Information Sciences 719 (2025) 122482

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

SCS: Subgraph contrastive supervised neural network for link prediction

Qiming Yang [a,b], Wei Wei [a,b,c,d,*], Ruizhi Zhang [a,b], Xiangnan Feng [e]

[a] School of Mathematical Sciences, Beihang University, Beijing, China
[b] Key Laboratory of Mathematics Informatics Behavioral Semantics, Ministry of Education, China
[c] Institute of Artificial Intelligence, Beihang University, Beijing, China
[d] School of Mathematics and Computer, Hengshui University, China
[e] Complexity Science Hub Vienna, Josefstädter Straße 39, 1080 Wien, Austria

ARTICLE INFO

Keywords:
Graph representation learning
Contrastive learning
Link prediction
Graph neural networks

ABSTRACT

Link prediction is a crucial task in network analysis that aims to predict missing or potential links between nodes, with applications spanning social sciences, biology, and computer science. State-of-the-art methods have successfully converted this problem into a binary graph classification task by extracting h-hop subgraph structures. However, this approach blocks information flow outside of h-hop subgraphs and requires additional memory. To address these limitations, we propose an end-to-end link prediction graph neural network incorporating a contrastive learning component. Specifically, we utilize cross-scale contrastive learning to entrench subgraph information by maximizing mutual information between h-hop subgraph information and node representations around the target link. Without explicitly extracting subgraph structures, the proposed method can update node representation with global information while obviating the requirements for additional memory. Extensive experimental results across both plain and attribute graphs demonstrate that our proposed method achieves consistently competitive performance, outperforming other state-of-the-art methods in most cases with satisfying computation cost and fast convergence.

1. Introduction

Graphs are a widely used and important data structure, which has garnered significant attention in recent research [1]. Extensive research has been conducted in various domains of graph data analysis, including but not limited to community detection [2], node classification [3], graph classification [4], and link prediction [5]. Specifically, link prediction is a crucial task that involves identifying potential links [6]. This task finds relevance among numerous real-world applications such as social network friend recommendation [7], targeted advertising in e-commerce networks [8], molecular network analysis [9], and knowledge graph completion [10].

As a basic approach, many heuristic methods have been proposed to tackle the link prediction problem by providing a similarity score for each unconnected node pair [6]. However, they usually have limited expressiveness and require extensive experimentation to select an appropriate method. Therefore, advanced methods such as WLNM [11], SEAL [12], and LGLP [13] have been proposed to automatically learn more sophisticated representations of network data, thereby overcoming the limitations of heuristic methods.

* Corresponding author at: School of Mathematical Sciences, Beihang University, Beijing, China.
E-mail addresses: asdyqm@buaa.edu.cn (Q. Yang), weiw@buaa.edu.cn (W. Wei), ruizhiz@buaa.edu.cn (R. Zhang), fengxiangnan@gmail.com (X. Feng).

https://doi.org/10.1016/j.ins.2025.122482
Received 17 September 2023; Received in revised form 1 July 2025; Accepted 2 July 2025

Available online 10 July 2025

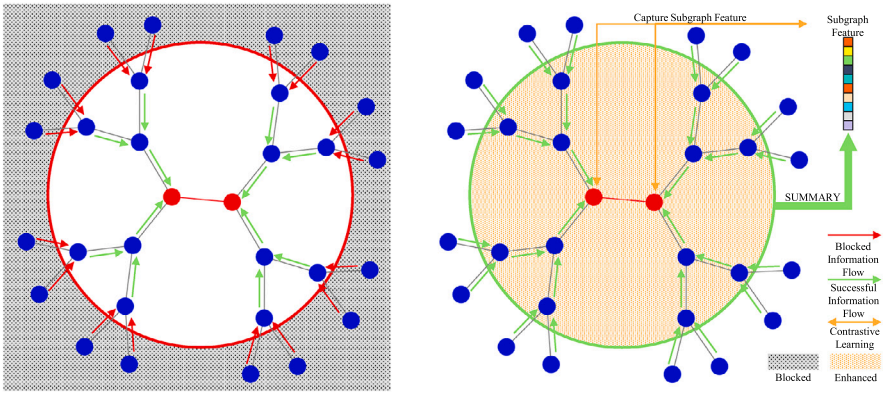

Fig. 1. Diagram of the drawback of information blocking of explicit subgraph extraction methods. On the contrary, contrastive learning could enhance subgraph information while maintaining the information flow from deeper regions.

Among them, SEAL and LGLP are state-of-the-art methods for link prediction on a variety of graphs [13]. They convert the link prediction task into a graph classification problem by extracting h-hop enclosing subgraphs around the target link. The main difference is that SEAL explicitly extracts the subgraph from the original graph, while LGLP transforms the subgraph structure into its corresponding line graph. To leverage subgraph embedding, SEAL introduces a sort pooling operation [14], which still suffers from drawbacks, including slow convergence, additional parameters, and partial node selection, leading to information loss. In contrast, LGLP overcomes these limitations through line graph transformation. These two methods highlight the potential of learning subgraph patterns. However, both methods are confined to local information within the h-hop subgraphs. They use manually designed DRNL [12] node labeling as the initial features to provide a proper starting point. However, their ability to learn features is hindered since information flow beyond the h-hop boundary is blocked. Additionally, both methods require extra memory to store subgraph data, especially for LGLP, where the line graph is usually much larger than the original subgraph, resulting in scalability issues.

In light of the constraints mentioned above, there is a pressing demand for a scalable model to integrate subgraph information into node representations while maintaining global information. Contrastive learning can align anchor samples with positive samples by approximating their semantic representations, which could be implemented in scalable models. With a cross-scale contrastive learning paradigm, we could inject subgraph-level information into end-node representations around the target link. Studies have shown that contrastive learning has improved performance in various tasks [15]. The success of contrastive learning on graph structures [16] demonstrates its potential to improve the accuracy of link prediction models. Contrastive learning enables the implicit capture of underlying subgraph patterns that may be difficult to define explicitly without requiring additional memory. Thus, we propose to incorporate contrastive learning as a regularization term in the end-to-end supervised link prediction model to improve its performance, efficiency, and generalizability (as shown in Fig. 1). Our contributions can be listed as follows,

- We analyze the limitation of explicit subgraph extraction learning patterns and envision the potential to introduce contrastive learning to implicitly capture subgraph information for link prediction problems without additional computation cost.
- We propose Subgraph Contrastive Supervised Neural Network (SCS), a supervised graph neural network with contrastive learning as a regularization term for the link prediction task, where subgraph information can be effectively learned through mutual information maximization.
- Our empirical analysis demonstrates that, in most cases, our proposed method outperforms other baseline methods, including previously state-of-the-art models. Our model achieves these results without additional memory and presents a satisfactory convergence speed. Additionally, our model reveals encouraging evidence that subgraph information can notably impact the link prediction task.
- Our method can achieve promising performance with the most straightforward graph neural network architectures, thus proving the superiority of our cross-scale contrastive learning paradigm.

2. Related work

2.1. Heuristic methods

Heuristic methods are widely used in link prediction due to their simplicity and efficiency. Representative examples include Common Neighbors [17] and Adamic-Adar [18], which estimate link likelihoods based on local structural features such as shared neighbors or degree overlap. These approaches are computationally lightweight, require no labeled data [6], and scale efficiently to large networks. However, their performance often degrades on complex or irregular graphs as they struggle to capture long-range dependencies or nonlinear patterns.

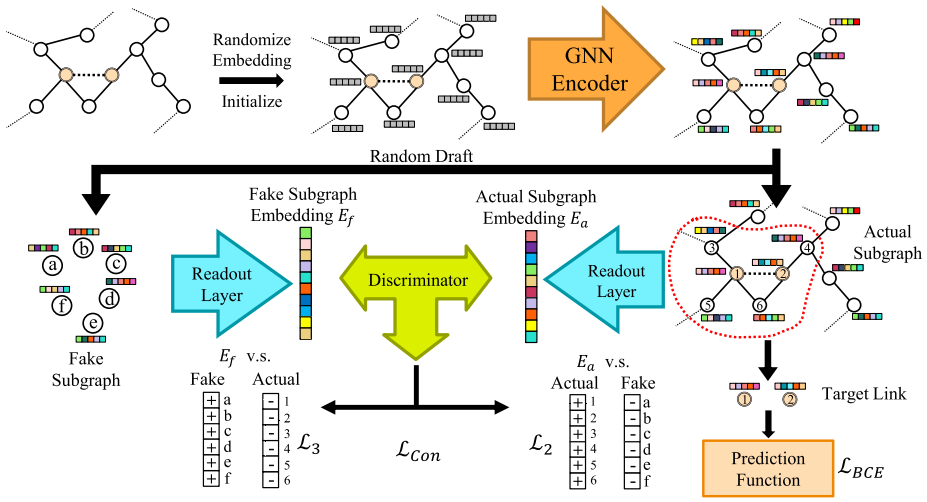

Fig. 2. Diagram of training process for SCS link prediction method.

2.2. Representation learning methods

Representation learning methods aim to learn expressive embeddings of nodes that capture structural dependencies in complex networks [19]. While early techniques such as matrix factorization [20] and probabilistic models [21] explicitly model latent structure, recent approaches often rely on Graph Neural Networks (GNNs) [3], which aggregate neighborhood features through message passing.

In link prediction, path-based methods like DeepWalk [22] and node2vec [23] generate embeddings via random walks, capturing proximity through sampling-based statistics. Despite their scalability, these methods often lack sensitivity to finer-grained local structures and generalize poorly to inductive settings.

Subgraph-based methods like WLNM [11], SEAL [12], and LGLP [13] formulate link prediction as a subgraph classification problem by extracting h-hop enclosing subgraphs around the target links and feeding them into GNN encoders. Recent efforts aim to improve subgraph extraction or encoding efficiency. SEAL+ [24] enhances SEAL by introducing enriched structural descriptors and improved pooling techniques for subgraph representation. HGLP [25] generalizes subgraph-based link prediction to hypergraph settings via hypergraph neural networks and tailored extraction schemes. Despite their strong empirical performance, subgraph-based methods suffer from high preprocessing overhead and limited scalability. These limitations underscore the need for alternative strategies to incorporate local structural context more efficiently.

2.3. Contrastive learning methods

Contrastive learning has become a prominent technique in graph representation learning, aiming to learn informative embeddings by pulling similar (positive and anchor) pairs closer and pushing dissimilar (negative and anchor) pairs apart [15,26]. Classic methods such as DGI [27], GraphCL [28], GRACE [29], and BGRL [30] primarily focus on node-level or graph-level representation tasks, often leveraging random perturbations or augmentation strategies to construct contrastive views. For instance, DGI maximizes mutual information between local patches and global summaries, while GRACE contrasts two randomly corrupted views of the same graph.

More recently, contrastive learning has been adopted in link prediction to improve representation quality. LGCL [31] contrasts node representations derived from line graphs and enclosing subgraphs within a multi-view ensemble to enhance structural alignment. ECLiP [32] learns separate edge embeddings and aligns them with the Hadamard product of node embeddings through contrastive loss. MCAS [33] extends subgraph-based approaches by incorporating multi-scale views and enforcing consistency across them via contrastive objectives.

While recent methods like LGCL and MCAS enhance structural representations by contrasting different views (e.g., line graphs or multi-hop subgraphs), their contrastive modules are typically supplementary, such as aligning distinct encoders. Contrastive learning in these settings is often not central to the model's training objective but rather an enhancement layer appended to existing architectures. This highlights an open opportunity to explore contrastive learning as a core optimization principle that directly shapes the embedding space for link prediction. In particular, we aim to capture relationships between nodes and their enclosing subgraphs without incurring the heavy computational requirements for explicit subgraph extraction.

These gaps motivate our proposed method. We propose a lightweight and scalable framework that uses contrastive learning to inject neighborhood-level information into node embeddings. Unlike prior methods, our approach avoids handcrafted heuristics and expensive subgraph extraction while preserving the advantages of subgraph reasoning and embedding space optimization.

3. Proposed method

3.1. Problem definition

In this paper, we mainly focus on undirected networks denoted as $G = (V, E)$, where $V = \{v_1, v_2, ..., v_i, ..., v_n\}$ stands for the set of nodes and $E \subseteq V \times V$ stands for the edge sets. For the attributed graph, we have attribute matrix $\Phi \in \mathbb{R}^{|V| \times N}$ where $|V|$ represents the cardinal of set V and N is the length of a feature. A network can also be presented in an adjacency matrix fashion. In the case of an undirected network, node v_i and v_j are connected if and only if the element a_{ij} in adjacency matrix $\mathbf{A}$ equals 1 and 0 otherwise.

The goal of link prediction is to learn a binary classification function f to predict whether the missing links are likely to be connected.

3.2. General framework

The diagram in Fig. 2 presents an overview of the training process of our proposed approach, which comprises an end-to-end graph neural network (GNN) encoder for addressing the link prediction task, along with a self-supervised contrastive component for enhancing the GNN encoder's capacity to learn "global" information pertinent to the target link. After training with the contrastive learning component, the learned GNN encoder is directly used to estimate connection probabilities for unseen node pairs in an end-to-end manner.

In a plain graph scenario, each node is endowed with an initial embedding obtained through a randomization process, while for an attributed graph, we naturally use the attribute matrix. This embedding is then passed through a graph neural network (GNN) model to generate learned representations, which are subsequently utilized for following link prediction tasks. Simultaneously, the readout layer harnesses genuine and spurious subgraph-level features, which a discriminator subsequently identifies.

In the following sections, we shall provide an in-depth exposition of our model, encompassing the graph neural network's architecture, the formulation of the contrastive task, and the loss function utilized for updating our model.

3.3. Supervised link prediction

Graph neural networks are effective models for learning graph structure and extracting node features. Typically, they require an embedding matrix $\mathbf{X} \in \mathbb{R}^{|V| \times d}$, where $h_v = \mathbf{X}[v, :]$ is the representation of node v and an adjacency matrix $\mathbf{A}$ to learn meaningful node representations. The overall computation of a GNN involves stacking multiple message-passing layers, where each layer's output serves as the next layer's input. The final node representations can then be used for downstream tasks such as node classification or link prediction.

Current GNN models follow the message-passing paradigm, which contains two major parts: aggregation and activation. Formally, the embedding of node v on l-th layer of a GNN can be described as,

$$n_v^{(l)} = AGGREGATE^{(l)}(h_v^{(l-1)}, \{h_u^{(l-1)} | u \in \mathcal{N}(v)\}), \tag{1}$$

$$h_v^{(l)} = ACTIVATE^{(l)}(h_v^{(l-1)}, n_v^{(l)}). \tag{2}$$

In it, the *AGGREGATE* operation combines neighboring node features (typically via weighted sum or average), while *ACTIVATE* applies a non-linear activation function (e.g., ReLU) to the aggregated result. In the aggregation process, GNN models will learn the patch representation derived from the target node's neighborhood information and its information. The activation step applies a non-linear function to guarantee the expressive power of GNN models.

In our proposed method, we utilize the GCN layer to learn the node embeddings. Formally, the operation performed by a single GCN layer can be expressed as,

$$H^{(l+1)} = \sigma(\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}H^{(l)}W^{(l)}), \tag{3}$$

where $H^{(l)}$ is the node feature matrix at layer l, specifically $H^{(0)} = \mathbf{X}$. $\tilde{A}$ is the adjacency matrix of the graph with self-loops, $\tilde{D}$ is the diagonal degree matrix of $\tilde{A}$, $W^{(l)}$ is the learnable weight matrix for the l-th layer, and σ is an activation function. This operation involves normalizing the adjacency matrix, multiplying it with the node feature matrix, and applying a linear transformation followed by a non-linear activation function.

Inspired by [34,35], after obtaining node representations from all GCN layers ($H^{(l)}, l = 1, 2, ..., L$ and the embedding of node v at the $l - th$ layer $h_v^{(l)} = H^{(l)}[v, :]$), the final embedding of a node summarizes different scales of patch information by concatenating the representations from all layers, which means,

$$h_v = CONCAT(\{h_v^{(l)}, l = 1, ..., L\}). \tag{4}$$

CONCAT is the operation that concatenates two or more tensors along a specified dimension, typically combining feature vectors from different layers or sources.

After acquiring node representations using a graph neural network, target link representations can be obtained by combining the representations of its end nodes. A straightforward approach is to concatenate the node representations of the linked nodes into a single feature vector. Another approach is fusing the node representations through a more sophisticated function. A widely used

function for this purpose is the Hadamard product, defined as the element-wise multiplication of two vectors. In our model, the final representation $h_{(u,v)}$ of link (u, v) can be formulated as,

$$h_{(u,v)} = CONCAT(h_u + h_v, h_u \odot h_v, h_u, h_v), \tag{5}$$

where $\odot$ denotes the Hadamard product. After getting the representation of links, we employ the Multilayer Perceptron (MLP) with only one hidden layer as the binary prediction function. That means the connection likelihood $p_{(u,v)}$ of non-existence link (u, v) are calculated as follows,

$$p_{(u,v)} = MLP(h_{(u,v)}). \tag{6}$$

Then, the loss of the supervised link prediction task can be formulated as a binary cross-entropy loss,

$$\begin{aligned}\mathcal{L}_1 = \mathcal{L}_{BCE} = -\frac{1}{|\mathcal{T}|}\sum_{(u,v)\in\mathcal{T}}\Big(& y_{(u,v)}\log(p_{(u,v)}) + \\ & (1-y_{(u,v)})\log(1-p_{(u,v)})\Big),\end{aligned} \tag{7}$$

where $\mathcal{T}$ stands for the training set, $|\mathcal{T}|$ stands for the number of elements in training set $\mathcal{T}$, and each instance (u, v) in it has a discrete actual label $y_{(u,v)} \in \{0, 1\}$ and a continuous predicted probability $p_{(u,v)} \in [0, 1]$, which is calculated by equation (6).

3.4. Contrastive component

Prior work has shown that transforming link prediction into subgraph classification yields strong empirical results [12,13,31]. However, these methods suffer from high computational cost and poor scalability, especially in dense graphs where the number of candidate links is $O(|V|^2)$. To address this, we propose an alternative that is more efficient than explicitly extracting subgraphs. Specifically, we utilize contrastive learning to encode local structural context into node embeddings implicitly.

In our framework, each node serves as an anchor, and the embedding of its surrounding subgraph acts as the positive sample. We encourage node embeddings to encode rich neighborhood information by aligning these representations. This idea aligns with the contrastive learning strategy in DGI [27], which maximizes mutual information between representations across different structural scales.

A key challenge in this setup is preventing over-smoothing caused by highly consistent anchor-positive pairs. To address this, we adopt a variance-guided negative sampling strategy, where embedding variance serves as a proxy for informativeness. Specifically, we generate fake subgraphs via random sampling to approximate the latent-space mean, encouraging node embeddings to remain diverse and capture meaningful structural differences.

We define a random draft function $\mathcal{RD}$ to generate a fake subgraph $\tilde{S}$ with the same number of nodes as a given real subgraph $\mathcal{S} = (\dot{\mathbf{X}}, \dot{\mathbf{A}}) \in (\mathbb{R}^{m\times d}, 0, 1^{m\times m})$. Specifically,

$$\begin{aligned}\mathcal{RD}(m) : (\mathbf{X}, \mathbf{A}) \in & (\mathbb{R}^{|V|\times d}, \{0,1\}^{|V|\times|V|}) \longrightarrow \\ & (\tilde{\mathbf{X}}, \tilde{\mathbf{A}}) \in (\mathbb{R}^{m\times d}, \{0,1\}^{m\times m}),\end{aligned} \tag{8}$$

where $(\mathbf{X}, \mathbf{A})$ is the original complete graph and $(\tilde{\mathbf{X}}, \tilde{\mathbf{A}})$ denotes the randomly sampled counterpart. These fake subgraphs serve as approximate observations of the latent space's global mean, acting as effective negatives that promote representational diversity in contrastive learning.

To obtain subgraph-level representations, we employ a readout function $\mathcal{R}$ that aggregates node embeddings from the local subgraph:

$$\vec{s} = \mathcal{R}(\mathcal{S}), \tag{9}$$

where $\vec{s}$ denotes the embedding of subgraph $\mathcal{S}$. Following this, we use the following objective as the contrastive loss,

$$\begin{aligned}\mathcal{L}_2 = -\frac{1}{|\mathcal{S}| + |\tilde{\mathcal{S}}|}\Bigg(& \sum_{v\in\mathcal{S}} \mathbb{E}_{(\dot{\mathbf{X}},\dot{\mathbf{A}})}[\log \mathcal{D}(h_v, \vec{s})] + \\ & \sum_{u\in\tilde{\mathcal{S}}} \mathbb{E}_{(\tilde{\mathbf{X}},\tilde{\mathbf{A}})}\Big[\log\Big(1 - \mathcal{D}(h_u, \vec{s})\Big)\Big]\Bigg).\end{aligned} \tag{10}$$

Empirical results in [28] show that a more challenging task might lead to a more powerful and robust model. Therefore, we not only want the model to recognize nodes from the actual subgraph (lower-right part of 2) but also to distinguish whether a node comes from the randomly generated fake one (lower-left part of 2). To achieve this, we introduce a dual objective as follows,

$$\mathcal{L}_3 = -\frac{1}{|\mathcal{S}| + |\tilde{\mathcal{S}}|}\left(\sum_{u\in\mathcal{S}} \mathbb{E}_{(\tilde{\mathbf{X}},\tilde{\mathbf{A}})}[\log \mathcal{D}(h_u, \tilde{\vec{s}})] + \sum_{v\in\mathcal{S}} \mathbb{E}_{(\tilde{\mathbf{X}},\tilde{\mathbf{A}})}\left[\log\left(1 - \mathcal{D}(h_v, \tilde{\vec{s}})\right)\right]\right). \tag{11}$$

In equations (10) and (11), $\mathbb{E}$ stands for the expectation, $\vec{s}$ and $\tilde{\vec{s}}$ are the summary representation of subgraph $\mathcal{S}$ and $\tilde{\mathcal{S}}$ respectively. $\mathcal{D}$ stands for a discriminator to quantify the likelihood of whether a node belongs to a given subgraph.

Finally, the contrastive loss is calculated as,

$$\mathcal{L}_{Con} = \mathcal{L}_2 + \mathcal{L}_3. \tag{12}$$

The contrastive component encourages node embeddings to align with their neighborhood structure while maintaining separation from randomly sampled negatives, which is crucial for link prediction [36]. This component in SCS leverages the fundamental hypothesis that nodes with more common neighbors are more likely to form links. According to Equation (10), if two unconnected nodes share many common neighbors, they are likely to co-occur in multiple subgraphs, promoting consistent embeddings and facilitating the discovery of potential links—an intuition consistent with classical heuristics such as common neighbor indices. In contrast, distant node pairs seldom appear together and are more often contrasted as negatives, reinforcing their separation.

Additionally, many real-world networks exhibit a scale-free property [37]. In our proposed SCS model, high-degree hub nodes appear in more subgraphs, thus contributing more to the contrastive loss and helping the model refine representations in structurally important regions. Such properties will be later verified in Section 4.3.4.

3.5. The proposed method

Our proposed method follows an end-to-end learning paradigm. As illustrated in Fig. 2, the model consists of two primary components: the GNN encoder and the contrastive learning module. An overview of these components is provided in Sections 3.3 and 3.4, with detailed implementation discussed subsequently. Initially, the method generates node embeddings based on Equations (3) and (5). These embeddings are further refined by the contrastive learning module, as defined in Equation (12), to produce diverse embeddings, thereby facilitating the downstream link prediction task.

The use of contrastive learning in this context aims to generate more diverse and informative node embeddings by increasing the variance of the representations. When generating fake subgraphs through random sampling, the embedding produced by the readout function is expected to correspond to the average of all node embeddings. According to Equation (10) and (11), the contrastive component tends to increase the embedding variance by pushing the current node and subgraph embeddings away from this average, thereby promoting greater diversity. This results in embeddings that capture more distinct information, ultimately improving the quality of the learned representations and facilitating the downstream link prediction task.

In detail, we choose the most straightforward architecture for the summary function and discriminator in model selection to avoid introducing unnecessary parameters. For the summary function in Equation (9), we adopt the average readout function since evidence in [27] shows that for small-scale networks, the average readout function has the potential to yield the best performance.

$$\mathcal{R}(\mathcal{S}) = \sigma\left(\frac{1}{|\mathcal{S}|}\sum_{v\in\mathcal{S}} h_v\right). \tag{13}$$

For the discriminator used in Equation (10) and (11), we select the same bilinear scoring function to calculate the matching likelihood,

$$\mathcal{D}(h_v, \vec{s}) = \sigma\left(h_v^T \mathbf{W} \vec{s}\right), \tag{14}$$

here $\mathbf{W}$ is a learnable matrix, and σ is the logistic sigmoid to convert the output value into a positive probability.

After all the architecture is fixed, the update of the model is operated by the following loss,

$$\mathcal{L} = \mathcal{L}_{BCE} + \alpha \cdot \mathcal{L}_{Con}, \tag{15}$$

where α is the hyperparameter to balance the contrastive regularization term. For calculation efficiency, we select a subset of the target links in each batch to implement the subgraph contrastive learning with a hyperparameter m. The pseudo-code of the training process for our model is illustrated in Algorithm 1.

4. Experiment

4.1. Datasets and baseline methods

In this paper, we perform our proposed methods Subgraph Contrastive Supervised Neural Network on 19 different datasets, including 14 plain graphs, namely BUP, CEG, UAL, SMG, EML, NSC, YST, UPG, KHN, ADV, GRQ, LDG, HPD, and ZWL[1] [38,39],

[1] https://noesis.ikor.org/datasets/link-prediction.

Algorithm 1 Pseudocode of training process for SCS link prediction method.

Input: Network $\mathcal{G}$, target link batch $\{(u_i, v_i), i = 1, ..., n\}$, original model $\Theta(\omega)$, number of sample in contrastive learning $m(\leq n)$
Output: Trained model $\Theta(\omega)$
contrastive learning subset $\{(u_j, v_j), j = 1, ..., m\} \leftarrow$ Random Draft($\{(u_i, v_i), i = 1, ..., n\}$, m)
for link (u, v) in $\{(u_i, v_i), i = 1, ..., n\}$ **do**
  for layer $l = 1$ to L **do**
    Update feature matrix $H^{(l)}$ via Eq. (3)
  end for
  Node feature matrix H via Eq. (4)
  Generate link feature $h_{(u,v)}$ via Eq. (5)
  Calculate $\mathcal{L}_{BCE}$ via Eq. (6) and Eq. (7)
  if $(u, v) \in \{(u_j, v_j), j = 1, ..., m\}$ **then**
    $S \leftarrow$ Extract h-hop subgraph around (u, v)
    $\tilde{S} \leftarrow$ Generate fake subgraph via Eq. (8)
    Get subgraph feature $\vec{s}$ and $\tilde{\vec{s}}$ via Eq. (13)
    Calculate $\mathcal{L}_{Con}$ via Eq. (14), Eq. (10) and Eq. (11)
  end if
  Calculate gradient $g \leftarrow \nabla_{\omega}(\mathcal{L}_{BCE} + \alpha \cdot \mathcal{L}_{Con})$
  Update model $\Theta(\omega)$ according to gradient g
end for
return $\Theta(\omega)$

Table 1
Summary information about plain graphs involved in our experiments. We select the exact 14 plain graphs that LGLP used in its experiment section.

Name	$\|V\|$	$\|E\|$	Avg. Degree	Network Type
BUP	105	441	8.4	Political Blog
CEG	297	2148	14.16	Biological
UAL	332	2126	12.81	Air Traffic
SMG	1024	4916	9.6	Co-authorship
EML	1133	5451	9.62	Shared Emails
NSC	1461	2742	3.75	Co-authorship
YST	2284	6646	5.82	Biological
UPG	4941	6594	2.669	Power Distribution
KHN	3772	12718	6.74	Co-authorship
ADV	5155	39285	15.24	Social
GRQ	5241	14484	5.53	Co-authorship
LDG	8324	41532	9.98	Co-authorship
HPD	8756	32331	7.38	Biological
ZWL	6651	54182	16.29	Co-authorship

Table 2
Summary information about attributed graphs used in the experiment.

Dataset	$\|V\|$	$\|E\|$	Avg. Degree	# Features
Cora	2,708	5,429	4.00	1,433
Citeseer	3,312	4,732	2.86	3,703
Pubmed	19,717	44,338	4.50	500
ogbl-collab	235,868	1,285,465	8.2	128
ogbl-ppa	576,289	30,326,273	73.7	58

and 5 attributed graphs, namely Cora, Citeseer, Pubmed[2] [40], and two large-scale real-world networks for scalability analysis: ogbl-collab [41] and ogbl-ppa [42].[3] These datasets are from a variety of domains with different scales. The details of these datasets are listed in Table 1 and 2. The code and datasets are available at https://github.com/Young0313/SCS.

We have selected diverse, representative methods to compare across various categories. For heuristic methods, we have chosen CN [17] and AA [18], local methods that operate on node neighborhoods, as well as SimRank [43], a global similarity-based method. For the unsupervised graph reconstruction learning methods, we have included DeepWalk [22], a random walk-based approach, and VGAE [44], a state-of-the-art graph deep learning method. For the category of supervised methods, we have selected SEAL [12] and LGLP [13], both of which are current state-of-the-art methods. As for the contrastive link prediction method, we include LGCL [31] to compare its performance against our approach.

For evaluation metrics, two commonly used metrics, Area Under Curve (AUC) and Average Precision (AP), are employed to evaluate the performance of binary classification models. The AUC ranges from 0.0 to 1.0, with a higher value indicating better

[2] https://github.com/kimiyoung/planetoid.
[3] https://ogb.stanford.edu/docs/linkprop.

Table 3
AUC performance comparison with other baseline methods (70% training links).

Method	BUP	CEG	UAL	SMG	EML	NSC	YST
CN	83.30(±2.12)	79.49(±0.88)	91.77(±0.36)	78.37(±0.24)	78.96(±0.30)	95.98(±0.72)	66.84(±0.35)
AA	84.16(±1.12)	81.84(±0.47)	92.25(±0.70)	78.83(±0.28)	79.14(±0.32)	96.27(±0.53)	66.63(±0.45)
SR	84.64(±1.54)	76.45(±1.10)	80.29(±1.46)	76.57(±1.06)	**85.90(±0.71)**	96.62(±0.29)	74.01(±0.53)
DW	81.27(±2.79)	74.78(±0.75)	68.86(±1.15)	69.13(±0.69)	74.61(±0.62)	96.07(±0.47)	62.54(±0.64)
VGAE	86.15(±0.78)	79.45(±0.69)	91.11(±0.55)	78.05(±0.69)	83.00(±0.75)	97.83(±0.60)	72.25(±0.32)
SEAL	81.28(±2.35)	72.96(±1.60)	88.02(±0.78)	83.61(±0.83)	73.18(±0.95)	96.36(±0.34)	79.10(±0.82)
LGLP	79.67(±1.37)	78.17(±1.00)	90.93(±0.39)	84.10(±0.52)	76.48(±0.40)	97.18(±0.33)	**83.03(±0.19)**
LGCL	80.89(±1.45)	77.89(±0.92)	91.73(±0.45)	85.10(±0.62)	78.23(±0.52)	97.93(±0.32)	82.46(±0.53)
SCS	**89.62(±0.23)**	**85.08(±0.20)**	**94.82(±0.26)**	**86.29(±0.31)**	84.36(±0.21)	**98.21(±0.05)**	80.51(±0.29)
Method	UPG	KHN	ADV	GRQ	LDG	HPD	ZWL
CN	55.77(±0.22)	74.44(±0.16)	86.32(±0.14)	86.37(±0.18)	83.12(±0.09)	68.81(±0.16)	89.57(±0.17)
AA	55.85(±0.22)	74.91(±0.44)	86.53(±0.16)	86.49(±0.46)	83.38(±0.08)	68.95(±0.19)	89.69(±0.09)
SR	58.49(±2.77)	77.81(±0.63)	85.37(±0.17)	**87.45(±0.29)**	89.58(±0.27)	80.77(±0.14)	**95.10(±0.08)**
DW	57.63(±0.75)	70.30(±0.45)	69.24(±0.34)	81.26(±0.27)	82.04(±0.09)	68.43(±0.16)	87.75(±0.06)
VGAE	55.71(±1.23)	72.25(±0.98)	82.67(±0.36)	79.14(±0.56)	76.52(±2.95)	70.65(±0.76)	88.21(±0.19)
SEAL	58.64(±1.07)	83.20(±0.33)	88.09(±0.18)	80.69(±0.19)	89.66(±0.16)	84.16(±0.41)	87.75(±0.37)
LGLP	56.45(±0.71)	83.55(±0.07)	89.24(±0.39)	79.53(±0.53)	86.40(±0.40)	84.37(±0.20)	86.69(±0.57)
LGCL	57.73(±0.85)	83.79(±0.23)	88.32(±0.28)	80.51(±0.34)	87.80(±0.26)	**84.46(±0.35)**	85.97(±0.26)
SCS	**58.73(±0.57)**	**84.33(±0.14)**	**90.07(±0.06)**	85.63(±0.17)	**89.69(±0.21)**	83.24(±0.19)	92.84(±0.12)

performance. A model with an AUC of 0.5 performs no better than random guessing, while a model with an AUC of 1.0 indicates perfect performance. Like AUC, AP ranges from 0.0 to 1.0, with a higher value indicating better performance. AP is handy when the positive class is rare or imbalanced.

4.2. Experimental setup

To evaluate our method's effectiveness, we conducted an empirical study by randomly selecting 70% of the existing edges in the graph as positive training samples. The remaining edges were masked during the training process and reserved for testing. Moreover, an equal number of non-existent links were randomly selected as negative samples. Furthermore, experiments with varying test ratios were conducted to demonstrate the effectiveness and robustness of our model.

To establish a baseline, we set the hyperparameters of the comparison methods to their reported values in their original papers. We fixed the importance factor of the Simrank index to 0.9 and set the output node embedding dimension of DeepWalk to 128. To ensure a fair comparison, we initialized the node embeddings for VGAE, SEAL, LGLP, LGCL, and our proposed SCS model using random values drawn from a normal distribution. We did not employ the DRNL labeling technique for SEAL, LGLP, and LGCL to compare the pure representation learning ability.

We trained VGAE for 200 epochs on each dataset with a learning rate of 0.01, while SEAL and LGCL were trained for 50 epochs. For LGLP and our proposed methods were trained for 15 epochs. To compare the convergence speed of SEAL, LGLP, and SCS, we configured all three methods with a batch size of 32. We randomly selected two samples in each batch for our proposed SCS method to perform the contrastive learning component. To directly evaluate the embedding learning ability, we employed the identical GNN encoder architecture in LGLP for our SCS model. For all experiments except hyperparameter analysis, our proposed method's balance ratio α is set to 0.1, and the subgraph size h for all three subgraph-based methods was set to 1.

4.3. Results and analysis

4.3.1. Link prediction on plain graphs

We conduct experiments on 14 plain graph datasets to compare the performance of our proposed method against baseline methods. We randomly split the original graph into training and testing sets with a 70%-30% ratio and repeated this process five times. We provide all graph neural network-based methods with randomly initialized node embedding information under the normalized distribution. We compare our proposed method against several representative baselines, including heuristic (CN [17], AA [18], SimRank [43]), unsupervised (DeepWalk [22], VGAE [44]), and supervised (SEAL [12], LGLP [13], LGCL [31]) approaches. Detailed descriptions and settings of these baselines are provided in Sections 4.1 and 4.2. We evaluate the results of the experiment using the area under the ROC curve (AUC) and average precision (AP), reporting the mean and standard deviation in Tables 3 and 4. We mark the best and second-best values in **bold** and underline, respectively. The results show that heuristic methods can achieve competitive performance in specific scenarios, such as the SimRank [43] method in the EML dataset and the Adamic-Adar index in the GRQ dataset. However, these methods do not consistently produce good results across different scenarios. Network reconstruction-based methods like DeepWalk [22] and VGAE [44] tend to achieve more satisfying results on smaller-scale networks.

Additionally, the state-of-the-art method VGAE [44] consistently outperforms DeepWalk, indicating its better expressive power in general. For supervised link prediction methods, we find that LGLP [13] can outperform SEAL [12] in most cases with lower

Table 4
AP performance comparison with other baseline methods (70% training links).

Method	BUP	CEG	UAL	SMG	EML	NSC	YST
CN	80.26(±2.13)	76.61(±1.01)	91.44(±0.24)	77.38(±0.25)	78.28(±0.25)	95.92(±0.73)	66.67(±0.35)
AA	83.27(±1.27)	81.22(±0.34)	92.93(±0.64)	79.76(±0.23)	79.34(±0.38)	96.28(±0.53)	66.71(±0.46)
SR	79.52(±2.61)	68.15(±1.48)	72.24(±2.18)	70.91(±1.70)	**86.32(±1.14)**	95.56(±0.13)	77.41(±0.64)
DW	76.97(±3.95)	72.73(±0.96)	64.64(±1.05)	69.46(±0.54)	76.08(±1.10)	96.16(±0.85)	64.19(±0.86)
VGAE	84.41(±0.48)	80.75(±0.63)	93.39(±0.42)	82.15(±0.55)	85.76(±0.70)	98.06(±0.46)	77.28(±0.11)
SEAL	78.32(±2.90)	72.66(±2.14)	86.89(±0.79)	83.18(±0.90)	70.64(±0.94)	95.18(±0.77)	76.74(±1.06)
LGLP	79.25(±3.22)	75.96(±1.69)	91.65(±0.54)	84.08(±1.05)	75.01(±0.49)	96.85(±0.41)	81.47(±0.49)
LGCL	78.97(±2.85)	76.21(±1.80)	90.98(±0.43)	85.23(±1.02)	73.01(±0.55)	97.23(±0.45)	79.85(±0.57)
SCS	**89.03(±0.27)**	**82.22(±0.17)**	**95.84(±0.15)**	**87.44(±0.54)**	85.19(±0.24)	**98.37(±0.08)**	**81.57(±0.30)**
Method	UPG	KHN	ADV	GRQ	LDG	HPD	ZWL
CN	55.76(±0.22)	73.84(±0.16)	85.96(±0.16)	86.35(±0.18)	82.79(±0.10)	68.65(±0.17)	89.29(±0.17)
AA	55.82(±0.24)	75.69(±0.44)	86.94(±0.13)	86.51(±0.45)	83.67(±0.08)	69.07(±0.19)	89.74(±0.10)
SR	68.51(±0.39)	76.73(±0.50)	82.84(±0.34)	**91.20(±0.19)**	88.24(±0.24)	83.46(±0.32)	**94.72(±0.17)**
DW	**69.82(±0.70)**	71.98(±0.57)	67.23(±0.40)	86.76(±0.20)	82.27(±0.17)	68.60(±0.37)	86.93(±0.08)
VGAE	58.11(±0.86)	74.95(±1.35)	84.30(±0.47)	82.72(±0.64)	77.05(±3.48)	71.59(±0.85)	89.19(±0.19)
SEAL	56.68(±0.85)	84.53(±0.54)	87.22(±0.20)	80.93(±0.39)	89.29(±0.14)	84.11(±0.37)	87.06(±0.48)
LGLP	56.16(±1.13)	84.97(±0.31)	89.01(±0.46)	81.90(±0.55)	86.60(±0.41)	84.79(±0.24)	85.50(±0.85)
LGCL	56.43(±1.05)	85.23(±0.42)	88.76(±0.35)	82.43(±0.46)	88.24(±0.29)	84.97(±0.35)	86.53(±0.76)
SCS	60.45(±0.47)	**87.01(±0.15)**	**90.62(±0.10)**	89.10(±0.10)	**91.16(±0.21)**	**85.10(±0.14)**	93.72(±0.13)

standard deviation values. LGCL's [31] contrastive learning framework matches the embeddings of subgraphs from SEAL and line graphs from LGLP, treating one as an anchor and the other as a positive. This approach tends to align two embedding spaces. While this may enhance the model's robustness, its embedding capability is still constrained by the limitations of the original models. It is reflected in the experimental results, where LGCL's performance is found to be quite similar to that of SEAL and LGLP. Based on these results, we may conclude that learning the embedding of links directly is better than learning graph embedding. Our proposed method, SCS, achieves better performance compared to other state-of-the-art methods, with leading performance in 10 out of 14 plain graph scenarios and remaining in the top two performance in almost all cases for AUC. Similarly, the performance in AP also leads to 11 out of 14 datasets, with top-two performance in most cases. The reported standard deviation value is the lowest in most cases, indicating that our proposed method is more stable than other baseline methods.

4.3.2. Model robustness analysis

To further investigate the learning ability of our proposed method, we vary the proportion of links used as training data, ranging from 30% to 90%. We present the AUC results in Fig. 3, compared with six competitive methods, excluding CN and DeepWalk [22] for clarity. The results show that our method consistently outperforms other baseline methods across various training ratios in most cases. This demonstrates that SCS is highly effective in leveraging the existing information within the network. Moreover, compared to other deep learning-based methods such as SEAL [12], LGLP [13], and LGCL [31], our model exhibits superior robustness and stronger performance, highlighting its ability to generalize across different network structures and training conditions. Furthermore, SCS is less sensitive to changes in training ratio, further validating its robustness.

We highlight the UPG dataset, which contains only 6,594 edges among 4,941 nodes, yielding an average degree below 2.7. Under low training ratios, all models yield AUC scores barely above 0.5, indicating performance close to random and reflecting the limited structural signals in such sparse networks. This suggests that meaningful structural patterns may be absent in extremely sparse networks. Despite the overall difficulty, our proposed SCS model outperforms subgraph-based methods like LGLP and LGCL across all ratios. While SimRank slightly outperforms it, it is worth noting that even the best-performing method reaches only 0.66 AUC when 90% of edges are used for training. Among all methods, SCS demonstrates consistently stable improvements across training ratios, in contrast to the noticeable fluctuations observed in models like SimRank [43] and LGCL. This highlights the robustness of our approach in scenarios where sparsity and noise severely challenge the reliability of structural cues.

4.3.3. Link prediction on attributed graphs

We also conduct experiments on three commonly used attributed graphs with detailed information about these graphs listed in Table 2. We compare our proposed method with other supervised methods, specifically SEAL [12], LGLP [13], and LGCL [31], to investigate the efficacy of different learning paradigms. We treat all attributed networks as undirected and randomly select 50% of all links as the training set for five repetitions. All methods use the attributed node embedding matrix Φ as the initial input $H^{(0)}$. We report the mean value and standard deviation of both AUC and AP measures in Table 5, with the best result in **bold** format.

Our findings demonstrate that our proposed method consistently outperforms other state-of-the-art approaches, with particularly significant performance gains on the Cora and Citeseer datasets. This indicates that our model utilizes node attribute information more effectively, leading to more expressive and discriminative embeddings. This improvement stems from the model's ability to integrate structural and attribute information more synergistically. By leveraging contrastive learning, our method enhances the distinction between node representations, resulting in higher variance and more informative embeddings. This allows the model to capture

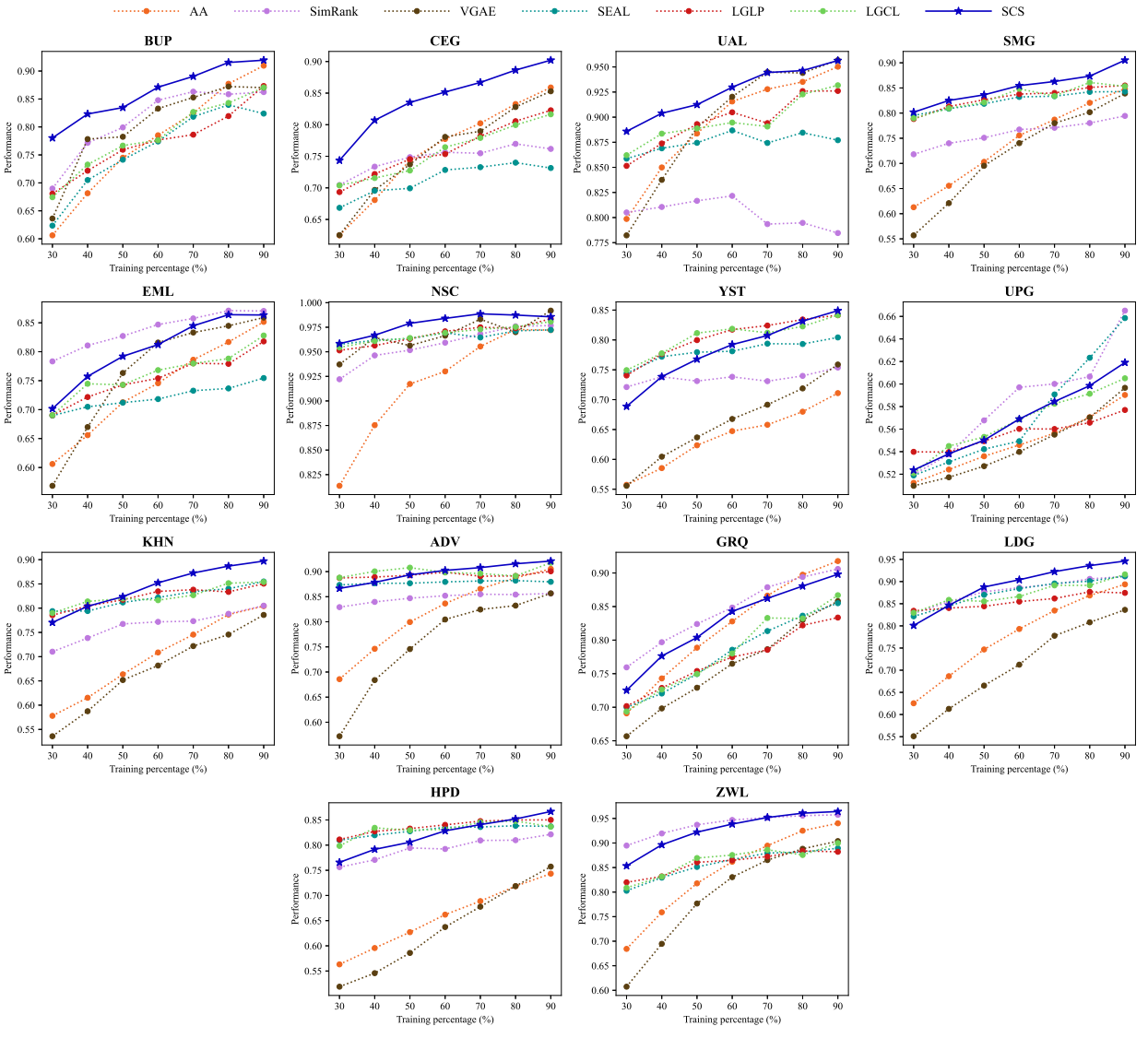

Fig. 3. AUC comparison on all datasets for AA, SimRank, VGAE, SEAL, LGLP, LGCL, and proposed SCS method with different settings of training ratio. For each dataset, we consider 30%, 40%, 50%, 60%, 70%, 80%, and 90% of all links in the network as the training set. SCS is reported in solid lines, while all other baseline methods are presented in dash lines form with alternative colors.

Table 5
Link prediction performance on attributed graph for SEAL, LGLP, LGCL, and proposed SCS methods.

Model	Cora	AUC Citeseer	Pubmed	Cora	AP Citeseer	Pubmed
SEAL	67.23(±0.72)	64.88(±1.39)	90.20(±1.10)	65.11(±1.62)	60.46(±1.49)	88.15(±1.55)
LGLP	73.17(±1.24)	70.01(±0.50)	91.86(±0.27)	71.34(±1.46)	70.47(±1.03)	90.44(±0.29)
LGCL	76.25(±1.45)	72.04(±0.85)	**92.01(±0.86)**	73.45(±1.72)	69.53(±1.25)	90.95(±1.07)
SCS	**86.95(±0.69)**	**86.35(±0.66)**	91.88(±0.25)	**87.84(±0.70)**	**87.49(±0.60)**	**91.31(±0.14)**

subtle differences in node characteristics and structural roles, ultimately improving its capability to generalize across different graph structures and enhance downstream link prediction performance.

4.3.4. Visualization analysis

We present the *t*-SNE [45] visualization result for the Citeseer dataset in Fig. 4. The initial embedding of links in this dataset exhibits chaotic behavior (Fig. 4a). Moreover, the outputs of SEAL and LGLP methods (Fig. 4b and 4c) show non-negligible overlap between positive and negative links, particularly the results produced by the SEAL method. Conversely, our proposed SCS method

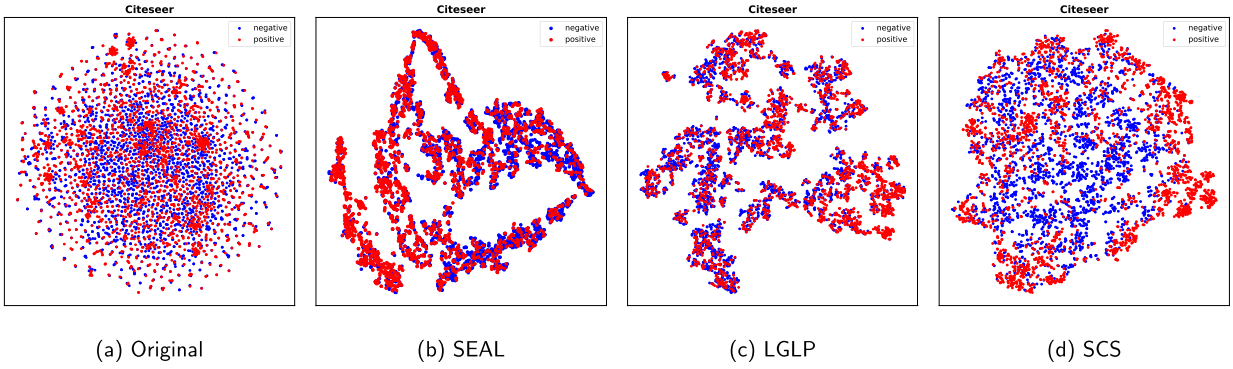

Fig. 4. Visualization of Citeseer attributed graph using t-SNE algorithm, showcasing positive links in red and negative links in blue.

Table 6
AUC performance of ablation analysis on LDG, ADV, and ZWL datasets.

Dataset	SCS (w/o contrastive)	SCS-h2	SCS-h3	SCS-L1	SCS-L3	SCS
LDG	0.8950	0.9087	0.8872	0.8879	0.8942	**0.9143**
ADV	0.8991	0.8925	0.8752	0.8845	0.8970	**0.9108**
ZWL	0.9277	0.9246	0.9140	0.9097	0.9179	**0.9391**

(Fig. 4d) effectively separates positive and negative links. The dispersion of positive samples can be attributed to the variance learning introduced by the contrastive component. As the model is trained to distinguish subgraph and node embeddings from the overall mean, it naturally drives embeddings toward higher variance representations. This results in a more dispersed distribution in the embedding space, and the pattern is visually evident in Fig. 4d.

4.3.5. Ablation study

To evaluate the effectiveness of key components in our proposed SCS model, we conduct ablation studies on three datasets, namely LDG, ADV, and ZWL, with 70% of all links as the training set. Specifically, we analyze the impact of three major factors: the contrastive learning component, the size of the extracted subgraph (determined by the hop size around the target link), and the number of graph convolutional layers in the GNN encoder. Table 6 reports the AUC scores of different model variants.

The variants include SCS, which represents the whole model; SCS (w/o contrastive), where the contrastive learning component is removed; SCS-hx, which extracts subgraphs with a hop size of x; and SCS-Lx, which uses a GNN encoder with x layers. The results provide insights into how each component contributes to the model's overall performance.

The experimental results show that all model variants perform worse than the original SCS. The significant performance drop in the model variant without contrastive learning highlights the critical role of the contrastive component in enhancing representation quality. Furthermore, the decline in performance across different variants can be attributed to the range of neighborhood information captured in node embeddings. Increasing these parameters (SCS-h2/h3/L3) expands the scope of neighborhood information incorporated into node embeddings, leading to an excessive influx of information. This results in overly complex and redundant representations, making it difficult for the model to learn distinct, high-variance embeddings, ultimately degrading downstream task performance. Conversely, reducing the number of GCN layers (SCS-L1) may limit the model's capacity to extract complex structural information, thereby weakening its expressive power and leading to suboptimal performance.

4.3.6. Scalability analysis

We conducted scalability analysis on five datasets: ADV, GRQ, ZWL [38,39], ogbl-collab [41] and ogbl-ppa [42]. For ADV, GRQ, and ZWL, we used 70% of the edges as the training set, and the remaining 30% of the edges were designated as the testing set. Regarding the two large-scale datasets, ogbl-collab and ogbl-ppa, we adhered to the default split provided by the OGB website, allocating approximately 75% and 70% of edges, respectively, for graph construction. Furthermore, to examine the scalability ability for all three subgraph learning-based models, we only fed 10% and 0.5% of all edges as the actual training edges into the training process and tested all models on 4% and 0.2% of edges, respectively. In this experiment, the SCS method's balance ratio α was set to 10.0, and the draft number m was set to 4. All experiments in this subsection were conducted on a computing platform equipped with two 32 vCPU Intel(R) Xeon(R) Platinum 8352V CPUs (2.10 GHz), along with two RTX 4090 GPUs (24 GB of VRAM each), and the platform was configured with a total of 240 GB of memory space.

Table 7 reports the memory usage, training time (for 15 epochs), and performance in terms of AUC and AP for all three methods. In addition, the value of AUC and AP on ADV, GRQ, and ZWL are omitted since they have been provided in Table 3, 4. The best value of each column is reported in **bold**, and OOM stands for out of memory. The results demonstrate that our proposed methods are both computationally and memory-efficient and perform well on large-scale real-world datasets.

Table 7
Scalability experiment result of three subgraph learning-based methods.

Dataset	ADV	GRQ	ZWL	ogbl-collab	ogbl-ppa	ADV	GRQ	ZWL	ogbl-collab	ogbl-ppa
	Training Time(s)(↓)					**Inference Time(s)(↓)**				
SEAL	602.46	213.30	821.23	5495	OOM	4.75	1.50	6.69	**15.518**	OOM
LGLP	**273.23**	**88.45**	**383.09**	**2952**	OOM	6.62	1.98	9.15	26.143	OOM
SCS	371.57	130.33	501.17	5049	**57831**	**4.04**	**1.13**	**6.32**	50.788	**760.12**
	Preprocess Time(s)(↓)					**Memory Usage(GB)(↓)**				
SEAL	174.05	63.36	205.14	629.78	OOM	9.02	4.33	11.71	55.53	OOM
LGLP	331.43	100.80	401.46	1567.3	OOM	23.85	6.13	24.36	130.45	OOM
SCS	**42.32**	**18.46**	**61.72**	**326.02**	**12248**	**3.34**	**2.26**	**2.75**	**8.39**	**42.37**
	AUC(↑)					**AP(↑)**				
SEAL	-	-	-	91.02	OOM	-	-	-	90.34	OOM
LGLP	-	-	-	90.17	OOM	-	-	-	90.31	OOM
SCS	-	-	-	**96.59**	**98.87**	-	-	-	**96.85**	**98.86**

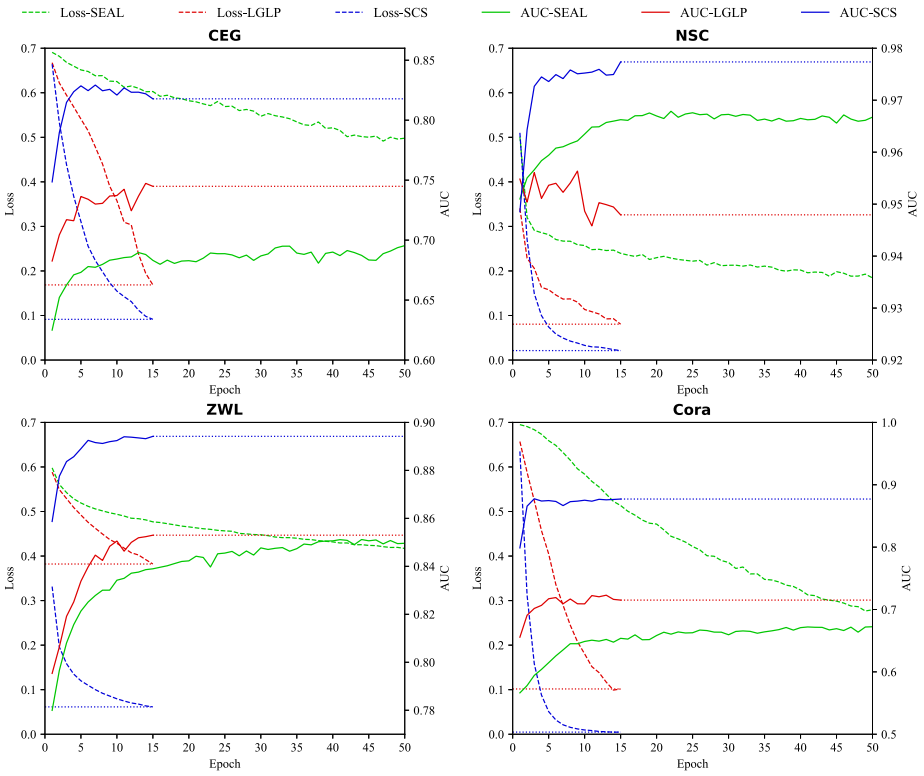

Fig. 5. Convergence speed analysis on CEG, NSC, ZWL, and Cora dataset.

4.3.7. Convergence speed analysis

As an end-to-end supervised link prediction model, our method acquires the edge embedding after getting the node representations through equation (5), where only graph convolution layers are implemented to extract this part of the feature. In the contrastive learning component, the pooling and discriminator are all in the uncomplicated matrix multiplication and activation formalism with few parameters. Thus, the convergence speed of our model is expected to match that of LGLP [13], which significantly outpaces the SEAL [12] model. To verify the convergence speed analysis of our proposed methods, we ran SEAL, LGLP, and SCS methods on four different plain graph datasets. We demonstrated the loss in the training process and the AUC value on the test dataset. Since a random draft is conducted in the subgraph contrast component, causing the unstable contrastive loss, we only report the BCE loss of our proposed SCS method, which could also help us directly compare the convergence speed under the same loss function. The results are listed in Fig. 5. The dashed lines represent the loss, while the AUC value is shown in solid lines. It can be observed that the SEAL method (green lines) requires more epochs to converge, while LGLP (red lines) and our proposed SCS (blue lines) reach convergence within 5 to 10 epochs.

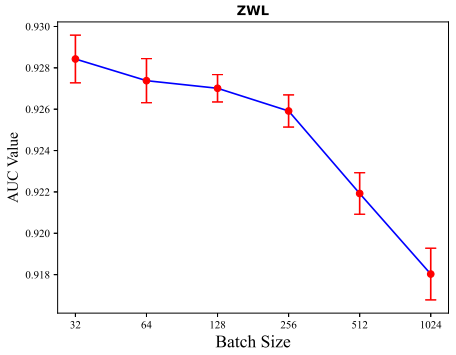

Fig. 6. AUC performance with different hyperparameters of batch size. Other hyperparameters remain the same, with the balance ratio $\alpha = 0.1$ and contrastive sample number $m = 2$.

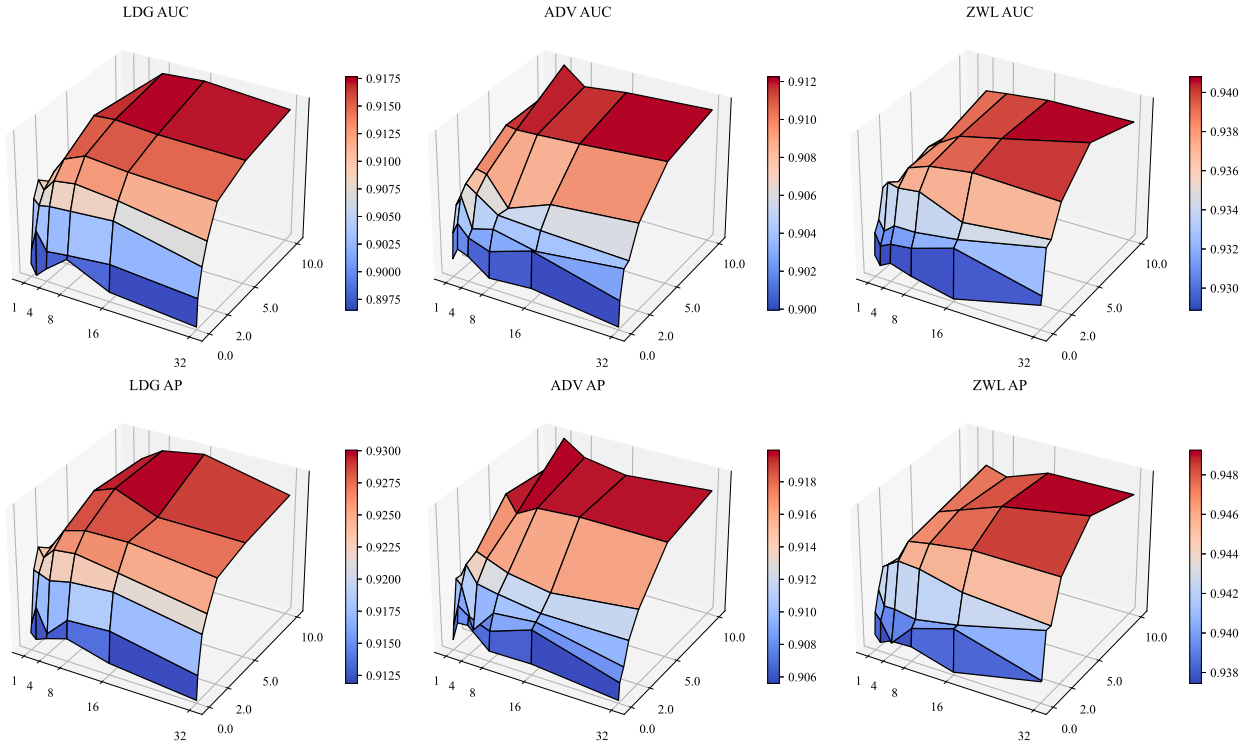

Fig. 7. AUC and AP performance across different combinations of hyperparameters: balance ratio α and draft number m in contrastive learning. The X-axis represents the draft number m, with values [1, 2, 4, 8, 16, 32], while the Y-axis denotes the balance ratio α, with values [0, 0.1, 0.5, 1.0, 2.0, 5.0, 10.0], with the batch size fixed at $n = 32$.

In addition, our proposed method achieves the lowest BCE loss on the training set, indicating its better learning and representation ability than SEAL and LGLP methods.

4.3.8. Hyperparameter analysis

Unlike traditional contrastive methods, our model does not treat all other samples in a batch as negative ones but generates a specific negative subgraph for each. As a result, the model's performance may decline slightly because a larger batch size implies fewer parameter updates. The experimental findings on batch size validate this inference, as elaborated in Fig. 6.

The hyperparameter experiment examines the impact of the balance ratio α and the draft number m within the contrastive learning component on link prediction performance. We analyzed this hyperparameter using the LDG, ADV, and ZWL datasets. As shown in Fig. 7, once m reaches a value of 8, further increases do not significantly affect the AUC and AP values, while the balance ratio α has a substantial impact. This result is expected, as subgraph information tends to be sufficient without requiring complete training. During the contrastive learning process, the loss described by Equation (12) is distributed equally among all nodes in a subgraph, making a higher balance ratio more critical for effectively extracting subgraph knowledge at each node. Furthermore, when the balance ratio α is set to zero, the contrastive learning component of the proposed SCS method becomes ineffective, resulting in the

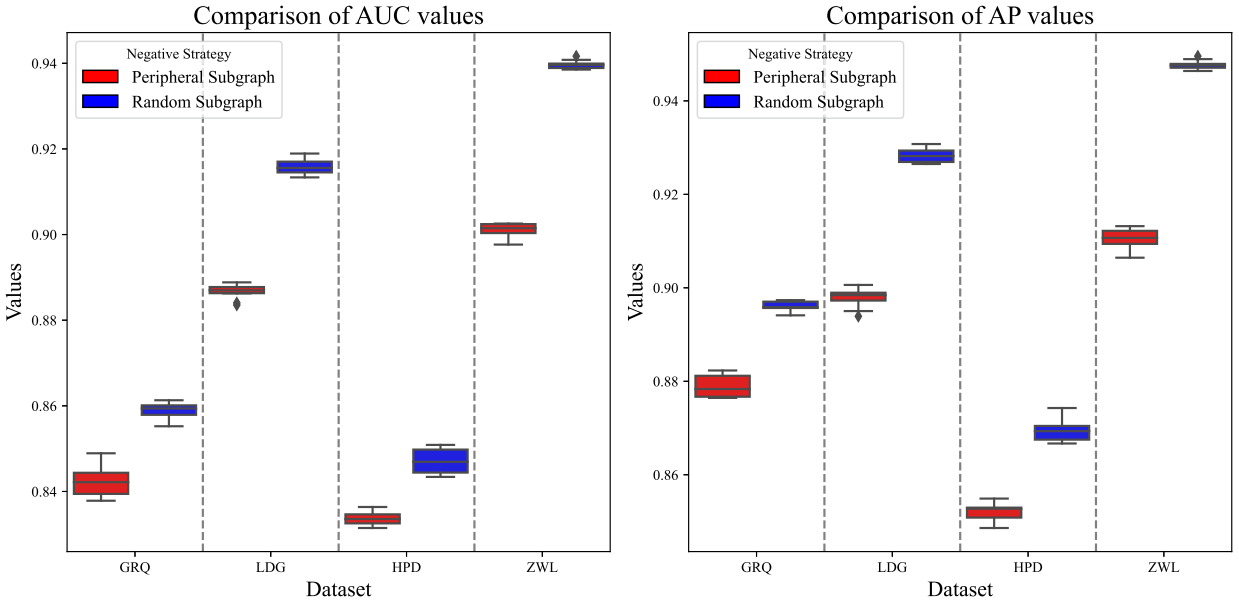

Fig. 8. AUC and AP performance across different negative sample strategies on four plain graphs.

poorest performance across all datasets. This finding reinforces the idea that subgraph learning offers valuable insights into the link prediction task and supports the effectiveness of our model. We recommend using a small sampling number m and a large balance ratio α to maximize contrastive learning effectiveness.

4.3.9. Negative generation

First, we define peripheral subgraphs (PS), the purpose of such subgraphs will be explained shortly. Here we define peripheral subgraphs $PS(S)$ of a given subgraph S as follows:

$$PS(S) = \{v_i | v_i \notin S, \text{ and } \min\{d(v_i, v_j) = 1\}, \forall v_j \in S\}, \tag{16}$$

where $d(v_i, v_j)$ stands for the distance between node v_i and node v_j. Essentially, this definition encapsulates the notion that the peripheral subgraph consists of nodes surrounding the given subgraph S.

The rationale behind introducing peripheral subgraphs lies in facilitating the learning process of subgraph feature extraction to discern whether nodes belong to the actual subgraph or not. Therefore, the information encapsulated in the surrounding layer of the subgraph (peripheral subgraphs) can be perceived as a form of *hard negative*, which might assist in training the model to distinguish between nodes within the actual subgraph and those outside.

We compared random negative sampling with using peripheral subgraphs as negative samples. Numerical results are demonstrated in Fig. 8, with a configuration of $m = 8$ and $\alpha = 10.0$. Our experiment revealed that the effectiveness of using peripheral subgraphs as negative samples was notably inferior to the random sample (Equation (8)) generation method employed previously.

This discrepancy arises because peripheral nodes used as negatives often appear in other actual subgraphs as positives. This overlap creates conflicting supervision signals, making random sampling a more reliable negative strategy.

It is worth emphasizing that the negative generation holds significant potential for further exploration. Our study merely scratches the surface with a simple numerical analysis. There is ample room for future research to delve deeper into the intricacies of negative sample generation methods and their impacts on contrastive learning.

4.3.10. Case study

In the fight against criminal networks, link prediction has emerged as a valuable analytical tool for uncovering hidden relationships and anticipating future interactions. In this case study, we examine the wiretap records criminal network (N_{WR}) from [46], focusing on its largest connected component comprising 176 nodes and 243 edges. Despite its small size, this network presents significant challenges due to its incomplete and potentially noisy structure, a common issue in criminal intelligence data where many connections remain unobserved.

To evaluate the predictive ability of our SCS model in such noisy scenarios, we randomly removed 10 edges and trained the model on the remaining links. As shown in Fig. 9, the cumulative distribution function (CDF) of predicted connection probabilities indicates that the removed (ground-truth) edges consistently rank among the highest-scoring predictions, whereas roughly 70% of unconnected node pairs are assigned scores below 0.5. This result highlights SCS's capacity to identify plausible links under high uncertainty without relying on dense connectivity.

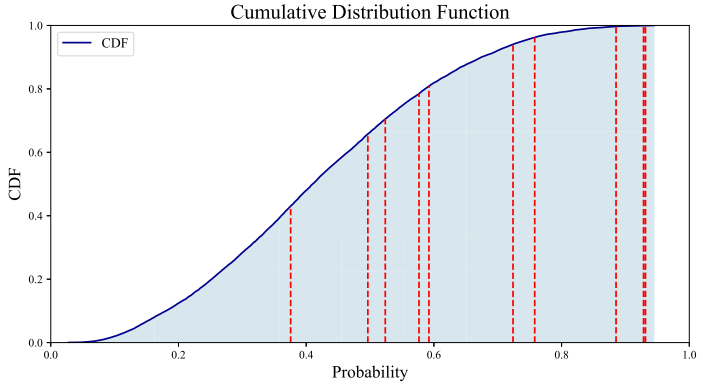

Fig. 9. The cumulative distribution function of edge connection probability yield by SCS method, connection probabilities of previously deleted test edges are highlighted in red.

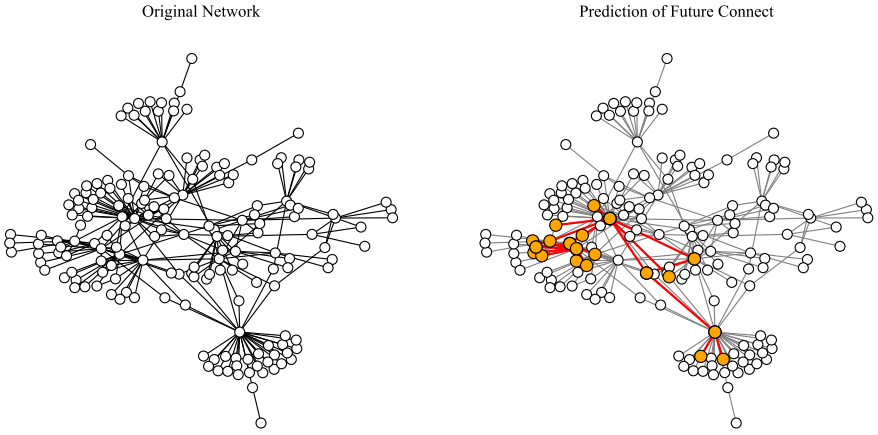

Fig. 10. The figure presents a comparison between the original graph (left) and the graph after SCS has been applied (right).

In noisy real-world networks, peripheral connections are often unreliable or missing. At the same time, hub nodes preserve more stable and meaningful relationships due to their higher connectivity and structural centrality. SCS implicitly prioritizes such nodes, as their frequent occurrence in subgraphs increases their impact on contrastive loss optimization.

Empirical results from the case study support this hypothesis. As shown in Fig. 10, predicted links predominantly cluster around hubs and dense regions. This structural preference allows SCS to recover critical substructures even under significant noise, reinforcing its robustness in sparse and imperfect real-world networks.

5. Conclusion and future work

This study presents a novel end-to-end graph neural network for link prediction that leverages contrastive learning to integrate neighborhood information without explicitly extracting subgraphs. Our method bypasses the costly enumeration of h-hop subgraph enumeration, reducing memory overhead and improving scalability. Extensive experiments on plain and attributed graphs demonstrate that SCS consistently outperforms state-of-the-art baselines while maintaining efficient training and fast convergence. A key contribution of our work is a variance-guided negative sampling strategy, which generates random subgraphs to approximate the latent space mean and promote more informative and discriminative embeddings. This approach offers a principled and effective solution to the challenge of contrastive supervision in structural learning tasks.

Looking forward, several extensions are worth exploring. First, our current subgraph encoder uses a simple average pooling strategy. More expressive readout functions, such as attention-based or structural-aware aggregators, may further enhance subgraph representation. Second, rather than relying on fixed-radius neighborhoods, future work could incorporate adaptive subgraph selection guided by the graph information bottleneck principle [47] to focus learning on task-relevant structure. Finally, simple heuristics such as Common Neighbors and SimRank remain surprisingly effective in sparse settings. Exploring ways to incorporate these domain-specific priors into neural architectures, potentially through contrastive formulations, offers a promising avenue for improving performance and interpretability in real-world networks.

CRediT authorship contribution statement

Qiming Yang: Writing – review & editing, Writing – original draft, Visualization, Software, Methodology, Formal analysis, Conceptualization. **Wei Wei:** Writing – review & editing, Writing – original draft, Supervision, Methodology, Funding acquisition. **Ruizhi Zhang:** Writing – original draft, Data curation, Conceptualization. **Xiangnan Feng:** Writing – review & editing, Writing – original draft, Validation, Project administration.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work

Acknowledgements

This work was supported by the National Natural Science Foundation of China (Grant Nos. 62276013, 62141605, 62050132), the Beijing Natural Science Foundation (Grant No. 1192012), and the Fundamental Research Funds for the Central Universities.

Data availability

Data will be made available on request.

References

[1] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, S.Y. Philip, A comprehensive survey on graph neural networks, IEEE Trans. Neural Netw. Learn. Syst. 32 (1) (2020) 4–24.
[2] S. Fortunato, Community detection in graphs, Phys. Rep. 486 (3–5) (2010) 75–174.
[3] T.N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, arXiv preprint, arXiv:1609.02907, 2016.
[4] J. Lee, I. Lee, J. Kang, Self-attention graph pooling, in: International Conference on Machine Learning, PMLR, 2019, pp. 3734–3743.
[5] L. Lü, T. Zhou, Link prediction in complex networks: a survey, Phys. A, Stat. Mech. Appl. 390 (6) (2011) 1150–1170.
[6] V. Martínez, F. Berzal, J.-C. Cubero, A survey of link prediction in complex networks, ACM Comput. Surv. 49 (4) (2016) 1–33.
[7] P. Wang, B. Xu, Y. Wu, X. Zhou, Link prediction in social networks: the state-of-the-art, arXiv preprint, arXiv:1411.5118, 2014.
[8] F. Xie, Z. Chen, J. Shang, X. Feng, J. Li, A link prediction approach for item recommendation with complex number, Knowl.-Based Syst. 81 (2015) 148–158.
[9] C. Zhao, S. Liu, F. Huang, S. Liu, W. Zhang, Csgnn: contrastive self-supervised graph neural network for molecular interaction prediction, in: IJCAI, 2021, pp. 3756–3763.
[10] S.M. Kazemi, D. Poole, Simple embedding for link prediction in knowledge graphs, Adv. Neural Inf. Process. Syst. 31 (2018).
[11] M. Zhang, Y. Chen, Weisfeiler-Lehman neural machine for link prediction, in: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2017, pp. 575–583.
[12] M. Zhang, Y. Chen, Link prediction based on graph neural networks, Adv. Neural Inf. Process. Syst. 31 (2018).
[13] L. Cai, J. Li, J. Wang, S. Ji, Line graph neural networks for link prediction, IEEE Trans. Pattern Anal. Mach. Intell. 44 (9) (2021) 5103–5113.
[14] M. Zhang, Z. Cui, M. Neumann, Y. Chen, An end-to-end deep learning architecture for graph classification, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 32, 2018.
[15] A. Jaiswal, A.R. Babu, M.Z. Zadeh, D. Banerjee, F. Makedon, A survey on contrastive self-supervised learning, Technologies 9 (1) (2020) 2.
[16] Y. Liu, M. Jin, S. Pan, C. Zhou, Y. Zheng, F. Xia, P. Yu, Graph self-supervised learning: a survey, IEEE Trans. Knowl. Data Eng. (2022).
[17] D. Liben-Nowell, J. Kleinberg, The link-prediction problem for social networks, J. Am. Soc. Inf. Sci. Technol. 58 (7) (2007) 1019–1031.
[18] L.A. Adamic, E. Adar, Friends and neighbors on the web, Soc. Netw. 25 (3) (2003) 211–230.
[19] D. Zhang, J. Yin, X. Zhu, C. Zhang, Network representation learning: a survey, IEEE Trans. Big Data 6 (1) (2018) 3–28.
[20] N.M. Ahmed, L. Chen, Y. Wang, B. Li, Y. Li, W. Liu, Deepeye: link prediction in dynamic networks based on non-negative matrix factorization, Big Data Min. Anal. 1 (1) (2018) 19–33.
[21] D. Heckerman, C. Meek, D. Koller, Probabilistic entity-relationship models, PRMs, and plate models, in: Introduction to Statistical Relational Learning 2007, 2007, pp. 201–238.
[22] B. Perozzi, R. Al-Rfou, S. Skiena, Deepwalk: online learning of social representations, in: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2014, pp. 701–710.
[23] A. Grover, J. Leskovec, node2vec: scalable feature learning for networks, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 855–864.
[24] R. Karami, S.M. Vahidipour, A. Rezvanian, Seal+: a subgraph-enhanced framework for link prediction with graph neural networks, J. Ind. Inf. Integr. (2025) 100802.
[25] L. Chen, Y. Zhao, A. Sajjanhar, Hypergraph neural networks based on enclosing subgraph extraction for link prediction, in: 2024 IEEE 30th International Conference on Parallel and Distributed Systems (ICPADS), IEEE, 2024, pp. 455–461.
[26] L. Wu, H. Lin, C. Tan, Z. Gao, S.Z. Li, Self-supervised learning on graphs: contrastive, generative, or predictive, IEEE Trans. Knowl. Data Eng. (2021).
[27] P. Velickovic, W. Fedus, W.L. Hamilton, P. Liò, Y. Bengio, R.D. Hjelm, Deep graph infomax, ICLR (Poster) 2 (3) (2019) 4.
[28] Y. You, T. Chen, Y. Sui, T. Chen, Z. Wang, Y. Shen, Graph contrastive learning with augmentations, Adv. Neural Inf. Process. Syst. 33 (2020) 5812–5823.
[29] Y. Zhu, Y. Xu, F. Yu, Q. Liu, S. Wu, L. Wang, Deep graph contrastive representation learning, arXiv preprint, arXiv:2006.04131, 2020.
[30] S. Thakoor, C. Tallec, M.G. Azar, M. Azabou, E.L. Dyer, R. Munos, P. Veličković, M. Valko, Large-scale representation learning on graphs via bootstrapping, arXiv preprint, arXiv:2102.06514, 2021.
[31] Z. Zhang, S. Sun, G. Ma, C. Zhong, Line graph contrastive learning for link prediction, Pattern Recognit. 140 (2023) 109537.
[32] L. Liu, Q. Xie, W. Wen, J. Zhu, M. Peng, Edge contrastive learning for link prediction, Inf. Process. Manag. 61 (6) (2024) 103847.
[33] Y. Yao, P. Guo, Z. Mao, Z. Ti, Y. He, F. Nian, R. Zhang, N. Ma, Multi-scale contrastive learning via aggregated subgraph for link prediction, Appl. Intell. 55 (6) (2025) 1–20.
[34] K. Xu, C. Li, Y. Tian, T. Sonobe, K.-i. Kawarabayashi, S. Jegelka, Representation learning on graphs with jumping knowledge networks, in: International Conference on Machine Learning, PMLR, 2018, pp. 5453–5462.

[35] F.-Y. Sun, J. Hoffmann, V. Verma, J. Tang, Infograph: unsupervised and semi-supervised graph-level representation learning via mutual information maximization, arXiv preprint, arXiv:1908.01000, 2019.
[36] S. Yun, S. Kim, J. Lee, J. Kang, H.J. Kim, Neo-gnns: neighborhood overlap-aware graph neural networks for link prediction, Adv. Neural Inf. Process. Syst. 34 (2021) 13683–13694.
[37] X. Zhong, H. Liang, On the scale-free property of citation networks: an empirical study, in: Companion Proceedings of the ACM on Web Conference 2024, 2024, pp. 541–544.
[38] D.J. Watts, S.H. Strogatz, Collective dynamics of 'small-world' networks, Nature 393 (6684) (1998) 440–442.
[39] M.E. Newman, The structure of scientific collaboration networks, Proc. Natl. Acad. Sci. 98 (2) (2001) 404–409.
[40] Z. Yang, W. Cohen, R. Salakhudinov, Revisiting semi-supervised learning with graph embeddings, in: International Conference on Machine Learning, PMLR, 2016, pp. 40–48.
[41] K. Wang, Z. Shen, C. Huang, C.-H. Wu, Y. Dong, A. Kanakia, Microsoft academic graph: when experts are not enough, Quant. Sci. Stud. 1 (1) (2020) 396–413.
[42] D. Szklarczyk, A.L. Gable, D. Lyon, A. Junge, S. Wyder, J. Huerta-Cepas, M. Simonovic, N.T. Doncheva, J.H. Morris, P. Bork, et al., String v11: protein–protein association networks with increased coverage, supporting functional discovery in genome-wide experimental datasets, Nucleic Acids Res. 47 (D1) (2019) D607–D613.
[43] G. Jeh, J. Widom, Simrank: a measure of structural-context similarity, in: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2002, pp. 538–543.
[44] T.N. Kipf, M. Welling, Variational graph auto-encoders, arXiv preprint, arXiv:1611.07308, 2016.
[45] L. Van der Maaten, G. Hinton, Visualizing data using t-sne, J. Mach. Learn. Res. 9 (11) (2008).
[46] G. Berlusconi, F. Calderoni, N. Parolini, M. Verani, C. Piccardi, Link prediction in criminal networks: a tool for criminal intelligence analysis, PLoS ONE 11 (4) (2016) e0154244.
[47] T. Wu, H. Ren, P. Li, J. Leskovec, Graph information bottleneck, Adv. Neural Inf. Process. Syst. 33 (2020) 20437–20448.