LCLRD: Link Prediction via Contrastive Learning With Relational Distillation

Yabing Yao, Ziyu Ti, Pingxia Guo, Zhiheng Mao, Yangyang He, Jianxin Tang, Fuzhong Nian, and Ning Ma

***Abstract*—Link prediction is widely used in various fields to predict the missing or potential future links between node pairs in the network. Knowledge distillation (KD) methods have been introduced for the link prediction task, demonstrating exceptional performance in inference acceleration. However, these methods primarily focus on supervised learning settings that rely on high-quality labels, and they fail to capture the rich structural features inherent in graph data within the teacher model effectively, resulting in suboptimal performance. To address the problem of reducing label dependency, we propose a link prediction method via contrastive learning with relational distillation (LCLRD) that aims to improve model performance under label scarcity. LCLRD incorporates contrastive learning into the teacher model for pretraining, enabling the teacher model to obtain high-quality node representations more effectively. Then, graph structural information is extracted from the teacher model via relational distillation and transferred to the student model. In addition, we leverage the Pearson correlation coefficient (PCC) as a new matching strategy to replace the Kullback–Leibler (KL) divergence, aiming to alleviate the prediction discrepancy between the student and the stronger teacher model. The experimental results show that LCLRD significantly outperforms other baseline methods on 12 datasets, demonstrating the superiority of this approach.**

***Index Terms*—Complex networks, contrastive learning, knowledge distillation (KD), link prediction.**

I. Introduction

LINK prediction is intended to infer the likelihood of a link between any node pair in complex networks [1], [2], [3], [4]. It holds substantial value in practical applications. For instance, in biological research, protein–protein interaction networks [5] are used for predicting possible associations between proteins. In addition, link prediction can be applied to online social platforms [6], e-commerce websites [7], and product recommendation networks [8], providing valuable insights for friend recommendation and product suggestion.

Owing to the exceptional representational capacity of graph neural networks (GNNs) [9], the encoder–decoder architecture is extensively used for link prediction approaches. In this architecture, the encoder is a GNN model that learns node embeddings, while the decoder is a multilayer perceptron (MLP) that uses these representations to predict links. The success of GNNs lies in their ability to iteratively aggregate information from neighboring nodes, thereby enriching node representations and capturing complex internode dependencies [10]. For example, SEAL [11] uses graph neural networks to model local and global dependencies among various nodes, enabling the capture of intricate features inherent in the graph structure. Line graph link prediction [12] proposes a line graph neural network model, which can prevent information loss during feature extraction by converting the input graph into its line graph. Yet, owing to graph dependency, GNNs suffer from considerable inference latency, increasing time complexity and computational costs. In other words, it is difficult for existing GNN-based methods to improve inference speed while maintaining the same prediction accuracy.

To address the tradeoff relationship between speed and performance of existing methods, some knowledge distillation (KD) models have been proposed, which distill the essential knowledge acquired by the teacher model into the student model using KD techniques [13], [14], [15], [16], [17]. For example, Graph2Feat [18] leverages KD learning framework to achieve inductive link prediction. GLNNs [15] transfer GNN knowledge to MLPs through KD framework, effectively eliminating the latency issue in extracting neighbor information during GNN inference. Existing methods based on KD have achieved remarkable progress in accelerating inference efficiency. However, these methods concentrate on supervised learning paradigms, which results in heavy reliance on labels and limited robustness. Hence, this study focuses on learning

Received 13 August 2024; revised 29 April 2025 and 21 July 2025; accepted 24 July 2025. This work was supported in part by the National Natural Science Foundation of China under Grant 62366030 and Grant 62162040, in part by Gansu Provincial Natural Science Foundation under Grant 23JRRA8222, in part by the Higher Education Innovation Fund Project of Gansu under Grant 2022A-022 and Grant 2024A-024, in part by the Open Project of Key Laboratory of Linguistic and Cultural Computing Ministry of Education under Grant KFKT202304, and in part by Gansu Provincial Science Fund for Distinguished Young Scholars under Grant 23JRRA766. *(Corresponding author: Yabing Yao.)*

Yabing Yao, Ziyu Ti, Pingxia Guo, Zhiheng Mao, Yangyang He, Jianxin Tang, and Fuzhong Nian are with the School of Computer and Communication, Lanzhou University of Technology, Lanzhou 730050, China (e-mail: yaoyabing@lut.edu.cn; tzy97666@163.com; guopingxia2024@163.com; mzh123452021@163.com; yangyanghe2023@163.com; tangjx@lut.edu.cn; gdnfz@lut.edu.cn).

Ning Ma is with the Key Laboratory of Linguistic and Cultural Computing Ministry of Education, Northwest Minzu University, Lanzhou 730030, China (e-mail: maning@xbmu.edu.cn).

Digital Object Identifier 10.1109/TCSS.2025.3593331

TABLE I
EVALUATE THE PERFORMANCE OF TEACHER AND STUDENT MODELS WITH VARYING LAYER DEPTHS USING AUC (AUPR) ON THE CORA DATASET

Layers	One	Two	Three	Four
Teacher	95.20(95.78)	95.15(95.23)	95.21(95.12)	95.10(95.01)
Student(KL)	92.73(93.01)	94.66(95.11)	86.95(86.16)	81.81(81.98)
Student(PCC)	**92.93(92.93)**	**95.48(95.63)**	**92.54(92.59)**	**86.26(85.72)**

Note: Bold entries indicate the best results.

high-quality node embeddings in label-scarce settings and further investigates alternative distillation strategies.

In numerous real-world application domains, acquiring data labels can be difficult or expensive. However, self-supervised learning (SSL) methods offer a possible solution, enabling models to enhance the robustness of node embeddings with fewer labels [19], [20]. SSL techniques can generally be divided into two main categories: generative learning and contrastive learning. Among these methods, contrastive learning aims to capture the relationship between the graph data by maximizing the consistency between the anchor node and the positive sample, while minimizing its similarity to negative samples, while minimizing similarity to negative samples.

In addition, the essence of KD relies on minimizing the Kullback–Leibler (KL) divergence [14] between the output probability distributions of the teacher and student models, thereby effectively transferring the teacher's knowledge to the student. However, KL divergence mainly measures the difference between two distributions. Minimizing KL divergence can lead to overfitting of the student model to the teacher model, especially when the teacher model is very powerful (such as employing a larger model size or using data augmentation strategies). As shown in Table I, the student model's performance degrades much faster than that of the teacher model. In this case, it may be challenging to recover the prediction accurately via KL divergence, which can result in the failure of vanilla KD. Compared to KL divergence, the Pearson correlation coefficient (PCC) [21] alleviates overfitting by focusing on the correlation between predictive distributions and is computationally less expensive. Therefore, we leverage the PCC as a new matching method to replace the KL divergence.

To sum up, this article aims to investigate how contrastive learning can be employed for pretraining and fine-tuning a teacher model to improve its generalization capability in link prediction tasks, while simultaneously minimizing the prediction gap between the student and teacher models by using the PCC. We propose a link prediction method based on contrastive learning for relational distillation, which addresses the problem of failing to capture rich graph structure features and overfitting of the student model to the teacher model. Specifically, contrastive learning is employed for pretraining the teacher model, and subsequently fine-tuning it with labeled data, thereby helping the teacher model learn high-quality node representations more effectively. Subsequently, the PCC is adopted as a distillation loss to better capture the deep structural dependencies encoded by the teacher model and accurately transfer them to the student model, thereby improving the student model's performance. In summary, the main contributions are as follows.

1) We design an innovative contrastive learning framework for KD, named a link prediction method via contrastive learning with relational distillation (LCLRD). In comparison to traditional distillation approaches, LCLRD learns more robust node representations with reduced information loss.
2) We propose a distribution matching strategy based on the PCC to alleviate the prediction discrepancy between the student model and a stronger teacher model.
3) We conduct extensive experiments on 12 public benchmark datasets, comparing our method against a range of competitive baselines. Experimental results demonstrate that LCLRD outperforms state-of-the-art methods in terms of both generalization and robustness. In addition, ablation studies and parameter sensitivity analysis validate the efficiency of our algorithm.

The remainder of this article is structured as follows. Section II shows the related works. In Section III, we introduce the preliminary definitions and notations. Section IV provides a detailed overview of our framework, including its pipeline and components. Section V analyzes a comprehensive evaluation of the experimental results. Section VI finally concludes our work.

II. RELATED WORK

This section presents a comprehensive review of prior research relevant to our work, including link prediction, KD, and SSL.

A. Link Prediction

Link prediction [22], [23] has attracted significant attention from the research community, due to its wide applications. In general, link prediction can be categorized into three primary approaches—heuristic methods, embedding methods, and graph representation learning methods.

1) Heuristic Methods: The essence of heuristic methods lies in leveraging neighborhood information to compute the similarity score between two target nodes [24]. According to the structural information used in the computation process, heuristic methods can be categorized into three classes, including first-order [1], [25], second-order [6], [26], and high-order heuristics [27], [28], [29] methods. Common neighbors (CN) [1] is a classic first-order heuristic method that relies solely on directly connected nodes (i.e., one-hop neighbors) to compute similarity. Adamic-Adar (AA) [6] scores similar common attributes of two nodes (two-hop neighbors) to predict links between nodes. Katz [27] utilizes the entire graph to calculate the similarity measure between two nodes. High-order heuristic methods generally achieve better performance than low-order ones but incur higher computational costs.

2) Embedding Methods: Embedding methods aim to transform high-dimensional sparse node features into low-dimensional dense vectors, which capture their inherent structural information within the network. Recent advances such as Node2vec [30] are proposed to learn node embeddings via the

skip-gram [31] framework. A recent study proposed by Xiao et al. [32] introduces a novel approach to link prediction for social networks, which addresses complex network features. LP-ROBIN [33] uses incremental node embeddings based on random walks to capture the dynamic network structure and predict new links. These methods can effectively capture high-quality features within the graph, leading to strong link prediction performance. However, their effectiveness can be hindered by sparse graph structures.

3) Graph Representation Learning Methods: To overcome the constraints of heuristic methods, graph representation learning methods [34] are proposed. These methods leverage both node attribute features and graph structural features to automatically mine the distribution of links within the network. In recent years, GNN-based methods have shown significant performance advantages for link prediction. For instance, Cai et al. [35] develop a multiscale method that employs subgraphs at multiple levels to capture graph structural features, leading to superior link prediction results. TGGDN [36] demonstrates effectiveness in the challenging task of dynamic link prediction on diverse datasets. Despite improving task performance, these methods often struggle to scale in real-world applications due to their high computational complexity.

B. KD

KD [14] seeks to distill the knowledge of a large complex model into a smaller model. Logit-matching [37] is one direct approach for KD. Its goal is to enable the smaller model to perform as well as the large complex model on a specific downstream task. Nevertheless, it is difficult to acquire the deeper knowledge of the large complex model. In addition, other works [18], [38], [39] also explore transferring knowledge from intermediate layers to enhance distillation performance, but these methods typically incur additional computational costs during training. For example, OFD [38] leverages multiple intermediate layers for KD, its reliance on additional convolutions to ensure feature alignment leads to increased training expenses. Contrastive representation distillation [39] introduces a contrastive loss for transferring pairwise relationships, but suffers from a high computational burden caused by the maintenance of a memory bank for large datasets such as ImageNet.

Recently, a few studies [40], [41], [42] propose the relational KD methods that aim to preserve relationships between representations of similar instances. Pairwise difference relational distillation [40] tackles the challenge of consistent ranking in person re-identification (Re-ID) tasks. It achieves this by minimizing pairwise similarity differences between student and teacher networks through a nonlinear KD loss. By employing relational distillation, multiclassification anomaly detection [41] effectively transfers knowledge about image relationships from a teacher to a student network. Linkless link prediction (LLP) [42] is proposed to specifically target the pairwise transfer of relational knowledge between nodes from teacher to student, facilitating the learning of complex relationships in the student model. Most relational distillation methods rely on KL divergence to align the predicted relationships between teacher and student. However, a stronger teacher model may not always lead to optimal student performance in KD. In some cases, the student's performance can even decrease. In contrast, our approach to KD prioritizes maintaining the teacher model's ranking of predictions, instead of accurately reproducing absolute prediction values. In this article, the PCC is adopted, which more effectively captures the relational topology of the graph, thus improving the performance of the student model.

C. Self Supervised Learning

Our study is also related to SSL. By crafting pretext tasks, SSL can extract rich information from data without manual labels, enabling a variety of applications for graph data. Existing SSL methods can be broadly categorized as either generative or contrastive, based on their learning objectives [43]. Generative learning focuses on learning graph embeddings by reconstructing feature or structural information on a graph. It typically focuses on local node features and relationships, which makes it difficult to capture the overall graph structure and long-distance dependencies. In contrast, contrastive learning can effectively capture the global graph structure by comparing the representations of nodes from different parts of the graph. It first defines the context of a node at different levels, including the node, subgraph, and graph levels. By contrasting the node representations under different levels of context, the model can learn more comprehensive node features. GraphCL [44] uses four types of augmentation to generate multiple augmented views of a network. By utilizing a shared encoder to learn graph embeddings and maximizing the mutual information between different views, the model captures the intrinsic features of the graph more effectively. Deep graph infomax [45] feeds the original network and the augmented network into two diverse encoders to obtain graph-level and node-level representations, thus improving the model's generalization ability. Wan et al. [46] propose a SSL architecture that adaptively selects the most informative negative samples for contrastive learning. This approach effectively suppresses noise within the negative samples, resulting in more robust node representations. Liao et al. [47] introduce RevGNN, a contrastive graph learning framework for academic reviewer recommendation, addressing label sparsity and false negatives by improving negative sampling with a pseudo negative-label strategy. Meanwhile, Feng et al. [48] propose CP-Prompt, a framework for domain-incremental continual learning that decouples contrastive and personalized information using twin-prompt strategies, which aligns with our goal of flexible KD between teacher and student models. In contrast to previous studies, we propose a framework that integrates contrastive learning with KD, effectively addressing the problem of long training time.

III. Problem Formulation

Following, we introduce notations for link prediction, KD, and SSL in line with previous works.

Definition 1 (Link Prediction): Formally, let $\mathcal{G} = (V, E, X)$ denote a simple network, where V and E represent the sets of nodes and edges, respectively. The node feature matrix

$X \in \mathbb{R}^{N\times d}$ contains the feature information for each node, with N being the number of nodes and d the dimension of node features. The adjacency matrix $A \in \mathbb{R}^{N\times N}$ is another way to represent the network, with $A(i,j)=1$ indicating a link between nodes i and j, and $A(i,j)=0$ otherwise. Consequently, a network can alternatively be described as $\mathcal{G}=(X,A)$. Link prediction aims to forecast potential future or missing links within a network by analyzing its structure.

Definition 2 (KD): In the framework of KD, when performing representation learning on node i, we denote the vectors obtained from the teacher and student models as $\mathbf{h}_i$ and $\hat{\mathbf{h}}_i$, respectively. During the training process, the true label of a target link between nodes i and j is represented by $y_{i,j} \in \{0,1\}$. The prediction probabilities for this true label by the teacher and student models are denoted by $\tilde{y}_{i,j}^{(t)}$ and $\tilde{y}_{i,j}^{(s)}$, respectively. In this work, we adopt the extensively applied encoder–decoder model for the link prediction task.

A. Teacher Model

For each node i at layer l, its embedding $\mathbf{h}_i^l$ is updated through a combination of the representations of its neighboring nodes $\mathcal{N}(i)$, where $j \in \mathcal{N}(i)$ and $\mathbf{h}_i^0 = x_i$, as follows

$$\mathbf{h}_i^l = U\text{PDATE}(\mathbf{h}_i^{l-1}, A\text{GGREGATE}(\{\mathbf{h}_j^{l-1}, j \in \mathcal{N}(i)\})). \quad (1)$$

In order to predict the link relationship between nodes, the decoder $\mathcal{D}$ obtains node representations from the last layer, and the link prediction scores for a node pair i and j can be described as

$$\tilde{y}_{i,j}^{(t)} = \sigma(\mathcal{D}(\mathbf{h}_i, \mathbf{h}_j)) \quad (2)$$

where $\sigma(\cdot)$ represents a Sigmoid function. Finally, the Hadamard product operation is adopted as the input of MLP decoder.

B. Student Model

The model is a stack of two linear layers, forming a simple multilayer perceptron architecture. The link prediction decoder can be written as

$$\tilde{y}_{i,j}^{(s)} = \sigma(\mathcal{D}(\hat{\mathbf{h}}_i, \hat{\mathbf{h}}_j)). \quad (3)$$

Definition 3 (SSL): Our objective is to learn an encoder function, denoted $\mathcal{E}$, that maps context subgraphs to latent node representations. Given the vth context subgraph with N' nodes and d features each node, represented by the feature matrix $\mathbf{X}_v \in \mathbb{R}^{N'\times d}$, and its corresponding adjacency matrix $\mathbf{A}_v \in \mathbb{R}^{N'\times N'}$ encoding connectivity information, the encoder generates a latent representation matrix $\mathbf{H} \in \mathbb{R}^{N'\times d'}$ for the nodes in the subgraph. The goal is for the encoder to capture node information by taking into account the connections between nodes within the subgraph.

IV. Methodology

In this part, as depicted in Fig. 1, we illustrate the overall architecture of our proposed LCLRD. LCLRD consists of the following two components.

1) Contrastive Learning as a Teacher Model for Pretraining: In KD, the pretrained teacher model plays a key role. The stronger the teacher model's performance, the richer the knowledge that can be distilled, effectively guiding the student model to learn. Introducing contrastive learning enhances the robustness of the teacher model's representations, thereby further improving the effectiveness of KD. During the entire teacher model pretrained process, there are two main stages: self-supervised pretraining and teacher model fine-tuning. It is important to note that the self-supervised pretraining stage does not require labeled data, while the fine-tuning stage relies on labeled data for training. In particular, in order to generate contrastive SSL samples, we leverage the importance scores of neighboring nodes to sample closely related nodes, thereby constructing context subgraphs. Subsequently, the contrastive learning model employs an encoder and readout function to obtain low-dimensional representations for both the center node and its corresponding local subgraph. Finally, the teacher model is optimized using a contrastive loss applied to the central node and subgraph representations, and the teacher logits are computed using a decoder.

2) Relational Distillation for Link Prediction: In link prediction, considering the limitations of both logit-based and representation-based matching methods, the student model cannot perform well due to insufficient knowledge extraction. Therefore, relational distillation serves as a mechanism for extracting rich knowledge from the teacher model, and effectively enhances the student model's ability in link prediction accuracy. Specifically, the logits obtained from both the teacher and student models are used to compute rank and distribution matching losses. Subsequently, joint training can be performed by optimizing the true label loss and the two matching losses within the LCLRD model.

In Sections IV-A and IV-B delve into the two primary components of our framework. The overall LCLRD algorithm is elaborated upon in Section IV-C. The computational complexity of LCLRD is analyzed in Section IV-D.

A. Contrastive Learning as a Teacher Model for Pretraining

The success of KD largely depends on the performance of the teacher model. Existing KD research mainly adopts a supervised approach that relies on labeled data for training. However, although this method can achieve good performance, it requires a substantial quantity of labeled data, which incurs significant costs. To explore methods for reducing reliance on labeled data, we introduced contrastive learning as a self-supervised pretraining step in the training of the teacher model. This approach, which does not require additional labeled data, enables the teacher model to capture rich semantic information from the data by learning to distinguish between positive and negative sample pairs. Subsequently, we fine-tune the pretrained teacher model using labeled data to better adapt it to specific downstream tasks. This strategy, which combines self-supervised pretraining with supervised fine-tuning, aims to improve model performance while reducing its dependency on

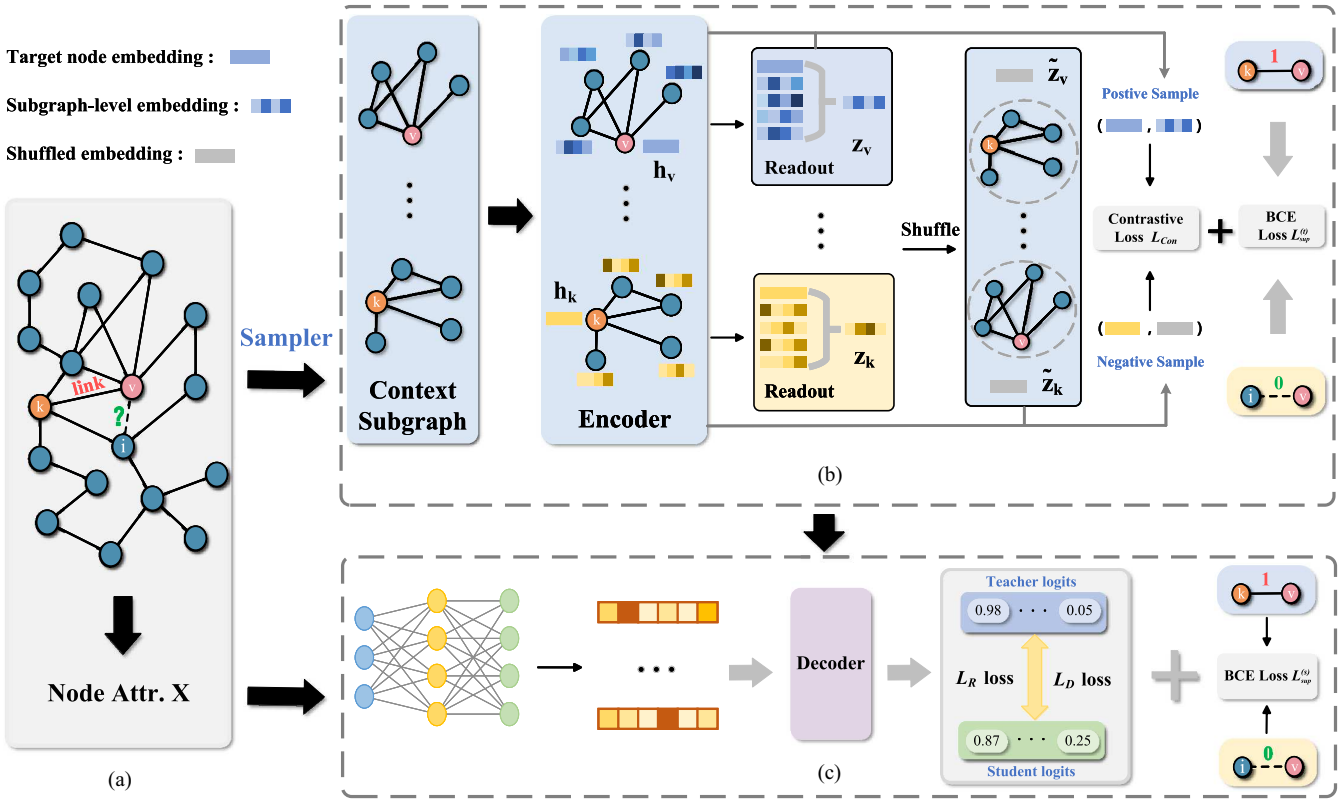

Fig. 1. Overall framework of LCLRD. The framework is composed of three parts: (a) input graph; (b) contrastive learning as a teacher model for pretraining; and (c) relational distillation for link prediction. In the component (b), we sample similar nodes from neighbors to build context subgraphs. Then, the GNN model learns low-dimensional embeddings for the target node and its neighbors, which is trained by maximizing the contrastive learning loss and the BCE loss. In the component (c), we use relational distillation to improve the student model's link prediction by learning from the teacher.

large-scale labeled datasets. In this article, we design a novel type of teacher model to address the issues of weak robustness and label scarcity. Specifically, this work proposes a GNN-based contrastive learning teacher model, which effectively captures high-quality node representations and improves its performance. The teacher model generally consists of three steps: *extracting the context subgraph of the target node*, *encoding node and subgraph representations*, and *computing contrastive loss and optimizing the model*.

1) Extracting the Context Subgraph of the Target Node: While various graph data augmentation methods exist, including edge perturbation and feature masking, this article uses subgraph sampling to generate instance pairs. Considering the diverse importance of different neighbors, the personalized pageRank (PPR) algorithm [49] is utilized for subgraph sampling. This method effectively selects crucial nodes for the subgraph based on a given target or seed node. Given a target node v, the subgraph sampler $\mathcal{S}$ utilizes the PPR algorithm as a means of quantifying the importance of neighboring nodes. The importance score matrix $\mathbf{Q}$ is denoted as

$$\mathbf{Q} = \lambda(\mathbf{I} - (1-\lambda)\cdot\hat{\mathbf{A}}) \tag{4}$$

where the parameter $\lambda \in [0, 1]$ is consistently assigned a value of 0.15. $\mathbf{I}$ is the identity matrix, and $\hat{\mathbf{A}} = \mathbf{A}\mathbf{D}^{-1}$ represents the column-normalized adjacency matrix, where $\mathbf{A}$ is the adjacency matrix and $\mathbf{D}$ is the degree matrix. For a given target node v, the subgraph sampler $\mathcal{S}$ selects the top-l scoring neighbors according to the score matrix $\mathbf{Q}$ to form a subgraph. These selected nodes can be indexed by

$$\mathrm{id} = \mathrm{top\_rank}(\mathbf{Q}(v,:), l) \tag{5}$$

where $\mathrm{top\_rank}(\cdot)$ denotes a function that identifies and returns the indices corresponding to the top-l among the neighbors of node v, and l denotes the size of the context subgraph. Given a node index, the subgraph sampler $\mathcal{S}$ extracts the context subgraph $\mathcal{G}_v$ of node v from the original graph. Its feature matrix $\mathbf{X}_v$ and adjacency matrix $\mathbf{A}_v$ are defined as follows:

$$\mathbf{X}_v = \mathbf{X}_{\mathrm{id},:}, \ \ \mathbf{A}_v = \mathbf{A}_{\mathrm{id},\mathrm{id}} \tag{6}$$

where id is used for indexing. The corresponding to the extracted subgraph indexed node features and adjacency matrix are represented by $\mathbf{X}_{\mathrm{id},:}$ and $\mathbf{A}_{\mathrm{id},\mathrm{id}}$, respectively. Consequently, the context subgraph $\mathcal{G}_v = (\mathbf{X}_v, \mathbf{A}_v)$ is derived from the original graph $\mathcal{G} = (\mathbf{X}, \mathbf{A})$ centered around node v.

2) Encoding Node and Subgraph Representations: Given a target node v, its context subgraph is denoted by $\mathcal{G}_v = (\mathbf{X}_v, \mathbf{A}_v)$, the encoder $\mathcal{E}$ transforms the input into a latent representations matrix $\mathbf{H}_v$, which is expressed as

$$\mathbf{H}_v = \mathcal{E}(\mathbf{X}_v, \mathbf{A}_v). \tag{7}$$

We adopt two layers of GCN as the encoder to obtain node representation by aggregating neighboring nodes. The target

node v is selected from the latent representation matrix $\mathbf{H}_v$ according to its index, and is denoted as

$$\mathbf{h}_v = \mathcal{P}(\mathbf{H}_v) \tag{8}$$

where $\mathcal{P}$ denotes the operation of obtaining the target node embedding. After obtaining the node representation $\mathbf{h}_v$, capturing the representation of the context subgraph of the target node v is crucial. Here, we leverage the readout function $\mathcal{R}$ to extract the subgraph-level representation. The goal is to aggregate the latent representations $\mathbf{H}_v$ in the subgraph into a subgraph-level embedding vector $\mathbf{z}_v$, which can be written as

$$\mathbf{z}_v = \mathcal{R}(\mathbf{H}_v). \tag{9}$$

So far, we obtain the embeddings of the target node and the context subgraph, which are the key foundations for generating positive and negative samples in contrastive learning, and lay the foundation for the model to learn graph structure and semantic information.

3) Computing Contrastive Loss and Optimizing the Model: The success of contrastive learning is largely influenced by the manner in which contrastive instance pairs are defined. For the target node v, we treat the pair of $\mathbf{h}_v$ and the context subgraph representation $\mathbf{z}_v$ as a positive sample pair, and the perturbed subgraph representation $\tilde{\mathbf{z}}_v$ as a negative sample pair. To create negative samples, we employ the perturbation function $\mathcal{C}$, which corrupts a group of subgraph representations, as follows:

$$\{\tilde{\mathbf{z}}_1, \tilde{\mathbf{z}}_2 ..., \tilde{\mathbf{z}}_S\} \sim \mathcal{C}(\{\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_S\}) \tag{10}$$

where S is the size of context subgraphs sampled. As to the objective function, the margin triplet loss [50] is employed to optimize the model, and the loss is expressed as

$$\mathcal{L}_{\text{Con}} = \frac{1}{S}\sum_{v=1}^{S}(\max(\sigma(\mathbf{h}_v\mathbf{z}_v) - \sigma(\mathbf{h}_v\tilde{\mathbf{z}}_v)) + \delta, 0) \tag{11}$$

where $\sigma(\cdot)$ is the sigmoid function and δ is margin value. Next, we transform the link prediction into a binary classification task, and use binary cross-entropy (BCE) to calculate the link prediction loss

$$\mathcal{L}_{\text{sup}}^{(t)} = -\frac{1}{|L|}\sum_{(i,j)\in L} y_{i,j}\log(\tilde{y}_{i,j}^{(t)}) + (1 - y_{i,j})\log(1 - \tilde{y}_{i,j}^{(t)}) \tag{12}$$

where $|L|$ denotes the total count of node pairs within the training dataset. The target link is assigned a binary label $y_{i,j} \in \{0, 1\}$, where 1 denotes that a link exists, while 0 denotes that a link does not exist. Finally, by combining self-supervised pretraining and fine-tuning the teacher model, we jointly optimized the teacher model and calculated the resulting total loss

$$\mathcal{L}^{(T)} = \mathcal{L}_{\text{sup}}^{(t)} + \mathcal{L}_{\text{Con}}. \tag{13}$$

B. Link Prediction With Relational Distillation

Building upon relational distillation, LLP [42] proposes a linkless approach to link prediction. It mainly uses KL divergence to evaluate the prediction difference between the student and teacher models. However, employing KL divergence can lead to the student model overfitting the teacher model, especially when the teacher model is complex (Fig. 4 demonstrates this phenomenon). To address this critical limitation, our proposed method adopts the PCC to mitigate the gap between the student and the more powerful teacher model. In this article, we introduce two matching strategies: rank matching and distribution matching.

Rank matching is a knowledge transfer technique that aligns the prediction ranks of the teacher and student models. It can effectively improve the student model's generalization. Specifically, the teacher and the student models predict the input data, and then calculate the similarity between their predicted rankings by a similarity measure. Finally, the student model's predictions are adjusted based on this similarity score, aiming to bring them closer to the teacher's rankings. Therefore, based on the LLP [42] method, we employ logits ranking from teacher GNNs to train students. Let u represent the anchor node, and $\mathcal{D}_u$ represent the group of neighboring nodes corresponding to u. $\mathcal{Y}_u^{(t)} = \left\{\tilde{y}_{u,i}^{(t)} \mid i \in \mathcal{D}_u\right\}$, obtained by (2), represents the teacher model's prediction probability for node u and each node in $\mathcal{D}_u$. Similarly, the prediction probabilities of the student model are denoted by $\mathcal{Y}_u^{(s)}$ via (3). That is

$$\mathcal{L}_R = \sum_{u\in V}\sum_{\left\{\tilde{y}_{u,i}^{(s)}, \tilde{y}_{u,j}^{(s)}\right\}\in\mathcal{Y}_u^{(s)}} \max(0, -p\cdot\left(\tilde{y}_{u,i}^{(s)} - \tilde{y}_{u,j}^{(s)}\right) + \epsilon) \tag{14}$$

where

$$p = \begin{cases} 1, & \text{if } \tilde{y}_{u,i}^{(t)} - \tilde{y}_{u,j}^{(t)} > \epsilon; \\ -1, & \text{if } \tilde{y}_{u,i}^{(t)} - \tilde{y}_{u,j}^{(t)} < -\epsilon; \\ 0, & \text{otherwise,} \end{cases} \tag{15}$$

where ϵ is the margin hyperparameter. Its detailed description and settings are given in [42].

Distribution-based matching is a technology that matches the softmax outputs of the teacher and student models. In particular, the output scores from both models are first transformed into probability distributions, after which a distance metric is applied to compute the discrepancy between the two distributions. Subsequently, the student model's predicted scores are adjusted based on the distance, resulting in greater alignment with the teacher model's predicted scores. In this article, we employ the PCC as a metric to compute the distance between two distributions. Since it can measure the linear correlation between two distributions, it is more suitable for the situation where there is a large difference in prediction accuracy. It is worth emphasizing that PCC is used here solely as a continuous similarity measure, not as a statistical test or for hypothesis inference. Therefore, this application does not involve multiple hypothesis testing, and is not subject to the multiple comparison problem (MCP) [51] as discussed in classical statistical analysis. Pearson's distance can be adopted as the metric

$$d_p(\mathbf{x}, \mathbf{y}) = 1 - \rho_p(\mathbf{x}, \mathbf{y}) \tag{16}$$

$\rho_p(\mathbf{x},\mathbf{y})$ specifically denotes the PCC, which is a measure of the correlation (linear relationship) between two variables.

$$\begin{aligned}\rho_p(\mathbf{x},\mathbf{y}) &= \frac{\mathrm{Cov}(\mathbf{x},\mathbf{y})}{\mathrm{Std}(\mathbf{x})\mathrm{Std}(\mathbf{y})} \\ &= \frac{\sum_{i=1}^{N}(x_i-\overline{x})(y_i-\overline{y})}{\sqrt{\sum_{i=1}^{N}(x_i-\overline{x})^2\sum_{i=1}^{N}(y_i-\overline{y})^2}}\end{aligned} \tag{17}$$

where $\mathrm{Cov}(\mathbf{x},\mathbf{y})$ is the covariance of $\mathbf{x}$ and $\mathbf{y}$, $\overline{x}$ and $\mathrm{Std}(\mathbf{x})$ denote the mean and standard derivation of $\mathbf{x}$, respectively. By this means, we describe the distribution-based matching, denoted as

$$\mathcal{L}_D = \frac{1}{B}\sum_{u=1}^{B} d_p(\mathcal{Y}_u^{(s)}, \mathcal{Y}_u^{(t)}) \tag{18}$$

where B represents the batch size. Next, we apply BCE to calculate the link prediction loss in the student model, denoted as

$$\mathcal{L}_{\mathrm{sup}}^{(s)} = -\frac{1}{|L|}\sum_{(i,j)\in L} y_{i,j}\log\left(\tilde{y}_{i,j}^{(s)}\right) + (1-y_{i,j})\log\left(1-\tilde{y}_{i,j}^{(s)}\right). \tag{19}$$

Joint optimization of the real label loss and two matching losses be employed during LCLRD training. Therefore, the overall loss of LCLRD can be expressed as

$$\mathcal{L}_{\mathrm{total}} = \alpha\cdot\mathcal{L}_{\mathrm{sup}}^{(s)} + \beta\cdot\mathcal{L}_R + \gamma\cdot\mathcal{L}_D \tag{20}$$

where α, β, and γ are tuning parameters that determine the influence of each loss term on the model's learning.

C. LCLRD: A Framework of Relational Distillation Based on Contrastive Learning

In this section, the overall algorithm of our proposed LCLRD architecture is presented. The algorithm consists of two primary stages: the pretraining phase of teacher model and training phase of student model. In the pretraining phase, the teacher model is trained using unsupervised contrastive learning. After that, during the training phase, structural knowledge is extracted from the teacher model into the student model through the application of two relationship matching strategies.

The entire procedure of our LCLRD framework is presented in Algorithm 1. For each pretraining epoch, we initially divide the node set V into multiple mini-batches. Then, during each iteration, a context subgraph is extracted for each node in the current mini-batch, and the corresponding node and subgraph representations are obtained. Next, a corruption operation is applied to generate negative samples by perturbing a set of subgraphs. Finally, the contrastive loss and the link prediction loss are jointly calculated. Backpropagation with gradient descent is employed to optimize the contrastive learning model's parameters. After the pretraining phase, the two matching losses and the link prediction loss are computed and jointly optimized over multiple training epochs through backpropagation, during which the model parameters are updated accordingly.

Algorithm 1: Overall Procedure of LCLRD.

Input: Original Graph $\mathcal{G}=(X,A)$, Number of training epochs: T, Batch size: B, Node set: V, The number of context subgraphs sampled: S.
Output: Total loss $\mathcal{L}_{total}$.
// Teacher model pre-training phase
for $t \in 1,2,\ldots,T$ **do**
  $\mathcal{B} \leftarrow$ (Randomly divide V into groups of size B)
  for each batch $b=(v_1,\ldots,v_B)\in\mathcal{B}$
    Sampler $\mathcal{S}\rightarrow\{(X_1,A_1),(X_2,A_2),\cdots,(X_S,A_S)\}$ via Eqs.(4), (5), (6).
    for all each context subgraph $\mathcal{G}_v=(\mathbf{X}_v,\mathbf{A}_v)$ **do**
      $\mathbf{H}_v$, $\mathbf{h}_v$, $\mathbf{z}_v$ via Eqs.(7), (8), (9).
    end for
    Generate negative examples by corrupting subgraph representations $\{\tilde{\mathbf{z}}_1,\tilde{\mathbf{z}}_2,...,\tilde{\mathbf{z}}_S\}\sim\mathcal{C}(\{\mathbf{z}_1,\mathbf{z}_2,...,\mathbf{z}_S\})$ via Eq.(10).
    Calculate $\mathcal{L}^{(T)}$ via Eqs.(11), (12), (13).
    Update the parameters of teacher model.
  end for
end for
// Student model training phase
for $t \in 1,2,\ldots,T$ **do**
  Calculate $\mathcal{L}_{total}$ via Eqs.(14), (18), (19), (20).
end for

D. Complexity Analysis

We can analyze the complexity of the LCLRD algorithm from the following two aspects.

1) Teacher Model Pretraining Phase: In the LCLRD method, the teacher model is pretrained using contrastive learning. When pretraining a teacher model, it is necessary to process a large amount of graph data and compute the embedding representations of each node and its neighboring nodes. Assuming the graph has N nodes and E edges, the teacher model's complexity is mainly proportional to the size of the graph. The computational complexity of this phase is mainly determined by the following two factors.

a) Contrastive learning: In contrastive learning, calculating the similarity or distance between node pairs typically requires $O(N)$ computations, and the embedding vector for each node may need to be updated multiple times iteratively. Assuming there are T training iterations, the computational complexity is $O(T\times N^2)$.

b) GCN layers: The teacher model typically follows a graph convolutional network (GCN) architecture, where the computational complexity of GCN is usually $O(N\times F\times L)$, where F is the feature dimension per layer and L is the number of GCN layers. Therefore, the complexity of the pretraining phase increases with the graph size, the number of layers, and the feature dimension.

In summary, the total complexity of the pretraining phase is: $O(T\times N^2)+O(N\times F\times L)\approx O(T\times N^2)$.

2) Student Model Inference Phase: During the inference phase of the student model, we primarily extract knowledge

from the pretrained teacher model through KD. The computational complexity of the inference phase for the student model is generally lower and mainly includes the following aspects.

a) Structure of the student model: The student model is usually smaller with fewer parameters, resulting in relatively lower computational complexity. The complexity of the student model is $O(N \times F')$, where F' is the feature dimension of the student model.

b) Distillation process: The student model learns by imitating the teacher model's outputs (such as soft labels and embedding representations), and this process itself does not significantly increase computational complexity. The computational cost of the distillation phase is mainly concentrated on computing the difference between the teacher model's and the student model's outputs, such as loss functions such as KL divergence.

Overall, total complexity of the inference phase is: $O(N \times F')$. The complexity of this stage is relatively low, but it is still related to the number of nodes and the feature dimension of the student model.

V. Experiments

In this section, we answer the following study problems by conducting comprehensive experiments.

1) *Q1:* How does our method perform in the link prediction task?
2) *Q2:* How robust is our method in the link prediction task?
3) *Q3:* What does LCLRD's performance on null models reveal about its robustness?
4) *Q4:* How does the proposed contrastive learning module benefit the overall model?
5) *Q5:* What is the utility of the proposed PCC in KD?
6) *Q6:* How do different modules affect in the link prediction task?
7) *Q7:* How do various hyperparameter settings impact the accuracy of this method?

A. Datasets and Experiment Setup

1) Datasets: We evaluate our method on 12 datasets, including Email, NetScience, Power, Cora, Citeseer, Pubmed, Squirrel, Chameleon, Computers, Photo, Wiki, and CS to illustrate the efficiency of our method. To ensure generalizability and robustness of our evaluation, we selected datasets based on the following criteria: a) inclusion of diverse network types, including citation networks, social networks, biological networks, and infrastructure networks; b) availability and completeness of node attribute information, enabling fair comparisons among attribute-based methods; and c) suitability for link prediction tasks, such as containing no excessive missing links or disconnected components. This selection strategy allows us to comprehensively evaluate model performance under varied structural and semantic conditions. All of the above networks can be downloaded from the following website: https://pytorch-geometric.readthedocs.io/en/latest/modules/datasets.html. The details of the datasets are shown in Table II.

TABLE II
Statistics of Datasets (Summary of Datasets Used in Our Experiments

Datasets	Nodes	Edges	Features	Type
Email	1133	5451	-	Social
NetScience	1589	2742	-	Social
Power	4941	6594	-	Electricity
Chameleon	2277	18 051	3132	Wikipedia
Wiki	2405	17 981	4973	Wikipedia
Cora	2708	5278	1433	Citation
Citeseer	3327	4552	3703	Citation
Squirrel	5201	108 537	3148	Wikipedia
Photo	7650	238 162	745	Copurchase
Computers	13 752	491 722	767	Copurchase
CS	18 333	163 788	6805	Coauthor
Pubmed	19 717	44 324	500	Citation

Note: The number of node, edge, feature, and graph type are provided for each dataset. The "-" represents a network with no attribute features, where the features are one-hot encoding vectors determined solely by the number of nodes.

2) Experiment Setup: The LCLRD architecture is deployed using PyTorch and PyTorch Geometric (PyG). All experiments are performed utilizing NVIDIA RTX A6000 (86-GB memory). In this experiment, the learning rate is uniformly set to 0.001 throughout all experiments. All methods employ an output dimension of 128. A two-layer network architecture is adopted for the teacher model. We train the model for Cora, Citeseer, Squirrel, Chameleon, and Wiki datasets with 500 epochs, and train on Pubmed, Computers, Photo, and CS datasets with 1000 epochs.

a) Evaluation metrics: For the purpose of assessing the results of our method and the baselines, we adopt the area under the curve (AUC) and the area under the precision-recall (AUPR) as evaluation metrics.

AUC [4] serves as a benchmark for gauging the link prediction algorithm's effectiveness. It represents the area under the receiver operating characteristic (ROC) curve, denoted as

$$AUC = \frac{n^{'} + 0.5 \times n^{''}}{n} \tag{21}$$

where n represents the total number of sample pairs, $n^{'}$ denotes the number of times a positive sample ranks higher than a negative sample, and $n^{''}$ represents the number of times a positive and a negative sample have the same ranking.

AUPR [52] is composed of recall on the x-axis and precision on the y-axis, denoted as

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{22}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{23}$$

where TP, TN, FP, and FN are the number of correctly predicted links, correctly predicted nonlinks, incorrectly predicted links, and incorrectly predicted nonlinks, respectively, in the link prediction task.

B. Baselines

To confirm the efficacy and superiority of the proposed framework, our method is compared with four types of baseline methods as follows.

1) Heuristic Methods:

a) CN [53] is a network structure-based node similarity metric that the more shared connections two nodes possess, the greater the likelihood of a direct connection being made.

b) Resource allocation (RA) [54] is a link prediction method based on RA, where the core idea is to predict the likelihood of a direct connection between two nodes based on the amount of shared resources.

c) Local path (LP) [26] is a link prediction method based on LPs, which considers all possible paths of length 2 and 3 between nodes. It calculates a score based on the number of these paths, with shorter paths contributing more weight due to an exponential decay factor.

d) Common neighbors degree penalization (CNDP) [55] is a link prediction method based on CN and distance propagation. It not only considers the number of shared neighbors between two nodes but also the propagation of distance between those neighbors. The more shared neighbors two nodes have and the closer these neighbors are to each other in the network, the higher the likelihood of a direct connection between the nodes.

e) Katz [27] takes into account all possible paths between nodes. It calculates a score based on the sum of these paths, with shorter paths contributing more weight due to an exponential decay factor.

2) Embedding Methods:

a) DeepWalk [56] employs random walks as a sampling strategy to learn node embeddings, and encodes these embeddings into vectors in a continuous vector space.

b) Node2vec [30] improves upon DeepWalk by utilizing biased random walks to learn node embeddings that better represent both local and global network structures.

3) Graph Representation Learning Methods:

a) SEAL [11] innovates by replacing fully connected neural networks with graph neural networks to capture structural features from local subgraphs. This enables learning from both subgraph topology and node attributes.

b) T-BGRL [57] is a simple and effective noncontrastive method for link prediction in graph SSL tasks.

c) Synergetic fusion based graph convolutional networks (SFGCN) [58] is a synergetic fusion-based GCN approach that integrates multilevel fusion mechanisms to enhance link prediction accuracy in social networks by combining structural, textual, and attribute data.

4) KD Methods:

a) DEAL [59] constructs comprehensive node representations by employing two encoders and an alignment mechanism to integrate information from different perspectives.

b) Graph2Feat [18] is an effective and powerful inductive link prediction method that combines the advantages of GNNs and MLPs, and significantly improves the inference speed while maintaining high accuracy.

c) LLP [42] a relational KD framework, it surpasses direct KD methods by focusing on relational knowledge around anchor nodes, achieving high accuracy and efficiency.

C. Results and Analysis

1) Comparison With Baselines (Q1): The performance of our method is evaluated through comparison with baseline methods across 12 datasets. We evaluated the performance of our proposed method by comparing it with baseline methods on 12 datasets. To evaluate model performance, each dataset is randomly divided into training and test sets for ten times. In each split, 80% of the links are selected for the training set, and the remaining 20% as the test set. The results of AUC and AUPR are shown in Tables III and IV, respectively, where the best results are highlighted in bold and the second-best are shown by underline.

As observed from the results, embedding-based methods consistently outperform heuristic methods across all datasets. For instance, Node2vec utilizes biased random walks to create node sequences for learning node representations in graphs, which makes it very suitable for processing various graph data, particularly in sparse network settings. However, this method relies on biased walking from prior knowledge, which hinders its effectiveness in dealing with dense networks and complex data. Graph representation learning methods show better performance than heuristic and network embedding-based methods in the link prediction task. This advantage stems from their ability to directly utilize the inherent structure of the graph. For instance, SFGCN achieves competitive results on certain datasets such as Chameleon and Squirrel, its overall performance is less stable across diverse graph structures. In particular, on sparse or perturbation-sensitive networks like Power and NetScience, its accuracy is significantly lower than that of our method. However, it is worth noting that in certain traditional datasets such as Email and NetScience, some heuristic methods demonstrate competitive or even superior performance compared to embedding-based and GNN-based methods. This phenomenon can be attributed to the structural simplicity and sparsity of these networks. In addition, deep learning models may suffer from over-smoothing or underfitting on small or sparse graphs, where the available supervision is limited and neighborhood information lacks diversity. In contrast, heuristics leverage explicit

TABLE III
COMPARISON OF PREDICTION ACCURACY UNDER THE AUC INDICATOR OF THIRTEEN METHODS ON 12 DATASETS

Method	Datasets	Email	NetScience	Power	Chameleon	Wiki	Cora	Citeseer	Squirrel	Photo	Computers	CS	Pubmed
Heuristic	CN	0.8512(±0.48)	0.9611(±0.72)	0.5884(±0.45)	0.8339(±0.54)	0.6954(±0.76)	0.6153(±0.21)	0.5852(±0.43)	0.5965(±0.53)	0.8816(±0.06)	0.8785(±0.76)	0.7153(±0.65)	0.6421(±0.78)
	RA	0.8489(±0.58)	0.9751(±0.32)	0.5874(±0.55)	0.8289(±0.24)	0.6894(±0.72)	0.6163(±0.41)	0.5832(±0.71)	0.5975(±0.43)	0.8796(±0.16)	0.8745(±0.76)	0.7122(±0.58)	0.6483(±0.97)
	LP	0.9154(±0.59)	0.9778(±0.92)	0.6165(±0.45)	0.8345(±0.32)	0.7031(±0.17)	0.6383(±0.39)	0.5868(±0.49)	0.5927(±0.51)	0.8827(±0.64)	0.8895(±0.73)	0.7134(±0.27)	0.6432(±0.96)
	CNDP	0.8635(±0.34)	0.9165(±0.62)	0.5973(±0.43)	0.8079(±0.21)	0.6991(±0.52)	0.6323(±0.86)	0.5802(±0.41)	0.5995(±0.93)	0.8796(±0.16)	0.8825(±0.06)	0.7052(±0.35)	0.6387(±0.29)
	Katz	0.9287(±0.45)	0.9789(±0.52)	0.6538(±0.72)	0.5915(±0.45)	0.5873(±0.98)	0.6460(±0.87)	0.6180(±0.45)	0.5734(±0.87)	0.8101(±0.45)	0.7374(±0.75)	0.7743(±0.92)	0.7563(±0.54)
Embedding	Deepwalk	0.8271(±0.43)	0.8932(±0.23)	0.7456(±0.87)	0.9042(±1.32)	0.8945(±0.76)	0.6381(±1.74)	0.6380(±0.65)	0.8543(±0.29)	0.8037(±0.42)	0.8409(±0.63)	0.7291(±0.76)	0.7885(±0.76)
	Node2vec	0.8471(±0.73)	0.9165(±0.21)	0.7621(±0.83)	0.9138(±0.26)	0.9022(±1.32)	0.6530(±1.43)	0.6063(±0.78)	0.9493(±0.92)	0.8129(±0.64)	0.8402(±0.06)	0.7204(±0.87)	0.7787(±0.76)
GRL	SEAL	0.9352(±0.37)	0.9691(±0.48)	0.8764(±0.98)	0.9376(±0.45)	0.9234(±0.63)	0.8770(±0.97)	0.8144(±1.34)	0.9435(±0.34)	-	-	-	0.9597(±0.08)
	T-BGRL	0.9376(±0.02)	0.9765(±0.87)	0.8976(±0.34)	0.9462(±0.65)	0.9341(±0.32)	0.8792(±0.23)	0.9275(±0.30)	0.9557(±0.21)	0.9730 (±0.40)	0.9636(±0.39)	0.9523(±0.62)	0.9376(±1.23)
	SFGCN	0.9348(±0.44)	0.9623(±0.73)	0.7664(±0.34)	0.9387(±0.14)	0.9272(±0.48)	0.8745(±0.37)	0.8429(±0.86)	0.9534(±0.39)	0.8623(±0.52)	0.8147(±0.11)	0.8138(±0.37)	0.8012(±0.84)
KD	DEAL	0.9233(±0.32)	0.9583(±0.24)	0.9163(±0.86)	0.9241(±0.14)	0.9341(±0.32)	0.8644(±0.76)	0.9374(±0.23)	0.9084(±0.21)	0.9663(±0.76)	0.9537(±1.32)	0.9827(±0.29)	0.9663(±0.54)
	Graph2Feat	0.9023(±0.23)	0.9532(±0.64)	0.9234(±0.43)	0.9559(±0.31)	0.9463(±0.34)	0.9145(±0.34)	0.9131(±0.86)	0.9339(±0.23)	0.9075(±0.53)	0.9227(±0.54)	0.9579(±0.75)	0.9233(±0.04)
	LLP	0.9532(±0.26)	0.9785(±0.34)	0.9434(±0.17)	0.9822(±0.01)	0.9631(±0.12)	0.9466(±0.33)	0.9324(±0.52)	0.9878(±0.01)	0.9468(±1.41)	0.9311(±0.98)	0.9799(±0.11)	0.9808(±0.08)
Ours	LCLRD	**0.9689(±0.86)**	**0.9864(±0.24)**	**0.9576(±0.12)**	**0.9867(±0.01)**	**0.9694(±0.21)**	**0.9607(±0.91)**	**0.9541(±0.21)**	**0.9915(±0.01)**	**0.9881(±0.06)**	**0.9851(±0.05)**	**0.9847(±0.04)**	**0.9829(±0.05)**

Note: Performance comparison with baselines on training percentages (80%) (AUC). "±" represents standard deviation, "-" represents failure to run.

TABLE IV
COMPARISON OF PREDICTION ACCURACY UNDER THE AUC INDICATOR OF THIRTEEN METHODS ON 12 DATASETS

Method	Datasets	Email	NetScience	Power	Chameleon	Wiki	Cora	Citeseer	Squirrel	Photo	Computers	CS	Pubmed
Heuristic	CN	0.5514(±0.76)	0.7511(±0.34)	0.5874(±0.02)	0.6523(±0.62)	0.6324(±0.96)	0.4745(±0.65)	0.4462(±0.21)	0.5764(±0.76)	0.8278(±0.78)	0.8168(±0.05)	0.6133(±0.32)	0.5969(±0.42)
	RA	0.5469(±0.34)	0.7266(±0.18)	0.5884(±0.87)	0.6512(±0.34)	0.6345(±0.33)	0.4587(±0.87)	0.4489(±0.45)	0.5783(±0.87)	0.8196(±0.48)	0.8075(±0.34)	0.6102(±0.81)	0.6483(±0.97)
	LP	0.6154(±0.94)	0.7063(±0.29)	0.6175(±0.54)	0.6598(±0.24)	0.6487(±0.77)	0.4865(±0.91)	0.4552(±0.98)	0.5827(±0.11)	0.8345(±0.43)	0.8234(±0.36)	0.6102(±0.77)	0.6916(±0.62)
	CNDP	0.6635(±0.49)	0.7145(±0.22)	0.5963(±0.36)	0.6345(±0.44)	0.6378(±0.26)	0.4604(±0.66)	0.4512(±0.13)	0.5795(±0.35)	0.8334(±0.61)	0.8334(±0.67)	0.7052(±0.35)	0.6183(±0.92)
	Katz	0.7257(±0.49)	0.7759(±0.92)	0.6548(±0.58)	0.5065(±0.71)	0.5002(±0.95)	0.5245(±0.27)	0.4893(±0.49)	0.4987(±0.43)	0.6169(±0.49)	0.5176(±0.85)	0.7289(±0.34)	0.7139(±0.51)
Embedding	Deepwalk	0.8253(±0.94)	0.8934(±0.22)	0.7451(±0.85)	0.9076(±0.40)	0.9005(±0.72)	0.6591(±1.54)	0.6989(±0.35)	0.8634(±0.14)	0.8020(±0.32)	0.8474(±0.75)	0.7467(±0.43)	0.8075(±0.45)
	Node2vec	0.8421(±0.44)	0.9211(±0.29)	0.7631(±0.82)	0.9151(±0.91)	0.9096(±0.51)	0.6908(±1.98)	0.6560(±0.64)	0.9556(±0.21)	0.8158(±0.63)	0.8487(±0.36)	0.7433(±0.83)	0.8053(±0.34)
GRL	SEAL	0.9313(±0.46)	0.9678(±0.33)	0.8754(±0.78)	0.9189(±0.19)	0.9065(±0.45)	0.8797(±0.98)	0.8145(±1.34)	0.9234(±0.62)	-	-	-	0.9608(±0.28)
	T-BGRL	0.9357(±0.21)	0.9754(±0.24)	0.8973(±0.58)	0.9514(±0.22)	0.9424(±0.21)	0.8904(±0.19)	0.9432(±0.34)	0.9523(±0.40)	0.9721 (±1.40)	0.9628(±0.43)	0.9525(±0.21)	0.9414(±0.23)
	SFGCN	0.9359(±0.93)	0.9612(±0.34)	0.7682(±0.69)	0.9372(±0.27)	0.9267(±0.38)	0.8732(±0.83)	0.8465(±0.06)	0.9521(±0.29)	0.8602(±0.83)	0.8162(±0.28)	0.8176(±0.47)	0.7998(±0.25)
KD	DEAL	0.9228(±0.45)	0.9538(±0.87)	0.9231(±0.58)	0.9301(±1.14)	0.9354(±0.12)	0.8047(±0.43)	0.9077(±0.87)	0.9123(±0.29)	0.9314(±0.22)	0.8994(±0.32)	0.9841(±1.29)	0.9314(±0.53)
	Graph2Feat	0.9027(±0.49)	0.9538(±0.35)	0.9231(±0.66)	0.9607(±0.02)	0.9583(±0.17)	0.9139(±0.32)	0.9177(±0.84)	0.9374(±0.39)	0.9014(±0.13)	0.9197(±0.27)	0.9562(±0.75)	0.9353(±0.25)
	LLP	0.9536(±0.44)	0.9782(±0.23)	0.9437(±0.23)	0.9833(±0.15)	0.9658(±0.07)	0.9510(±0.58)	0.9397(±0.43)	0.9891(±0.02)	0.9354(±1.54)	0.9321(±1.00)	0.9783(±0.16)	0.9780(±0.12)
Ours	LCLRD	**0.9683(±0.49)**	**0.9866(±0.92)**	**0.9572(±0.28)**	**0.9878(±0.03)**	**0.9737(±0.09)**	**0.9592(±0.20)**	**0.9583(±0.29)**	**0.9918(±0.01)**	**0.9856(±0.08)**	**0.9829(±0.07)**	**0.9854(±0.07)**	**0.9800(±0.05)**

Note: Performance comparison with baselines on training percentages (80%) (AUPR). "±" represents standard deviation, "-" represents failure to run.

local topology and remain robust in low-resource settings. Compared to graph representation learning methods, KD approaches show improved performance across all 12 datasets, demonstrating their effectiveness in enhancing model accuracy. Experimental results consistently justify that our LCLRD method achieves superior capability compared to all baseline methods in terms of both evaluation metrics. This achievement is attributed to the fact that this method can learn high-quality node representation to fulfill the best capability in the link prediction task.

2) Model Robustness Analysis (Q2): To demonstrate the effectiveness of our method with insufficient training data, we carry out experiments on 12 datasets using training set sizes ranging from 30% to 80% of the total data, with the rest allocated for testing. Experiments are conducted using different training percentages. The results are illustrated in Figs. 2 and 3 for AUC and AUPR, respectively.

The experiments show that LCLRD yields better results in comparison to other methods. To validate the model's robustness and generalization, we compare LCLRD with heuristic

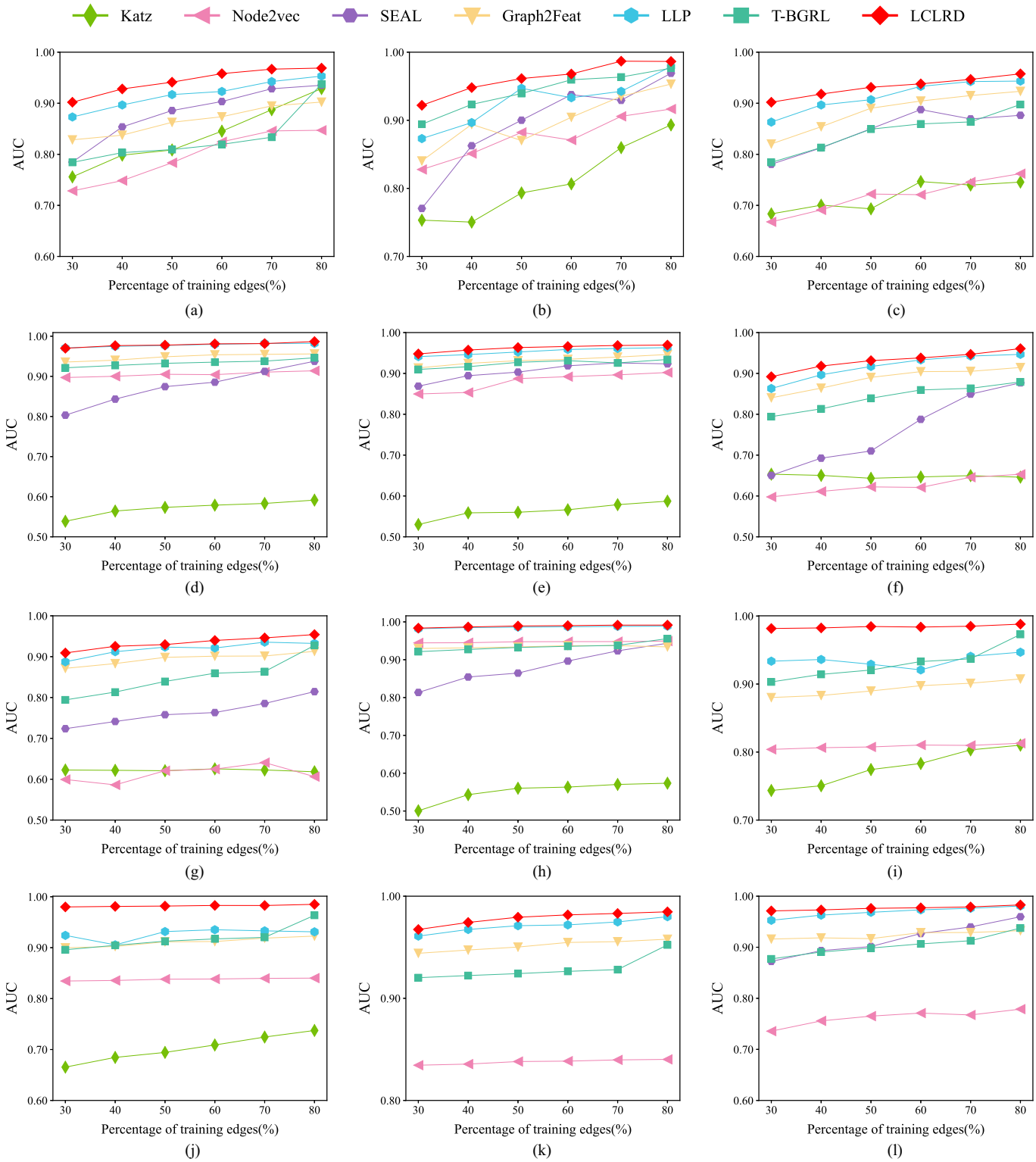

Fig. 2. Robustness analysis on 12 datasets. The predictive performance of seven methods training ratios is measured by AUC. Each data point is calculated by averaging the results of 20 independent runs. The x-axis displays the training set ratio, ranging from 0.2 to 0.8, while the y-axis represents the AUC value. Among the 12 datasets, seven methods in four datasets are not fully presented. Specifically, SEAL fail to run on the computers and photo datasets, Katz and SEAL fail to run on the CS dataset, and Katz fail to run on the Pubmed dataset. (a) Email. (b) NetScience. (c) Power. (d) Chameleon. (e) Wiki. (f) Cora. (g) Citeseer. (h) Squirrel. (i) Photo. (j) Computers. (k) CS. (l) pubmed.

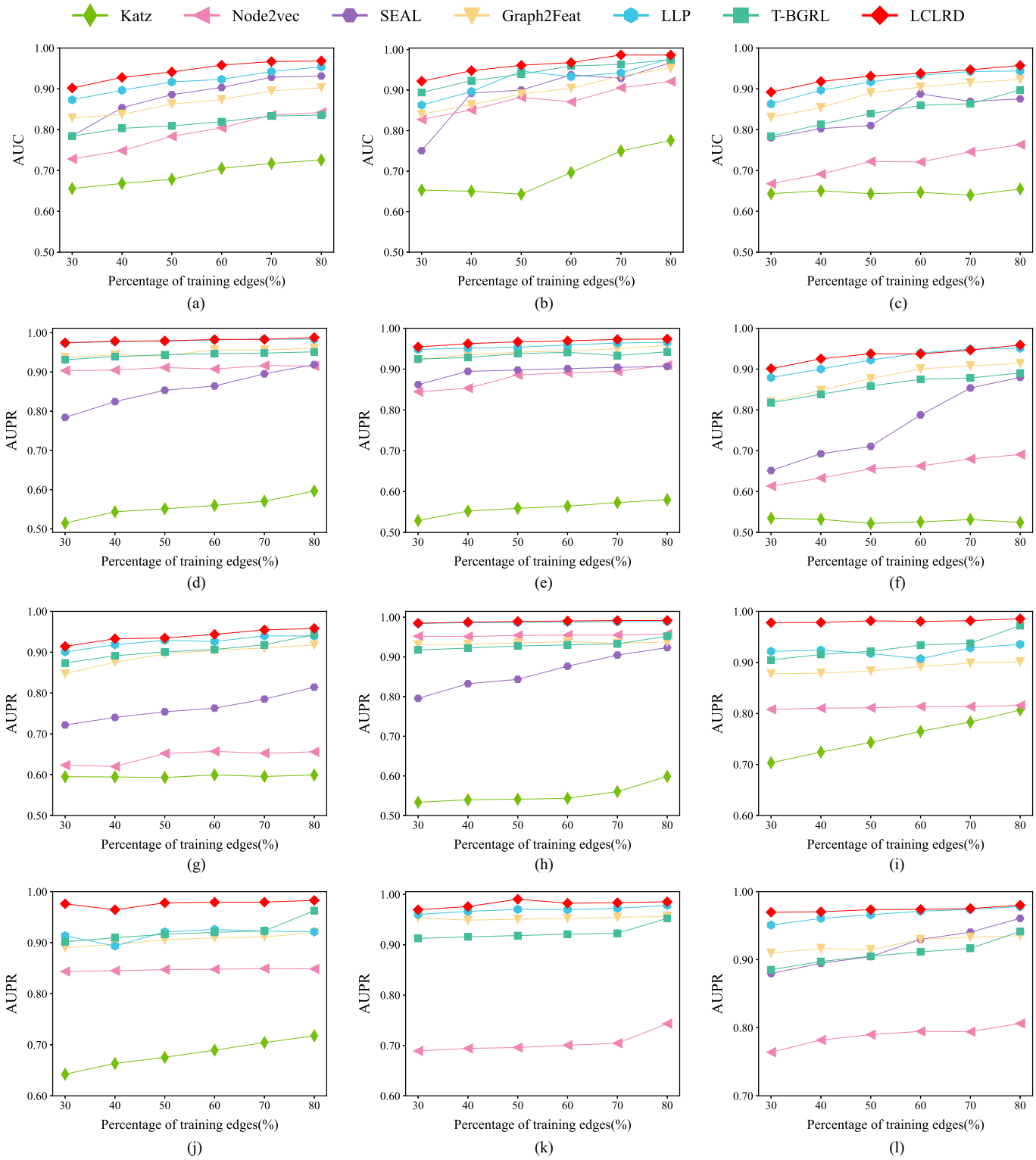

Fig. 3. Robustness analysis on 12 datasets. The predictive performance of seven methods training ratios is measured by AUPR. Each data point is calculated by averaging the results of 20 independent runs. The x-axis displays the training set ratio, ranging from 0.2 to 0.8, while the y-axis represents the AUPR value. Among the 12 datasets, seven methods in four datasets are not fully presented. Specifically, SEAL fail to run on the computers and photo datasets, Katz and SEAL fail to run on the CS dataset, and Katz fail to run on the Pubmed dataset. (a) Email. (b) NetScience. (c) Power. (d) Chameleon. (e) Wiki. (f) Cora. (g) Citeseer. (h) Squirrel. (i) Photo. (j) Computers. (k) CS. (l) pubmed.

TABLE V
PERFORMANCE COMPARISON OF DIFFERENT LINK PREDICTION METHODS ON BA NETWORKS, EVALUATED USING AUC (AUPR) METRICS

Nodes	2000		5000		10000	
	BA	Null model	BA	Null modell	BA	Null model
CN	0.5685 (0.5618)	0.6034 (0.6145)	0.5481 (0.5429)	0.5835 (0.5838)	0.5306 (0.5284)	0.5735 (0.5753)
RA	0.4562 (0.5048)	0.5589 (0.5476)	0.4744 (0.5023)	0.5457 (0.5518)	0.4881 (0.5081)	0.5357 (0.5402)
LP	0.4586 (0.5004)	0.5179 (0.5237)	0.4767 (0.5001)	0.5045 (0.5234)	0.4843 (0.5034)	0.4963 (0.5345)
CNDP	0.5573 (0.5431)	0.5993 (0.5918)	0.5425 (0.5282)	0.5935 (0.5902)	0.5231 (0.5149)	0.5835 (0.5798)
Katz	0.7637 (0.7702)	**0.8034 (0.8123)**	0.7722 (0.7823)	**0.8135 (0.8297)**	0.7736 (0.7834)	**0.8176 (0.8187)**
Deepwalk	0.6845 (0.6834)	0.6695 (0.6676)	0.6898 (0.6868)	0.6574 (0.6534)	0.7035 (0.7023)	0.6466 (0.6426)
Node2vec	0.6943 (0.6947)	0.6674 (0.6656)	0.7054 (0.7051)	0.6732 (0.6745)	0.7134 (0.7128)	0.6773 (0.6763)
SEAL	0.8945 (0.8949)	0.7264 (0.7234)	0.9134 (0.9138)	0.7545 (0.7574)	0.9345 (0.9341)	0.7847 (0.7858)
T-BGRL	0.9138 (0.9134)	0.7345 (0.7365)	0.9383 (0.9378)	0.7664 (0.7648)	0.9476 (0.9464)	0.7903 (0.7922)
SFGCN	0.9243 (0.9213)	0.7342 (0.7371)	0.9375 (0.9368)	0.7624 (0.7619)	0.9421 (0.9403)	0.7743 (0.7741)
DEAL	0.8934 (0.8933)	0.6902 (0.6923)	0.9182 (0.9145)	0.7276 (0.7265)	0.9389 (0.9386)	0.7701 (0.7723)
Graph2Feat	0.8897 (0.8893)	0.6945 (0.6995)	0.9034 (0.9039)	0.7365 (0.7368)	0.9373 (0.9376)	0.7693 (0.7682)
LLP	0.9345 (0.9334)	0.6853 (0.6862)	0.9478 (0.9458)	0.7464 (0.7453)	0.9642 (0.9648)	0.7912 (0.7934)
LCLRD	**0.9456 (0.9451)**	0.6734 (0.6742)	**0.9643 (0.9648)**	0.7034 (0.7044)	**0.9765 (0.9769)**	0.7698 (0.7686)

Note: Bold entries indicate the best results and the second-best are shown by underline.

methods, embedding methods, graph representation learning methods, and KD methods. Our experiments' results indicate that the efficacy of heuristic, embedding, and graph representation learning methods tends to stabilize as the training set scale varies, but they decrease compared to KD methods. On the NetScience dataset, we observe that heuristic methods (e.g., Katz) show significant performance fluctuations as the training edge ratio increases from 30% to 80%. This instability is mainly due to their strong dependence on local structural features, which are unreliable when the graph is sparse. In contrast, LCLRD maintains consistently high performance, benefiting from its ability to extract stable representations through contrastive learning and relational distillation, even under severe edge sparsity. In particular, On the CS dataset, the KD method shows excellent performance, effectively reducing inference time while simultaneously enhancing the overall performance of the model. Despite demonstrating notable advantages in enhancing link prediction performance, this method's strong reliance on labeled data, and its efficiency increases slowly with increasing training dataset size limit its applicability. This further demonstrates our method's effectiveness. Our experiments clearly demonstrate that LCLRD is insensitive to the quantity of training samples, and outperforms other baseline methods with various levels of network sparsity.

3) Structural Robustness Evaluation With Null Models (Q3): The previous robustness analysis primarily focused on label sparsity, while in this section, we further investigate the model's stability under structural perturbations. Inspired by the null model testing methodology introduced in [60] and [61], we evaluate whether LCLRD can effectively distinguish between genuine structural patterns and randomized ones. To this end, we construct synthetic graphs at three different scales (2000, 5000, and 10 000 nodes) using the Watts–Strogatz (WS) and Barabási–Albert (BA) models to simulate structurally perturbed scenarios. In addition, we apply a randomized edge rewiring strategy (RE) [62] to construct null models while preserving the degree distribution of each node. Specifically, we perform edge swaps by randomly selecting two edges and switching their endpoints, which disrupts the global structure without altering node degrees. The number of such rewiring operations is set equal to the number of nodes in the graph. This allows us to isolate the influence of structural information on link prediction performance.

The experimental results, as shown in Tables V and VI, highlight distinct sensitivities of various approaches to structural perturbations. Heuristic methods, which rely on shallow, local statistics rather than meaningful topological features, exhibit stable performance under null models. Embedding-based methods show moderate performance drops, especially on scale-free (BA) graphs, suggesting a partial sensitivity to underlying structure. GNN-based models experience more pronounced degradation, reflecting their higher dependence on meaningful topology. For instance, SFGCN exhibits strong performance on original BA and WS graphs, indicating its capacity to learn from graph structures. However, it suffers significant accuracy drops under the null models—particularly on BA-10000 and WS-10000 settings—suggesting that its structural generalization is less robust than that of our method. This instability may stem from its structure-free design, which limits its ability to resist structural perturbations. Notably, our proposed LCLRD model, which integrates contrastive learning and relational KD, demonstrates the most significant performance decline across both WS and BA null graphs. This substantial drop indicates a strong reliance on genuine structural signals, highlighting its structural sensitivity and superior generalization when applied to real-world networks. In practical terms, this also implies potential vulnerability to adversarial manipulations, where deliberately randomized or misleading inputs may mislead the model. Thus, while LCLRD excels in capturing real structure, future work should consider strategies to enhance its robustness against adversarial attacks.

TABLE VI
PERFORMANCE COMPARISON OF DIFFERENT LINK PREDICTION METHODS ON WS NETWORKS, EVALUATED USING AUC (AUPR) METRICS

Nodes	2000		5000		10000	
	WS	Null model	WS	Null modell	WS	Null model
CN	0.4933 (0.5238)	0.6084 (0.6283)	0.4993 (0.4995)	0.6473 (0.6421)	0.5045 (0.5044)	0.6834 (0.6842)
RA	0.4934 (0.5028)	0.5937 (0.6021)	0.4973 (0.5056)	0.6383 (0.6439)	0.4983 (0.5006)	0.6813 (0.6931)
LP	0.4895 (0.5003)	0.6035 (0.6237)	0.4934 (0.5012)	0.6395 (0.6472)	0.4965 (0.4987)	0.6583 (0.6598)
CNDP	0.4818 (0.5089)	0.5935 (0.6193)	0.4831 (0.4998)	0.6353 (0.6472)	0.5003 (0.5021)	0.6684 (0.6676)
Katz	0.5374 (0.5445)	0.6395 (0.6502)	0.5352 (0.5393)	0.6846 (0.6878)	0.5349 (0.5371)	0.7482 (0.7491)
Deepwalk	0.5942 (0.5935)	0.5912 (0.5932)	0.6245 (0.6252)	0.6203 (0.6231)	0.6492 (0.6486)	0.6431 (0.6428)
Node2vec	0.6034 (0.6024)	0.6002 (0.6014)	0.6432 (0.6442)	0.6289 (0.6264)	0.6593 (0.6583)	0.6521 (0.6522)
SEAL	0.8745 (0.8735)	0.6748 (0.6798)	0.9034 (0.9031)	0.7034 (0.7045)	0.9342 (0.9352)	0.7362 (0.7344)
T-BGRL	0.8672 (0.8662)	**0.7394 (0.7387)**	0.8913 (0.8924)	**0.7424 (0.7429)**	0.9274 (0.9285)	**0.7582 (0.7579)**
SFGCN	0.8843 (0.8813)	0.7043 (0.7021)	0.9275 (0.9265)	0.7264 (0.7219)	0.9328 (0.9303)	0.7441 (0.7451)
DEAL	0.8578 (0.8573)	0.7046 (0.7066)	0.8743 (0.8752)	0.7284 (0.7276)	0.8876 (0.8879)	0.7362 (0.7359)
Graph2Feat	0.8655 (0.8659)	0.6973 (0.6987)	0.8976 (0.8972)	0.7238 (0.7246)	0.9176 (0.9166)	0.7382 (0.7369)
LLP	0.9239 (0.9242)	0.7074 (0.7048)	0.9472 (0.9469)	0.7183 (0.7179)	0.9631 (0.9638)	0.7284 (0.7268)
LCLRD	**0.9345 (0.9342)**	0.6703 (0.6732)	**0.9564 (0.9568)**	0.6923 (0.6938)	**0.9687 (0.9679)**	0.7163 (0.7157)

Note: Bold entries indicate the best results and the second-best are shown by underline.

TABLE VII
LINK PREDICTION EVALUATION METRICS MEASURED BY AUC (AUPR) ON 12 DATASETS WITH DIFFERENT GNN TEACHERS (MLP, GCN, GAT, SAGE, AND LCLRD)

Datasets	MLP	GCN	GAT	SAGE	LCLRD
Email	0.9234 (0.9188)	0.9543 (0.9541)	0.9340 (0.9294)	0.9484 (0.9493)	**0.9621 (0.9616)**
NetScience	0.9409 (0.9448)	0.9647 (0.9641)	0.9545 (0.9532)	0.9748 (0.9753)	**0.9803 (0.9806)**
Power	0.8934 (0.9032)	0.9295 (0.9265)	0.9334 (0.9384)	0.9248 (0.9272)	**0.9576 (0.9534)**
Chameleon	0.9592 (0.9612)	0.9739 (0.9740)	0.9706 (0.9670)	0.9818 (0.9809)	**0.9841 (0.9865)**
Wiki	0.9503 (0.9557)	0.9586 (0.9617)	0.9592 (0.9638)	0.9591 (0.9636)	**0.9608 (0.9643)**
Cora	0.8921 (0.8902)	0.9089 (0.9196)	0.9210 (0.9309)	0.9476 (0.9545)	**0.9506 (0.9563)**
Citeseer	0.9261 (0.9315)	0.9329 (0.9428)	0.9289 (0.9381)	0.9505 (0.9546)	**0.9563 (0.9602)**
Squirrel	0.9777 (0.9774)	0.9817 (0.9826)	0.9809 (0.9803)	0.9868 (0.9877)	**0.9891 (0.9899)**
Photo	0.9607 (0.9505)	0.9722 (0.9671)	0.9621 (0.9500)	0.9711 (0.9643)	**0.9756 (0.9709)**
Computers	0.9522 (0.9437)	0.9661 (0.9636)	0.9498 (0.9344)	0.9718 (0.9670)	**0.9749 (0.9704)**
CS	0.9622 (0.9545)	0.9671 (0.9659)	0.9662 (0.9638)	0.9740 (0.9775)	**0.9848 (0.9779)**
Pubmed	0.9209 (0.9088)	0.9575 (0.9541)	0.9270 (0.9184)	0.9448 (0.9490)	**0.9833 (0.9816)**

Note: Bold entries indicate the best results and the second-best are shown by underline.

4) Efficiency Analysis: We concentrate on analyzing the performance enhancement of the teacher model by contrastive learning and evaluating the utility of the PCC in KD to deal with the following two crucial questions.

a) How does the proposed contrastive learning module benefit the overall model? (Q4): As discussed in Section I, although KD improves inference speed, it is difficult to learn robust node embedding owing to the deficiency of label information. As shown in Table VII, we compare our method with KD approaches using MLP, GCN, GAT, and SAGE as teacher models on 12 datasets. It can be observed that leveraging contrastive learning in the teacher model of LCLRD leads to superior link prediction results, compared to LCLRD with SAGE as the teacher model. The reason is that, unlike

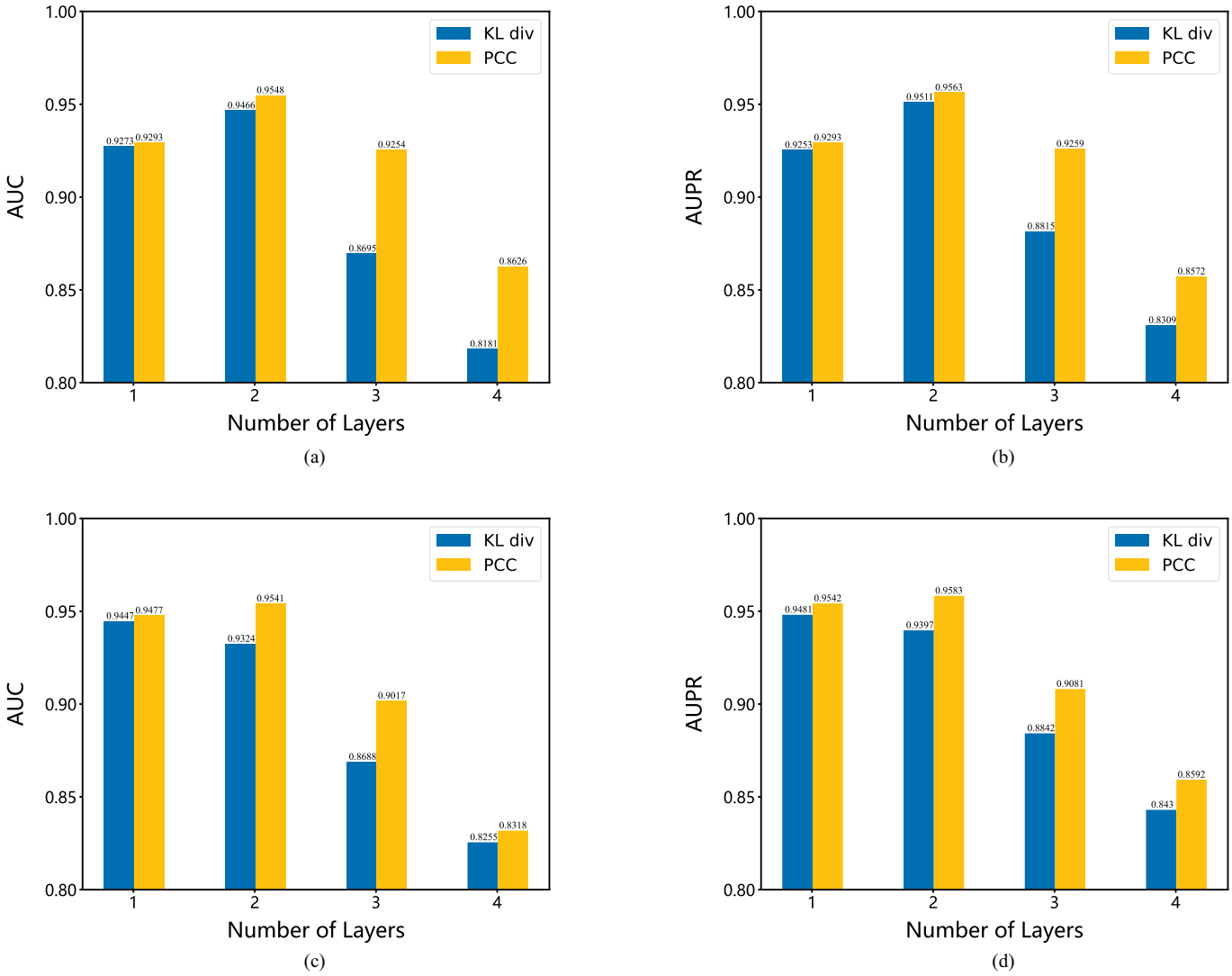

Fig. 4. We Compare the performance of KD models with various layers applying KL divergence and PCC as loss functions, under the metrics of AUC and AUPR. (a) Cora-AUC. (b) Cora-AUPR. (c) Citeseer-AUC. (d) Citeseer-AUPR.

traditional supervised learning, contrastive learning does not need a lot of labeled data, and can effectively utilize data without labels for model training. Consequently, the better the teacher model's performance, the better the learning effect of the student model.

b) What is the utility of the proposed PCC in KD? (Q5): We explore the performance differences of two KD methods, the KL divergence and the PCC, at different layers of the model. As shown in Fig. 4, on the Cora dataset, with increasing model layers, both methods exhibit a performance decline trend after reaching a peak at the second layer. This may be because with the increase of network depth, the model's internal representation becomes more complicated. Thereby, the training difficulty is increased, leading to a decline in final performance. However, it is crucial to note that in the third and fourth layers, the performance of the KL divergence method is obviously behind the PCC method. This phenomenon shows that when the teacher model is relatively complex, the PCC method may be more effective than the KL divergence method for KD. This is attributed to the PCC method's ability to extract key knowledge information more robustly and transfer it to the student model.

D. Ablation Study (Q6)

In this section, we conduct a comprehensive investigation into the impact of different components in the link prediction task. The study is conducted under three settings: 1) using only the KD model; 2) using the KD model with contrastive learning (KD + CL); and 3) using the complete LCLRD model. The experimental results are shown in Fig. 5.

Comparison between the KD and KD + CL models across 12 datasets reveals a significant performance improvement in link prediction when contrastive learning is incorporated. This is primarily attributed to the contrastive learning's ability to empower models to obtain robust representations, enabling effective learning even without labeled data. The teacher model's robustness positively impacts the student model's learning efficacy (efficiency analysis experiments validate this phenomenon). On the dataset of Photo, a significant performance deficit between the KD + CL and the LCLRD models is observed. This disparity

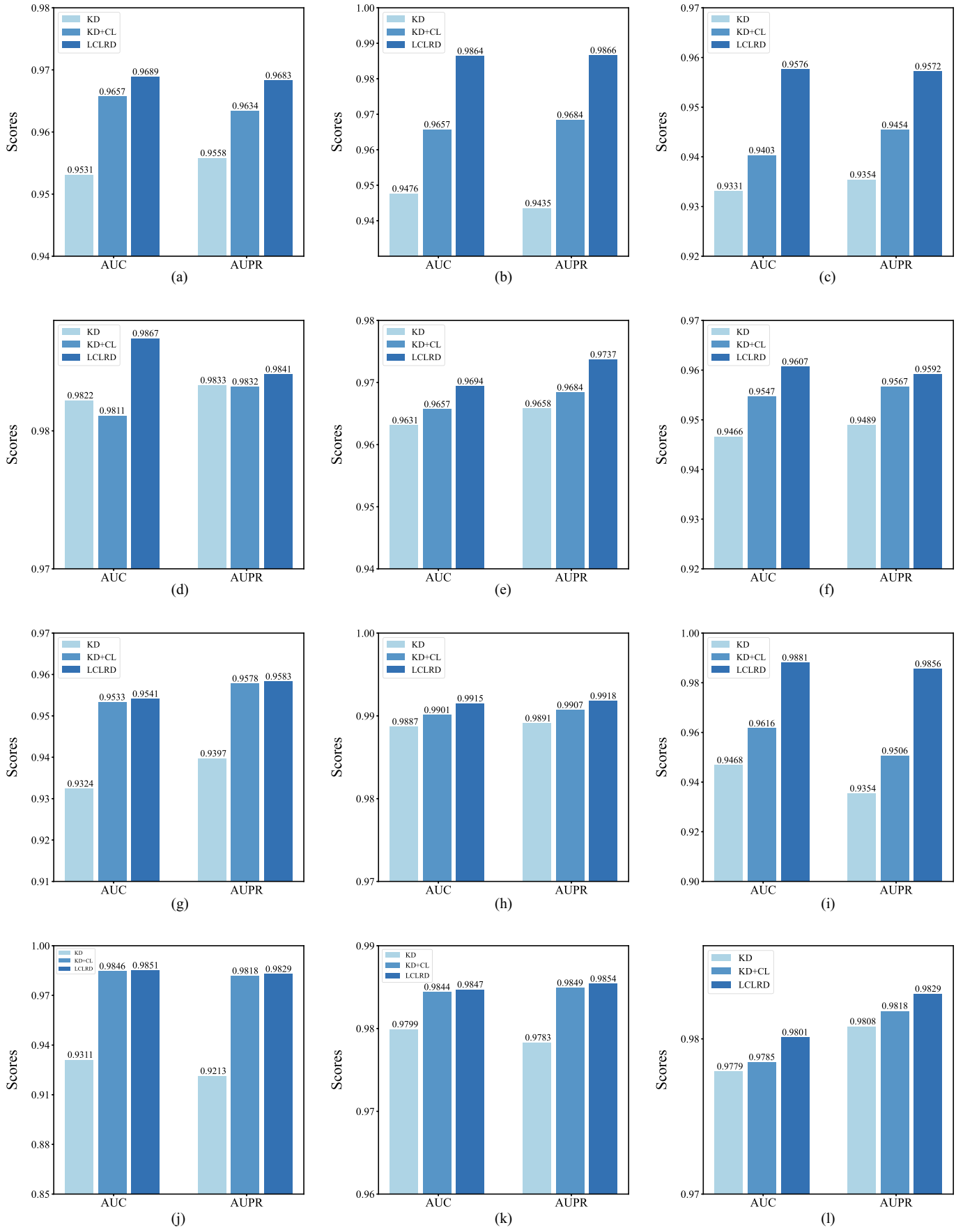

Fig. 5. Ablation study on 12 datasets. AUC and AUPR were used for link prediction tasks on KD, KD + CL, and LCLRD, respectively. (a) Email. (b) NetScience. (c) Power. (d) Chameleon. (e) Wiki. (f) Cora. (g) Citeseer. (h) Squirrel. (i) Photo. (j) Computers. (k) CS. (l) pubmed.

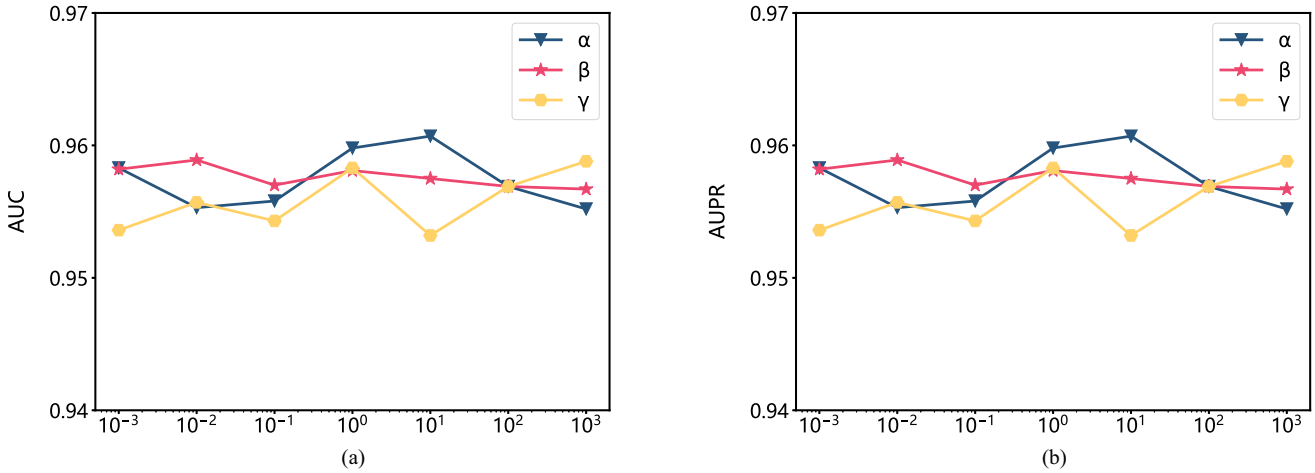

Fig. 6. This article presents the experimental results of a parameter sensitivity analysis. The hyperparameters α, β, and γ correspond to the true label loss, rank matching loss, and distribution matching loss, respectively. (a) Cora-AUC. (b) Cora-AUPR.

can be explained by the fact that stronger teacher models not only have larger model capacities but also employ more effective training strategies. Contrastive learning, as an excellent strategy for training models, exhibits significant differences in performance between KL divergence and PCC when the number of model layers remains constant. Experimental results demonstrate that the LCLRD model consistent performance gains on all 12 datasets when employing the same two-layer GCN model architecture. In addition, the results show that the PCC can realize better results when the teacher model adopts stronger strategies.

E. Parametric Sensitivity Analysis (Q7)

The three hyperparameters α, β and γ, in (18), are essential to our framework. Due to efficiency limitations, we are only able to execute experiments on the small-scale Cora dataset. Fig. 6 shows the experimental results of AUC and AUPR obtained when fixing the other two parameters. To study how different values of α affect model performance, we set $\beta = 1$, $\gamma = 1$ and vary α from $\{10^{-3}, 10^{-2}, 10^{-1}, 10^{0}, 10^{1}, 10^{2}, 10^{3}\}$. Similarly, the same evaluation is also carried out for β and γ.

As shown in Fig. 6, setting α, β, and γ to 1 produces satisfying results. It is observed that the LCLRD's performance is not significantly affected by the selection of these parameters. This means that LCLRD method is not sensitive to the choice of parameters, and can obtain better performance in a flexible range of parameters, reducing the difficulty of hyperparameter tuning. We also note that LCLRD's performance is more influenced by the values of α and γ, especially on Cora dataset. This phenomenon indicates that model performance is influenced by both true label and distribution matching loss, further evidence for our method's effectiveness.

F. Failure Case Analysis and Discussion

Although the proposed LCLRD model demonstrates superior predictive performance across 12 public datasets, we also observe that in certain specific scenarios, the model's performance improvement is limited, or even degraded. A deeper analysis of these cases can help us better understand the applicability boundaries of LCLRD and provide insights for future research.

1) Challenges From Graph Structural Complexity and Heterogeneity: In datasets such as Chameleon and Squirrel, which exhibit significant structural heterogeneity and low node homophily, LCLRD shows limited advantages over some baseline methods. We attribute this to the difficulty contrastive learning faces in constructing representative positive and negative pairs when neighboring nodes exhibit substantial semantic differences. In such graphs, context subgraphs generated through local sampling may fail to accurately capture global semantic structures, thus limiting the effectiveness of contrastive learning.

2) Impact of Network Depth on Student Model Performance: As shown in Fig. 4 and Table I, increasing the number of layers in the teacher model leads to a noticeable performance drop in the student model when using KL divergence as the distillation objective. Although introducing the PCC alleviates overfitting to some extent, deeper network structures can still cause degradation in the distillation effect due to increased representation complexity and unstable gradient propagation. This indicates that traditional distillation strategies face adaptability challenges when the teacher model is overly strong.

3) Limitations of the Subgraph Sampling Strategy: Although this study introduces the PPR algorithm to enhance subgraph sampling quality, in scenarios with uneven node distribution or sparse connections, important neighbors may still be missed. This sampling uncertainty directly affects the quality of positive and negative pairs, thereby influencing the performance of contrastive learning and the final representation quality.

4) Generalization to Sparse and Structurally Distinctive Networks: In real-world applications, many networks exhibit structural sparsity or distinctive topologies, such as tree-like

or long-cycle formations. Our method demonstrates strong potential in such settings. For *sparse networks* (e.g., Twitter graphs and sexual contact networks), the contrastive learning framework in LCLRD enhances representation robustness by alleviating the reliance on dense connectivity. This is supported by performance gains observed on sparse datasets such as NetScience and Power. For *tree-like and long-cycle networks*, the proposed *relational distillation module*, which integrates ranking-based alignment and PCC-based similarity matching, effectively captures both hierarchical dependencies and long-range structural correlations. These capabilities suggest that LCLRD is well-suited for generalization across a wide range of topologically diverse networks.

VI. Conclusion

This article proposes a novel method, named LCLRD, which is based on the KD framework of contrastive learning to capture robust representation for link prediction. Compared to current KD methods and some classical heuristic, and embedding methods, it achieves state-of-the-art performances in most cases. The effectiveness of generating high-quality embeddings by contrastive learning is deeply studied. We find that contrastive learning can capture deeper semantic information between nodes and make the generated representations more robust, which enables LCLRD to achieve better performance on complex link prediction task. In addition, methods for selecting distance metrics to calculate probability distributions are explored from the KD's perspective. It is discovered that appropriate distance metrics can effectively measure the discrepancy between the student and teacher models, resulting in further improving the efficiency of KD.

The proposed LCLRD, combining contrastive learning and KD, achieves outstanding performance in link prediction tasks. We find that contrastive learning captures deeper semantic information between nodes, making the generated representations more robust and thereby improving the model's performance in complex scenarios. Furthermore, we explore the potential applications of LCLRD in various real-world domains, including social network analysis, recommendation systems, and biological network research. In social network analysis, LCLRD can be used for friend recommendation and community detection; in recommendation systems, it can accurately predict user interests, enhancing recommendation quality; in biological network research, it aids in discovering potential protein interactions. In addition, we analyze the impact of different distance metrics on model performance from the perspective of KD and finding that the appropriate metric can effectively measure the discrepancy between the student and teacher model, further improving knowledge transfer efficiency. In the future, we plan to further optimize the computational efficiency of LCLRD and explore its applicability in sparse data environments to expand its application in more practical tasks.

Data Availability Statement

All source code is publicly available at https://github.com/yabingyao/LCLRD4LinkPrediction.

References

[1] D. Liben-Nowell and J. Kleinberg, "The link prediction problem for social networks," in *Proc. 12th Int. Conf. Inf. Knowl. Manage.*, 2003, pp. 556–559.

[2] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *Proc. 15th Int. Conf. Semantic Web (ESWC)*, Heraklion, Crete, Greece: Springer, 2018, pp. 593–607.

[3] M. Al Hasan, V. Chaoji, S. Salem, and M. Zaki, "Link prediction using supervised learning," in *Proc. Workshop Link Anal., Counter-Terrorism Secur. (SDM)*, vol. 30, 2006, pp. 798–805.

[4] L. Lü and T. Zhou, "Link prediction in complex networks: A survey," *Physica A*, vol. 390, no. 6, pp. 1150–1170, 2011.

[5] E. M. Airoldi, D. Blei, S. Fienberg, and E. Xing, "Mixed membership stochastic blockmodels," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 21, 2008, pp. 1981–2014.

[6] L. A. Adamic and E. Adar, "Friends and neighbors on the web," *Social Netw.*, vol. 25, no. 3, pp. 211–230, 2003.

[7] H. Wang et al., "Knowledge-aware graph neural networks with label smoothness regularization for recommender systems," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2019, pp. 968–977.

[8] J. You, Y. Wang, A. Pal, P. Eksombatchai, C. Rosenburg, and J. Leskovec, "Hierarchical temporal convolutional networks for dynamic recommender systems," in *Proc. World Wide Web Conf.*, 2019, pp. 2236–2246.

[9] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1025–1035.

[10] Z. Zhang, P. Cui, and W. Zhu, "Deep learning on graphs: A survey," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 1, pp. 249–270, Jan. 2022.

[11] M. Zhang and Y. Chen, "Link prediction based on graph neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, 2018, pp. 5171–5181.

[12] L. Cai, J. Li, J. Wang, and S. Ji, "Line graph neural networks for link prediction," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 9, pp. 5103–5113, Sep. 2022.

[13] J. Gou, B. Yu, S. J. Maybank, and D. Tao, "Knowledge distillation: A survey," *Int. J. Comput. Vis.*, vol. 129, no. 6, pp. 1789–1819, 2021.

[14] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015, *arXiv:1503.02531*.

[15] S. Zhang, Y. Liu, Y. Sun, and N. Shah, "Graph-less neural networks: Teaching old MLPs new tricks via distillation," 2021, *arXiv:2110.08727*.

[16] W. Zheng, E. W. Huang, N. Rao, S. Katariya, Z. Wang, and K. Subbian, "Cold Brew: Distilling graph node representations with incomplete or missing neighborhoods," 2021, *arXiv:2111.04840*.

[17] Y. Hu, H. You, Z. Wang, Z. Wang, E. Zhou, and Y. Gao, "Graph-MLP: Node classification without message passing in graph," 2021, *arXiv:2106.04051*.

[18] A. E. Samy, Z. T. Kefato, and S. Girdzijauskas, "Graph2Feat: Inductive link prediction via knowledge distillation," in *Companion Proc. ACM Web Conf.*, 2023, pp. 805–812.

[19] L. Wu, H. Lin, C. Tan, Z. Gao, and S. Z. Li, "Self-supervised learning on graphs: Contrastive, generative, or predictive," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 4, pp. 4216–4235, Apr. 2023.

[20] Y. Liu et al., "Graph self-supervised learning: A survey," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 6, pp. 5879–5900, Jun. 2023.

[21] K. Pearson, "VII. Mathematical contributions to the theory of evolution.—III. Regression, heredity, and panmixia," in *Proc. Philos. Trans. Roy. Soc. London. Ser. A, Containing Papers Math. Phys. Character*, 1896, pp. 253–318.

[22] Y. Yao et al., "Deep non-negative matrix factorization with edge generator for link prediction in complex networks," *Appl. Intell.*, vol. 54, no. 1, pp. 592–613, 2024.

[23] Y. Yao et al., "Link prediction based on the mutual information with high-order clustering structure of nodes in complex networks," *Physica A*, vol. 610, no. 2, 2023, Art. no. 128428.

[24] A. Kumar, S. S. Singh, K. Singh, and B. Biswas, "Link prediction techniques, applications, and performance: A survey," *Physica A*, vol. 553, no. 6, 2020, Art. no. 124289.

[25] I. Ahmad, M. U. Akhtar, S. Noor, and A. Shahnaz, "Missing link prediction using common neighbor and centrality based parameterized algorithm," *Sci. Rep.*, vol. 10, no. 1, 2020, Art. no. 364.

[26] F. Aziz, H. Gul, I. Muhammad, and I. Uddin, "Link prediction using node information on local paths," *Physica A*, vol. 557, 2020, Art. no. 124980.

[27] L. Katz, "A new status index derived from sociometric analysis," *Psychometrika*, vol. 18, no. 1, pp. 39–43, 1953.
[28] S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine," *Comput. Netw. ISDN Syst.*, vol. 30, nos. 1–7, pp. 107–117, 1998.
[29] G. Wang, Y. Wang, J. Li, and K. Liu, "A multidimensional network link prediction algorithm and its application for predicting social relationships," *J. Comput. Sci.*, vol. 53, 2021, Art. no. 101358.
[30] A. Grover and J. Leskovec, "Node2vec: Scalable feature learning for networks," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2016, pp. 855–864.
[31] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 26, 2013, pp. 3111–3119.
[32] Y. Xiao, R. Li, X. Lu, and Y. Liu, "Link prediction based on feature representation and fusion," *Inf. Sci.*, vol. 548, pp. 1–17, 2021, doi: 10.1016/j.ins.2020.09.039.
[33] E. P. Barracchia, G. Pio, A. Bifet, H. M. Gomes, B. Pfahringer, and M. Ceci, "LP-robin: Link prediction in dynamic networks exploiting incremental node embedding," *Inf. Sci.*, vol. 606, pp. 702–721, 2022, doi: 10.1016/j.ins.2022.05.079.
[34] F. Hu, Y. Zhu, S. Wu, W. Huang, L. Wang, and T. Tan, "GraphAIR: Graph representation learning with neighborhood aggregation and interaction," *Pattern Recognit.*, vol. 112, 2021, Art. no. 107745.
[35] L. Cai and S. Ji, "A multi-scale approach for graph link prediction," in *Proc. AAAI Conf. Artif. Intell.*, vol. 34, no. 4, 2020, pp. 3308–3315.
[36] D. Huang and F. Lei, "Temporal group-aware graph diffusion networks for dynamic link prediction," *Inf. Process. Manage.*, vol. 60, no. 3, 2023, Art. no. 103292.
[37] S. Sun, W. Ren, J. Li, R. Wang, and X. Cao, "Logit standardization in knowledge distillation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2024, pp. 15731–15740.
[38] B. Heo, J. Kim, S. Yun, H. Park, N. Kwak, and J. Y. Choi, "A comprehensive overhaul of feature distillation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 1921–1930.
[39] Y. Tian, D. Krishnan, and P. Isola, "Contrastive representation distillation," 2019, *arXiv:1910.10699*.
[40] Y. Xie, H. Wu, Y. Lin, J. Zhu, and H. Zeng, "Pairwise difference relational distillation for object re-identification," *Pattern Recognit.*, vol. 152, 2024, Art. no. 110455.
[41] Z. Li, Y. Ge, X. Yue, and L. Meng, "MCAD: Multi-classification anomaly detection with relational knowledge distillation," *Neural Comput. Appl.*, vol. 36, pp. 1–15, 2024.
[42] Z. Guo et al., "Linkless link prediction via relational distillation," in *Proc. Int. Conf. Mach. Learn.*, PMLR, 2023, pp. 12012–12033.
[43] X. Liu et al., "Self-supervised learning: Generative or contrastive," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 1, pp. 857–876, Jan. 2023.
[44] Y. You, T. Chen, Y. Sui, T. Chen, Z. Wang, and Y. Shen, "Graph contrastive learning with augmentations," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, pp. 5812–5823, 2020.
[45] P. Veličković, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, and R. D. Hjelm, "Deep Graph Infomax," 2018, *arXiv:1809.10341*.
[46] S. Wan et al., "Boosting graph contrastive learning via adaptive sampling," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 11, pp. 15971–15983, Nov. 2024.
[47] W. Liao, Y. Zhu, Y. Li, Q. Zhang, Z. Ou, and X. Li, "RevGNN: Negative sampling enhanced contrastive graph learning for academic reviewer recommendation," *ACM Trans. Inf. Syst.*, vol. 43, no. 1, pp. 1–26, 2024.
[48] Y. Feng et al., "CP-prompt: Composition-based cross-modal prompting for domain-incremental continual learning," in *Proc. 32nd ACM Int. Conf. Multimedia*, 2024, pp. 2729–2738.
[49] G. Jeh and J. Widom, "Scaling personalized web search," pp. 271–279, 2003, doi: 10.1145/775152.775191.
[50] F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: A unified embedding for face recognition and clustering," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 815–823.
[51] D. B. Duncan, "A Bayesian approach to multiple comparisons," *Technometrics*, vol. 7, no. 2, pp. 171–222, 1965.
[52] J. Davis and M. Goadrich, "The relationship between precision-recall and roc curves," in *Proc. 23rd Int. Conf. Mach. Learn.*, 2006, pp. 233–240.
[53] M. E. Newman, "Clustering and preferential attachment in growing networks," *Phys. Rev. E*, vol. 64, no. 2, 2001, Art. no. 025102.
[54] T. Zhou, L. Lü, and Y.-C. Zhang, "Predicting missing links via local information," *Eur. Phys. J. B*, vol. 71, pp. 623–630, 2009, doi: 10.1140/epjb/e2009-00335-8.
[55] S. Rafiee, C. Salavati, and A. Abdollahpouri, "CNDP: Link prediction based on common neighbors degree penalization," *Physica A*, vol. 539, 2020, Art. no. 122950.
[56] B. Perozzi, R. Al-Rfou, and S. Skiena, "DeepWalk: Online learning of social representations," pp. 701–710, 2014, doi: 10.1145/2623330.2623732.
[57] W. Shiao, Z. Guo, T. Zhao, E. E. Papalexakis, Y. Liu, and N. Shah, "Link prediction with non-contrastive learning," 2022, *arXiv:2211.14394*.
[58] S.-W. Lee et al., "SFGCN: Synergetic fusion-based graph convolutional networks approach for link prediction in social networks," *Inf. Fusion*, vol. 114, 2025, Art. no. 102684.
[59] Y. Hao, X. Cao, Y. Fang, X. Xie, and S. Wang, "Inductive link prediction for nodes having only attribute information," 2020, *arXiv:2007.08053*.
[60] K. Shang, T. Li, M. Small, D. Burton, and Y. Wang, "Link prediction for tree-like networks," *Chaos, Interdisciplinary J. Nonlinear Sci.*, vol. 29, no. 6, 2019.
[61] K. Shang and M. Small, "Link prediction for long-circle-like networks," *Phys. Rev. E*, vol. 105, no. 2, 2022, Art. no. 024311.
[62] S. Maslov and K. Sneppen, "Specificity and stability in topology of protein networks," *Science*, vol. 296, no. 5569, pp. 910–913, 2002.