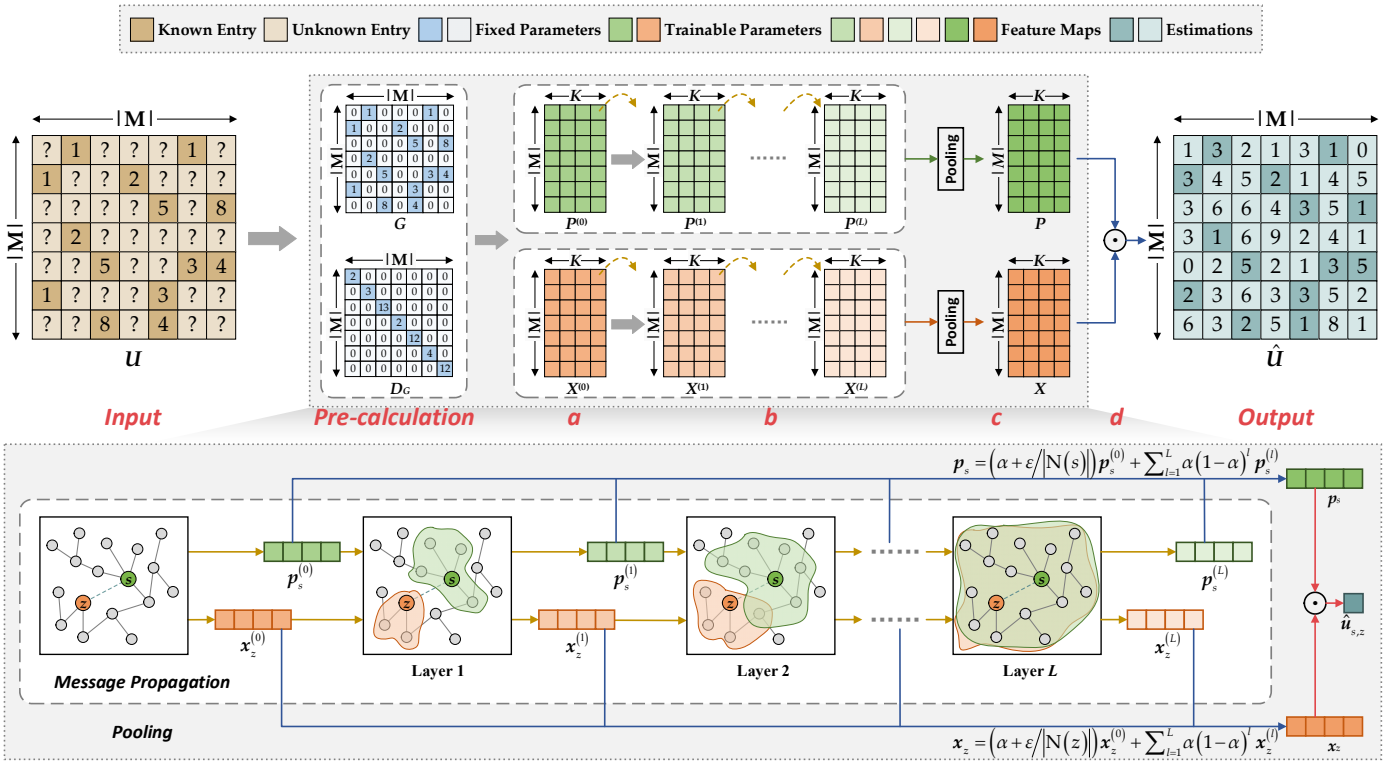


Fig. 3. Overall structure of the proposed GLCPN. It first takes an HDI matrix (U) as input, while simultaneously pre-calculating the weighted adjacency matrix (G) and diagonal matrix (D_G). Subsequently, it: a) initializes two node feature matrices to be trained ($P^{(0)}$, $X^{(0)}$); b) iteratively performs linear message propagation to obtain the latent feature matrices ($P^{(1)}$, $P^{(2)}$,…, $P^{(L)}$, $X^{(1)}$, $X^{(2)}$,…, $X^{(L)}$) capturing high-order connectivity; c) assembles node features of different orders into final representations (P, X) with the locality-enhanced pooling method; and d) predicts the missing entries via inner product and outputs $\hat{U}$. Best viewed in color.

matrices are denoted by uppercase italic letters, e.g., U; 3) vectors are denoted by bold lowercase italic letters, e.g., $\boldsymbol{u}$; 4) scalars are represented by lowercase italic letters, e.g., u; 5) functions are denoted by bold lowercase italic Roman letters, e.g., $\boldsymbol{\mu}(\cdot)$; 6) algorithm costs are denoted by uppercase script letters, e.g., $\mathcal{U}$; 7) fields of numbers and feature spaces are denoted using double-struck letters, e.g., $\mathbb{U}$; 8) the elements in the i-th row of a matrix U is represented as $\boldsymbol{u}_i$; 9) the element in the i-th row and j-th column of a matrix U is denoted as $u_{i,j}$; 10) the n-th entry in a set is denoted as n. Except for a few key notations such as layer number (L), feature dimension (K), batch size (B), and the maximum number of training epochs (T), we adhere to the aforementioned rules. For more notation details, Table SI in the Supplementary File (SF)[1] lists all the symbols involved in this paper.

B. Problem Formulation

An HDI matrix describing a certain type of relationship among a massive set of nodes is the fundamental input for a certain LFA model, where the interactions could be unipartite or bipartite. For easy understanding, we primarily introduce the related concepts in the form of unipartite interactions. Commonly, an HDI matrix is defined as follows [4], [6]-[10].

***Definition* 1. (*An HDI matrix*).** Given a node set M, $U^{|\mathrm{M}|\times|\mathrm{M}|}$ is a matrix, whose each entry $u_{s,z}$ quantifies the interaction between two nodes s, $z\in$M. Let Λ be U's known entry set, O be the unknown one, and U is an HDI matrix if $|\Lambda|\ll|\mathrm{O}|$.

LFA models have been shown to be effective in performing representation learning in the HDI matrix, by leveraging the latent features of nodes to seek its optimal approximation. Generally, an LFA model is defined as [2], [4]-[10]:

***Definition* 2. (*An LFA model*).** Given U, an LFA model only leverages Λ to build its rank-K approximation $\hat{U}$, in which $\hat{u}_{s,z}$ is the estimation for two nodes s, $z\in$M such that the generated loss (e.g., $\sum_{u_{s,z}\in\Lambda}(u_{s,z}-\hat{u}_{s,z})^2$) is minimized, and $K\ll|\mathrm{M}|$.

III. The GLCPN Model

This section introduces the GLCPN model, whose overall architecture is illustrated in Fig. 3. In general, given a target HDI matrix, GLCPN performs the representation learning by completing the following four sequential phases:

(a) **Initialization for Node Representations:** GLCPN first initializes the node representations and takes a symmetrically normalized adjacency matrix as the input.

(b) **Extraction of High-Order Connectivity:** GLCPN then extracts the high-order connectivity from the HDI data by iteratively performing linear graph message propagation.

(c) **Locality-Enhanced Holistic Pooling.** Afterward, GLCPN aggregates the representations of each node itself and its neighbors by a novel pooling scheme to obtain the resultant node representations for estimation.

(d) **Model Prediction and Optimization.** At last, it performs model optimization and prediction.

In what follows, the details of GLCPN will be introduced.

A. Initialization for Node Representations

Empirical MF-based LFA models [2], [4], [10]-[12], [15]-[18] decompose an HDI matrix into two feature matrices as the node representations. In this paper, we follow previous

[1]https://github.com/Oak-B/GLCPN/blob/main/GLCPN-Supplementary%20File.pdf

settings and define the node representations as two matrices $P^{(0)}$, $X^{(0)}\in\mathbb{R}^{|\mathrm{M}|\times K}$, where K is the dimensionality of latent feature space. The parameters in $P^{(0)}$ and $X^{(0)}$ are randomly initialized and optimized to a stable state in an end-to-end manner, making GLCPN suitable for the lack of initial node features in HDI data across various scenarios.

When performing the estimation for two specified nodes s and z, a conventional LFA model calculates as:

$$\hat{u}_{s,z} = \boldsymbol{p}_s^{(0)} \circ \boldsymbol{x}_z^{(0)}, \tag{1}$$

where $\hat{u}_{s,z}$ denotes the estimation for $u_{s,z}$ in an HDI matrix U, $\boldsymbol{p}_s^{(0)}$, $\boldsymbol{x}_z^{(0)}\in\mathbb{R}^K$ are the representations of s and z corresponding to the s-th row and z-th row parameters in $P^{(0)}$ and $X^{(0)}$, and $\circ$ indicates an interaction operator such as inner product or fully-connected network. It is worth noting that this approach only considers the node representations of s and z themselves, which is insufficient as shown in Fig. 2. In contrast, our proposed GLCPN explicitly aggregates their neighborhood information on the entire graph to achieve more accurate estimation while maintaining the efficiency.

B. Extraction of High-Order Connectivity

Graph neural networks can effectively learn in non-Euclidean structure (e.g., graphs), whose key idea is message propagation [19]-[22]. The conventional nonlinear message propagation along the graph structure in HDI data is defined as follows.

For a connected node pair (s, z) in the graph, the $(l+1)$-th layer node representations of s and z can be generated as:

$$\begin{cases} \boldsymbol{p}_s^{(l+1)} = \boldsymbol{\sigma}\left(W_P^{(l)} \cdot \mathrm{AGG}\left(\boldsymbol{p}_s^{(l)}, \left\{\boldsymbol{p}_i^{(l)} \mid i \in \mathrm{N}(s)\right\}\right)\right), \\ \boldsymbol{x}_z^{(l+1)} = \boldsymbol{\sigma}\left(W_X^{(l)} \cdot \mathrm{AGG}\left(\boldsymbol{x}_z^{(l)}, \left\{\boldsymbol{x}_i^{(l)} \mid i \in \mathrm{N}(z)\right\}\right)\right), \end{cases} \tag{2}$$

where $\boldsymbol{p}_s^{(l+1)}\in P^{(l+1)}$ and $\boldsymbol{x}_z^{(l+1)}\in X^{(l+1)}$ are the $(l+1)$-th layer's node representations of s and z, $\boldsymbol{p}_s^{(l)}$, $\boldsymbol{p}_i^{(l)}\in P^{(l)}$ and $\boldsymbol{x}_z^{(l)}$, $\boldsymbol{x}_i^{(l)}\in X^{(l)}$ are the ones in the l-th layer, $\mathrm{N}(s)$ and $\mathrm{N}(z)$ are the first hop neighbor sets corresponding to s and z, $\mathrm{AGG}(\cdot)$ indicates some aggregation function for the last layer features, and $\boldsymbol{\sigma}(\cdot)$ is a nonlinear activation function, e.g., Rectified Linear Unit (ReLU). $W_P^{(l)}$, $W_X^{(l)}\in\mathbb{R}^{K^{(l+1)}\times K^{(l)}}$ are the learned transform matrices of filtering parameters in $P^{(l)}$ and $X^{(l)}$, and $K^{(l+1)}$, $K^{(l)}$ are the feature dimensions of $(l+1)$-th and l-th layers, which are uniformly set to K in our study. Several previous studies have focused on designing complicated message propagation strategies [21]-[24], which are mostly variants of the graph convolutional network [19]. Following the idea of graph convolutional networks, Eq. (2) is rewritten as the following:

$$\begin{cases} \boldsymbol{p}_s^{(l+1)} = \boldsymbol{\sigma}\left(W_P^{(l)}\boldsymbol{p}_s^{(l)} + \sum\limits_{i\in\mathrm{N}(s)} \dfrac{1}{\sqrt{|\mathrm{N}(s)|\cdot|\mathrm{N}(i)|}} W_P^{(l)}\boldsymbol{p}_i^{(l)}\right), \\ \boldsymbol{x}_z^{(l+1)} = \boldsymbol{\sigma}\left(W_X^{(l)}\boldsymbol{x}_z^{(l)} + \sum\limits_{i\in\mathrm{N}(z)} \dfrac{1}{\sqrt{|\mathrm{N}(z)|\cdot|\mathrm{N}(i)|}} W_X^{(l)}\boldsymbol{x}_i^{(l)}\right). \end{cases} \tag{3}$$

In Eq. (3), $1/\sqrt{|\mathrm{N}(s)|\cdot|\mathrm{N}(i)|}$ and $1/\sqrt{|\mathrm{N}(z)|\cdot|\mathrm{N}(i)|}$ are the symmetrically normalized forms that reduce numerical instabilities and decay the propagation with the path length.

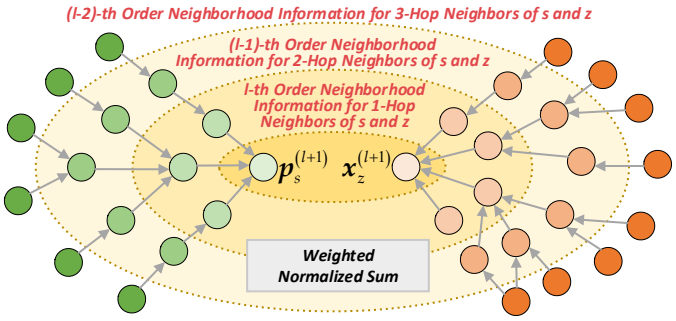


Fig. 4. An example illustrating how high-order connectivity information for nodes s and z is captured. Note that the self-connection, feature transformation, and nonlinear activation are omitted, and only the weighted normalized sum of neighborhood information in the previous layer is propagated to the next layer. This simplifies the message propagation rules for fine training of node representations and assigns different weights to different neighbors for attending over their features.

And the layer-wise nonlinear message propagation rules in Eqs. (2) and (3) can be reformulated into the matrix form:

$$\begin{cases} P^{(l+1)} = \boldsymbol{\sigma}\left(\left(\hat{A}+I\right)\cdot P^{(l)}\cdot\left(W_P^{(l)}\right)^T\right), \\ X^{(l+1)} = \boldsymbol{\sigma}\left(\left(\hat{A}+I\right)\cdot X^{(l)}\cdot\left(W_X^{(l)}\right)^T\right), \end{cases} \tag{4}$$

where $\hat{A}=D^{-1/2}AD^{-1/2}$ is A's symmetrically normalized matrix, in which each element $\hat{a}_{s,i}$ is equal to $1/\sqrt{|\mathrm{N}(s)|\cdot|\mathrm{N}(i)|}$ in Eq. (3). A is the $|\mathrm{M}|\times|\mathrm{M}|$ adjacency matrix of the given HDI matrix U. D is the degree matrix of A, in which the i-th diagonal entry d_{ii} denotes the number of first hop neighbors of i. And I is the $|\mathrm{M}|\times|\mathrm{M}|$ identity matrix. By iteratively performing Eq. (4), the L-th layer representations $P^{(L)}$ and $X^{(L)}$ containing high-order connectivity information are output to estimate the missing entries.

Note that the propagation rules in Eqs. (3) and (4) just simply incorporate general graph convolutional networks into MF. When dealing with an HDI matrix, the following issues should be taken into account: First, the node representations in the 0-th layer are randomly initialized and optimized, which are the fundamental optimization parameters. However, they are challenging to train due to the transformation matrix of each layer and nonlinear activation function, resulting in the loss of accuracy. Second, in HDI data, the connection strength between any two nodes varies, necessitating assigning different weights to the neighbors when performing message propagation. Third, the 0-th layer node representations, i.e., the node representations themselves, are key components in the final output, which should be specially considered to mitigate the issues of over-smoothing and node independence. Thus, the vanilla nonlinear message propagation rules in Eqs. (3) and (4) need further improvement.

Efficient Linear Message Propagation. Based on the above analyses and prior studies [27], [31]-[33], the transformation matrices and nonlinearities in conventional graph convolutional layers are unnecessary and even detrimental. Thus, in the proposed approach, we omit them and propose our linear message propagation strategy (Fig. 4), which generates the representations for $(l+1)$-th layer following the procedure below:

$$\begin{cases} \boldsymbol{p}_s^{(l+1)} = \sum\limits_{i\in \mathrm{N}(s)} \dfrac{1}{\sqrt{\left|\mathrm{N}(s)\right|\cdot\left|\mathrm{N}(i)\right|}} \boldsymbol{p}_i^{(l)}, \\ \boldsymbol{x}_z^{(l+1)} = \sum\limits_{i\in \mathrm{N}(z)} \dfrac{1}{\sqrt{\left|\mathrm{N}(z)\right|\cdot\left|\mathrm{N}(i)\right|}} \boldsymbol{x}_i^{(l)}, \end{cases} \tag{5}$$

where the layer structure is simplified by removing the transformation matrix $W_P^{(l)}$, $W_X^{(l)}$ and the nonlinear activation function $\boldsymbol{\sigma}(\cdot)$ shown in Eq. (4). Thus, the node representations $P^{(l+1)}$ and $X^{(l+1)}$ at the (l+1)-th layer are obtained by aggregating the normalized sum of the l-th layer representations of their neighbors, which essentially captures the (l+1)-th order neighborhood features. Leveraging the strategy shown in Eq. (5), we can achieve lower training difficulty, i.e., higher computational efficiency and representation accuracy.

Priori Convolution Incorporating Node Collaboration. In HDI data, a quantifiable index reflecting the correlation or connection strength is assigned to the interaction between two nodes in the form of weight, e.g., the confidence of interactome between proteins [1], the certain quality of cloud services [6], the binding affinity in Drug-Protein Interactions (DPIs) [37], the intensity of scientific collaboration [2], and the user contact rate in social networks [3]. Such fundamental node-node interactions are commonly neglected and only used in calculating the optimization loss in previous work [2], [4], [9]-[12]. To enable the proposed approach to capture such information, we propose a novel priori convolution operator:

$$\begin{cases} \boldsymbol{p}_s^{(l+1)} = \sum\limits_{i\in \mathrm{N}(s)} \dfrac{g_{s,i}}{\sqrt{w(s)\cdot w(i)}} \boldsymbol{p}_i^{(l)}, \\ \boldsymbol{x}_z^{(l+1)} = \sum\limits_{i\in \mathrm{N}(z)} \dfrac{g_{z,i}}{\sqrt{w(z)\cdot w(i)}} \boldsymbol{x}_i^{(l)}, \end{cases} \tag{6}$$

where $g_{s,i}\in G$ and $g_{z,i}\in G$ are regarded as the priori convolution coefficients assigned to the neighbors, G is the weighted adjacency matrix of U, and $w(s)$, $w(z)$, and $w(i)$ respectively denote the sums of the weights assigned to the neighbors of s, z, and i. Based on the analysis of autoencoders [13], [18], [25], the proposed operator can be seen as the implicit encoding for the input HDI data.

Overall, the linear message propagation rules in Eq. (6) are reformulated into the matrix form as:

$$\begin{cases} P^{(l+1)} = \hat{G}P^{(l)}, \\ X^{(l+1)} = \hat{G}X^{(l)}, \end{cases} \tag{7}$$

where $\hat{G}=D_G^{-1/2} G D_G^{-1/2}$, D_G is G's degree matrix. As shown in Fig. 3, given a sparse normalized matrix G, the message propagation operation only requires a fast sparse matrix computation, which is effective in dealing with an HDI matrix.

C. Locality-Enhanced Pooling

As discussed in [19]-[23], deep GNNs suffer from over-smoothing or gradient vanishing when stacking multiple graph convolutional layers. To mitigate this limitation, some GNNs incorporate skip-connections for representation learning, such as concatenating each layer with the final layer [34], residually connecting the consecutive layers [35], or combining both [32]. In the final output phase, the common layer pooling approaches are *Stack*, *Concat*, *Sum*, and *Mean* [24]-[27], [38]. However, these techniques lack specific consideration for the 0-th layer node features. The 0-th layer representations are initialized as trainable latent features, which possibly confront challenges of over-smoothing or gradient vanishing. Thus, we propose a locality-enhanced pooling scheme (Fig. 3), which enables the proposed model to learn expressive representations for downstream tasks.

Locality-Enhanced Residual Learning. To preserve the original features while propagating messages, existing methods attempt to include controlled self-connections at each layer:

$$\begin{cases} \boldsymbol{p}_s^{(l+1)} = \alpha\boldsymbol{p}_s^{(0)} + (1-\alpha) \sum\limits_{i\in \mathrm{N}(s)} \dfrac{g_{s,i}}{\sqrt{\left|w(s)\right|\cdot\left|w(i)\right|}} \boldsymbol{p}_i^{(l)}, \\ \boldsymbol{x}_z^{(l+1)} = \alpha\boldsymbol{p}_z^{(0)} + (1-\alpha) \sum\limits_{i\in \mathrm{N}(z)} \dfrac{g_{z,i}}{\sqrt{\left|w(z)\right|\cdot\left|w(i)\right|}} \boldsymbol{x}_i^{(l)}, \end{cases} \tag{8}$$

where α is the teleport probability that controls the reserving of the starting features, i.e., self-node representations. Note that the self-connections in Eq. (8) are to link the 0-th layer features instead of the l-th layer ones for better balancing the representations of each node itself and its neighbors. In matrix form, Eq. (8) can be reformulated as follows:

$$\begin{cases} P^{(l+1)} = \alpha P^{(0)} + (1-\alpha)\hat{G}P^{(l)}, \\ X^{(l+1)} = \alpha X^{(0)} + (1-\alpha)\hat{G}X^{(l)}. \end{cases} \tag{9}$$

By stacking multiple propagation layers, the final L-th feature matrix $P^{(L)}$ is obtained:

$$\begin{aligned} P^{(L)} &= \alpha P^{(0)} + (1-\alpha)\hat{G}P^{(L-1)} \\ &= \alpha P^{(0)} + \alpha(1-\alpha)\hat{G}P^{(0)} + (1-\alpha)^2\hat{G}^2P^{(L-2)} \\ &= \alpha P^{(0)} + \alpha(1-\alpha)\hat{G}P^{(0)} \\ &\quad + \alpha(1-\alpha)^2\hat{G}^2P^{(0)} + \cdots + (1-\alpha)^L\hat{G}^LP^{(0)}. \end{aligned} \tag{10}$$

Similarly, $X^{(L)}$ is calculated as:

$$\begin{aligned} X^{(L)} &= \alpha X^{(0)} + \alpha(1-\alpha)\hat{G}X^{(0)} \\ &\quad + \alpha(1-\alpha)^2\hat{G}^2X^{(0)} + \cdots + (1-\alpha)^L\hat{G}^LX^{(0)}. \end{aligned} \tag{11}$$

Based on the analysis in Eqs. (9)-(11), we can observe that the resulting features of the L-th layer is the integration of all information from the first L-order neighborhoods. However, when $\alpha\to 0$, the final representation is dominated by the neighborhood of L-th order, which is detrimental for the subsequent predictive performances. To overcome this drawback, we propose Locality-Enhanced Residual Learning:

$$\begin{cases} P^{(L)} = \sum\limits_{l=0}^{L} \alpha(1-\alpha)^l\hat{G}^lP^{(0)}, \\ X^{(L)} = \sum\limits_{l=0}^{L} \alpha(1-\alpha)^l\hat{G}^lX^{(0)}. \end{cases} \tag{12}$$

Holistic Pooling. By examining Eqs. (6), (7), and (12), it is seen that the propagation rules in Eq. (12) can be interpreted as iteratively using Eq. (6) and integrating the representations from all layers to obtain the final representations of each node:

$$\begin{cases} \boldsymbol{p}_s = \sum_{l=0}^{L} \alpha(1-\alpha)^l \boldsymbol{p}_s^{(l)}, \\ \boldsymbol{x}_z = \sum_{l=0}^{L} \alpha(1-\alpha)^l \boldsymbol{x}_z^{(l)}, \end{cases} \tag{13}$$

where $\boldsymbol{p}_s \in P$ and $\boldsymbol{x}_z \in X$ are the final representations of s and z used to estimate the missing interaction between s and z. To further improve the capability of the proposed GLCPN, the output layer can be calculated as:

$$\begin{cases} \boldsymbol{p}_s = \left(\alpha + \frac{\varepsilon}{|\mathrm{N}(s)|}\right)\boldsymbol{p}_s^{(0)} + \sum_{l=1}^{L} \alpha(1-\alpha)^l \boldsymbol{p}_s^{(l)}, \\ \boldsymbol{x}_z = \left(\alpha + \frac{\varepsilon}{|\mathrm{N}(z)|}\right)\boldsymbol{x}_z^{(0)} + \sum_{l=1}^{L} \alpha(1-\alpha)^l \boldsymbol{x}_z^{(l)}, \end{cases} \tag{14}$$

where $\varepsilon \in (0, 1)$ is a learnable irrational. The linear message propagation rule in Eq. (6) and the locality-enhanced holistic pooling mechanism in Eq. (14) focus on retaining the node representations themselves, while efficiently aggregating their high-order connectivity information.

D. Model Prediction and Optimization

By linearly extracting the high-order connectivity from the interaction graph, we obtain the final node representations P and X. Based on them, we perform the estimation as:

$$\hat{u}_{s,z} = \boldsymbol{p}_s \circ \boldsymbol{x}_z. \tag{15}$$

Herein, the inner product is chosen as our interaction operator ($\circ$) for simplicity and efficiency [2], [7].

To minimize the differences between the estimated values and the truth ones, the commonly adopted Euclidean distance-based objective function [2], [4], [6], [10] is used:

$$\boldsymbol{\tau}\left(P^{(0)}, X^{(0)}\right) = \sum_{u_{s,z} \in \Lambda} \left(u_{s,z} - \hat{u}_{s,z}\right)^2 + \lambda\left(\left\|P^{(0)}\right\|_F^2 + \left\|X^{(0)}\right\|_F^2\right), \tag{16}$$

where Λ is the known entry set in U, λ controls the L_2 regularization strength to prevent model overfitting.

E. Algorithm Design and Analysis

The proposed GLCPN can be efficiently trained by utilizing back propagation, as Algorithm 1 illustrates. Its time cost consists of two important factors including initialization and GNN-incorporated linear learning. Consequently, the time cost of Algorithm 1 comes to:

$$\mathcal{T}_1 = \Theta\left(|\Lambda| \times K \times L \times \lceil |\Lambda| / B \rceil \times t\right). \tag{17}$$

And the storage cost of Algorithm 1 involves four main factors: a) Data cache: $\Theta(|\Lambda|)$; b) Arrays caching parameters: $\Theta(|\mathrm{M}| \times K)$; c) Arrays caching the adjacency matrix: $\Theta(|\Lambda|)$; d) Arrays caching intermediate variables: $\Theta(|\mathrm{M}| \times K)$. To sum up, the storage cost of Algorithm 1 is calculated as:

$$\mathcal{S}_1 = \Theta\left(K \times \max\left\{|\Lambda|/K, |\mathrm{M}|\right\}\right), \tag{18}$$

which scales linearly with a target HDI matrix's known entry count and the number of latent features. According to the above analysis, it is seen that the proposed GLCPN achieves high efficiency in computation and storage when handling the HDI matrix.

ALGORITHM 1. GLCPN

Input: Λ, M

Operation

/* **Initialization** */

Initialize $P^{(0)}$, $X^{(0)}$, K, L, B, T, η, λ, α, t=1, b=1

/* **GNN-Incorporated Linear Learning** */

Construct sparse G and D_G based on the input HDI matrix;
Calculate $\hat{G}$ according to $\hat{G}=D_G^{-1/2} G D_G^{-1/2}$ in a sparse manner;
while not converge and $t \le T$ **do**
  for b=1 to $\lceil |\Lambda|/B \rceil$
    for l=1 to L
      Calculate $P^{(l)}$ and $X^{(l)}$ according to formula (7);
    end for
    Calculate P and X according to formula (14);
    Sample a batch of entries Φ from Λ;
    for $u_{s,z}$ in Φ
      Calculate $\hat{u}_{s,z}$ according to formula (15);
      Calculate the single loss according to $(u_{s,z}-\hat{u}_{s,z})^2$;
    end for
    Accumulate all the losses in Φ;
    Calculate the gradients of $P^{(0)}$ and $X^{(0)}$ according to Eq. (16);
    Update $P^{(0)}$ and $X^{(0)}$ based on the calculated gradients;
  end for
end while

/* **Operation Ending** */

Output: $P^{(0)}$ and $X^{(0)}$

IV. Model Discussions

The theoretical analysis is conducted in this section to provide an in-depth understanding of GLCPN.

A. Comparison with Related Approaches

Herein, we discuss the relation and differences between GLCPN and its closely related methods, i.e., MF [7], GAT [20], LightGCN [27], and PageRank [34]. The discussions are shown in Sec. II.A of SF.

B. Expressivity Analysis based on 1-WL Test

To learn more effective representations from HDI data, a learning model $\boldsymbol{\xi}(\cdot)$ is expected to approximate the true but priori unknown mapping $\boldsymbol{\xi}^*(\cdot)$ within a certain range. The model's expressivity refers to the estimation of this range, providing an important measure of learning potential. There are two commonly adopted approaches to enhancing the expressivity: naively increasing the model's complexity and introducing inductive biases into the model while maintaining complexity. Considering that HDI data can be organized as graphs and GNNs have proven to be effective in handling graph-structured data due to possessing a general inductive bias known as permutation invariance [19]-[20]. Thus, the learning capacity of a GNN-based model on HDI data is reflected by its expressivity on graphs. GNNs generally adhere to the basic assumption that prediction targets are independent of node order in the graph. And existing research suggests an equivalence between the ability to distinguish non-isomorphic features and the approximation of permutation-invariant functions [39]-[40]. Hence, unlike expressivity in terms of function approximation, the expressivity of GNNs is predominantly concerned with how effectively the model distinguishes different graph structures leading to distinct representations. Previous studies demonstrate that message passing GNNs are as powerful as the 1-WL test in terms of

TABLE I
PROPERTIES OF TESTING CASES.

Dataset	T-208964	T-227321	T-9544	T-9796	Astro-Ph	Cond-Mat	Cond-Mat-2003	Cond-Mat-2005	Throughput	Kiba
Entry Count	1,559,616	1,120,030	15,597,778	12,149,608	242,502	95,188	240,058	351,382	1,831,253	118,254
Node Count	5,565	7,963	20,462	19,664	16,706	16,726	31,163	40,421	6,164	2,340
Density	5.04%	1.77%	3.73%	3.14%	0.09%	0.03%	0.02%	0.02%	92.74%	24.46%
Category	Protein-Protein Interactions					Scientific Collaborations			QoS	DPI

distinguishing ability, and some representatives, e.g., GCNs and GATs, have been shown to be unable to distinguish certain structures [36], [39]. Thus, to study GLCPN's representation learning ability, the theoretical validation lies in whether the adopted aggregation and readout functions of GLCPN are homogeneous to the 1-WL test. To further clarify this, we illustrate several examples to show the distinguishing effect of different aggregators in Fig. S1 of SF.

Specifically, we have the theorem showing that the graph structures under certain conditions are not distinguishable for the neighborhood aggregation function in Eq. (6) as follows.

***Theorem* 1.** In the countable feature space $\mathbb{X}$, the aggregation function in Eq. (6) is represented as $\boldsymbol{\delta}(c, \mathrm{X}) = \sum_{x \in \mathrm{X}} \hat{g}_{c,x} \cdot x$, where c is the center node's feature, $\mathrm{X} \in \mathbb{X}$ is a multiset containing the feature vectors of neighbor nodes. For all X and the strategy in (6), $\boldsymbol{\delta}(c_1, \mathrm{X}_1) = \boldsymbol{\delta}(c_2, \mathrm{X}_2)$ iff $\mathrm{X}_1 = \{\Gamma, \mu_1\}$, $\mathrm{X}_2 = \{\Gamma, \mu_2\}$, and $\kappa \cdot \sum_{y=x, y \in \mathrm{X}_1} g_{c_1,y} \Big/ \sqrt{w(y)} = \sum_{y=x, y \in \mathrm{X}_2} g_{c_2,y} \Big/ \sqrt{w(y)}$, for $\kappa = \sqrt{w(c_2)/w(c_1)}$ and $x \in \Gamma$.

We leave the proofs in Sec. II.B of SF. From ***Theorem* 1**, it is seen that $\boldsymbol{\delta}(\cdot)$ maps different multisets (X_1 and X_2) into the same embeddings if and only if they have the same underlying set (Γ) and each sum of normalized weights between the center node and its neighbors possessing the same node features is proportional (κ). However, by adopting the pooling mechanism in Eq. (14), GLCPN can distinguish all different structures possessing certain topological properties mentioned in ***Theorem* 1**. Thus, we have the following corollary.

***Corollary* 1.** Let $\boldsymbol{\omega}(\cdot)$ be the locality-enhanced holistic pooling function in Eq. (14) that operates on a multiset $\mathrm{X} \in \mathbb{X}$, where $\mathbb{X}$ is the countable node feature space. There exists an $\mathbb{X}$ such that $\boldsymbol{\omega}(\cdot)$ can distinguish all the different multisets that were not correctly distinguished by Eq. (6) previously.

Based on the above analysis, we conclude that GLCPN's expressive ability is strong enough to enable it to learn effective node representations used for downstream tasks.

C. Spectral Expressivity Analysis

Another perspective on understanding the representation learning capacity of GNNs is through the analysis of their spectral filtering characteristics, which is closely related to graph signal processing. Based on previous studies [19], [41], the simplified graph convolutional operation of GLCPN can be viewed as applying a fixed low-pass filter and performing Laplacian smoothing, where more flexible polynomial filter coefficients indicate higher spectral expressivity. Hence, we present the form of GLCPN's propagation mechanism in the graph spectral domain to demonstrate that how its locality-enhanced pooling scheme improves its expressivity. The analysis details are shown in Sec. II.C of SF.

V. EXPERIMENTS

In this section, we conduct extensive empirical studies on ten real-world datasets to evaluate the proposed GLCPN. We first compare the performance of GLCPN with several state-of-the-art methods in terms of accuracy and efficiency. Then, we present the impacts of different hyperparameter settings, including the depth of propagation layer L, the dimensionality of latent feature space K, and the teleport probability α. Third, we perform ablation studies to analyze the impacts of novel modules proposed in this paper, including the linear message propagation rule, the priori convolution operator, and the locality-enhanced pooling mechanism.

A. General Experimental Settings

Datasets. Ten real-world industrial HDI datasets raised from four distinct applications are used in our experiments. Their properties, including the known entry count, node count, density, and category are summarized in Table I. These datasets have been frequently adopted to measure the performance of an LFA model for PPI prediction [1], [2], co-authorship analysis [5], [42], service selection [6], [17], and Drug-Target Affinity (DTA) prediction [37], [43]. Note that more details of them can be found in Sec. III.A of SF, and we have placed the original URLs for obtaining them as well as the prepared datasets on GitHub[1].

Evaluation Metrics. In this paper, we mainly consider evaluating the capability of representation learning to HDI data raised from various types of scenarios. A well-established assessment technique is to measure the accuracy of estimating the missing entries in an HDI matrix [4]-[11]. Accordingly, we use several widely-adopted evaluation metrics, including three regression error metrics and one ranking metric. They are the Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), coefficient of determination (i.e., R^2), and Normalized Discounted Cumulative Gain (NDCG) [4]-[11], [44]. Their calculation details are shown in Sec. III.B of SF.

Models for Comparisons. Seventeen advanced approaches are compared with GLCPN, which are either MF-, DNN-, or GNN-based LFA models. They include NeuMF (2017) [7], MetaMF (2020) [16], GC-MC (2018) [25], NGCF (2019) [24], LightGCN (2020) [27], LR-GCCF (2020) [35], SGL-ED (2021) [38], DGCN-HN (2021) [32], LightGCN$_{r\text{-}AdjNorm}$ (2022) [45], HMLET (2022) [8], HCCF (2022) [28], GTN (2022) [29], CIGCN (2023) [46], JMP-GCF (2023) [33], LightGCL (2023) [31], MGDN (2024) [47], and PopGo (2024) [48]. We have published our codes for GLCPN on GitHub[2], and the details of the baselines are described in Sec. III.C of SF.

[1] https://github.com/Oak-B/GLCPN/blob/main/GLCPN-Datasets.zip
[2] https://github.com/Oak-B/GLCPN/blob/main/GLCPN-Codes.zip

TABLE II
PERFORMANCE ON PPI PREDICTION.

Method	*T-208964*				*T-227321*			
	RMSE ↓	**MAE ↓**	**R^2 ↑**	**NDCG ↑**	**RMSE ↓**	**MAE ↓**	**R^2 ↑**	**NDCG ↑**
SGL-ED	0.1495±3.9E-4	0.1088±3.0E-4	0.3342±1.4E-3	0.8456±1.2E-3	0.1437±4.2E-4	0.1029±2.7E-4	0.4759±3.5E-4	0.8163±1.5E-4
LightGCN$_{r\text{-}AdjNorm}$	0.1185±3.4E-4	0.0837±2.1E-4	0.5776±1.3E-4	0.8529±6.8E-4	0.1179±3.2E-4	0.0813±1.3E-4	0.6462±3.0E-3	0.8224±2.6E-3
HMLET	0.1202±2.8E-4	0.0849±1.8E-4	0.5694±2.2E-3	0.8591±9.7E-4	0.1200±2.9E-4	0.0831±1.4E-4	0.6358±9.0E-4	0.8198±3.4E-4
HCCF	0.1299±2.1E-4	0.0940±2.4E-4	0.4969±2.0E-3	0.8520±1.3E-3	0.1329±5.4E-4	0.0939±3.2E-4	0.5494±3.3E-3	0.8177±3.8E-3
GTN	0.1271±3.5E-4	0.0906±2.0E-4	0.5202±2.4E-3	0.8477±1.3E-3	0.1267±3.4E-4	0.0895±1.6E-4	0.5925±9.6E-4	0.8182±3.3E-3
CIGCN	0.1247±2.4E-4	0.0890±8.7E-5	0.5364±1.6E-3	0.8498±7.0E-4	0.1220±2.7E-4	0.0849±1.5E-4	0.6233±2.4E-3	0.8201±3.5E-3
JMP-GCF	0.1682±1.3E-3	0.1175±2.1E-3	0.2269±1.6E-3	0.7136±7.8E-4	0.1636±1.7E-3	0.1208±2.4E-3	0.4101±2.9E-3	0.7724±4.7E-3
LightGCL	0.1324±4.6E-4	0.0957±2.8E-4	0.4809±1.2E-3	0.7470±2.0E-3	0.1284±2.6E-4	0.0907±1.8E-4	0.5831±1.7E-3	0.7205±3.2E-3
MGDN	0.1248±4.4E-4	0.0893±2.4E-4	0.5359±2.6E-3	0.8474±6.9E-4	0.1213±4.0E-4	0.0844±2.1E-4	0.6270±3.1E-3	0.8189±3.4E-3
PopGo	0.1298±3.6E-4	0.0928±2.9E-4	0.4979±2.8E-3	0.8460±5.0E-4	0.1280±4.5E-4	0.0900±2.6E-4	0.5846±3.6E-3	0.8169±3.5E-3
GLCPN (Ours)	**0.1161±3.7E-4**	**0.0810±3.2E-4**	**0.6030±1.6E-3**	**0.8676±6.0E-4**	**0.1164±2.8E-4**	**0.0800±9.7E-5**	**0.6663±2.4E-3**	**0.8257±3.6E-3**

TABLE III
PERFORMANCE ON CP PREDICTION.

Method	*Astro-Ph*				*Cond-Mat-2005*			
	RMSE ↓	**MAE ↓**	**R^2 ↑**	**NDCG ↑**	**RMSE ↓**	**MAE ↓**	**R^2 ↑**	**NDCG ↑**
SGL-ED	0.4636±1.7E-2	0.1724±3.5E-3	0.2057±1.4E-2	0.6564±4.3E-4	0.8210±1.7E-2	0.3437±2.1E-3	0.1114±1.2E-2	0.6133±1.9E-3
LightGCN$_{r\text{-}AdjNorm}$	0.4698±1.7E-2	0.1815±3.7E-3	0.1983±1.0E-2	0.6621±1.7E-3	0.8186±1.8E-2	0.3508±1.7E-3	0.1133±9.5E-3	0.6141±3.7E-4
HMLET	0.4764±1.7E-2	0.1833±3.9E-3	0.1597±9.1E-3	0.6608±7.5E-4	0.8286±1.7E-2	0.3488±2.0E-3	0.0884±1.1E-2	0.6161±1.8E-3
HCCF	0.4972±1.6E-2	0.2148±2.8E-3	0.0890±5.7E-3	0.6499±1.5E-3	0.8498±1.9E-2	0.3681±3.7E-3	0.0437±1.3E-2	0.6058±2.1E-3
GTN	0.4744±1.7E-2	0.1921±3.5E-3	0.1709±1.1E-2	0.6596±7.9E-4	0.8178±1.8E-2	0.3540±1.8E-3	0.1104±1.1E-2	0.6155±1.8E-3
CIGCN	0.4756±1.8E-2	0.1758±3.7E-3	0.1636±5.0E-3	0.6611±8.3E-4	0.8356±1.8E-2	0.3454±2.5E-3	0.0736±1.2E-2	0.6163±1.8E-3
JMP-GCF	0.4823±1.7E-2	0.2052±3.9E-3	0.1435±9.6E-3	0.6521±2.8E-4	0.8242±1.7E-2	0.3640±1.7E-3	0.1013±1.3E-2	0.6132±2.0E-3
LightGCL	0.4751±1.8E-2	0.1845±3.9E-3	0.1692±1.0E-2	0.6060±1.0E-3	0.8235±1.8E-2	0.3508±2.0E-3	0.1006±9.2E-3	0.5709±2.3E-3
MGDN	0.4690±1.7E-2	0.1817±3.7E-3	0.1891±1.1E-2	0.6604±9.4E-4	0.8196±1.7E-2	0.3496±2.5E-3	0.1098±1.2E-2	0.6168±1.7E-3
PopGo	0.4734±1.6E-2	0.1890±2.9E-3	0.1738±1.2E-2	0.6592±1.2E-3	0.8226±1.7E-2	0.3510±1.8E-3	0.1034±8.6E-3	0.6164±1.8E-3
GLCPN (Ours)	**0.4527±1.7E-2**	**0.1628±3.7E-3**	**0.2521±9.4E-3**	**0.6656±6.0E-4**	**0.8015±1.8E-2**	**0.3200±2.4E-3**	**0.1476±1.5E-2**	**0.6196±1.7E-3**

Training Settings and Experimental Environment. For fair comparisons, the established settings are adopted in our experiments, which are shown in Sec. III.D of SF. Our general training settings for each model like data splitting methods, unique configurations to GLCPN such as the teleport probability α, and implementation details of the comparison methods are all included in it. Besides, it is worth noting that we implement all the experiments in Python 3.7, and deploy them on a server with one 2.4-GHz Intel Xeon 4214R CPU, four NVIDIA RTX 3090 GPUs, and 128-GB RAM.

B. Comparison with State-of-the-Art Approaches

For the in-depth comparison, we consider four distinct learning tasks, including PPI prediction, Collaboration Potential (CP) prediction, QoS estimation, and DTA prediction. These tasks play important roles in diverse real-world applications such as disease research, optimization of research resources, service selection, and drug discovery. Owing to space constraints, we have presented the achieved RMSE, MAE, R^2, and NDCG of eleven recently proposed approaches, including GLCPN, in Tables II-V. Besides, the supplementary comparison results regarding accuracy and efficiency are appended to Tables SII-SIV of SF. Based on these records, the Friedman test [50] is conducted to further verify the superiority of GLCPN, whose results are listed in Tables SV-SVI.

i. Results of PPI Prediction

Due to the better modeling for high-order connectivity in HDI data, the PPI prediction performance of GLCPN is significantly higher than that of its peers. For example, considering the performance on *T-208964* shown in Table II, GLCPN achieves the RMSE at 0.1161, MAE at 0.0810, R^2 at 0.6030, and NDCG at 0.8676, which are better than the best comparison models by 2.03%, 3.23%, 4.21%, and 0.98%. GLCPN can consistently achieve the highest accuracy compared to seventeen advanced methods. These results clearly demonstrate the superior performance of GLCPN in PPI prediction tasks.

ii. Results of CP Prediction

In CP prediction, the proposed GLCPN also substantially outperforms other baselines and even achieves more accuracy gains compared to PPI prediction. As shown in Table III, GLCPN achieves the MAE at 0.1628 on *Astro-Ph* and 0.3200 on *Cond-Mat-2005*, which are 5.57% and 6.90% lower than the lowest MAE achieved by other methods. More evidently, the R^2 improvements on these two datasets reach 18.41% and 23.24%, suggesting that GLCPN is capable of better capturing the trends in incomplete collaborations, thereby enhancing the predictive performance. It is undeniable that the RMSE of GLCPN on *Cond-Mat* is inferior to GC-MC (Table SII), while its MAE is far superior. Hence, the stability of GLCPN in dealing with HDI data with outliers needs to be improved in future work. Overall, GLCPN's performance in CP prediction tasks is also top-notch compared with the state-of-the-art.

iii. Results of QoS Estimation

As Table IV demonstrates, to assess the performance of GLCPN across a wide range of density conditions, we construct four testing cases based on *Throughput*, following the practices of the service computing community [6], [17]. From the results, we observe that GLCPN can adapt to various densities and always outperforms the comparison models in all

TABLE IV
PERFORMANCE ON QOS ESTIMATION.

Method	*Throughput* (Den.=10%)				*Throughput* (Den.=20%)			
	RMSE ↓	**MAE ↓**	**R^2 ↑**	**NDCG ↑**	**RMSE ↓**	**MAE ↓**	**R^2 ↑**	**NDCG ↑**
SGL-ED	$0.4978_{\pm 1.5E-3}$	$0.2005_{\pm 1.9E-3}$	$0.7986_{\pm 1.2E-3}$	$0.6776_{\pm 2.9E-3}$	$0.4278_{\pm 1.4E-3}$	$0.1613_{\pm 4.2E-4}$	$0.8512_{\pm 9.5E-4}$	$0.7362_{\pm 1.8E-3}$
LightGCN$_{r\text{-}AdjNorm}$	$0.5129_{\pm 3.4E-4}$	$0.1860_{\pm 5.2E-4}$	$0.7862_{\pm 2.8E-4}$	$0.6256_{\pm 1.4E-3}$	$0.4726_{\pm 3.8E-5}$	$0.1726_{\pm 4.6E-4}$	$0.8185_{\pm 2.9E-5}$	$0.6711_{\pm 4.0E-4}$
HMLET	$0.4907_{\pm 3.9E-3}$	$0.1944_{\pm 3.4E-3}$	$0.8043_{\pm 3.1E-3}$	$0.6789_{\pm 2.8E-3}$	$0.4246_{\pm 1.2E-4}$	$0.1617_{\pm 8.6E-5}$	$0.8535_{\pm 8.2E-5}$	$0.7398_{\pm 3.0E-3}$
HCCF	$0.6006_{\pm 1.9E-2}$	$0.2503_{\pm 1.2E-2}$	$0.7065_{\pm 1.8E-2}$	$0.6022_{\pm 4.2E-3}$	$0.4660_{\pm 3.9E-3}$	$0.1901_{\pm 1.7E-3}$	$0.8235_{\pm 2.9E-3}$	$0.7192_{\pm 3.0E-3}$
GTN	$0.5512_{\pm 1.2E-3}$	$0.2331_{\pm 5.2E-4}$	$0.7531_{\pm 1.1E-3}$	$0.6190_{\pm 3.1E-3}$	$0.4564_{\pm 1.1E-3}$	$0.1821_{\pm 3.6E-4}$	$0.8307_{\pm 8.5E-4}$	$0.6869_{\pm 3.7E-3}$
CIGCN	$0.4972_{\pm 2.2E-3}$	$0.2029_{\pm 1.9E-3}$	$0.7876_{\pm 1.8E-3}$	$0.6623_{\pm 1.6E-3}$	$0.4251_{\pm 7.1E-4}$	$0.1614_{\pm 1.5E-3}$	$0.8411_{\pm 4.9E-4}$	$0.7328_{\pm 3.1E-3}$
JMP-GCF	$0.6594_{\pm 2.1E-2}$	$0.3377_{\pm 2.2E-2}$	$0.6462_{\pm 2.3E-2}$	$0.5868_{\pm 1.4E-2}$	$0.5747_{\pm 1.7E-3}$	$0.2499_{\pm 6.1E-4}$	$0.7315_{\pm 1.6E-3}$	$0.6355_{\pm 1.2E-3}$
LightGCL	$0.6626_{\pm 3.8E-3}$	$0.3453_{\pm 2.4E-3}$	$0.6431_{\pm 4.1E-3}$	$0.5429_{\pm 3.0E-3}$	$0.5169_{\pm 2.3E-3}$	$0.2494_{\pm 1.0E-3}$	$0.7828_{\pm 1.9E-3}$	$0.6671_{\pm 3.7E-3}$
MGDN	$0.4904_{\pm 2.8E-3}$	$0.1886_{\pm 1.1E-3}$	$0.8045_{\pm 2.2E-3}$	$0.6781_{\pm 2.4E-3}$	$0.4246_{\pm 1.2E-3}$	$0.1578_{\pm 7.2E-4}$	$0.8534_{\pm 8.6E-4}$	$0.7406_{\pm 7.0E-3}$
PopGo	$0.5400_{\pm 2.9E-2}$	$0.2213_{\pm 9.2E-3}$	$0.7623_{\pm 2.6E-2}$	$0.6592_{\pm 2.0E-2}$	$0.4424_{\pm 9.8E-4}$	$0.1650_{\pm 1.1E-3}$	$0.8409_{\pm 7.1E-4}$	$0.7181_{\pm 2.7E-3}$
GLCPN (Ours)	$\mathbf{0.4605_{\pm 5.1E-4}}$	$\mathbf{0.1588_{\pm 1.1E-4}}$	$\mathbf{0.8277_{\pm 3.8E-4}}$	$\mathbf{0.6928_{\pm 1.4E-3}}$	$\mathbf{0.4021_{\pm 3.9E-4}}$	$\mathbf{0.1365_{\pm 2.4E-4}}$	$\mathbf{0.8686_{\pm 2.6E-4}}$	$\mathbf{0.7486_{\pm 1.1E-3}}$
Method	*Throughput* (Den.=30%)				*Throughput* (Den.=40%)			
	RMSE ↓	**MAE ↓**	**R^2 ↑**	**NDCG ↑**	**RMSE ↓**	**MAE ↓**	**R^2 ↑**	**NDCG ↑**
SGL-ED	$0.3948_{\pm 1.1E-3}$	$0.1484_{\pm 3.5E-4}$	$0.8733_{\pm 7.3E-4}$	$0.7612_{\pm 2.7E-3}$	$0.3749_{\pm 6.8E-4}$	$0.1396_{\pm 5.9E-4}$	$0.8858_{\pm 4.1E-4}$	$0.7768_{\pm 2.6E-3}$
LightGCN$_{r\text{-}AdjNorm}$	$0.4641_{\pm 9.6E-5}$	$0.1704_{\pm 1.2E-3}$	$0.8249_{\pm 7.3E-5}$	$0.6811_{\pm 2.4E-4}$	$0.4609_{\pm 1.4E-4}$	$0.1694_{\pm 1.2E-3}$	$0.8273_{\pm 1.1E-4}$	$0.6842_{\pm 2.4E-4}$
HMLET	$0.3913_{\pm 1.3E-5}$	$0.1470_{\pm 3.3E-4}$	$0.8756_{\pm 8.0E-6}$	$0.7622_{\pm 2.0E-3}$	$0.3726_{\pm 1.5E-3}$	$0.1349_{\pm 1.1E-3}$	$0.8872_{\pm 8.9E-4}$	$0.7694_{\pm 3.4E-3}$
HCCF	$0.4222_{\pm 2.6E-3}$	$0.1684_{\pm 2.1E-3}$	$0.8551_{\pm 1.8E-3}$	$0.7528_{\pm 2.2E-3}$	$0.4110_{\pm 3.0E-3}$	$0.1613_{\pm 1.6E-3}$	$0.8627_{\pm 2.0E-3}$	$0.7634_{\pm 3.0E-3}$
GTN	$0.4243_{\pm 7.4E-4}$	$0.1651_{\pm 4.0E-4}$	$0.8536_{\pm 5.1E-4}$	$0.7214_{\pm 2.4E-3}$	$0.4090_{\pm 9.8E-4}$	$0.1553_{\pm 2.7E-4}$	$0.8641_{\pm 6.6E-4}$	$0.7310_{\pm 1.2E-3}$
CIGCN	$0.3926_{\pm 1.2E-3}$	$0.1470_{\pm 5.8E-4}$	$0.8446_{\pm 7.7E-4}$	$0.7381_{\pm 2.1E-3}$	$0.3731_{\pm 4.4E-4}$	$0.1375_{\pm 1.9E-4}$	$0.8456_{\pm 2.7E-4}$	$0.7360_{\pm 8.0E-4}$
JMP-GCF	$0.5828_{\pm 1.1E-2}$	$0.2538_{\pm 6.6E-3}$	$0.7238_{\pm 1.1E-2}$	$0.6387_{\pm 6.6E-3}$	$0.5878_{\pm 4.7E-3}$	$0.2584_{\pm 3.1E-3}$	$0.7191_{\pm 4.4E-3}$	$0.6436_{\pm 2.2E-3}$
LightGCL	$0.4557_{\pm 1.7E-3}$	$0.2048_{\pm 6.0E-4}$	$0.8312_{\pm 1.3E-3}$	$0.7225_{\pm 6.8E-3}$	$0.4209_{\pm 2.3E-3}$	$0.1791_{\pm 7.5E-4}$	$0.8560_{\pm 1.5E-3}$	$0.7481_{\pm 7.0E-3}$
MGDN	$0.3949_{\pm 6.0E-4}$	$0.1478_{\pm 4.2E-4}$	$0.8732_{\pm 3.9E-4}$	$0.7656_{\pm 2.0E-3}$	$0.3750_{\pm 8.9E-4}$	$0.1406_{\pm 3.3E-4}$	$0.8857_{\pm 5.4E-4}$	$0.7851_{\pm 1.8E-3}$
PopGo	$0.4087_{\pm 1.1E-3}$	$0.1500_{\pm 8.0E-4}$	$0.8642_{\pm 7.5E-4}$	$0.7393_{\pm 2.1E-3}$	$0.3865_{\pm 1.4E-3}$	$0.1376_{\pm 8.7E-4}$	$0.8786_{\pm 8.9E-4}$	$0.7571_{\pm 1.5E-3}$
GLCPN (Ours)	$\mathbf{0.3759_{\pm 2.3E-4}}$	$\mathbf{0.1256_{\pm 9.6E-5}}$	$\mathbf{0.8852_{\pm 1.4E-4}}$	$\mathbf{0.7706_{\pm 1.0E-3}}$	$\mathbf{0.3606_{\pm 1.5E-4}}$	$\mathbf{0.1190_{\pm 1.3E-4}}$	$\mathbf{0.8943_{\pm 8.7E-5}}$	$\mathbf{0.7881_{\pm 1.3E-3}}$

TABLE V
PERFORMANCE ON DTA PREDICTION.

Method	*Kiba*			
	RMSE ↓	**MAE ↓**	**R^2 ↑**	**NDCG ↑**
SGL-ED	$0.0232_{\pm 2.1E-4}$	$0.0140_{\pm 3.3E-5}$	$0.3043_{\pm 1.3E-2}$	$0.9674_{\pm 4.0E-5}$
LightGCN$_{r\text{-}AdjNorm}$	$0.0230_{\pm 7.4E-5}$	$0.0140_{\pm 4.6E-5}$	$0.3176_{\pm 4.4E-3}$	$0.9675_{\pm 3.9E-5}$
HMLET	$0.0213_{\pm 8.0E-4}$	$0.0146_{\pm 3.5E-4}$	$0.4127_{\pm 4.4E-2}$	$0.9683_{\pm 1.2E-4}$
HCCF	$0.0258_{\pm 1.5E-4}$	$0.0176_{\pm 8.5E-5}$	$0.1376_{\pm 1.0E-2}$	$0.9671_{\pm 1.3E-4}$
GTN	$0.0221_{\pm 1.3E-3}$	$0.0156_{\pm 7.8E-4}$	$0.3670_{\pm 7.4E-2}$	$0.9682_{\pm 8.4E-5}$
CIGCN	$0.0219_{\pm 4.6E-4}$	$0.0149_{\pm 4.0E-4}$	$0.3822_{\pm 2.6E-2}$	$0.9680_{\pm 1.3E-4}$
JMP-GCF	$0.0228_{\pm 1.5E-4}$	$0.0153_{\pm 1.1E-4}$	$0.3266_{\pm 8.7E-3}$	$0.9672_{\pm 6.8E-5}$
LightGCL	$0.0225_{\pm 1.2E-3}$	$0.0136_{\pm 3.6E-4}$	$0.3441_{\pm 7.1E-2}$	$0.9684_{\pm 1.5E-4}$
MGDN	$0.0222_{\pm 8.7E-5}$	$0.0154_{\pm 8.4E-5}$	$0.3641_{\pm 5.0E-3}$	$0.9679_{\pm 5.0E-5}$
PopGo	$0.0214_{\pm 3.2E-5}$	$0.0141_{\pm 2.1E-5}$	$0.4106_{\pm 1.8E-3}$	$0.9665_{\pm 3.3E-5}$
GLCPN (Ours)	$\mathbf{0.0197_{\pm 1.2E-4}}$	$\mathbf{0.0124_{\pm 5.2E-5}}$	$\mathbf{0.5001_{\pm 5.9E-3}}$	$\mathbf{0.9689_{\pm 3.3E-5}}$

metrics. It is also worth noting that GLCPN may achieve more significant performance gains under sparser conditions. Taking the RMSE as an example, as the data density decreases from 40% to 10%, the performance improvements of GLCPN are observed to be 3.22%, 3.94%, 5.30%, and 6.10%, respectively. On other metrics, the achieved outcomes are similar. Considering that in real-world scenarios, various data, including QoS data, are highly incomplete. Thus, this characteristic of GLCPN holds positive significance.

iv. Results of DTA Prediction

As the notable results on *Kiba* shown in Table V, GLCPN exhibits superior performance over other approaches in the task of DTA prediction. Specifically, with the RMSE of 0.0197, MAE of 0.0124, and R^2 of 0.5001, the proposed GLCPN outperforms the second-best methods by 7.51%, 8.82%, and 17.48%. Additionally, GLCPN outperforms other comparison models when evaluated by NDCG. It should be noted that the absolute differences in drug-protein affinity scores in *Kiba* are generally small, resulting in the improvement of GLCPN not being as substantial as RMSE, MAE, and R^2. In summary, GLCPN is more effective in DTA prediction tasks than state-of-the-art methods.

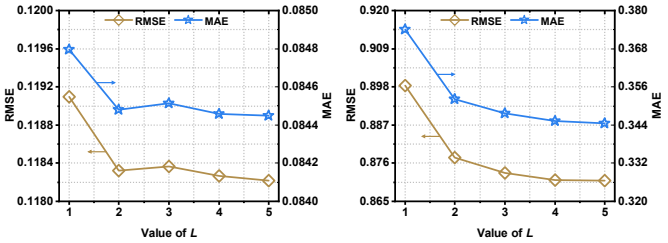
C. Sensitivity Analysis of Hyperparameters

Considering the unique hyperparameters of the proposed GLCPN, including L, K, and α, their settings might influence its performance. Hence, we conduct sensitivity tests and depict corresponding results in Figs. S2-S7 of SF. Here, we show representative results in Figs. 5-10. Based on them, we have the following findings.

First, explicitly capturing high-order neighborhood information significantly boosts the representation ability of the proposed GLCPN, i.e., the estimation accuracy increases with a higher L. For instance, as plotted in Fig. 5(a), on *T-9796*, as L increases from 1 to 5 while other hyperparameters are fixed, the RMSE of GLCPN decreases from 0.1191 to 0.1182 and the MAE decreases from 0.0848 to 0.0845. Similar situations can be found in other cases (Fig. S2), indicating the importance of high-order connectivity. In general, GLCPN achieves the best performance when L is set to 3 to 5.

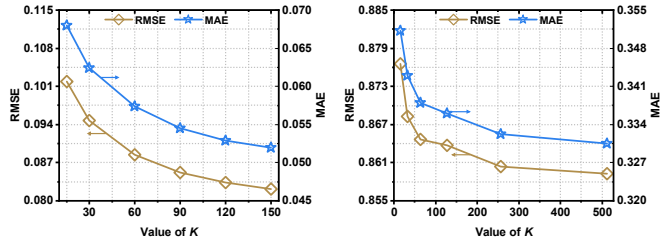
Second, increasing K vastly improves the estimation accuracy of GLCPN. Taking the testing case *T-9544* as an example (Fig. 6(a)), when K increases from 15 to 150, the corresponding RMSE decreases from 0.1019 to 0.0822 with the estimation error gap of 19.33%. When MAE is considered, it decreases from 0.0680 to 0.0520, where the estimation error gap is achieved at 23.53%. Generally, when K is set around 100, GLCPN can achieve a robust performance in most cases.

Third, appropriate settings of α are critical for GLCPN to obtain high estimation accuracy. For instance, as depicted in Fig. 7(a), on *T-9544*, as α varies from 0.04 to 0.12, the RMSE decreases from 0.1112 to 0.0815 with the error gap of 26.71%, and the MAE decreases by 28.40%. When α varies from 0.12 to 0.3, the RMSE of GLCPN increases from 0.0815 to 0.0876 with the RMSE increment of 6.96% and the MAE increases by



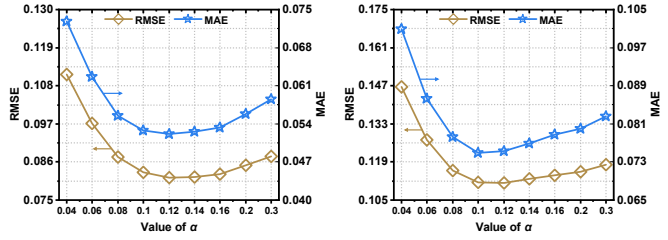
(a) Errors on *T-9796* (b) Errors on *Cond-Mat-2005*

Fig. 5. Errors of GLCPN as L varies.



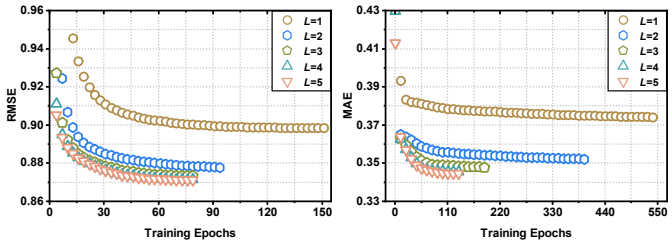
(a) Errors on *T-9544* (b) Errors on *Cond-Mat-2005*

Fig. 6. Errors of GLCPN as K varies.



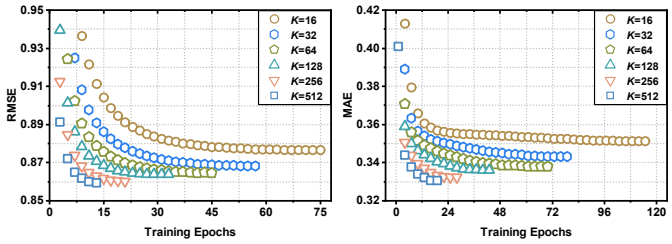
(a) Errors on *T-9796* (b) Errors on *T-227321*

Fig. 7. Errors of GLCPN as α varies.



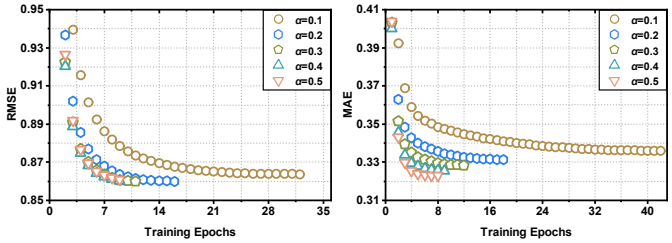
(a) RMSE curves (b) MAE curves

Fig. 8. Convergence curves of GLCPN as L varies.



(a) RMSE curves (b) MAE curves

Fig. 9. Convergence curves of GLCPN as K varies.



(a) RMSE curves (b) MAE curves

Fig. 10. Convergence curves of GLCPN as α varies.

10.77%. In most cases, α can be easily set as 0.1 to obtain a satisfactory performance.

It is also worth noting that the convergence rate of the proposed approach is greatly influenced by the settings of L, K, and α. Due to the increased complexity with the rise in L and K, GLCPN more rapidly captures the patterns specified in the target HDI data. Additionally, as L increases, each node receives messages from more distant neighbors, facilitating faster information propagation across the network. Moreover, an increase of α may result in smaller weight updates, allowing the model to achieve a stable state more quickly. Hence, as depicted in Figs. 8-10, S2, S4, and S6, the convergence rate of GLCPN decreases with the increase of L, K, and α in most testing cases. For instance, considering the RMSE on *Cond-Mat-2005*, as L increases from 1 to 5 (Fig. 8(a)), the epochs decrease from 149 to 77; as K increases from 16 to 512 (Fig. 9(a)), the training epochs decrease from 73 to 12; and as α increases from 0.1 to 0.5 (Fig. 10(a)), the converging epochs decrease from 150 to 11.

D. Ablation and Effectiveness Analysis

Aiming at investigating the effectiveness of the proposed fundamental components in GLCPN, i.e., the linear message propagation, the priori convolution operator, and the locality-enhanced pooling, we conduct a series of ablation studies and document the results with care. Tables VI-IX present a part of the results regarding RMSE, and more results are appended to Tables SVII-SXIII of SF.

i. Module Ablation Study

To validate the significance of the proposed modules and their combinations, we first conduct a module ablation study. Specifically, seven variants of GLCPN are obtained by removing different components while retaining the others, and subsequently their performance along with GLCPN on four tasks is tested. The corresponding results have been listed in Tables VI and SVII. It can be observed from the experimental results that GLCPN achieves the lowest errors in all testing cases, which demonstrates that each component of GLCPN is indispensable for its high representation learning performance on HDI data. For example, on *Throughput* (Den.=10%), the RMSE of GLCPN is 0.4588, which is substantially lower than 0.4637, 0.4666, 0.6311, 0.4772, 0.5075, 0.6974, and 0.6270 achieved by its variants. Besides, each module of GLCPN may not function independently, showing that their combined effects pose considerable impacts on its performance. This dependency is partially due to the fact that they are designed cumulatively for the commonly encountered problems in representation learning in HDI data [32], [45]-[46].

ii. Impact of Linear Message Propagation

For evaluating the effects of linear message propagation, we instantiate several variants of GLCPN, including GLCPN-MF (i.e., the MF model), which omits the graph convolutional operation, GLCPN-A that incorporates nonlinearities, GLCPN-T that implements the feature transformation, and GLCPN-A&T that adopts both. Their training and validation

TABLE VI
RESULTS OF MODULE ABLATIONS.

Method	*T-208964*	*T-227321*	*Astro-Ph*	*Cond-Mat-2005*	*Throughput*	*Kiba*
GLCPN-w/o-L	0.1192±2.7E-4	0.1172±2.8E-4	0.4604±6.2E-4	0.8828±1.2E-3	0.4637±1.4E-3	0.0209±1.8E-4
GLCPN-w/o-P	0.1179±2.8E-5	0.1152±3.4E-5	0.4493±5.6E-4	0.8723±9.4E-4	0.4666±6.3E-4	0.0203±1.7E-4
GLCPN-w/o-H	0.1648±7.2E-5	0.1564±1.5E-5	0.4502±5.4E-4	0.8686±1.2E-3	0.6311±7.4E-4	0.0507±1.2E-4
GLCPN-w/o-L&P	0.1182±1.8E-4	0.1179±1.6E-4	0.4564±8.2E-4	0.8809±1.1E-3	0.4772±5.0E-3	0.0210±3.7E-4
GLCPN-w/o-L&H	0.1497±6.0E-4	0.1407±1.5E-3	0.4480±9.9E-4	0.8695±9.7E-4	0.5075±9.2E-4	0.0209±6.4E-4
GLCPN-w/o-P&H	0.1816±1.8E-5	0.1775±5.3E-5	0.4680±2.2E-4	0.8878±7.3E-4	0.6974±3.5E-4	0.0504±2.5E-4
GLCPN-w/o-L&P&H	0.1570±3.9E-4	0.1489±5.8E-4	0.4504±8.7E-4	0.8705±5.2E-4	0.6270±2.1E-2	0.0206±3.8E-4
GLCPN	**0.1163±3.4E-5**	**0.1141±8.5E-5**	**0.4443±1.2E-3**	**0.8633±5.2E-4**	**0.4588±9.7E-4**	**0.0202±2.7E-4**

L stands for the linear message propagation; P stands for the priori convolution operator; H stands for the locality-enhanced holistic pooling; and w/o is the abbreviation of "without".

TABLE VII
ABLATION RESULTS REGARDING LINEAR MESSAGE PROPAGATION.

Method	*T-208964*		*Cond-Mat-2005*	
	Training Error	Validation Error	Training Error	Validation Error
GLCPN-MF	**0.0701±7.6E-4**	0.1195±2.2E-4	**0.4985±4.0E-2**	0.8859±5.7E-2
GLCPN-A	0.0873±8.0E-5	0.1125±4.2E-4	0.7624±2.3E-2	0.8676±5.8E-2
GLCPN-T	0.1009±2.5E-2	0.1284±9.8E-3	0.8054±1.3E-2	0.8694±5.6E-2
GLCPN-A&T	0.0752±1.3E-3	0.1141±2.7E-4	0.7741±6.3E-2	0.8612±5.8E-2
GLCPN	0.0866±9.3E-5	**0.1119±3.6E-4**	0.6009±3.5E-2	**0.8388±5.7E-2**

MF stands for matrix factorization, i.e., no graph convolution; A stands for the nonlinear activation; and T stands for the feature transformation.

TABLE VIII
ABLATION RESULTS REGARDING PRIORI CONVOLUTION OPERATOR.

Method	*Cond-Mat-2005*			
	Validation Error	Training Epochs	Time Cost Per Epoch	Total Time Cost
GLCPN-B	0.8502±5.8E-2	17±1.96	2.13±0.01	37±4.10
GLCPN-SA	0.8630±5.6E-2	10±2.50	6.85±0.20	71±16.57
GLCPN	**0.8388±5.7E-2**	**10±1.47**	**2.13±0.01**	**21±3.17**

B stands for the binary adjacency matrix; and SA stands for the self-attention mechanism.

TABLE IX
ABLATION RESULTS REGARDING LOCALITY-ENHANCED POOLING.

Method	*T-208964*	*T-227321*	*Astro-Ph*	*Cond-Mat-2005*
GLCPN-S	0.1677±8.5E-4	0.1593±6.8E-4	0.4418±7.4E-3	0.8593±5.8E-2
GLCPN-SS	0.1168±3.0E-4	0.1161±5.8E-4	0.4443±9.2E-3	0.8593±5.4E-2
GLCPN-M	0.1128±2.3E-4	0.1139±4.8E-4	0.4297±7.7E-3	0.8396±5.7E-2
GLCPN-C	0.1200±2.9E-4	0.1191±4.9E-4	0.4378±8.8E-3	0.8492±5.8E-2
GLCPN	**0.1119±3.6E-4**	**0.1112±4.3E-4**	**0.4273±8.1E-3**	**0.8388±5.7E-2**

S stands for that a model only outputs the single final layer; SS stands for that a model only outputs the final layer but with self-loop message propagation; M stands for that a model outputs the mean of all the layers; and C stands for that a model concatenates the feature transformation.

errors are recorded in Tables VII, SVIII, and SIX. As shown in these tables, the linear message propagation substantially improves the estimation accuracy of GLCPN. It is also observed that nonlinear activation and feature transformation impose negative effects on the generalization performance, i.e., both are detrimental due to the training difficulty. On the other hand, the training error of GLCPN-MF is far lower than that of GLCPN in most cases, and the validation error is versa. This indicates that linear message propagation plays a similar role as graph regularization to mitigate the overfitting effect faced by MF models [41], thus improving the representation learning ability in HDI data.

iii. Impact of Priori Convolution Operator

Considering the priori convolution operator, we implement GLCPN-B by using the binary adjacency matrix and GLCPN-SA by adaptively learning the importance of two connected nodes based on the self-attention mechanism [20], [30], which is introduced in Sec. II.A.ii of SF. The validation errors, training epochs, time cost per epoch, and total time cost are summarized in Tables VIII, SX, and SXI. As expected, the validation errors of GLCPN are far lower than those of GLCPN-B, demonstrating that the priori convolution operator significantly enhances its learning capability. As for GLCPN-SA, its learning process for attention increases the difficulty of training in obtaining desired node features, thus resulting in accuracy loss. Moreover, adaptively learning the importance between nodes is inevitably more time-consuming than using the priori graph convolutions. Thus, when capturing the neighborhood information in the latent feature analysis for HDI data, using the priori convolution operator to distinguish the importance of neighbors is more applicable.

iv. Impact of Locality-Enhanced Pooling

In terms of the locality-enhanced pooling strategy, we validate its positive impacts compared to the variants that only output the single final layer without and with self-loops (GLCPN-S and GLCPN-SS), and the variants that adopt the mean and concatenation of all the layers as output (GLCPN-M and GLCPN-C). The ablation results are presented in Tables IX, SXII, and SXIII. From the tables, we observe that GLCPN achieves lower errors than GLCPN-S and GLCPN-SS without pooling, revealing that the performance of GLCPN evidently benefits from its equipped pooling method. Besides, owing to the fact that GLCPN enhances the impacts of local features, it yields more accuracy improvement than conventional pooling methods do. For instance, on *T-227321*, GLCPN achieves the RMSE at 0.1112 (Table IX), which is respectively 30.19%, 4.22%, 2.37%, and 6.63% lower than those achieved by GLCPN-S, GLCPN-SS, GLCPN-M, and GLCPN-C. We can find similar outcomes in other test cases, which solidly verify the effectiveness of the locality-enhanced pooling.

E. Summary

Based on the presented results and analysis, we summarize the following pros and cons of GLCPN.

Pros: First, GLCPN models the node-node collaboration and the salience of nodes themselves, thereby achieving high estimation accuracy for the missing data in the HDI matrix. Second, GLCPN simplifies the model structure and removes unnecessary complexity, thus possessing competitively high computational efficiency. Third, the linear structure is easily

deployed and followed.

Cons: GLCPN is GPU-dependent, and its hyperparameters need careful tuning like other graph neural network-based methods [27], [32]-[35].

VI. Background and Related Work

This section briefly reviews two topics that are closely related to the proposed GLCPN, including latent feature analysis and graph neural networks.

A. Latent Feature Analysis

Approaches pertaining to LFA are acknowledged as potent techniques for understanding HDI data. They have been widely used for diverse data analyzing tasks, such as PPI prediction [1], [2], co-authorship analysis [5], [42], cloud service selection [6], [17], and drug discovery [37], [43]. An LFA model is only built on the known entries of an HDI matrix (i.e., the weighted interactions in a target network) and commonly adopts MF to reconstruct its low-rank approximation [2], [4], [6]-[13]. During this process, the resultant latent features are learned as effective representations of nodes, determining the performance of downstream tasks.

Existing LFA methods can generally be bifurcated into two categories based on whether nonlinear modeling is applied to HDI data. The linear methodology encompasses a diverse range of methods, including double factorization-based symmetric and nonnegative LFA [4], joint nonnegative MF [9], multilayer-structured prediction-sampling-based model [10], Alternating-Direction-Method of Multipliers (ADMM)-integrated model [2], block-diagonal guided symmetric LFA [11], implicit correlation-regularized probabilistic MF [12], and data-characteristic-aware LFA [6]. On the other hand, nonlinear examples include neural MF [8], light convolutional autoencoder-based LFA [13], fast nonnegative autoencoder-based one [50], outer product-incorporated neural LFA [14], deep variational MF-based model [15], federated learning-incoporated meta LFA [16], neighborhood-integrated deep MF [17], and collaborative denoising autoencoder-based LFA [18].

***Remark* 1**. These LFA-based approaches designed towards HDI data describing weighted networks have been proven effective [2], [4], [6]-[18], but they have scarcely devoted attention to capturing the high-order connectivity within HDI data. Moreover, previous research has substantiated that in many scenarios, due to the difficulties in training complex models, nonlinear models often perform inferiorly compared to their linear peers in both accuracy and efficiency [2], [4], [6]. Our proposed GLCPN model explores the capture of high-order connectivity in HDI data through linear message propagation, thereby filling a gap in this research field.

B. Graph Neural Networks

GNNs have emerged as powerful tools for processing non-Euclidean data such as graphs [19]-[29]. By leveraging the interaction relationships among nodes, GNNs are able to capture complex patterns that traditional neural networks may overlook. Typically, they operate by propagating information across the graph structure, enabling each node to gather information from its neighbors. GNNs have proven effective in a variety of applications, including text classification [19], social network analysis [20], molecule classification [51], and conversation generation [36].

A Series of advanced GNNs, e.g., graph convolutional networks [19], graph attention networks [20], [30], hyperbolic GNNs [21], non-local GNNs [22], PageRank-based GNNs [34], augmentation and sampling GNNs [52], label-enhanced GNNs [53], cluster-aware GNNs [54], polarized message-passing GNNs [36], and hypergraph neural networks [23] have been proposed. Several efforts incorporate MF into GNNs, aiming at enhancing the learning capabilities of node representations. Representative approaches include neural graph collaborative filtering [24], graph convolutional matrix completion [25], graph convolutional network-based MF [26], light graph convolutional network [27], Markov graph diffusion network [47], hypergraph contrastive collaborative filtering [28], light graph contrastive learning model [31], channel-independent graph convolutional network [46], and graph trend filtering network [29].

***Remark* 2**. Incorporating GNNs into the process of LFA may well handle the node feature dependencies, but existing approaches overlook the inherent variability in connection strength among nodes. Therefore, to better cope with highly sparse weighted networks, GLCPN introduces a novel priori convolution operator, which essentially learns informative features within an HDI matrix in an autoencoder form. Besides, we recognize that when combining GNNs with MF, the individuality of nodes and local similarity between nodes can be weakened, resulting in a loss of expressivity and increased training difficulties. To address this critical issue, locality-enhanced pooling is additionally proposed to further enhance its LFA performance, which is significantly different from existing methods and expected to facilitate the performance of GNNs in other tasks [36], [51].

To summarize, the proposed GLCPN addresses the critical challenges in MF- and GNN-based methods by appropriately tailoring the coalescence of linear message propagation, priori convolution, and locally enhanced pooling for representation learning in HDI data. To the best of our knowledge, GLCPN is the first attempt to simultaneously consider the previously mentioned modules for learning meaningful representations in HDI data for various real-world tasks. The empirical results presented in this paper have demonstrated the notable performance of GLCPN, indicating that the coalescence of the previously mentioned modules can significantly enhance representation learning in HDI data.

VII. Conclusions

This paper presents a novel LFA model, namely GLCPN, to learn expressive representations in HDI data. Unlike existing methods, it proposes an efficient linear message propagation rule to extract the information carried by high-order neighborhood, where the priori convolution operator and locality-enhanced pooling mechanism further improve its performance. For the first time, we theoretically analyze the expressiveness of the GNN-incorporated LFA model (i.e., GLCPN), providing more insights into designing effective approaches to representation learning in HDI data. Extensive comparisons have been conducted between the proposed GLCPN and state-of-the-art approaches on ten large-scale

industrial HDI datasets. The notable results obtained demonstrate that the proposed GLCPN has higher estimation accuracy and computational efficiency.

In future work, we will consider improving attention mechanisms, introducing transfer learning, and integrating pre-trained models as measures to address the issues of informational deficiency and uneven interaction strength in HDI data. Besides, the study of temporal graph representation learning has garnered significant attention in recent years [55]-[57]. It is worth considering enhancing the proposed GLCPN to construct effective dynamic GNNs and developing pooling mechanisms tailored to dynamic graph representation learning.

Acknowledgments

This work was supported by the National Key Research and Development Program of China under Grant 2018AAA0102002, the National Natural Science Foundation of China under Grants U20B2064 and U21B2043, and the CAAI-Huawei MindSpore Open Fund under Grant CAAIXSJLJJ-2021-035A.

References

[1] D. Szklarczyk, A. L. Gable, D. Lyon, A. Junge, S. Wyder, J. HuertaCepas, M. Simonovic, N. T. Doncheva, J. H. Morris, P. Bork, L. J. Jensen, and C. Mering, "STRING v11: protein-protein association networks with increased coverage, supporting functional discovery in genome-wide experimental datasets," *Nucleic Acids Research*, vol. 47, pp. D607-D613, 2019.

[2] X. Luo, Y. Zhong, Z. Wang, and M. Li, "An alternating-direction-method of multipliers-incorporated approach to symmetric non-negative latent factor analysis," *IEEE Trans. on Neural Networks and Learning Systems*, vol. 34, no. 8, pp. 4826-4840, 2023.

[3] H. Roghani, and A. Bouyer, "A fast local balanced label diffusion algorithm for community detection in social networks," *IEEE Trans. on Knowledge and Data Engineering*, vol. 35, no. 6, pp. 5472-5484, 2023.

[4] M. Li, and Y. Song, "An improved non-negative latent factor model for missing data estimation via extragradient-based alternating direction method," *IEEE Trans. on Neural Networks and Learning Systems*, vol. 34, no. 9, pp. 5640-5653, 2023.

[5] M. E. J. Newman, "Fast algorithm for detecting community structure in networks," *Physical Review E*, vol. 69, no. 6, pp. 066133, 2004.

[6] D. Wu, X. Luo, M. Shang, Y. He, G. Wang, and X. Wu, "A data-characteristic-aware latent factor model for web services QoS prediction," *IEEE Trans. on Knowledge and Data Engineering*, vol. 34, no. 6, pp. 2525-2538, 2022.

[7] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proc. of the 26th Int. Conf. on World Wide Web*, Perth, Australia, 2017, pp. 173-182.

[8] T. Kong, T. Kim, J. Jeon, J. Choi, Y.-C. Lee, N. Park, and S.-W. Kim, "Linear, or non-linear, that is the question," in *Proc. of the 15th ACM Int. Conf. on Web Search and Data Mining*, Tempe, USA, 2022, pp. 517-525.

[9] X. Ma, D. Dong, and Q. Wang, "Community detection in multi-layer networks using joint nonnegative matrix factorization," *IEEE Trans. on Knowledge and Data Engineering*, vol. 31, no. 2, pp. 273-286, 2019.

[10] D. Wu, X. Luo, Y. He, and M. Zhou, "A prediction-sampling-based multilayer-structured latent factor model for accurate representation of high-dimensional and sparse data," *IEEE Trans. on Neural Networks and Learning Systems*, vol. 35, no. 3, pp. 3845-3858, 2024.

[11] Y. Qin, G. Feng, Y. Ren, and X. Zhang, "Block-diagonal guided symmetric nonnegative matrix factorization," *IEEE Trans. on Knowledge and Data Engineering*, vol. 35, no. 3, pp. 2313-2325, 2023.

[12] L. Yao, X. Wang, Q. Z. Sheng, B. Benatallah, and C. Huang, "Mashup recommendation by regularizing matrix factorization with API co-invocations," *IEEE Trans. on Services Computing*, vol. 14, no. 2, pp. 502-515, 2021.

[13] X. Li, K. Xie, X. Wang, G. Xie, K. Li, J. Cao, D. Zhang, and J. Wen, "A light-weight and robust tensor convolutional autoencoder for anomaly detection," *IEEE Trans. on Knowledge and Data Engineering*, vol. 36, no. 9, pp. 4346-4360, 2024.

[14] X. He, X. Du, X. Wang, F. Tian, J. Tang, and T.-S. Chua, "Outer product-based neural collaborative filtering," in *Proc. of the 27th Int. Joint Conf. on Artificial Intelligence*, Stockholm, Sweden, 2018, pp. 2227-2233.

[15] X. Shen, B. Yi, H. Liu, W. Zhang, Z. Zhang, S. Liu, and N. Xiong, "Deep variational matrix factorization with knowledge embedding for recommendation system," *IEEE Trans. on Knowledge and Data Engineering*, vol. 33, no. 5, pp. 1906-1918, 2021.

[16] Y. Lin, P. Ren, Z. Chen, Z. Ren, D. Yu, Jun Ma, M. Rijke, and X. Cheng, "Meta matrix factorization for federated rating predictions," in *Proc. of the 43rd Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, Xi'an, China, 2020, pp. 981-990.

[17] G. Zou, J. Chen, Q. He, K. Li, B. Zhang, and Y. Gan, "NDMF: neighborhood-integrated deep matrix factorization for service QoS prediction," *IEEE Trans. on Network and Service Management*, vol. 17, no. 4, pp. 2717-2730, 2020.

[18] Y. Wu, C. DuBois, A. Zheng, and M. Ester, "Collaborative denoising auto-encoders for top-N recommender systems," in *Proc. of the 9th ACM Int. Conf. on Web Search and Data Mining*, San Francisco, USA, 2016, pp. 153-162.

[19] T. N. Kipf, and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. of the 5th Int. Conf. on Learning Representations*, Toulon, France, 2017.

[20] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *Proc. of the 6th Int. Conf. on Learning Representations*, Vancouver, Canada, 2018.

[21] A. Vyas, N. Choudhary, M. Khatir, and C. K. Reddy, "GraphZoo: a development toolkit for graph neural networks with hyperbolic geometries," in *Proc. of the 31st Web Conf.*, Lyon, France, 2022, pp. 184-188.

[22] M. Liu, Z. Wang, and S. Ji, "Non-local graph neural networks," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 44, no. 12, pp. 10270-10276, 2022.

[23] Y. Gao, Y. Feng, S. Ji, and R. Ji, "HGNN$^+$: general hypergraph neural networks," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 45, no. 3, pp. 3181-3199, 2023.

[24] X. Wang, X. He, M. Wang, F. Feng, and T.-S. Chua, "Neural graph collaborative filtering," in *Proc. of the 42nd Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, Paris, France, 2019, pp. 165-174.

[25] R. Berg, T. N. Kipf, and W. Max, "Graph convolutional matrix completion," in *Proc. of the 24th ACM SIGKDD Int. Conf. on Knowledge Discovery & Data Mining*, London, UK, 2018.

[26] P. Han, P. Yang, P. Zhao, S. Shang, Y. Liu, J. Zhou, X. Gao, and P. Kalnis, "GCN-MF: disease-gene association identification by graph convolutional networks and matrix factorization," in *Proc. of the 25th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, Anchorage, USA, 2019, pp. 705-713.

[27] X. He, K. Deng, X. Wang, Y. li, Y. Zhang, and M. Wang, "LightGCN: simplifying and powering graph convolution network for recommendation," in *Proc. of the 43rd Int. ACM SIGIR conf. on research and development in Information Retrieval*, Xi'an, China, 2020, pp. 639-648.

[28] L. Xia, C. Huang, Y. Xu, J. Zhao, D. Yin, and J. X. Huang, "Hypergraph contrastive collaborative filtering," in *Proc. of the 45th Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, Madrid, Spain, 2022, pp. 70-79.

[29] W. Fan, X. Liu, W. Jin, X. Zhao, J. Tang, and Q. Li, "Graph trend filtering networks for recommendation," in *Proc. of the 45th Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, Madrid, Spain, 2022, pp. 112-121.

[30] T. He, Y. S. Ong, and L. Bai, "Learning conjoint attentions for graph neural nets," in *Proc. of the 34th Annual Conf. on Neural Information Processing Systems*, Virtual Event, 2021, pp. 2641-2653.

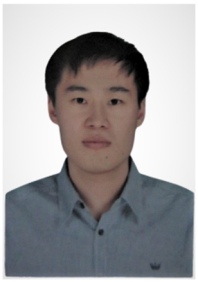
[31] X. Cai, C. Huang, L. Xia, and Xubin Ren, "LightGCL: simple yet effective graph contrastive learning for recommendation," in *Proc. of the 11th Int. Conf. on Learning Representations*, Kigali, Rwanda, 2023.

[32] W. Guo, Y. Yang, Y. Hu, C. Wang, H. Guo, Y. Zhang, R. Tang, W. Zhang, and X. He, "Deep graph convolutional networks with hybrid normalization for accurate and diverse recommendation", in *Proc. of 3rd Workshop on Deep Learning Practice for High-Dimensional Sparse Data with KDD*, Virtual Event, China, 2021.

[33] K. Liu, F. Xue, X. He, D. Guo, and R. Hong, "Joint multi-grained

popularity-aware graph convolution collaborative filtering for recommendation," *IEEE Trans. on Computational Social Systems*, vol. 10, no. 1, pp. 72-83, 2023.
[34] J. Klicpera, A. Bojchevski, and S. Günnemann, "Predict then propagate: graph neural networks meet personalized PageRank," in *Proc. of the 7th Int. Conf. on Learning Representations*, New Orleans, USA, 2019.
[35] L. Chen, L. Wu, R. Hong, K. Zhang, and M. Wang, "Revisiting graph based collaborative filtering: a linear residual graph convolutional network approach," in *Proc. of the 34th AAAI Conf. on Artificial Intelligence*, New York, USA, 2020, pp. 27-34.
[36] T. He, Y. Liu, Y.S. Ong, X. Wu, and X. Luo, "Polarized message-passing in graph neural networks," *Artificial Intelligence*, vol. 331, pp. 104-129, 2024.
[37] M. Kalemati, M. Zamani Emani, and S. Koohi, "BiComp-DTA: Drug-target binding affinity prediction through complementary biological-related and compression-based featurization approach," *PLOS Computational Biology*, vol. 19, no. 3, pp. e1011036, 2023.
[38] J. Wu, X. Wang, F. Feng, X. He, L. Chen, J. Lian, and X. Xie, "Self-supervised graph learning for recommendation," in *Proc. of the 44th Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, Virtual Event, Canada, 2021, pp. 726-735.
[39] K. Xu, W. Hu, J. Leskovec, S. Jegelka, "How powerful are graph neural networks," in *Proc. of the 7th Int. Conf. on Learning Representations*, New Orleans, USA, 2019.
[40] Waïss Azizian, and Marc Lelarge, "Expressive power of invariant and equivariant graph neural networks," in *Proc. of the 9th Int. Conf. on Learning Representations*, Virtual Event, Austria, 2021.
[41] M. Zhu, X. Wang, C. Shi, H. Ji, and P. Cui, "Interpreting and unifying graph neural networks with an optimization framework," in *Proc. of the 30th ACM Web Conf.*, Ljubljana, Slovenia, 2021, pp. 1215-1226.
[42] T. A. Davis, and Y. Hu. "The University of Florida sparse matrix collection," *ACM Trans. on Mathematical Software*, vol. 38, no. 1, 2011.
[43] J. Tang, A. Szwajda, S. Shakyawar, T. Xu, P. Hintsanen, K. Wennerberg, and T. Aittokallio, "Making sense of large-scale kinase inhibitor bioactivity data sets: a comparative and integrative analysis," *Journal of Chemical Information and Modeling*, vol. 54, no. 3, pp. 735-743, 2014.
[44] A. Kroll, Y. Rousset, X. Hu, N. A. Liebrand, and M. J. Lercher, "Turnover number predictions for kinetically uncharacterized enzymes using machine and deep learning," *Nature Communications*, vol. 14, no. 1, pp. 4139, 2023.
[45] M. Zhao, L. Wu, Y. Liang, L. Chen, J. Zhang, Q. Deng, K. Wang, X. Shen, T. Lv, and R. Wu, "Investigating accuracy-novelty performance for graph-based collaborative filtering," in *Proc. of the 45th Int. ACM SIGIR Conf. on Research and Development in Information Retrieval*, Madrid, Spain, 2022, pp. 50-59.
[46] T. Zhu, L. Sun, and G. Chen, "Embedding disentanglement in graph convolutional networks for recommendation," *IEEE Trans. on Knowledge and Data Engineering*, vol. 35, no. 1, pp. 431-442, 2023.
[47] J. Hu, B. Hooi, S. Qian, Q. Fang, and C. Xu, "MGDCF: distance learning via Markov graph diffusion for neural collaborative filtering," *IEEE Trans. on Knowledge and Data Engineering*, vol. 36, no. 7, pp. 3281-3296, 2024.
[48] A. Zhang, W. Ma, J. Zheng, X. Wang, and T.-S. Chua, "Robust collaborative filtering to popularity distribution shift," *ACM Trans. on Information Systems*, vol. 42, no. 3, pp. 1-25, 2024.
[49] X. Glorot, and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. of the 13th Int. Conf. on Artificial Intelligence and Statistics,* Sardinia, Italy, 2010, pp. 249-256.
[50] F. Bi, T. He, and X. Luo, "A fast nonnegative autoencoder-based approach to latent feature analysis on high-dimensional and incomplete data," *IEEE Trans. on Services Computing*, vol. 17, no. 3, pp. 733-746, 2024.
[51] X. Jiang, R. Zhu, P. Ji, and S. Li, "Co-embedding of nodes and edges with graph neural networks," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 45, no. 6, pp. 7075-7086, 2023.
[52] H. Zhou, W. Huang, Y. Chen, T. He, G. Cong, and Y. S. Ong, "Road network representation learning with the third law of geography," *arXiv preprint arXiv:2406.04038*, 2024.
[53] L. Yu, L. Sun, B. Du, T. Zhu, and W. Lv, "Label-enhanced graph neural network for semi-supervised node classification," *IEEE Trans. on Knowledge and Data Engineering*, vol. 35, no. 11, pp. 11529-11540, 2023.
[54] H. Zhou, T. He, Y.-S. Ong, G. Cong, and Q. Chen, "Differentiable clustering for graph attention," *IEEE Trans. on Knowledge and Data Engineering*, vol. 36, no. 8, pp. 3751-3764, 2024.
[55] E. Rossi, B. Chamberlain, F. Frasca, D. Eynard, F. Monti, and M. Bronstein, "Temporal graph networks for deep learning on dynamic graphs," *arXiv preprint arXiv:2006.10637*, 2020.
[56] M. Liu, Y. Liu, K. Liang, S. Wang, S. Zhou, and X. Liu, "Deep temporal graph clustering," in *Proc. of the 12th Int. Conf. on Learning Representations*, Vienna, Austria, 2024.
[57] M Liu, K Liang, Y Zhao, W Tu, S Zhou, X Gan, X Liu, and K He, "Self-supervised temporal graph learning with temporal and structural intensity alignment," *IEEE Trans. on Neural Networks and Learning Systems*, DOI: 10.1109/TNNLS.2024.3386168, 2024.

Fanghui Bi received the B.S. degree in Software Engineering from the Northeastern University, Shenyang, China, in 2020. He is currently pursuing his Ph.D. degree in computer science at the University of Chinese Academy of Sciences, Beijing, China. His research interests include Big Data Analytics and Graph Neural Networks.



Tiantian He (*Member*, IEEE) received the Ph.D. degree in Computer Science from the Hong Kong Polytechnic University in 2017. Currently, He is a Senior Scientist at Center for Frontier AI Research (CFAR), Institute of High Performance Computing (IHPC), Singapore Institute of Manufacturing Technology (SIMTech), Agency for Science, Technology and Research (A*STAR). His research interests include artificial intelligence, machine learning, data-centric transfer optimization, and data mining.

Yew-Soon Ong (*Fellow*, IEEE) received the Ph.D. degree in artificial intelligence in complex design from the University of Southampton, Southampton, U.K., in 2003. He is a President's Chair Professor of Computer Science with Nanyang Technological University (NTU), Singapore, and holds the position of Chief Artificial Intelligence Scientist with the Agency for Science, Technology and Research, Singapore. At NTU, he serves as the Director of the Data Science and Artificial Intelligence Research and the Co-Director of the Singtel-NTU Cognitive and Artificial Intelligence Joint Laboratory. His research interest is in artificial and computational intelligence. Dr. Ong has received several IEEE outstanding paper awards and was listed as a Thomson Reuters Highly Cited Researcher and among the World's Most Influential Scientific Minds. He is the Founding Editor-in-Chief of the IEEE TRANSACTIONS ON EMERGING TOPICS IN COMPUTATIONAL INTELLIGENCE and an Associate Editor of IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, IEEE TRANSACTIONS ON CYBERNETICS, and IEEE TRANSACTIONS ON ARTIFICIAL INTELLIGENCE.

Xin Luo (*Fellow*, IEEE) received the B.S. degree in computer science from the University of Electronic Science and Technology of China, Chengdu, China, in 2005, and the Ph.D. degree in computer science from the Beihang University, Beijing, China, in 2011. He is currently a Professor of Data Science and Computational Intelligence with the College of Computer and Information Science, Southwest University, Chongqing, China. He has authored or coauthored over 300 papers (including over 140 IEEE Transactions/Journal papers) in the areas of his interests. His research interests include Artificial Intelligence, Data Science and Neural Networks. Dr. Luo was the recipient of the Outstanding Associate Editor Award from IEEE Access in 2018, IEEE/CAA Journal of Automatica Sinica in 2020, and from IEEE Transactions on Neural Networks and Learning Systems in 2022. He is currently serving as an Associate Editor for IEEE Transactions on Neural Networks and Learning Systems, and IEEE/CAA Journal of Automatica Sinica. His page is https://scholar.google.com/citations?user=hyGlDs4AAAAJ&hl=zh-TW.